



---

The following document contains information on Cypress products. Although the document is marked with the name “Spansion” and “Fujitsu”, the company that originally developed the specification, Cypress will continue to offer these products to new and existing customers.

**Continuity of Specifications**

There is no change to this document as a result of offering the device as a Cypress product. Any changes that have been made are the result of normal document improvements and are noted in the document history page, where supported. Future revisions will occur when appropriate, and changes will be noted in a document history page.

**Continuity of Ordering Part Numbers**

Cypress continues to support existing part numbers. To order these products, please use only the Ordering Part Numbers listed in this document.

**For More Information**

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

**About Cypress**

Cypress (NASDAQ: CY) delivers high-performance, high-quality solutions at the heart of today’s most advanced embedded systems, from automotive, industrial and networking platforms to highly interactive consumer and mobile devices. With a broad, differentiated product portfolio that includes NOR flash memories, F-RAM™ and SRAM, Traveo™ microcontrollers, the industry’s only PSoC® programmable system-on-chip solutions, analog and PMIC Power Management ICs, CapSense® capacitive touch-sensing controllers, and Wireless BLE Bluetooth® Low-Energy and USB connectivity solutions, Cypress is committed to providing its customers worldwide with consistent innovation, best-in-class support and exceptional system value.

**FR60Lite**  
32-BIT MICROCONTROLLER  
**MB91265A Series**  
**HARDWARE MANUAL**



# **FR60Lite**

## **32-BIT MICROCONTROLLER**

# **MB91265A Series**

# **HARDWARE MANUAL**

<p>Be sure to refer to the “Check Sheet” for the latest cautions on development.</p>
--

“Check Sheet” is seen at the following support page

URL : <http://www.fujitsu.com/global/services/microelectronics/product/micom/support/index.html>

“Check Sheet” lists the minimal requirement items to be checked to prevent problems beforehand in system development.

**FUJITSU LIMITED**



# PREFACE

## ■ Purpose of this document and intended reader

We sincerely thank you for your continued use of Fujitsu semiconductor products.

The MB91265A series is a line of single-chip microcontrollers based on a 32-bit high-performance RISC CPU and integrating a variety of I/O resources and bus control mechanisms for embedded control applications which require high-performance, high-speed CPU processing. While focusing on external bus access to support vast address space to be accessed by the 32-bit CPU, the MB91265A series contains RAM (for data) to speed up instruction execution by the CPU.

The MB91265A series is designed to be best suited for embedded applications which require high-performance processing power of the CPU, such as digital video cameras, navigation systems, and DVD players.

The MB91265A series is a line of CPUs in the FR60Lite implemented by enhancing bus access for faster applications based on the FR30/40 of CPUs.

This manual describes the functions and operations of the MB91265A series for engineers who develop products using the MB91265A series. Please read through this manual.

For more information on various instructions, refer to "Instruction Manual".

Note: FR, the abbreviation of FUJITSU RISC controller, is a line of products of FUJITSU Limited.

## ■ Trademark

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

## ■ Organization of this document

This manual consists of the following 20 chapters and appendix.

### **CHAPTER 1 OVERVIEW**

This chapter provides basic information required to understand the MB91265A series, and covers features, a block diagram, and package dimension.

### **CHAPTER 2 HANDLING THE DEVICE**

This chapter provides precautions on handling the MB91265A series.

### **CHAPTER 3 CPU AND CONTROL UNIT**

This chapter provides basic information required to understand the CPU core functions of the MB91265A series. It covers architecture, specifications, and instructions.

### **CHAPTER 4 I/O PORT**

This chapter explains the overview of the I/O ports and the configuration and functions of registers.

### **CHAPTER 5 INTERRUPT CONTROLLER**

This chapter explains the overview of the interrupt controller, the configuration and functions of registers, and interrupt controller operation.

### **CHAPTER 6 EXTERNAL INTERRUPT AND NMI CONTROLLER**

This chapter describes the overview of the external interrupt and NMI controller, the configuration and functions of registers, and operation of the external interrupt and NMI controller.

### **CHAPTER 7 REALOS-RELATED HARDWARE**

The REALOS-related hardware is used by the real-time OS. Accordingly, these functions cannot be used by user programs if using REALOS. This chapter explains the overview of the delayed interrupt module and bit search module, the configuration and functions of registers, and the operation of the delayed interrupt module and bit search module.

### **CHAPTER 8 RELOAD TIMER**

This chapter describes the overview of the reload timer, the configuration and functions of registers, and the reload timer operation.

### **CHAPTER 9 TIMING GENERATOR**

This chapter explains the overview of the timing generator, the configuration and functions of registers, and operation of the timing generator.

### **CHAPTER 10 PPG**

This chapter explains the overview of the PPG, the configuration and functions of registers, and the PPG operation.

### **CHAPTER 11 PWC (PULSE WIDTH COUNT)**

This chapter explains the overview of the pulse width counter (PWC), the register configuration and functions and the counter operation.

### **CHAPTER 12 MULTIFUNCTION TIMER**

This chapter explains the overview of the multifunction timer, the configuration and functions of registers, and operation of the multifunction timer.

### **CHAPTER 13 U-TIMER (16-BIT TIMER FOR UART BAUD RATE GENERATION)**

This chapter explains the overview of the U-TIMER, the configuration and functions of registers, and U-TIMER operation.

## **CHAPTER 14 UART**

This chapter explains the overview of the UART, the configuration and functions of registers, and UART operation.

## **CHAPTER 15 C-CAN**

This chapter explains the functions and the operation of C-CAN.

## **CHAPTER 16 8/10-BIT A/D CONVERTER**

This chapter explains the overview of the 8/10-bit A/D converter, the configuration and functions of registers, and 8/10-bit A/D converter operation.

## **CHAPTER 17 16-BIT MAC**

This chapter explains the overview of 16-BIT MAC, the configuration and functions of registers, and the operation of 16-BIT MAC.

## **CHAPTER 18 DMAC (DMA CONTROLLER)**

This chapter explains the overview of the DMA controller (DMAC), the configuration and functions of registers, and DMAC operation.

## **CHAPTER 19 FLASH MEMORY**

This chapter explains the overview of the flash memory, the configuration and functions of registers, and the flash memory operation.

## **CHAPTER 20 SERIAL PROGRAMMING CONNECTION**

This chapter explains the basic configuration for serial programming, pins used for serial on-board programming, the connection example for serial programming, and system configuration for flash microcontroller programmer.

## **APPENDIX**

The appendix describes pin states in each CPU state, notes on using the little-endian areas, a list of FR family instructions, and notes on using MB91265A series.



- The contents of this document are subject to change without notice.  
Customers are advised to consult with sales representatives before ordering.
- The information, such as descriptions of function and application circuit examples, in this document are presented solely for the purpose of reference to show examples of operations and uses of FUJITSU semiconductor device; FUJITSU does not warrant proper operation of the device with respect to use based on such information. When you develop equipment incorporating the device based on such information, you must assume any responsibility arising out of such use of the information. FUJITSU assumes no liability for any damages whatsoever arising out of the use of the information.
- Any information in this document, including descriptions of function and schematic diagrams, shall not be construed as license of the use or exercise of any intellectual property right, such as patent right or copyright, or any other right of FUJITSU or any third party or does FUJITSU warrant non-infringement of any third-party's intellectual property right or other right by using such information. FUJITSU assumes no liability for any infringement of the intellectual property rights or other rights of third parties which would result from the use of information contained herein.
- The products described in this document are designed, developed and manufactured as contemplated for general use, including without limitation, ordinary industrial use, general office use, personal use, and household use, but are not designed, developed and manufactured as contemplated (1) for use accompanying fatal risks or dangers that, unless extremely high safety is secured, could have a serious effect to the public, and could lead directly to death, personal injury, severe physical damage or other loss (i.e., nuclear reaction control in nuclear facility, aircraft flight control, air traffic control, mass transport control, medical life support system, missile launch control in weapon system), or (2) for use requiring extremely high reliability (i.e., submersible repeater and artificial satellite).  
Please note that FUJITSU will not be liable against you and/or any third party for any claims or damages arising in connection with above-mentioned uses of the products.
- Any semiconductor devices have an inherent chance of failure. You must protect against injury, damage or loss from such failures by incorporating safety design measures into your facility and equipment such as redundancy, fire protection, and prevention of over-current levels and other abnormal operating conditions.
- Exportation/release of any products described in this document may require necessary procedures in accordance with the regulations of the Foreign Exchange and Foreign Trade Control Law of Japan and/or US export control laws.
- The company names and brand names herein are the trademarks or registered trademarks of their respective owners.

# CONTENTS

<b>CHAPTER 1</b>	<b>OVERVIEW .....</b>	<b>1</b>
1.1	Overview .....	2
1.2	Block Diagram .....	4
1.3	Pin Assignment .....	5
1.3.1	Package Dimension .....	6
1.4	List of Pin Functions .....	7
1.5	I/O Circuit Type .....	13
<b>CHAPTER 2</b>	<b>HANDLING THE DEVICE .....</b>	<b>17</b>
2.1	Precautions on Handling the Device .....	18
<b>CHAPTER 3</b>	<b>CPU AND CONTROL UNIT .....</b>	<b>21</b>
3.1	Memory Space .....	22
3.2	Memory Map .....	23
3.3	Internal Architecture .....	24
3.4	Basic Programming Model .....	29
3.4.1	Registers .....	30
3.5	Data Structure .....	36
3.6	Memory Map .....	38
3.7	Divergence Instructions .....	39
3.8	EIT (Exception, Interruption, and Trap) .....	42
3.9	Operating Mode .....	56
3.10	Reset (Device Initialization) .....	58
3.10.1	Reset Level .....	59
3.10.2	Reset Factor .....	60
3.10.3	Reset Sequence .....	62
3.10.4	Oscillation Stabilization Wait Time .....	63
3.10.5	Reset Operation Mode .....	65
3.11	Clock Generation Control .....	66
3.11.1	Selection of the Source Clock .....	67
3.11.2	PLL Control .....	68
3.11.3	Oscillation Stabilization Wait and PLL Lock Wait Time .....	69
3.11.4	Clock Distribution .....	71
3.11.5	Clock Divider .....	72
3.11.6	Block Diagram of Clock Generation Control Unit .....	73
3.11.7	Explanation of Register Details for Clock Generation Control Unit .....	74
3.11.8	Peripheral Circuit Functions in the Clock Controller .....	87
3.12	Device State Control .....	90
<b>CHAPTER 4</b>	<b>I/O PORT .....</b>	<b>97</b>
4.1	Basic Block Diagram of the I/O Port .....	98
4.1.1	I/O Port with Pull-up Resistor .....	99

4.2	Registers of the I/O Port .....	100
4.3	Analog Input Port .....	106
<b>CHAPTER 5</b>	<b>INTERRUPT CONTROLLER .....</b>	<b>109</b>
5.1	Overview .....	110
5.2	Interrupt Controller Registers .....	111
5.3	Block Diagram .....	113
5.4	Register Details Explanation .....	114
5.5	Operation of the Interrupt Controller .....	116
<b>CHAPTER 6</b>	<b>EXTERNAL INTERRUPT AND NMI CONTROLLER .....</b>	<b>125</b>
6.1	External Interrupt and NMI Controller .....	126
<b>CHAPTER 7</b>	<b>REALOS-RELATED HARDWARE .....</b>	<b>135</b>
7.1	Delayed Interrupt Module .....	136
7.2	Bit Search Module .....	138
<b>CHAPTER 8</b>	<b>RELOAD TIMER .....</b>	<b>143</b>
8.1	Overview .....	144
8.2	Block Diagram .....	145
8.3	Reload Timer Register .....	146
8.4	Operation of Reload Timer .....	150
8.5	Notes on Using the 16-bit Reload Timer .....	154
<b>CHAPTER 9</b>	<b>TIMING GENERATOR .....</b>	<b>155</b>
9.1	Timing Generator .....	156
<b>CHAPTER 10</b>	<b>PPG .....</b>	<b>161</b>
10.1	Overview .....	162
10.2	Block Diagram .....	165
10.3	Register of PPG .....	169
10.4	Operation Explanation .....	174
<b>CHAPTER 11</b>	<b>PWC (PULSE WIDTH COUNT) .....</b>	<b>179</b>
11.1	Overview .....	180
11.2	PWC Block Diagram .....	181
11.3	PWC Register .....	182
11.4	Operation Explanation .....	189
<b>CHAPTER 12</b>	<b>MULTIFUNCTION TIMER .....</b>	<b>201</b>
12.1	Overview .....	202
12.2	Block Diagram .....	204
12.3	Pins of the Multifunction Timer .....	211
12.4	Multifunction Timer Register .....	212
12.4.1	Compare Clear Buffer Register (CPCLRBH0 to CPCLRBH2, CPCLRBL0 to CPCLRBL2) / Compare Clear Register (CPCLRHO to CPCLR H2, CPCLRL0 to CPCLRL2) .....	218

12.4.2	Timer Data Register(TCDTH0 to TCDTH2, TCDTL0 to TCDTL2)	220
12.4.3	Timer State Control Register(TCCSH0 to TCCSH2, TCCSL0 to TCCSL2)	221
12.4.4	A/D Trigger Control Register (ADTRGC)	226
12.4.5	Output Compare Buffer Register (OCCPBH0 to OCCPBH5, OCCPBL0 to OCCPBL5) / Output Compare Register (OCCPH0 to OCCPH5, OCCPL0 to OCCPL5)	228
12.4.6	Compare Control Register(OCSH0 to OCSH5, OCSL0 to OCSL5)	230
12.4.7	Compare Mode Control Register (OCMOD)	236
12.4.8	Input Capture Data Register(IPCPH0 to IPCPH3, IPCPL0 to IPCPL3)	238
12.4.9	Input Capture State Control/PPG Output Control Register (ICSH23, ICSL23, PICSH01, PICSL01)	239
12.4.10	16-bit Dead Timer Register(TMRRH0 to TMRRH2, TMRRL0 to TMRRL2)	247
12.4.11	16-bit Dead Timer Control Register (DTCR0 to DTCR2)	248
12.4.12	Waveform Control Register (SIGCR1/SIGCR2)	257
12.4.13	A/D activation compare register (ADCOMP1, ADCOMP2, ADCOMPC1, ADCOMPC2)	260
12.4.14	Free-run Timer Selector Register (FSR0 to FSR2)	262
12.5	Multifunction Timer Interrupt	264
12.6	Operation of the Multifunction Timer	267
12.6.1	Operation of 16-bit Free-run Timer	268
12.6.2	Operation of 16-bit Output Compare	275
12.6.3	16-bit Input Capture Operation	286
12.6.4	Waveform Generator Operation	288
12.6.4.1	Operation of Timer Mode	292
12.6.4.2	Operation during Dead Time Timer Mode	293
12.6.4.3	DTTI pin Control Operation	297
12.6.5	A/D Activation Compare Operation	299
12.7	Notes on Using the Multifunction Timers	301
12.8	Example Program for Multifunction Timer	303

## **CHAPTER 13 U-TIMER (16-BIT TIMER FOR UART BAUD RATE GENERATION)**

		<b>307</b>
13.1	Overview	308
13.2	Operation Explanation	312

## **CHAPTER 14 UART 313**

14.1	Overview	314
14.2	Register Details Explanation	317
14.3	Operation of UART	324
14.4	Application Example	330
14.5	Examples of Baud Rate and U-TIMER Reload Value Settings	332

## **CHAPTER 15 C-CAN 333**

15.1	Features of C-CAN	334
15.2	Block Diagram of C-CAN	335
15.3	Registers of C-CAN	336
15.4	Function of C-CAN Register	342
15.4.1	Overall Control Register	343
15.4.1.1	CAN Control Register (CTRLR)	344

15.4.1.2	CAN Status Register (STATR)	347
15.4.1.3	CAN Error Counter (ERRCNT)	350
15.4.1.4	CAN Bit Timing Register (BTR)	351
15.4.1.5	CAN Interrupt Register (INTR)	352
15.4.1.6	CAN Test Register (TESTR)	353
15.4.1.7	CAN Prescaler Extended Register (BRPER)	355
15.4.2	Message Interface Register	356
15.4.2.1	IFx Command Request Register (IFxCREQ)	357
15.4.2.2	IFx Command Mask Register (IFxCMSK)	359
15.4.2.3	IFx Mask Registers 1 and 2 (IFxMSK1, IFxMSK2)	364
15.4.2.4	IFx Arbitration Registers 1 and 2 (IFxARB1, IFxARB2)	365
15.4.2.5	IFx Message Control Register (IFxMCTR)	366
15.4.2.6	IFx Data Registers A1, A2, B1 and B2 (IFxDTA1, IFxDTA2, IFxDTB1, IFxDTB2)	367
15.4.3	Message Object	369
15.4.4	Message Handler Register	375
15.4.4.1	CAN Transmission Request Registers (TREQR1, TREQR2)	376
15.4.4.2	CAN Data Renewal Registers (NEWDT1, NEWDT2)	378
15.4.4.3	CAN Interrupt Pending Registers (INTPND1, INTPND2)	380
15.4.4.4	CAN Message Enable Registers (MSGVAL1, MSGVAL2)	382
15.4.5	CAN Clock Prescaler Register	384
15.5	C-CAN Function	385
15.5.1	Message Object	386
15.5.2	Message Transmission Operation	388
15.5.3	Message Reception Operation	390
15.5.4	Function of FIFO Buffer	393
15.5.5	Function of Interrupt	395
15.5.6	Bit Timing	396
15.5.7	Test mode	399
15.5.8	Software Initialization	403
15.5.9	CAN Clock Prescaler	404

## **CHAPTER 16 8/10-BIT A/D CONVERTER ..... 407**

16.1	Overview	408
16.2	Configuration	409
16.3	Pin	412
16.4	Register	414
16.4.1	A/D Channel Control Register (ADCH)	415
16.4.2	A/D Mode Setting Register (ADMD)	417
16.4.3	A/D Control Status Register (ADCS)	420
16.4.4	A/D Data Register (ADCD)	423
16.4.5	Analog Input Control Register (AICR)	424
16.5	Interrupt	425
16.6	Operation Explanation	426
16.7	A/D Conversion Data Protection Function	430
16.8	Notes on Using 8/10-bit A/D Converter	431

<b>CHAPTER 17 16-BIT MAC .....</b>	<b>433</b>
17.1 Features .....	434
17.2 Defined Instruction .....	438
17.3 Explanation of Register .....	439
17.4 Operation Explanation .....	445
17.5 Instruction Detail Explanation .....	449
<b>CHAPTER 18 DMAC (DMA CONTROLLER) .....</b>	<b>455</b>
18.1 Overview .....	456
18.2 Register Details Explanation .....	459
18.3 Operation of DMAC (DMA Controller) .....	472
18.3.1 Setting up Transfer Requests .....	474
18.3.2 Transfer Sequence .....	475
18.3.3 General DMA Transfers .....	477
18.4 Operation Flowchart .....	486
18.5 Data Path .....	488
<b>CHAPTER 19 FLASH MEMORY .....</b>	<b>491</b>
19.1 Overview .....	492
19.2 Registers .....	498
19.2.1 Flash Memory Status Register (FLCR) .....	499
19.2.2 Flash Memory Wait Register (FLWC) .....	501
19.3 Access Mode of Flash Memory .....	503
19.4 Automatic Algorithm Startup Procedures .....	505
19.5 Automatic Algorithm Execution States .....	510
19.6 Dual-operations .....	515
<b>CHAPTER 20 SERIAL PROGRAMMING CONNECTION .....</b>	<b>517</b>
20.1 Overview .....	518
<b>APPENDIX .....</b>	<b>523</b>
APPENDIX A I/O Map .....	524
APPENDIX B Interrupt Vector .....	535
APPENDIX C Pin States in Each CPU State .....	538
APPENDIX D Notes when Little Endian Region is used .....	540
APPENDIX E Instruction List .....	545
APPENDIX F Notes on Using .....	560
<b>INDEX .....</b>	<b>563</b>











# Main changes in this edition

Page	Changes (For details, refer to main body.)
2	1.1 Overview is changed. ( "MB91F267NA" →"MB91F267NA/MB91267NA" )
3	1.1 Overview ● Internal Peripheral Functions is changed. ( "Table 1.1-1 Internal Peripheral Functions" is added. ) ( "• C-CAN 32MSB: 1 channel (loaded in only MB91F267NA)" is deleted. )
4	1.2 Block Diagram Figure1.2-1 Block Diagram is changed. ( "MB91F267NA only" →"MB91F267NA / MB91267NA" )
5	1.3 Pin Assignment ■ LQFP-64 (MB91F267A/MB91F267NA/MB91267A/MB91267NA) is changed. ( "MB91266A" →"MB91267A/MB91267NA" ) ( "only MB91F267NA" →"MB91F267NA/MB91267NA" )
7	1.4 List of Pin Functions Table 1.4-1 List of Pin Functions (1 / 6) is changed. ( "terminal" →"pin" )
8	1.4 List of Pin Functions Table 1.4-1 List of Pin Functions (2 / 6) is changed. ( "terminal" →"pin" )
9	1.4 List of Pin Functions Table 1.4-1 List of Pin Functions (3 / 6) is changed. ( "terminal" →"pin" )
10	1.4 List of Pin Functions Table 1.4-1 List of Pin Functions (4 / 6) is changed. ( "terminal" →"pin" )
11	1.4 List of Pin Functions Table 1.4-1 List of Pin Functions (5 / 6) is changed. ( "terminal" →"pin" )
12	1.4 List of Pin Functions Table 1.4-1 List of Pin Functions (6 / 6) is changed. ( "terminal" →"pin" ), ( "only MB91F267NA" →"MB91F267NA / MB91267NA" )
18	2.1 Precautions on Handling the Device ■ Precautions on Handling Device is changed. ( "terminal" →"pin" )



Page	Changes (For details, refer to main body.)
23	3.2 Memory Map ■ MB91F267A/MB91F267NA/MB91267A/MB91267NA Memory Map is changed. ( "MB91F267NA" →"MB91F267NA/MB91267A/MB91267NA" ), ( "MB91F267A/MB91F267NA" →"MB91F267A/MB91F267NA/MB91267A/MB91267NA" )
49	3.8 EIT (Exception, Interruption, and Trap) Table 3.8-3 Vector Table (3 / 3) is changed. ( "only MB91F267NA" →"MB91F267NA/MB91267NA" )
56	3.9 Operating Mode ■ Operating Mode is changed. ( "terminals" →"pins" )
59	3.10.1 Reset Level ■ Set Initialization Reset (INIT) is changed. ( "terminal" →"pin" )
60	3.10.2 Reset Factor ■ $\overline{\text{INIT}}$ Pin Input (Set Initialization Reset Pin) is changed. ( "terminal" →"pin" )
64	3.10.4 Oscillation Stabilization Wait Time ■ Select Oscillation Stabilization Wait Time is changed. ( "terminal" →"pin" )
69	3.11.3 Oscillation Stabilization Wait and PLL Lock Wait Time ■ Wait Time after Power Supply is Turned on is changed, ( "terminal" →"pin" )
76	3.11.7 Explanation of Register Details for Clock Generation Control Unit ■ STCR: Standby Control Register is changed. ( "terminal" →"pin" )
77	3.11.7 Explanation of Register Details for Clock Generation Control Unit ■ STCR: Standby Control Register is changed. ( "terminal" →"pin" )
81	3.11.7 Explanation of Register Details for Clock Generation Control Unit ■ TBCR: Time-base Counter Control Register is changed. ( "* Restrictions" →"Note" )
91	3.12 Device State Control ■ State of Device and Each Transition is changed. ( "terminal" →"pin" )
103	4.2 Registers of the I/O Port ■ Port Function Registers (PFR: PFR0, PFR1, PTFR0) is changed. ( "MB91F267NA" →"MB91F267NA/MB91267NA" )

Page	Changes (For details, refer to main body.)
104	4.2 Registers of the I/O Port  Port Function Registers (PFR: PFR0, PFR1, PTFR0) is changed. ( "MB91F267NA" →"MB91F267NA/MB91267NA" )
105	4.2 Registers of the I/O Port  Port Function Registers (PFR: PFR0, PFR1, PTFR0) is changed. ( "MB91F267NA" →"MB91F267NA/MB91267NA" )
117	5.5 Operation of the Interrupt Controller Table 5.5-1 Relationship among Interrupt Factors, Interrupt Numbers, and Interrupt Levels (1 / 3) is changed. ( "only MB91F267NA" →"MB91F267NA/MB91267NA" )
118	5.5 Operation of the Interrupt Controller Table 5.5-1 Relationship among Interrupt Factors, Interrupt Numbers, and Interrupt Levels (2 / 3) is changed. ( "only MB91F267NA" →"MB91F267NA/MB91267NA" )
120	5.5 Operation of the Interrupt Controller  Generation criteria is changed. ( "passed" →"issued" )
132	6.1 External Interrupt and NMI Controller Figure 6.1-6 Operational Sequence for Returning from STOP State by External Interrupt is changed. ( ""1" enable (Set before switching to STOP mode)" → ""1" (Set to enable before switching to STOP mode)" )
138	7.2 Bit Search Module is changed. ( "Searches for "0", "1"" →"Bit search module searches for "0", "1"" )
139	7.2 Bit Search Module  0-detect data register (BSD0) ( "irregular. The read value is indeterminate." →"undefined. The read value is undefined." )
139	7.2 Bit Search Module  1-detect data register (BSD1) is changed. ( "value written." →"written value." )
139	7.2 Bit Search Module  1-detect data register (BSD1) is changed. ( "irregular" →"undefined" )
139	7.2 Bit Search Module  Change point detection data register (BSDC) is changed. ( "value written" →"written value" )
139	7.2 Bit Search Module  Change point detection data register (BSDC) is changed. ( "The initial value by reset is irregular." →"The initial value by reset is undefined." )

Page	Changes (For details, refer to main body.)
139	7.2 Bit Search Module ● Change point detection data register (BSDC) is changed. ( "The read value is indeterminate." → "The read value is undefined." )
143	CHAPTER 8 RELOAD TIMER is changed. ( "Precautions on Using the 16-bit Reload Timer" → "Notes on Using the 16-bit Reload Timer" )
144	8.1 Overview ■ Reload Timer Registers is changed. ( "Not initialized" → "Undefined value" ), ( "Undefined bit" → "Undefined" )
146	8.3 Reload Timer Register ■ Control Status Register (TMCSR: TMCSR0 to TMCSR2) is changed. ( "ch.1" → "The ch.1" )
147	8.3 Reload Timer Register ■ Control Status Register (TMCSR: TMCSR0 to TMCSR2) is changed. ( "count continues when the counter" → "counting continues when the counter value" ), ( "count halts when the counter" → "counting halts when the counter value" )
148	8.3 Reload Timer Register ■ Control Status Register (TMCSR: TMCSR0 to TMCSR2) is changed. ( "Output interdiction" → "Output disabled" ), ( "starts the count." → "starts the counting." )
149	8.3 Reload Timer Register ■ TMR Register (16-bit Timer Register) is changed. ( "irregular" → "undefined" ), ( "Not initialized" → "Undefined value" ), ( "Undefined bit" → "Undefined" )
151	8.4 Operation of Reload Timer ■ Underflow Operation is changed. ( "count" → "counting" )
154	8.5 Notes on Using the 16-bit Reload Timer is changed. ( "Precautions" → "Notes" )
154	8.5 Notes on Using the 16-bit Reload Timer ■ Notes is changed. ( "cleared at the same timing" → "cleared at the same time" )
159	9.1 Timing Generator ■ Operation Overview of Timing Generator is changed. ( "Operational" → "Operation" )
159	9.1 Timing Generator Figure 9.1-2 Operation/stop Timing of 8-bit Counter is changed., ( "Stop counter" → "Stop counting" ), ( "Stop counter by overflow" → "Stop counting by overflow" )

Page	Changes (For details, refer to main body.)
162	10.1 Overview is changed. ( "8 8-bit" →"8 × 8-bit"), ( "16 8-bit" →"16 × 8-bit" ), ( 8 external pulse outputs" →"8 × external pulse outputs" ) ( "8 interrupt output" →"and 8 × interrupt output" )
163	10.1 Overview ■ Registers of the PPG Timer is changed. ( "Not initialized" →"Undefined value" ), ( "Undefined bit" →"Undefined" )
164	10.1 Overview ● Reload register: 8-bit PPG mode is changed. ( "Not initialized" →"Undefined" )
164	10.1 Overview ● Reload register: 16-bit PPG mode is changed ( "Not initialized" →"Undefined" )
169	10.3 Register of PPG ■ PPGCn Register (PPGn Operation Mode Control Register) n=0,1,2,3,4,5,6,7 is changed. ( "Not initialized" →"Undefined"), ( "PPG 2 channels" →"PPG 2 channel"),
169	10.3 Register of PPG ■ PPGCn Register (PPGn Operation Mode Control Register) n=0,1,2,3,4,5,6,7 is changed. ( "The read and write are possible." →"Reading and writing are permitted." )
171	10.3 Register of PPG ■ PRLl/PRLH Register (Reload Register: PRLl0 to PRLl7/PRLH0 to PRLH7) is changed. ( "X: Undefined" is added )
172	10.3 Register of PPG ■ Output Inversion Register (REVC) is changed. ( "outputs, they also invert the initial levels." →"outputs simply, they also invert the initial levels." )
173	10.3 Register of PPG ■ GATE Function Control Register (GATEC) is changed. ( "Undefined bit" →"Undefined" ), ( "X: Undefined value" is added )
175	10.4 Operation Explanation ● PPG output operation is changed, ( "TRG register is set to 0. After having stopped, the pulse output holds L level" → "TRG register writes "0". After having stopped, the pulse output holds "L" level" )
182	11.3 PWC Register ■ PWC Control/Status Register (PWCSR: PWCSR0) is changed, ( "This bit starts, restarts, and stops" →"These bits start, restart, and stop" )
182	11.3 PWC Register ■ PWC Control/Status Register (PWCSR: PWCSR0) is changed. ( "R/W" is added )








Page	Changes (For details, refer to main body.)
182	11.3 PWC Register <b>■</b> PWC Control/Status Register (PWCSR: PWCSR0) is changed. ( "- : Undefined" is deleted )
183	11.3 PWC Register <b>■</b> PWC Control/Status Register (PWCSR: PWCSR0) is changed. ( "bit-processing" →"bit manipulation" ), ( "status must be in operation" ) →× "status will be in operation surely" )
185	<b>■</b> PWC Control/Status Register (PWCSR: PWCSR0) [bit5, bit4] PIS1,PIS0: Pulse width count input pin select bits is changed. ( "terminal" →"pin" ), ( "Read and write" ) →× "Reading and writing" )
186	<b>■</b> PWC Control/Status Register (PWCSR: PWCSR0) [bit2 to bit0] MOD2 to MOD0: Operation mode/measurement edge select bits is changed. ( "count mode" →"measurement mode" )
187	11.3 PWC Register <b>■</b> PWC Data Buffer Register (PWCR: PWCR0) is changed. ( "Reading" →"Only reading" )
188	11.3 PWC Register <b>■</b> Ratio of Dividing Frequency Control Register (PDIVR: PDIVR0) is changed. ( "Not initialized" →"Undefined value" ), ( "Undefined bit" →"Undefined" )
191	11.4 Operation Explanation <b>■</b> Select Operation Mode is changed. ( "Selects" →"Select" ), ( "measurement" →"measurement to" )
192	11.4 Operation Explanation <b>■</b> Starting and Stopping of the Pulse Width Count is changed. ( "other" →"by the bigger unit" )
193	11.4 Operation Explanation <b>■</b> Details of Pulse Width Count Operation is changed. ( "count end" →"measurement end interrupt" )
195	11.4 Operation Explanation ● Measurement mode and count operation is changed. ( "operation occurs" →"operations are executed" ), ( "be unpredictable" →"not be guaranteed" )
196	11.4 Operation Explanation ● Pulse width/cycle measurement range is changed. ( "frequency division" →"Divided by" ), ( "occurred" →"generation" ), ( "termination" →"end interrupt" )

Page	Changes (For details, refer to main body.)
198	11.4 Operation Explanation ● STRT bit and STOP bit of the PWCSR register is changed. ( "■PWCSR register" →"■ PWC Control/Status Register (PWCSR: PWCSR0)" ), ( "bit-processing" →"bit manipulation" )
198	11.4 Operation Explanation ● Minimum input pulse width is changed. ( "be unpredictable" →"not be guaranteed" )
213	12.4 Multifunction Timer Register ■ 16-bit Free-run Timer Register is changed. ( "Not initialized" →"Undefined value" ), ( "Undefined bit" →"Undefined" )
214	12.4 Multifunction Timer Register ■ 16-bit Output Compare Register is changed. ( "Not initialized" →"Undefined value" ), ( "Undefined bit" →"Undefined" )
215	12.4 Multifunction Timer Register ■ 16-bit Free-run Timer Register is changed, ( "Not initialized" →"Undefined value" ), ( "Undefined bit" →"Undefined" )
216	12.4 Multifunction Timer Register ■ Waveform Generator Register is changed, ( "Not initialized" →"Undefined value" ), ( "Undefined bit" →"Undefined" )
217	12.4 Multifunction Timer Register ■ A/D Activation Compare Register is changed, ( "Not initialized" →"Undefined value" ), ( "Undefined bit" →"Undefined" )
217	12.4 Multifunction Timer Register ■ Free-run Timer Selector Register is changed, ( "Not initialized" →"Undefined value" ), ( "Undefined bit" →"Undefined" )
238	12.4.8 Input Capture Data Register (IPCPH0 to IPCPH3, IPCPL0 to IPCPL3) ■ Input Capture Data Register (IPCPH0 to IPCPH3, IPCPL0 to IPCPL3) is changed. ( "Not initialized" →"Undefined" )
261	12.4.13 A/D activation compare register (ADCOMP1, ADCOMP2, ADCOMPC1, ADCOMPC2) ■ Control Register 1, 2 (ADCOMPC1, ADCOMPC2) is changed. ( "Not initialized" →"Undefined value" ), ( "Undefined bit" →"Undefined" )
262	12.4.14 Free-run Timer Selector Register (FSR0 to FSR2) ■ Free-run Timer Selector Register (FSR0 to FSR2) is changed. ( "Not initialized" →"Undefined value" ), ( "Undefined bit" →"Undefined" )
310	13.1 Overview ● U-TIMER Control register (UTIMC: UTIMC0, UTIMC1) is changed. ( "Reserved" →"Undefined" )

Page	Changes (For details, refer to main body.)
313	CHAPTER 14 UART is changed. ( "baud rate and U-TIMER reload value settings" → "Baud Rate and U-TIMER Reload Value Settings" )
321	14.2 Register Details Explanation ■ Serial Input Data Register (SIDR: SIDR0, SIDR1) /Serial Output Data Register (SODR: SODR0, SODR1) is changed. ( "Not initialized" → "Undefined" )
332	14.5 Examples of Baud Rate and U-TIMER Reload Value Settings is changed. ( "Examples of baud rate and U-TIMER reload value settings" → "Examples of Baud Rate and U-TIMER Reload Value Settings" )
334	15.1 Features of C-CAN ■ Features of C-CAN is changed. ( "MBit/s" → "Mbps" )
340	15.3 Registers of C-CAN Table 15.3-3 List of Message Handler Register is changed. ( "Reserved area supporting for more than 32 message buffers" is deleted )
344	15.4.1.1 CAN Control Register (CTRLR) ■ Function of Registers is changed. ( ":Reserved bits" → "- : Reserved bits" )
358	15.4.2.1 IFx Command Request Register (IFxCREQ) ■ Function of Registers is changed. ( "[bit5 to bit0] Message Number: Message number (for 128-message buffer CAN)" is deleted. )
369	15.4.3 Message Object is changed. ( "(up to 128 depending on the implementation)" is deleted )
377	15.4.4.1 CAN Transmission Request Registers (TREQR1, TREQR2) ■ Function of Registers is changed. ( "Refer to the following table for the check..." is deleted. )
379	15.4.4.2 CAN Data Renewal Registers (NEWDT1, NEWDT2) ■ Function of Registers is changed. ( "Refer to the following table for the check..." is deleted. )
381	15.4.4.3 CAN Interrupt Pending Registers (INTPND1, INTPND2) ■ Function of Registers is changed. ( "Refer to the following table for the check..." is deleted. )
383	15.4.4.4 CAN Message Enable Registers (MSGVAL1, MSGVAL2) ■ Function of Registers is changed. ( "Refer to the following table for the check..." is deleted. )
384	15.4.5 CAN Clock Prescaler Register ■ CAN Clock Prescaler Register is changed. ( "- : Undefined bit" is deleted )

Page	Changes (For details, refer to main body.)
407	CHAPTER 16 8/10-BIT A/D CONVERTER is changed. ( "16.8 Precautions when Using 8/10-bit A/D Converter" → "16.8 Notes on Using 8/10-bit A/D Converter" )
422	16.4.3 A/D Control Status Register (ADCS) Table 16.4-3 Function of Each Bit in A/D Control Status Register (ADCS) (2 / 2) is changed. ( "Undefined" →"Undefined bit" )
426	16.6 Operation Explanation Figure 16.6-1 Settings for Single-shot Conversion Mode is changed. ( "Bits used" →"Used bit" )
429	16.6 Operation Explanation Figure 16.6-3 Settings for Stop Conversion Mode is changed. ( "Bits used" →"Used bit" )
431	16.8 Notes on Using 8/10-bit A/D Converter is changed. ( "Precautions when" →"Notes on" )
439	17.3 Explanation of Register Figure17.3-1 DSP-CSR is changed. ( "Undefined bit" →"Undefined" )
442	17.3 Explanation of Register ■ DSP Program Counter (DSP-PC) is changed. ( "Not initialized" →"Undefined value" ), ( "Undefined bit" →"Undefined" )
443	17.3 Explanation of Register ■ DSP Delay Register (DSP-LY) is changed. ( "Undefined bit" →"Undefined" )
444	17.3 Explanation of Register ■ DSP Variable Monitor Register (DSP-OT0 to DSP-OT7) is changed. ( "Not initialized" →"Undefined" )
459	18.2 Register Details Explanation ■ DMAC-ch.0,ch.1,ch.2,ch.3,ch.4 Control Status Registers A[DMACA: DMACA0 to DMACA4] is changed. ( "-: Undefined" is added. )
459	18.2 Register Details Explanation ■ DMAC-ch.0,ch.1,ch.2,ch.3,ch.4 Control Status Registers A[DMACA: DMACA0 to DMACA4] is changed. ( "An asterisk (*) indicates bits that will affect operation..." is deleted. )
462	18.2 Register Details Explanation ■ DMAC-ch.0,ch.1,ch.2,ch.3,ch.4 Control Status Registers A[DMACA: DMACA0 to DMACA4] is changed. ( "Reserved" →"-" )



Page	Changes (For details, refer to main body.)
470	18.2 Register Details Explanation  DMAC-ch.0,ch.1,ch.2,ch.3,ch.4 Overall DMAC Control Register [DMACR] is changed. ( "-: Undefined" is added. )
471	18.2 Register Details Explanation  DMAC-ch.0,ch.1,ch.2,ch.3,ch.4 Overall DMAC Control Register [DMACR] is changed. ( Reserved" →"- " )
488	18.5 Data Path is changed. ( "Shown" →"shown below" )
492	19.1 Overview  Flash Memory Outline is changed. ( "Embedded Erase <sup>TM</sup> and Embedded Program <sup>TM</sup> are trademarks" → "Embedded Algorithm <sup>TM</sup> is a trademark" )
499	19.2.1 Flash Memory Status Register (FLCR)  Flash Memory Status Register (FLCR) is changed. ( "- : Undefined bit" is deleted. )
499	19.2.1 Flash Memory Status Register (FLCR)  Flash Memory Status Register (FLCR) is changed. ( "(reserved)" →"- : Reserved bit" )
500	19.2.1 Flash Memory Status Register (FLCR)  Restrictions is changed. ( "(reserved)" →"- : Reserved bit" )
501	19.2.2 Flash Memory Wait Register (FLWC)  Flash Memory Wait Register (FLWC) is changed. ( "- : Undefined bit" is deleted. ), ( "(reserved)" →"- : Reserved bit (s)" )
523	APPENDIX is changed. ( "APPENDIX F Precautions when Using" →"APPENDIX F Notes on Using" )
524	APPENDIX A I/O Map Note: is changed. ( "disable" →"disabled" )
533	APPENDIX A I/O Map Table A-1 I/O Map (9/9) is changed. ( "Reserved (>32..128 Message buffer)" is deleted. )
534	APPENDIX A I/O Map Table A-1 I/O Map (9/9) is changed. ( "only MB91F267NA" →"MB91F267NA/MB91267NA" )

Page	Changes (For details, refer to main body.)
537	APPENDIX B Interrupt Vector Table B-1 Vector Table (3 / 3) is changed. ( "only MB91F267NA" →"MB91F267NA/MB91267NA" )
539	APPENDIX C Pin States in Each CPU State Table C-1 Single-chip Mode is changed. ( "only MB91F267NA" →"MB91F267NA/MB91267NA" )
560	APPENDIX F Notes on Using is changed. ( "Precautions when" →"Notes on" )

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



# ***CHAPTER 1***

---

# ***OVERVIEW***

**This chapter provides basic information required to understand the MB91265A series, and covers features, a block diagram, and package dimension.**

- 1.1 Overview
- 1.2 Block Diagram
- 1.3 Pin Assignment
- 1.4 List of Pin Functions
- 1.5 I/O Circuit Type

## 1.1 Overview

The MB91265A series is general-purpose Fujitsu 32-bit RISC microcontrollers designed for embedded control applications that require high-speed processing. The series uses the FR60Lite CPU which is compatible with the FR family.  
C-CAN (1 channel) is loaded in MB91F267NA/MB91267NA.

### ■ Features

#### ● FR60Lite CPU

- 32-bit RISC, load/store architecture, 5-stage pipeline
- Maximum operating frequency: 33MHz (for a 4.192MHz source oscillation and 8 multiplication of source oscillation (PLL clock multiplier))
- 16-bit fixed-length instructions (basic instruction)
- Instruction execution speed: 1 instruction per cycle
- Memory to memory transfer instructions, bit manipulation instructions, and barrel shift instructions: Instructions optimized for embedded applications
- Function entry and exit instructions, multi-load/store instructions for register content: Instructions intended for use with C language
- Register interlock function: Makes writing assembly language programs easier
- Built-in multiplier/support on the instruction level
  - 32-bit multiplication with sign: 5 cycles
  - 16-bit multiplication with sign: 3 cycles
- Interrupts (PC/PS backup): 6 cycles (16 priority levels)
- Uses Harvard architecture for simultaneous program access and data access
- The instruction is interchangeable with the FR family

#### ● Internal Peripheral Functions

**Table 1.1-1 Internal Peripheral Functions**

	MB91V265A	MB91F267A	MB91F267NA	MB91267A	MB91267NA
	Evaluation product	Flash memory product		MASK ROM product	
Package	PGA-401 (Lead pitch : 2.54 mm interstitial)	LQFP-64 (Lead pitch : 0.65 mm)			
ROM/Flash capacity	External SRAM	128 Kbytes			
RAM capacity	24 Kbytes	4 Kbytes			
C-CAN	1 channel	None	1 channel	None	1 channel

- A/D converter (successive comparison type)
  - Resolution: 10 bits: 4 channels  $\times$  1 unit, 7 channels  $\times$  1 unit
  - Conversion time: 1.2 $\mu$ s (minimum conversion time for a 33MHz system clock)  
1.35 $\mu$ s (minimum conversion time for a 20MHz system clock)
- External interrupt input: 8 channels
- Bit search module (for use with REALOS)
  - Function that searches for the position of the first bit in a word that changes "1" to "0", starting from the MSB (upper bit).
- UART (full-duplex, double-buffered): 2 channels
  - Parity on/off selectable
  - Asynchronous (start-stop synchronization) or clock synchronous communications selectable
  - Separate internal dedicated baud rate timer (U-TIMER) for each channel
  - An external clock can be used as the transfer clock
  - Parity, frame, and overrun error detection function
- 8/16-bit PPG timer: 8 channels (8-bit) or 4 channels (16-bit)
- Timing generator
- 16-bit reload timer: 3 channels (cascade mode available, no output on reload timer 0)
- 16-bit free-run timer: 3 channels
- 16-bit PWC timer: 1 channel
- Input capture: 4 channels (Linked to free-run timer)
- Output compare: 6 channels (Linked to free-run timer)
- Waveform generator
 

A wide range of different waveforms can be generated using the output compare output, 16-bit PPG timer 0, and 16-bit dead timer.
- Sum of products macro
 

RAM : instruction RAM (I-RAM) 256  $\times$  16-bit  
           coefficient RAM (X-RAM) 64  $\times$  16-bit  
           variable RAM (Y-RAM) 64  $\times$  16-bit

Execution of 1 cycle MAC (16-bit  $\times$  16-bit + 40 bits)

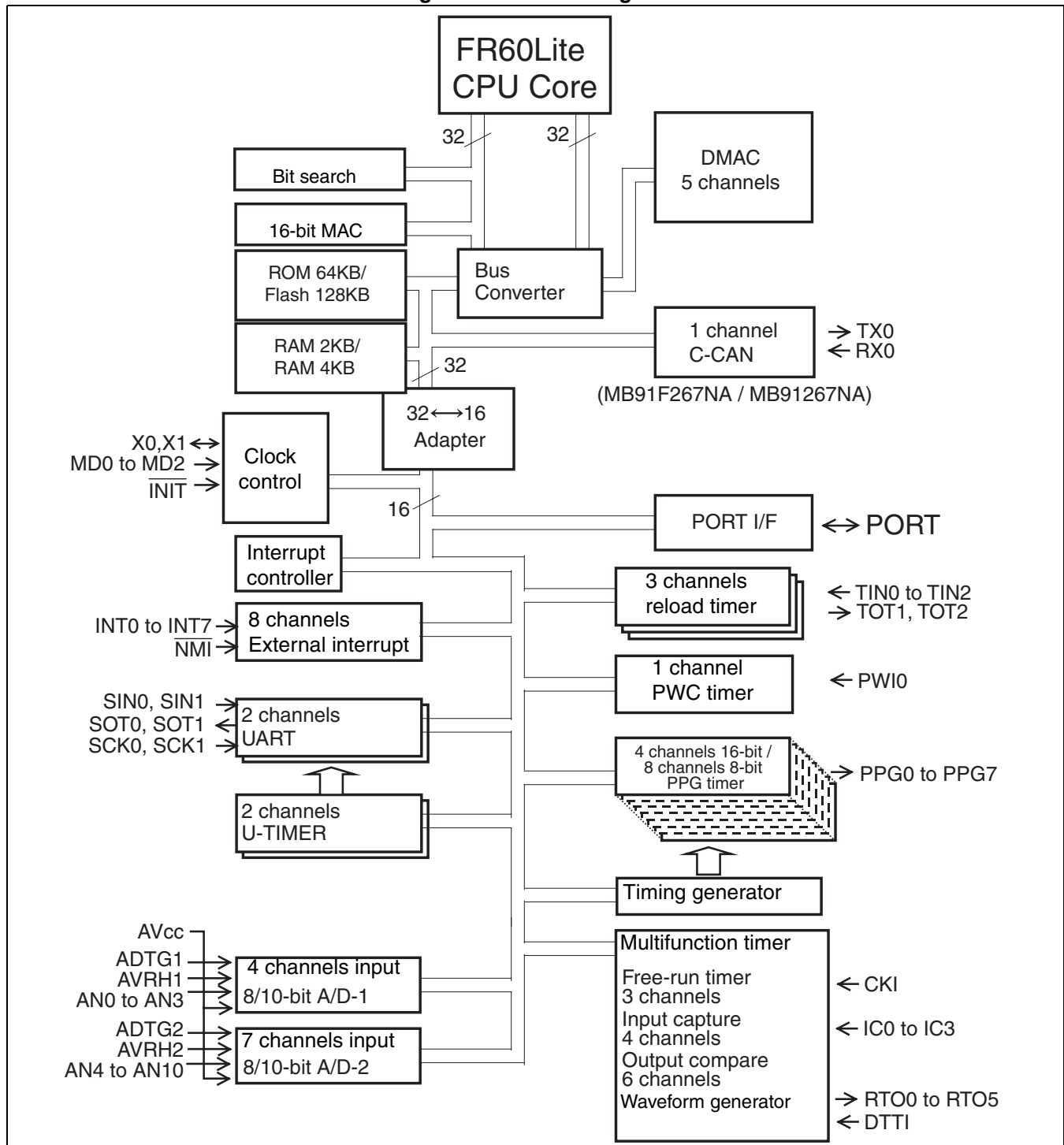
Operation result are extracted rounded from 40 to 16 bits
- DMAC (DMA Controller) : 5 channels
  - Transfer can be initiated by an internal peripheral interrupt or by software
- Watchdog timer
- Low-power consumption mode
  - Sleep and stop function
- Package: LQFP-64
- CMOS technology: 0.35 $\mu$ m
- Power supply: 1 power supply (V<sub>cc</sub>=4.0V to 5.5V)

## 1.2 Block Diagram

Figure1.2-1 shows the block diagram of the MB91265A series.

### ■ Block Diagram

Figure1.2-1 Block Diagram

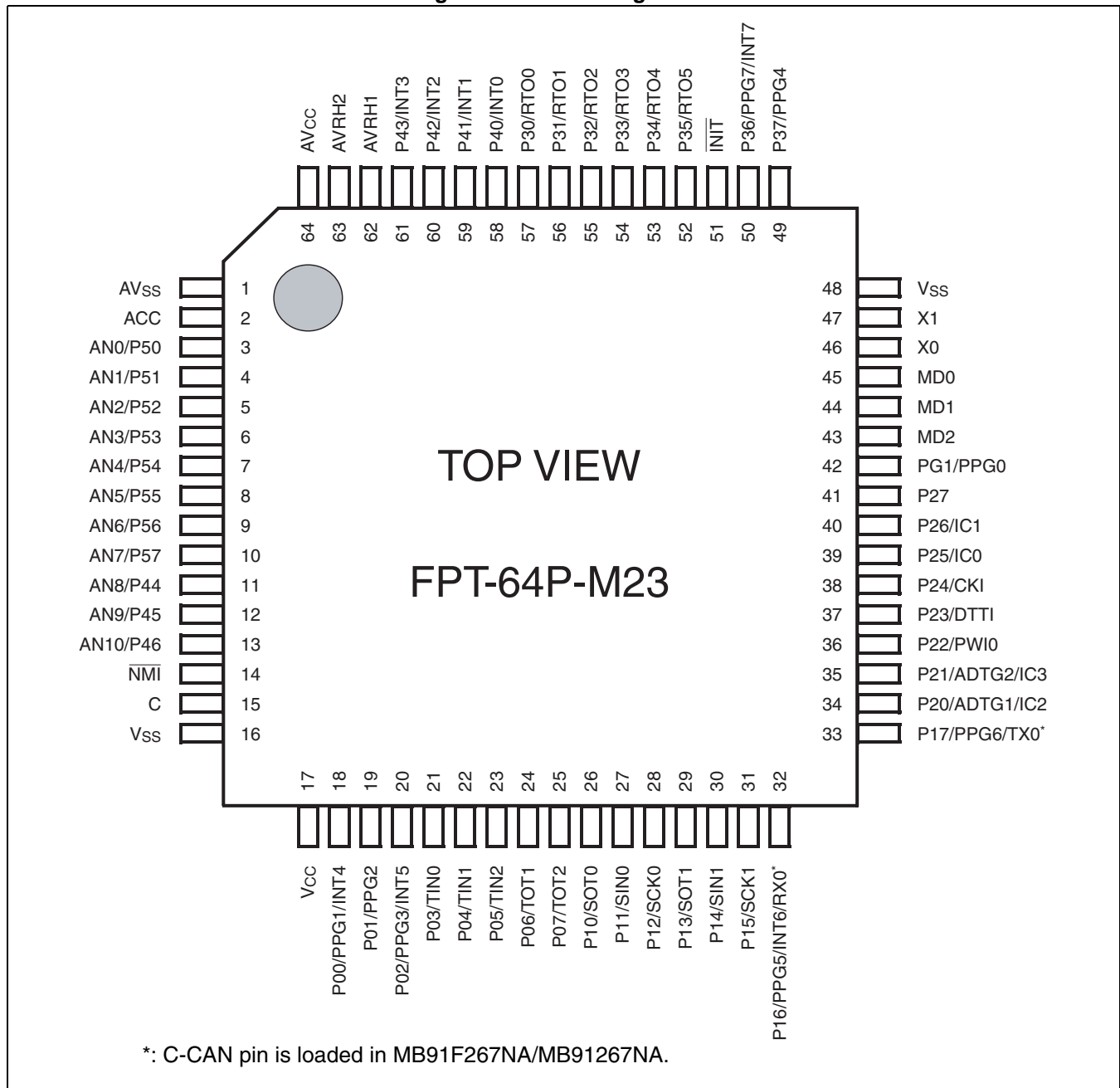


## 1.3 Pin Assignment

This section shows the pin assignment and package dimension of FPT-64P-M23.

### ■ LQFP-64 (MB91F267A/MB91F267NA/MB91267A/MB91267NA)

Figure1.3-1 Pin Assignment



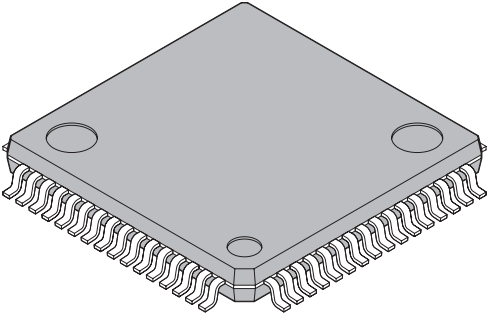


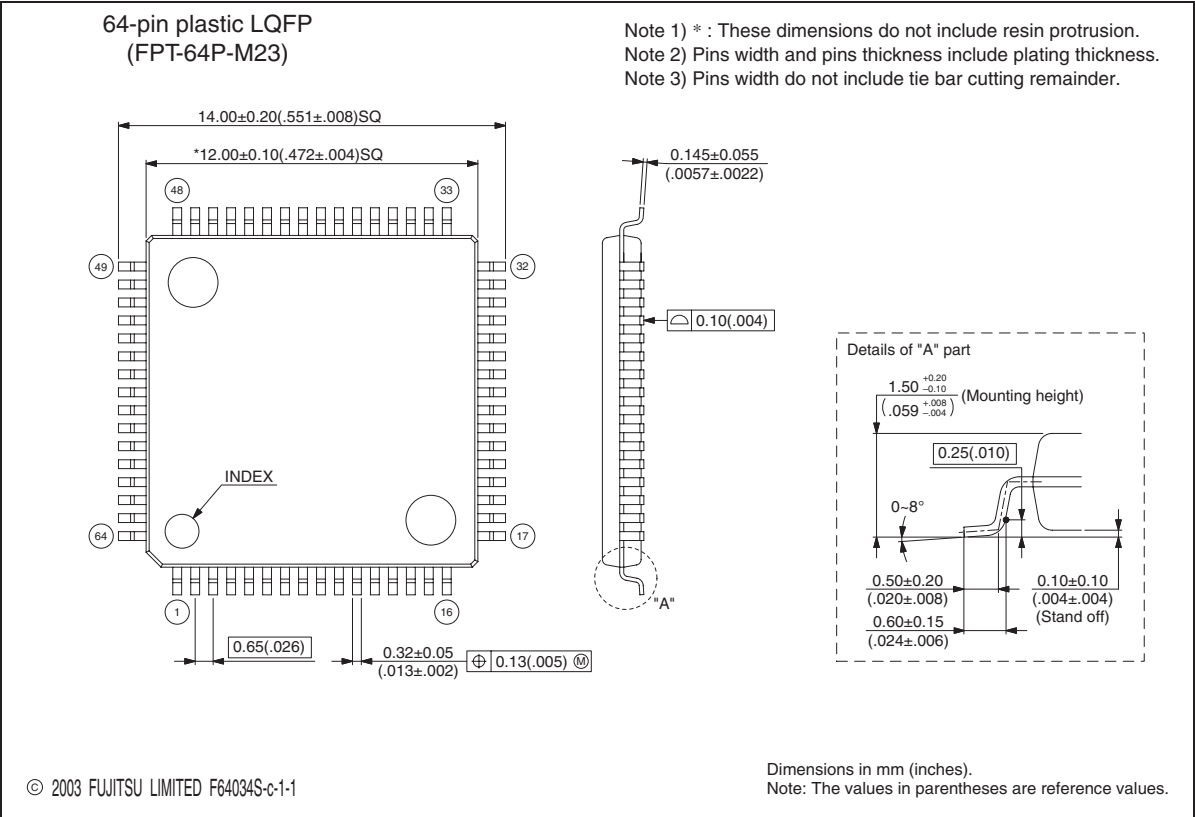
1.3.1 Package Dimension

Figure1.3-2 shows the package dimension of FPT-64P-M23.

■ Package Dimension of FPT-64P-M23

Figure1.3-2 Package Dimension

<div>64-pin plastic LQFP</div>  <div>(FPT-64P-M23)</div>	Lead pitch	0.65 mm
	Package width × package length	12.0 × 12.0 mm
	Lead shape	Gullwing
	Sealing method	Plastic mold
	Mounting height	1.70 mm MAX
	Code (Reference)	P-LFQFP64-12×12-0.65



Please confirm the latest Package dimension by following URL.  
<http://edevic.fujitsu.com/fj/DATASHEET/ef-ovpklv.html>

## 1.4 List of Pin Functions

Table 1.4-1 lists the pin functions.

### ■ List of Pin Functions

Please refer to the pin assignment.

Table 1.4-1 List of Pin Functions (1 / 6)

Pin Number	Pin Name	I/O Circuit Type*1	Functional description
3	AN0	G	Analog input pin of A/D converter 1. This function becomes valid when set the corresponding AICR1 register to analog input.
	P50		General purpose input/output port. This function becomes valid when analog input is set to disabled.
4	AN1	G	Analog input pin of A/D converter 1. This function becomes valid when set the corresponding AICR1 register to analog input.
	P51		General purpose input/output port. This function becomes valid when analog input is set to disabled.
5	AN2	G	Analog input pin of A/D converter 1. This function becomes valid when set the corresponding AICR1 register to analog input.
	P52		General purpose input/output port. This function becomes valid when analog input is set to disabled.
6	AN3	G	Analog input pin of A/D converter 1. This function becomes valid when set the corresponding AICR1 register to analog input.
	P53		General purpose input/output port. This function becomes valid when analog input is set to disabled.
7	AN4	G	Analog input pin of A/D converter 2. This function becomes valid when set the corresponding AICR2 register to analog input.
	P54		General purpose input/output port. This function becomes valid when analog input is set to disabled.
8	AN5	G	Analog input pin of A/D converter 2. This function becomes valid when set the corresponding AICR2 register to analog input.
	P55		General purpose input/output port. This function becomes valid when analog input is set to disabled.
9	AN6	G	Analog input pin of A/D converter 2. This function becomes valid when set the corresponding AICR2 register to analog input.
	P56		General purpose input/output port. This function becomes valid when analog input is set to disabled.
10	AN7	G	Analog input pin of A/D converter 2. This function becomes valid when set the corresponding AICR2 register to analog input.
	P57		General purpose input/output port. This function becomes valid when analog input is set to disabled.
11	AN8	G	Analog input pin of A/D converter 2. This function becomes valid when set the corresponding AICR2 register to analog input.
	P44		General purpose input/output port. This function becomes valid when analog input is set to disabled.

**Table 1.4-1 List of Pin Functions (2 / 6)**

Pin Number	Pin Name	I/O Circuit Type*1	Functional description
12	AN9	G	Analog input pin of A/D converter 2. This function becomes valid when set the corresponding AICR2 register to analog input.
	P45		General purpose input/output port. This function becomes valid when analog input is set to disabled.
13	AN10	G	Analog input pin of A/D converter 2. This function becomes valid when set the corresponding AICR2 register to analog input.
	P46		General purpose input/output port. This function becomes valid when analog input is set to disabled.
14	$\overline{\text{NMI}}$	H	$\overline{\text{NMI}}$ (Non Maskable Interrupt) input pin.
18	INT4	E	External interrupt input pin. Since this input is used as required while the corresponding external interrupt is enabled, the port output must remain off unless intentionally used.
	PPG1		Output pin of PPG timer 1. This function becomes valid when output of PPG timer 1 is set to enabled.
	P00		General purpose input/output port. This function becomes valid when output of PPG timer 1 and external interrupt input are set to disabled.
19	PPG2	D	Output pin of PPG timer 2. This function becomes valid when output of PPG timer 2 is set to enabled.
	P01		General purpose input/output port. This function becomes valid when output of PPG timer 2 is set to disabled.
20	INT5	E	External interrupt input pin. Since this input is used as required while the corresponding external interrupt is enabled, the port output must remain off unless intentionally used.
	PPG3		Output pin of PPG timer 3. This function becomes valid when output of PPG timer 3 is set to enabled.
	P02		General purpose input/output port. This function becomes valid when output of PPG timer 3 and external interrupt input are set to disabled.
21	TIN0	D	External trigger input pin of reload timer 0. Since this input is used as required while the trigger input is enabled, the port output must remain off unless intentionally used.
	P03		General purpose input/output port. This function becomes valid when external clock input of reload timer 0 is set to disabled.
22	TIN1	D	External trigger input pin of reload timer 1. Since this input is used as required while the trigger input is enabled, the port output must remain off unless intentionally used.
	P04		General purpose input/output port. This function becomes valid when external clock input of reload timer 1 is set to disabled.
23	TIN2	D	External trigger input pin of reload timer 2. Since this input is used as required while the trigger input is enabled, the port output must remain off unless intentionally used.
	P05		General purpose input/output port. This function becomes valid when external clock input of reload timer 2 is set to disabled.
24	TOT1	D	Output pin of reload timer 1. This function becomes valid when output of reload timer 1 is set to enabled.
	P06		General purpose input/output port. This function becomes valid when output of reload timer 1 is set to disabled.
25	TOT2	D	Output pin of reload timer 2. This function becomes valid when output of reload timer 2 is set to enabled.
	P07		General purpose input/output port. This function becomes valid when output of reload timer 2 is set to disabled.

**Table 1.4-1 List of Pin Functions (3 / 6)**

Pin Number	Pin Name	I/O Circuit Type*1	Functional description
26	SOT0	D	UART0 data output pin. This function becomes valid when data output of UART0 is set to enabled.
	P10		General purpose input/output port. This function becomes valid when data output of UART0 is set to disabled.
27	SIN0	D	UART0 data input pin. Since this input is used as required while the UART0 input is enabled, the port output must remain off unless intentionally used.
	P11		General purpose input/output port. This function becomes valid when data input of UART0 is set to disabled.
28	SCK0	D	UART0 clock input/output pin. This function becomes valid when clock output of UART0 is set to enabled.
	P12		General purpose input/output port. This function becomes valid when clock output of UART0 is set to disabled.
29	SOT1	D	UART1 data output pin. This function becomes valid when data output of UART1 is set to enabled.
	P13		General purpose input/output port. This function becomes valid when data output of UART1 is set to disabled.
30	SIN1	D	UART1 data input pin. Since this input is used as required while the UART1 input is enabled, the port output must remain off unless intentionally used.
	P14		General purpose input/output port. This function becomes valid when data input of UART1 is set to disabled.
31	SCK1	D	UART1 clock input/output pin. This function becomes valid when clock output of UART1 is set to enabled.
	P15		General purpose input/output port. This function becomes valid when clock output of UART1 is set to disabled.
32	INT6	E	External interrupt input pin. Since this input is used as required while the corresponding external interrupt is enabled, the port output must remain off unless intentionally used.
	PPG5		Output pin of PPG timer 5. This function becomes valid when output of PPG timer 5 is set to enabled.
	RX0		RX0 input pin of C-CAN0 (MB91F267NA/MB91267NA) . Since this input is used as required, the port output must remain off unless intentionally used.
	P16		General purpose input/output port. This function becomes valid when output of PPG timer 5 and RX0 input*2 of C-CAN0 are set to disabled.
33	PPG6	D	Output pin of PPG timer 6. This function becomes valid when output of PPG timer 6 is set to enabled.
	TX0		TX0 output pin of C-CAN0 (MB91F267NA/MB91267NA) . This function becomes valid when TX0 output of C-CAN0 is set to enabled.
	P17		General purpose input/output port. This function becomes valid when output of PPG timer 6 and TX0 output*2 of C-CAN0 are set to disabled.

**Table 1.4-1 List of Pin Functions (4 / 6)**

Pin Number	Pin Name	I/O Circuit Type*1	Functional description
34	ADTG1	D	External trigger input pin of A/D converter 1. Since this input is used as required while it selects as A/D activation trigger cause, the port output must remain off unless intentionally used.
	IC2		Trigger input pin of input capture 2. The port can serve as an input when set for input with the setting of the input capture trigger input. When the port is used for input capture input, this input is used as required. The port output must therefore remain off unless intentionally used.
	P20		General purpose input/output port. This function becomes valid when the setting of the external trigger input of A/D converter 1 or the setting of the input capture trigger input is set to disabled.
35	ADTG2	D	External trigger input pin of A/D converter 2. Since this input is used as required while it selects as A/D activation trigger cause, the port output must remain off unless intentionally used.
	IC3		Trigger input pin of input capture 3. The port can serve as an input when set for input with the setting of the input capture trigger input. When the port is used for input capture input, this input is used as required. The port output must therefore remain off unless intentionally used.
	P21		General purpose input/output port. This function becomes valid when the setting of the external trigger input of A/D converter or the setting of the input capture trigger input is set to disabled.
36	PW10	D	Pulse width counter input of PWC timer 0. This function becomes valid when pulse width counter input of PWC timer 0 is set to enabled.
	P22		General purpose input/output port. This function becomes valid when pulse width counter input of PWC timer 0 is set to disabled.
37	DTTI	D	Control input signal of multi-function timer waveform generator output RTO0 to RTO5. This function becomes valid when DTTI pin input is set to enabled.
	P23		General purpose input/output port. This function becomes valid when input of DTTI pin is set to disabled.
38	CKI	D	External clock input pin of free-run timer. Since this input is used as required while the port is used for external clock input pin of free-run timer, the port output must remain off unless intentionally used.
	P24		General purpose input/output port. This function becomes valid when external clock input of free-run timer is set to disabled.
39	IC0	D	Trigger input pin of input capture 0. The port can serve as an input when set for input with the setting of the trigger input of input capture 0. When the port is used for input capture input, this input is used as required. The port output must therefore remain off unless intentionally used.
	P25		General purpose input/output port. This function becomes valid when trigger input of input capture 0 is set to disabled.
40	IC1	D	Trigger input pin of input capture 1. The port can serve as an input when set for input with the setting of the trigger input of input capture 1. When the port is used for input capture input, this input is used as required. The port output must therefore remain off unless intentionally used.
	P26		General purpose input/output port. This function becomes valid when trigger input of input capture 1 is set to disabled.
41	P27	D	General purpose input/output port.

**Table 1.4-1 List of Pin Functions (5 / 6)**

Pin Number	Pin Name	I/O Circuit Type*1	Functional description
42	PPG0	D	Output pin of PPG timer 0. This function becomes valid when output of PPG timer 0 is set to enabled.
	PG1		General purpose input/output port. This function becomes valid when output of PPG timer 0 is set to disabled.
43	MD2	H,K	Mode pin 2. Setting this pin determines the basic operation mode. Connect to V <sub>CC</sub> or V <sub>SS</sub> . The circuit type of flash memory models is K.
44	MD1	H,K	Mode pin 1. Setting this pin determines the basic operation mode. Connect to V <sub>CC</sub> or V <sub>SS</sub> . The circuit type of flash memory models is K.
45	MD0	H	Mode pin 0. Setting this pin determines the basic operation mode. Connect to V <sub>CC</sub> or V <sub>SS</sub> .
46	X0	A	Clock (oscillation) output pin.
47	X1	A	Clock (oscillation) input pin.
49	PPG4	D	Output pin of PPG timer 4. This function becomes valid when output of PPG timer 4 is set to enabled.
	P37		General purpose input/output port. This function becomes valid when output of PPG timer 4 is set to disabled.
50	INT7	E	External interrupt input pin. Since this input is used as required while the corresponding external interrupt is enabled, the port output must remain off unless intentionally used.
	PPG7		Output pin of PPG timer 7. This function becomes valid when output of PPG timer 7 is set to enabled.
	P36		General purpose input/output port. This function becomes valid when output of PPG timer 7 is set to disabled.
51	$\overline{\text{INIT}}$	I	External reset input pin.
52	RTO5	J	Waveform generator output pin of multi-function timer. This pin outputs waveform set at the waveform generator. This function becomes valid when waveform generator output of multi-function timer is set to enabled.
	P35		General purpose input/output port. This function becomes valid when output of waveform generator is set to disabled.
53	RTO4	J	Waveform generator output pin of multi-function timer. This pin outputs waveform set at the waveform generator. This function becomes valid when waveform generator output of multi-function timer is set to enabled.
	P34		General purpose input/output port. This function becomes valid when output of waveform generator is set to disabled.
54	RTO3	J	Waveform generator output pin of multi-function timer. This pin outputs waveform set at the waveform generator. This function becomes valid when waveform generator output of multi-function timer is set to enabled.
	P33		General purpose input/output port. This function becomes valid when output of waveform generator is set to disabled.
55	RTO2	J	Waveform generator output pin of multi-function timer. This pin outputs waveform set at the waveform generator. This function becomes valid when waveform generator output of multi-function timer is set to enabled.
	P32		General purpose input/output port. This function becomes valid when output of waveform generator is set to disabled.

**Table 1.4-1 List of Pin Functions (6 / 6)**

Pin Number	Pin Name	I/O Circuit Type*1	Functional description
56	RTO1	J	Waveform generator output pin of multi-function timer. This pin outputs waveform set at the waveform generator. This function becomes valid when waveform generator output of multi-function timer is set to enabled.
	P31		General purpose input/output port. This function becomes valid when output of waveform generator is set to disabled.
57	RTO0	J	Waveform generator output pin of multi-function timer. This pin outputs waveform set at the waveform generator. This function becomes valid when waveform generator output of multi-function timer is set to enabled.
	P30		General purpose input/output port. This function becomes valid when output of waveform generator is set to disabled.
58	INT0	E	External interrupt input pin. Since this input is used as required while the corresponding external interrupt is enabled, the port output must remain off unless intentionally used.
	P40		General purpose input/output port. This function becomes valid when external interrupt input is set to disabled.
59	INT1	E	External interrupt input pin. Since this input is used as required while the corresponding external interrupt is enabled, the port output must remain off unless intentionally used.
	P41		General purpose input/output port. This function becomes valid when external interrupt input is set to disabled.
60	INT2	E	External interrupt input pin. Since this input is used as required while the corresponding external interrupt is enabled, the port output must remain off unless intentionally used.
	P42		General purpose input/output port. This function becomes valid when external interrupt input is set to disabled.
61	INT3	E	External interrupt input pin. Since this input is used as required while the corresponding external interrupt is enabled, the port output must remain off unless intentionally used.
	P43		General purpose input/output port. This function becomes valid when external interrupt input is set to disabled.

\*1: For the I/O circuit type, refer to Section "1.5 I/O Circuit Type".

\*2: C-CAN is set in MB91F267NA/MB91267NA.

[Power supply pin and GND pin]

Pin Number	Pin Name	Functional description
16, 48	V <sub>SS</sub>	GND pins. Apply equal potential to all of the pins.
17	V <sub>CC</sub>	Power supply pin. Apply equal potential to all of the pins.
64	AV <sub>CC</sub>	Analog power supply pin for A/D converter.
63	AVRH2	Analog reference power supply pin for A/D converter 2.
62	AVRH1	Analog reference power supply pin for A/D converter 1.
1	AV <sub>SS</sub>	Analog GND pin for A/D converter.
15	C	Condenser connection pin for internal regulator.
2	ACC	Condenser connection pin for analog.

# 1.5 I/O Circuit Type

Table 1.5-1 shows the I/O circuit type.

■ I/O Circuit Type

Table 1.5-1 I/O Circuit Type (1 / 3)

Classification	Circuit Type	Remark
A		<ul style="list-style-type: none"><li>• Oscillation feedback resistance for high speed (main clock oscillation): approx. 1MΩ</li></ul>
D		<ul style="list-style-type: none"><li>• CMOS level output</li><li>• CMOS level hysteresis input</li><li>• With standby control</li><li>• With Pull-up control</li></ul> <p>Pull-up resistance = approx. 50kΩ(Typ)</p> <p>IOL = 4mA</p>
E		<ul style="list-style-type: none"><li>• CMOS level output</li><li>• CMOS level hysteresis input</li><li>• With standby control</li><li>• With Pull-up control</li></ul> <p>Pull-up resistance = approx. 50kΩ(Typ)</p> <p>IOL = 4mA</p>



**Table 1.5-1 I/O Circuit Type (2 / 3)**

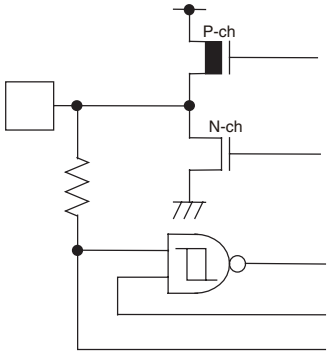
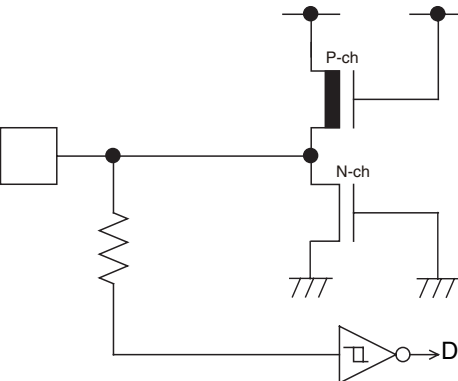
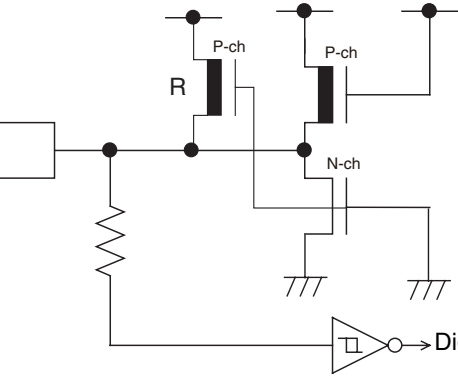
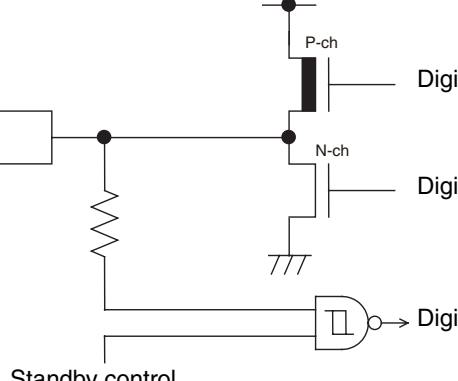
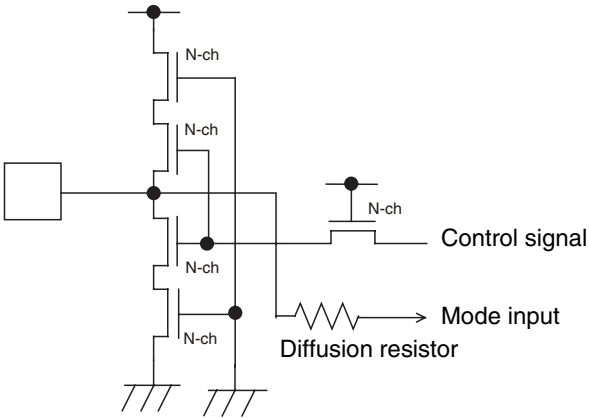
Classification	Circuit Type	Remark
G	 <p>Digital output</p> <p>Digital output</p> <p>Digital input</p> <p>Standby control</p> <p>Analog input</p>	<ul style="list-style-type: none"> <li>• Analog/CMOS level hysteresis input/output pin</li> <li>• CMOS level output</li> <li>• CMOS level hysteresis input (attached with standby control)</li> <li>• Analog input (Analog input is enabled when AICR's corresponding bit is set to "1".)</li> </ul> <p><math>I_{OL}=4mA</math></p>
H	 <p>Digital input</p>	<ul style="list-style-type: none"> <li>• CMOS level hysteresis input</li> <li>• Without standby control</li> </ul>
I	 <p>Digital input</p>	<ul style="list-style-type: none"> <li>• CMOS level hysteresis input</li> <li>• With pull-up resistor Pull-up resistance = approx. <math>50k\Omega</math> (Typ)</li> <li>• Without standby control</li> </ul>
J	 <p>Digital output</p> <p>Digital output</p> <p>Digital input</p> <p>Standby control</p>	<ul style="list-style-type: none"> <li>• CMOS level output</li> <li>• CMOS level hysteresis input</li> <li>• With standby control</li> </ul> <p><math>I_{OL} = 12mA</math></p>

Table 1.5-1 I/O Circuit Type (3 / 3)

Classification	Circuit Type	Remark
K		Flash memory product only <ul style="list-style-type: none"><li>• CMOS level input</li><li>• High voltage control for test of FLASH</li></ul>



# ***CHAPTER 2***

---

# ***HANDLING THE DEVICE***

**This chapter provides precautions on handling the MB91265A series.**

## **2.1 Precautions on Handling the Device**

## 2.1 Precautions on Handling the Device

---

**This section explains the precautions on handling the device.**

---

### ■ Precautions on Handling Device

#### ● Preventing latch-up

Latch up phenomenon may occur with CMOS IC, when a voltage higher than  $V_{CC}$  or lower than  $V_{SS}$  is applied to either the input or output pins, or when a voltage is applied between  $V_{CC}$  and  $V_{SS}$  that exceeds the rated voltage. When latch up occurs, a significant power-supply current surge results, which may damage some elements due to the excess heat, so great care must be taken to ensure that the maximum rating is never exceeded during use.

#### ● About the processing of an unused input pin

If unused input pins are kept being opened, may cause erroneous operation, so they should be pulled up or pulled down.

#### ● Power pins

If multiple  $V_{CC}$  or  $V_{SS}$  pins are provided, although the pins that are meant to be at the same potential are connected internally in the device design to prevent misoperation such as latch-up, always connect these to the appropriate power supply or ground externally to ensure that the overall output current rating is achieved, to minimize spurious radiation, and to prevent misoperation of strobe signals due to a rise in the ground level and other such problems. Also try to ensure that connection to the  $V_{CC}/V_{SS}$  on this device is at the lowest impedance possible from the current supply source.

In addition, Fujitsu will recommend the ceramic capacitor of about 0.1 $\mu$ F to be connected as bypass capacitor between  $V_{CC}$  and  $V_{SS}$  near this device.

#### ● Crystal oscillator circuit

The noise near X0 and X1 pins becomes original of the malfunction of this device. Design the printed circuit board such that X0, X1, the crystal oscillator (or ceramic oscillator), and the bypass capacitor to ground are located as close to the device as possible.

It is also strongly recommended that the printed circuit board artwork surrounds the X0 and X1 pins with ground to promote stable operation.

Please ask the crystal maker to evaluate the oscillational characteristics of the crystal and this device.

#### ● About mode pins (MD0 to MD2)

Please tie directly to  $V_{CC}$  or  $V_{SS}$  and use these pins. In order to prevent erroneous entry to test mode due to noise, the pattern length between each mode pin and  $V_{CC}$  or  $V_{SS}$  on the printed circuit board should be as short as possible, and they should be connected at low impedance.

#### ● Turning on the power

Always use the  $\overline{\text{INIT}}$  pin to perform a settings initialization reset (INIT) immediately after turning on the power.

Also, to ensure the oscillation stabilization wait time for the oscillation circuit and the stabilization wait time for the regulator immediately after turning on the power, connect the "L" level input to the  $\overline{\text{INIT}}$  pin for the stabilization wait time required by the oscillation circuit. (When an INIT is triggered by the  $\overline{\text{INIT}}$  pin, the oscillation stabilization wait time is initialized to its minimum value.)

- Power-on sequence

Turn on the power supply in the sequence  $V_{CC} \rightarrow AV_{CC} \rightarrow AVR_H$ , and turn off the power in the reverse sequence.

If not using the A/D converter, connect  $AV_{CC}=V_{CC}$  and  $AV_{SS}=V_{SS}$ .

- Source oscillation input when turning on the power

When turning on the power, always input the clock until the oscillation stabilization wait time is released.

- Note on PLL clock mode operation

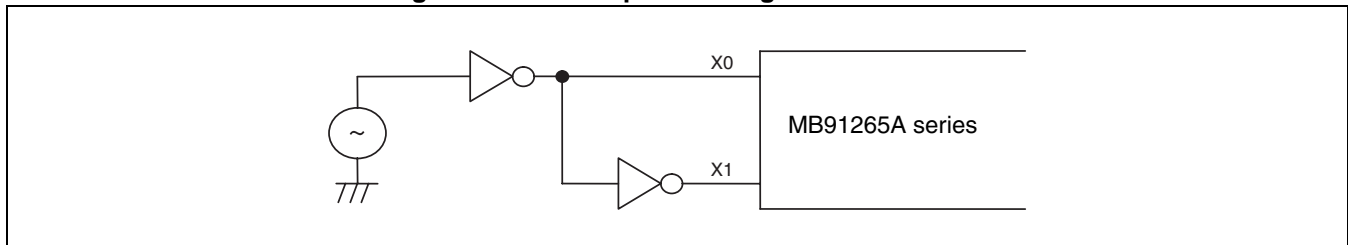
On this microcontroller, if in case the crystal oscillator breaks off or an external reference clock input stops while the PLL clock mode is selected, a self-oscillator circuit contained in the PLL may continue its operation at its self-running frequency. However, Fujitsu will not guarantee results of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

- Using an external clock

When using an external clock, you must always input clock signals to X0 pin and clock signals with opposite phase from X0 pin to X1 pin. However, as the X1 pin halts with an output at the "H" level during STOP mode, insert a resistor of approximately  $1k\Omega$  externally to prevent a conflict between the two outputs if using STOP mode (oscillation stop mode).

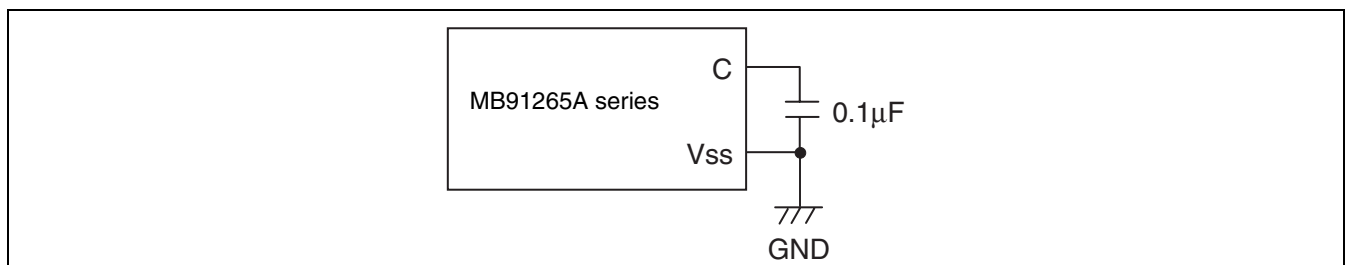
The figure below shows an example of how to use an external clock.

**Figure 2.1-1 Example of Using External Clock**



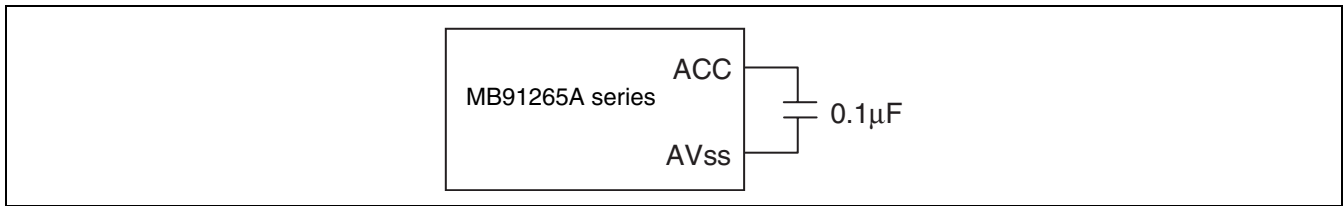
- C pin

As this device includes an internal regulator, always connect a bypass capacitor of approximately  $0.1\mu F$  to the C pin for use by the regulator.



### ● ACC pin

As the device has an internal A/D converter, a capacitor of approximately  $0.1\mu\text{F}$  must be connected between the ACC pin and  $\text{AV}_{\text{SS}}$  pin.



### ● Synchronous Mode Software Reset

When using the synchronous mode software reset, the following two conditions must be satisfied before setting the SRST bit in the STCR (standby control register) to "0".

- Set interrupt enable flag (I-Flag) to interrupts disabled (I-Flag=0)
- NMI not used

# **CHAPTER 3**

---

## ***CPU AND CONTROL UNIT***

**This chapter provides basic information required to understand the CPU core functions of the MB91265A series. It covers architecture, specifications, and instructions.**

- 3.1 Memory Space
- 3.2 Memory Map
- 3.3 Internal Architecture
- 3.4 Basic Programming Model
- 3.5 Data Structure
- 3.6 Memory Map
- 3.7 Divergence Instructions
- 3.8 EIT (Exception, Interruption, and Trap)
- 3.9 Operating Mode
- 3.10 Reset (Device Initialization)
- 3.11 Clock Generation Control
- 3.12 Device State Control



## 3.1 Memory Space

---

The logical address space of the MB91265A series is 4 Gbytes ( $2^{32}$  addresses) and the CPU performs linear access.

---

### ■ Direct Addressing Area

The undermentioned area of the address space is used for I/O.

This area is called the direct addressing area and the operand address can be specified directly in the instruction.

A direct area is different depending on the size of the accessed data as follows.

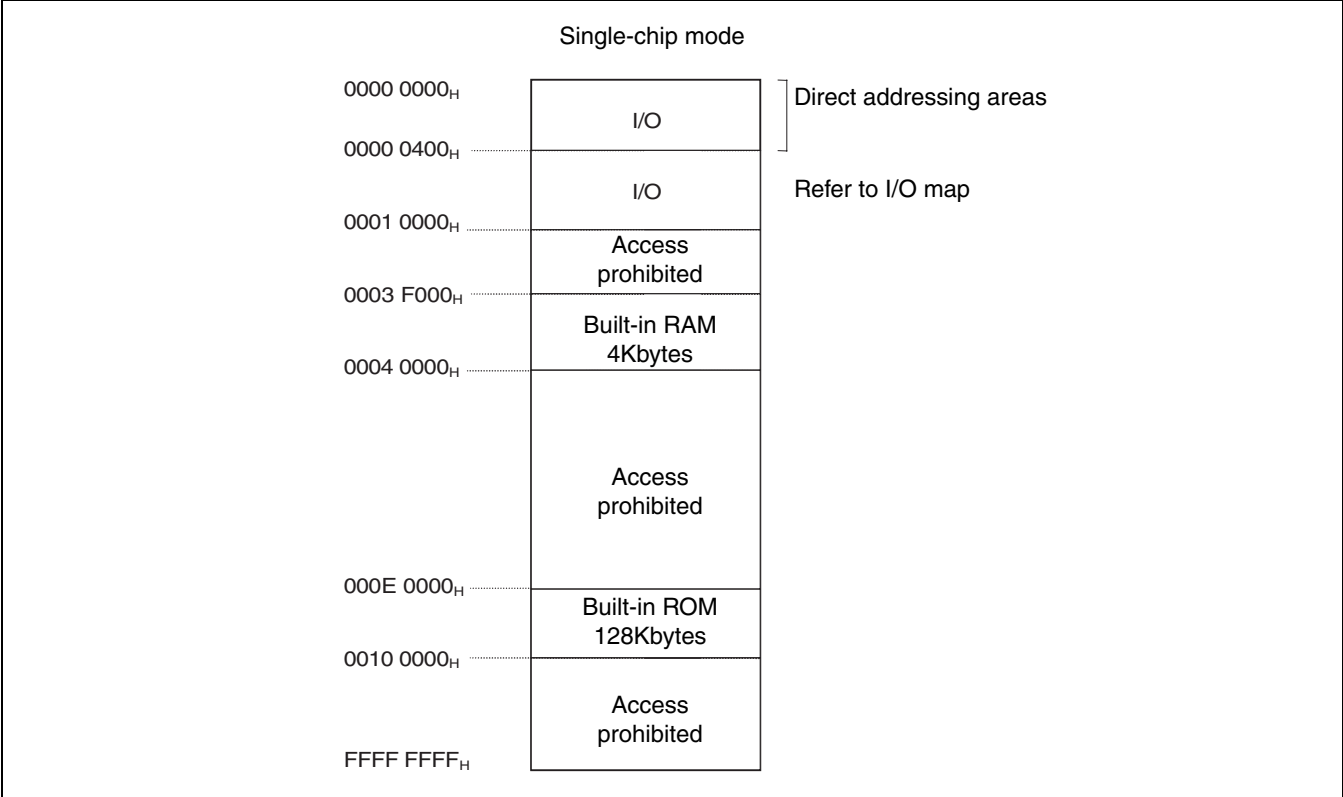
- Byte data access : 000<sub>H</sub> to 0FF<sub>H</sub>
- Half word data access : 000<sub>H</sub> to 1FF<sub>H</sub>
- Word data access : 000<sub>H</sub> to 3FF<sub>H</sub>

### 3.2 Memory Map

This section shows the memory map of the MB91265A series.

#### ■ MB91F267A/MB91F267NA/MB91267A/MB91267NA Memory Map

Figure 3.2-1 MB91F267A/MB91F267NA/MB91267A/MB91267NA Memory Map



## 3.3 Internal Architecture

---

**As well as adopting RISC architecture, the MB91265A series CPU is a high performance core featuring high function commands for embedded applications.**

---

### ■ Features of Internal Architecture

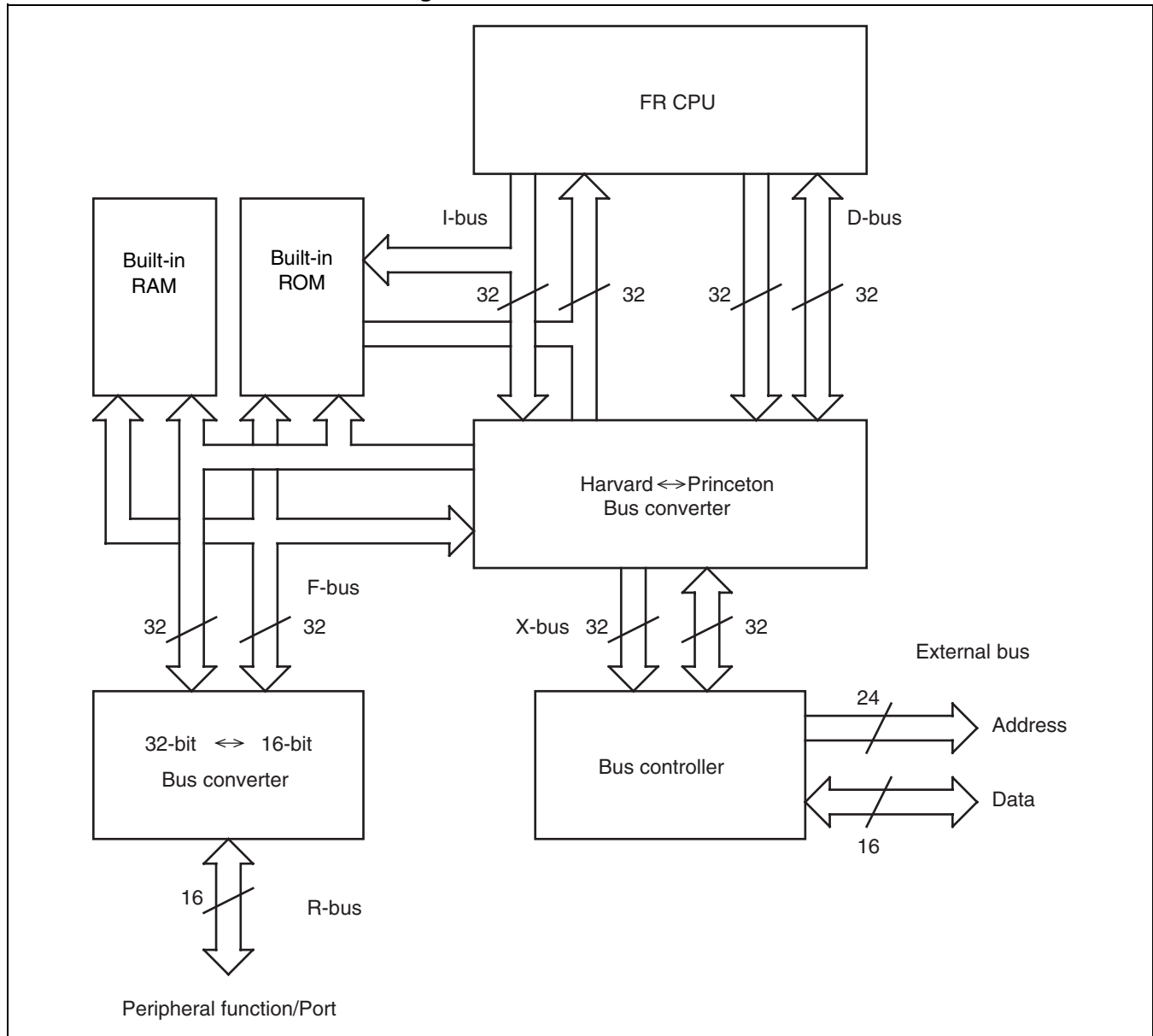
- Adoption of RISC architecture  
Basic instruction: one instruction one cycle
- 32-bit architecture  
General-purpose registers: 32 bits × 16 registers
- Linear memory space of 4GB
- Installing of multipliers  
32-bit by 32-bit multiplication: 5 cycles  
16-bit by 16-bit multiplication: 3 cycles
- Reinforcement of interruption processing function  
High-speed response speed (6 cycles)  
Support of multiple interruption  
Level mask function (16 levels)
- Reinforcement of instruction for I/O operation  
Memory-to-memory transfer instruction  
Bit manipulation instruction
- High code efficiency  
Basic instruction word length: 16 bits
- Low-power consumption  
Sleep mode, stop mode
- Clock division ratio setting function

## ■ Internal Architecture

The CPU of the FR family uses the Harvard architecture with separate instruction bus and data bus.

A 32-bit  $\leftrightarrow$  16-bit bus converter is connected to the 32-bit bus (F-bus) to provide an interface between the CPU and peripheral resources. A Harvard  $\leftrightarrow$  Princeton bus converter is connected to the I-bus and D-bus to provide an interface between the CPU and the bus controller.

**Figure 3.3-1 Internal Architecture**



Note:

External bus function is not supported.

## ■ CPU

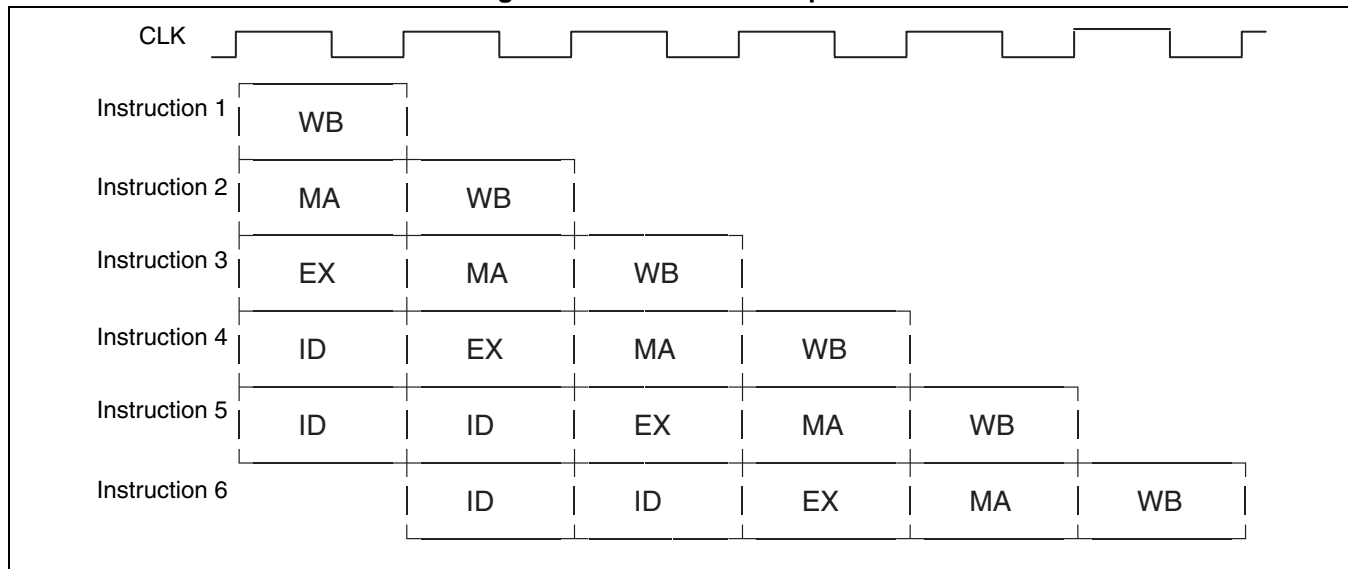
CPU is compactly implementation for the FR architecture for 32-bit RISC.

A five-stage instruction pipeline is used to enable execution of one instruction per cycle.

The pipeline is composed of the following stages.

- Instruction fetch (IF)..... : The instruction address is output, and the instruction is fetched.
- Instruction decipherment (ID) :The decipherment does the fetched instruction. The register is read.
- Execution (EX) ..... : The operation is executed.
- Memory access (MA) ..... : Loading into or storing the memory is accessed.
- Write-back (WB) ..... : Writes the result of operation (or loaded memory data) to a register.

**Figure 3.3-2 Instruction Pipeline**



The instruction is never in any order executed. Accordingly, if instruction A enters the pipeline before instruction B, instruction A always reaches write-back stage before instruction B.

As a rule, the instruction is executed at the speed of one instruction per cycle. A number of cycles are required to execute commands for the load store commands accompanying memory wait, branch commands that do not have delay slots and multiple cycle commands. Also, instruction execution speed drops if supply of instructions is delayed.

## ■ 32-bit ↔ 16-bit Bus Converter

Acts as an interface between the F-bus which uses high-speed 32-bit access and the R-bus which uses 16-bit access to enable the CPU to access data from the internal peripheral circuits.

The bus converter converts 32-bit access from the CPU into two 16-bit accesses and performs R-bus access. Restrictions on the access width apply for some internal peripheral circuits.

## ■ Harvard ↔ Princeton Bus Converter

Arbitrates instruction access and data access from the CPU to provide a smooth interface to the external buses.

In CPU, the instruction bus and the data bus are the independent Harvard architecture structures. On the other hand, the bus controller that controls the external bus has a single-bus Princeton architecture structure. This bus converter ranks the priority order for command access and data access of the CPU and controls access to the bus controller. This mechanism causes the prioritizing of external bus access to be optimized.

## ■ Overview of Instructions

FR family supports logic operation, bit operation, and direct addressing commands that are optimized for embedded applications, as well as the command system of the normal RISC. See Appendix for the list of the instruction set. Excellent memory efficiency is achieved because each instruction is 16-bit long (some instructions are 32 or 48 bits in length).

The instruction set can be divided into the following function groups.

- Arithmetic operation
- Loading and store
- Divergence
- Logical operation and bit operation
- Direct addressing
- The others

### ● Arithmetic operation

It has standard arithmetic operation commands (addition, subtraction, and comparison) and shift commands (logic shift and arithmetic operation shift). Operations with carry that are used for multi-word length operations and operations that do not change the flag which are convenient for address calculations are enabled for addition and subtraction.

32-bit  $\times$  32-bit and 16-bit  $\times$  16-bit multiplication instructions and a 32-bit/32-bit step division instruction are also provided.

Immediate value transfer instructions for setting immediate values to registers and register-to-register transfer instructions are also provided.

All arithmetic operation instructions use the multiplication and division register and the general-purpose registers in the CPU to perform the operation.

### ● Loading and store

Load and store instructions read or write data in external memory. They are also used to read or write to the peripheral resources (I/O) on the chip.

Loading and store have three kinds of access lengths of the byte, the half-word, and the word. In addition to normal indirect register memory addressing, memory addressing is also possible for certain commands such as indirect displacement registers and indirect increment/decrement registers.

### ● Divergence

It is an instruction of the divergence, the call, the interruption, and the return. There are two types of branch commands; one type features a delay slot while the other does not. They can be optimized in accordance with the purpose. See Section "3.7 Divergence Instructions" for details of the divergence instruction.

### ● Logical operation and bit operation

Logical operation commands can perform AND, OR, and EOR logical operations between general-purpose registers or between a general-purpose register and the memory (and I/O). Moreover, the bit operation instruction can operate the content of the memory (and I/O) directly. The register of the memory addressing is generally indirect.

- Direct addressing

Direct addressing commands are used to access between I/O and general-purpose registers or between I/O and the memory. The I/O address can be accessed quickly and efficiently by direct specification within the command instead of indirectly to the register. Indirect memory addressing to the register with register increment/decrement is also enabled for some commands.

- The others

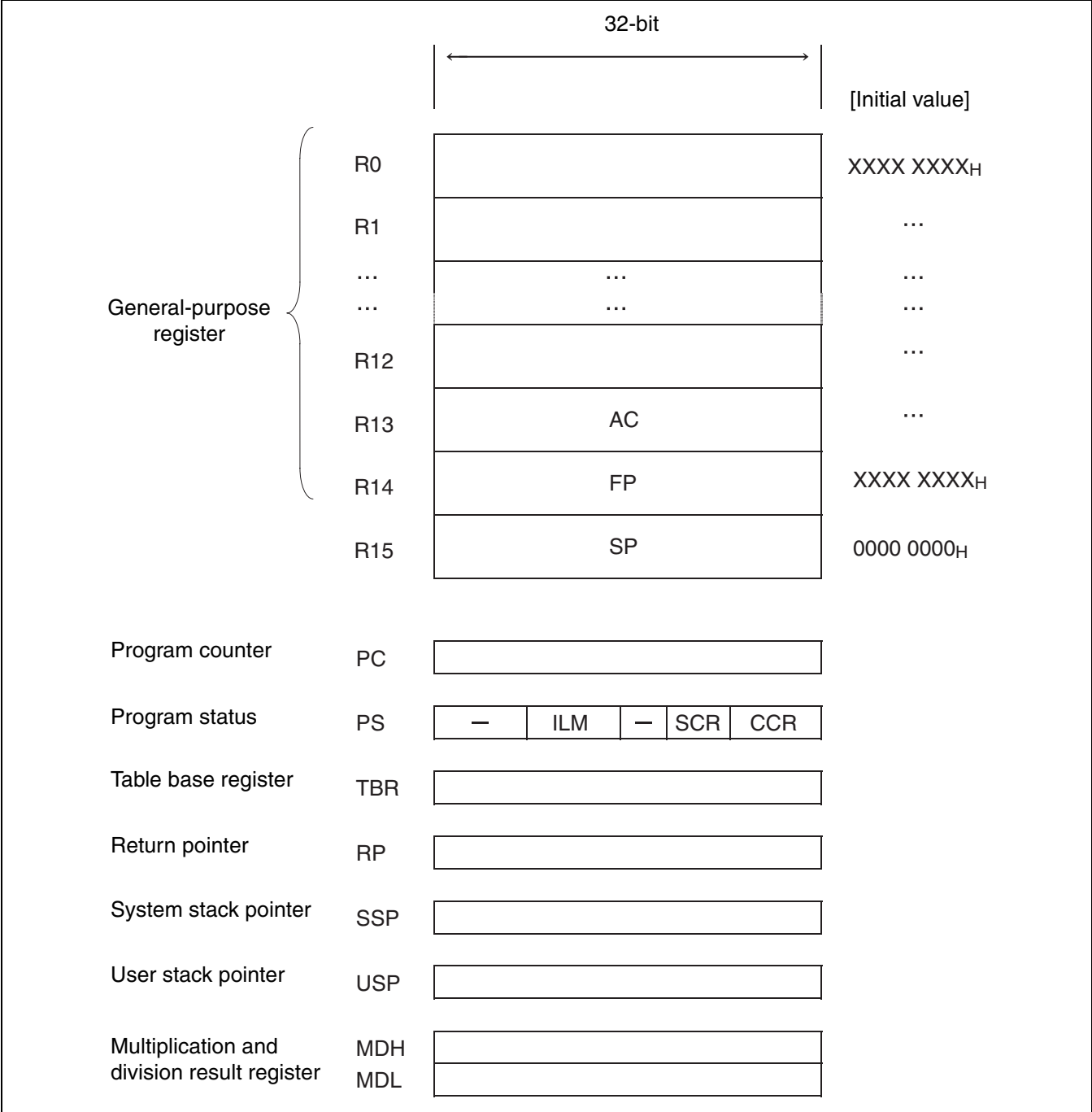
Other instructions include instructions for setting the flags in the PS register, performing stack operations, and performing sign and zero-extended operations. Function entry and exit instructions for use with high-level languages and register multi-load/store instructions are also provided.

### 3.4 Basic Programming Model

This section explains the basic programming model and each register.

■ Basic Programming Model

Figure 3.4-1 Basic Programming Model



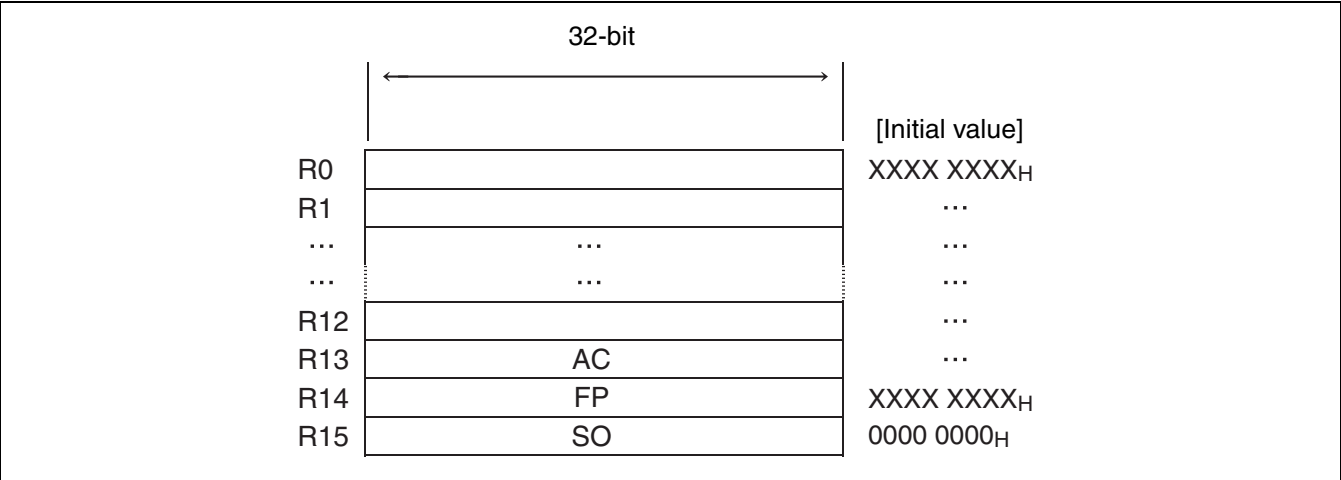


### 3.4.1 Registers

This section explains each register.

#### ■ General-purpose Register

Figure 3.4-2 General-purpose Register



Registers R0 to R15 are a general-purpose register. They are used as accumulator for various types of operation and as pointer for memory access.

The following of the 16 registers are expected to have special uses, so some commands are emphasized.

R13: Virtual accumulator

R14: Frame pointer

R15: Stack pointer

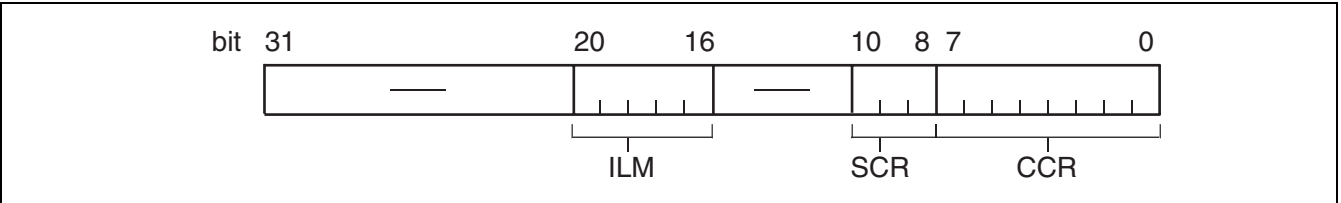
R0 to R14 of the initial value by reset are irregular. R15 is 00000000H (SSP value).

#### ■ PS (Program Status)

This register retains the program status and is separated into three parts, namely, ILM, SCR, and CCR.

All bits undefined in figure are reservation bit. Reading always returns "0".

Writing has no effect.



## ■ CCR (Condition Code Register)

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
-	-	S	I	N	Z	V	C	--00XXXX <sub>B</sub>

[bit5] S : Stack flag

The stack pointer used as R15 is specified.

Value	Description
0	SSP is used as R15. Automatically goes to "0" when an EIT occurs. (However, the value saved on the stack is the value before the bit is cleared.)
1	USP is used as R15.

Cleared to "0" by a reset.

Set to "0" when executing the RETI instruction.

[bit4] I : Interrupt enable flag

Permission and interdiction of the user interruption demand are controlled.

Value	Description
0	User interruption interdiction. Cleared to "0" when the INT instruction is executed. (However, the value saved on the stack is the value before the bit is cleared.)
1	User interruption permission. Masking of user interrupt requests is controlled by the value stored in the ILM.

Cleared to "0" by a reset.

[bit3] N : Negative flag

Indicates the sign when an operation result is represented as a two's-complement integer.

Value	Description
0	It is indicated that operation result was a positive value.
1	It is indicated that operation result was a negative value.

Initial state by reset is irregular.

[bit2] Z : Zero flag

It is shown operation result was 0.

Value	Description
0	It is indicated that operation result was the values other than "0".
1	It is indicated that operation result was "0".

Initial state by reset is irregular.

**[bit1] V : Overflow flag**

Operands used for calculations are defined as integers expressed in complements of "2", and whether or not an overflow occurs as a result of the calculation is indicated.

Value	Description
0	It is indicated not to have caused the overflow as a result of the operation.
1	It is indicated to have caused the overflow as a result of the operation.

Initial state by reset is irregular.

**[bit0] C : Carrying flag**

Indicates whether an operation resulted in a borrow or a carry from the most significant bit.

Value	Description
0	It is indicated that neither a carry nor a borrow occurred.
1	It is indicated that a carry or a borrow occurred.

Initial state by reset is irregular.

**■ SCR (System Condition Code Register)**

				Initial value
bit10	bit9	bit8		
D1	D0	T		XX0 <sub>B</sub>

**[bit10, bit9] D1, D0: Flag for step division**

The middle data of step division execution time is maintained.

Do not change while executing the division processing.

During step division when other processing is performed, re-start of the step division is guaranteed by saving and returning to the PS register value.

Initial state by reset is irregular.

Set based on the value of the dividend and divisor during the execution of the DIV0S instruction.

Forcibly cleared by execution of the DIV0U instruction.

**[bit8] T: Step trace trap flag**

It is a flag which specifies whether to make the step trace trap effective.

Value	Description
0	Step trace trap invalidity.
1	Step trace trap effective. In this case, all user NMIs and user interrupts are disabled.

Initialized to "0" by a reset.

The emulator uses the function of the step trace trap. When the emulator is used, the use cannot be done in user program.

## ■ ILM

bit20	bit19	bit18	bit17	bit16	Initial value
ILM4	ILM3	ILM2	ILM1	ILM0	01111 <sub>B</sub>

This register stores the interrupt level mask value and the value set in the ILM is used as the level mask.

The interrupt request to be input to the CPU is accepted only when its interrupt level is stronger than the level indicated by this ILM.

As for the level value, 0 (00000<sub>B</sub>) is the strongest, and 31 (11111<sub>B</sub>) is the weakest.

There is a limitation in the value which can be set from the program.

### When former value is 16 to 31

New values can only be set in the range 16 to 31. Executing an instruction that sets a value between 0 and 15 results in (specified value + 16) being transferred.

### When former value is 0 to 15

Any value of 0 to 31 can be set.

Initialized to 15 (01111<sub>B</sub>) by reset.

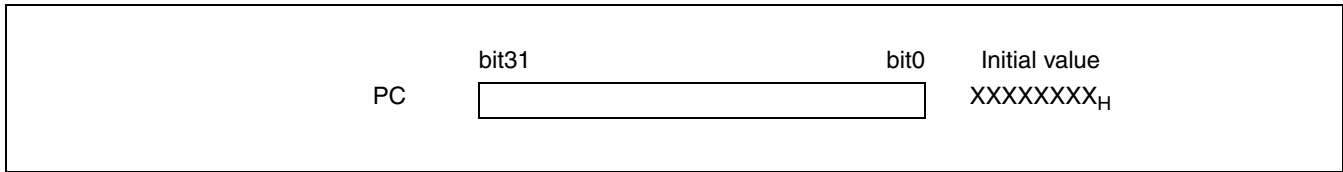
### [Notes of PS register]

As some instructions pre-process the PS register, the following exception operations may cause a break to occur in an interrupt processing routine when using the debugger or the updating of the PS flag.

In either case, the system is designed to re-execute correctly after the EIT returns and therefore processing before and after the EIT is executed correctly.

- The following operations may occur when (a) user interrupt/NMI is received, (b) step execution is performed, (c) break occurs in a data event or emulator menu in an instruction immediately preceding DIV0U/DIV0S instruction.
  - D0 and D1 flags precede and are renewed.
  - EIT processing routine (user interruption, NMI, or emulator) is executed.
  - After returning from EIT, DIV0U/DIV0S instructions are executed and the D0 and D1 flags are updated to the same value as (1).
- When each ORCCR/STILM/MOV Ri and PS instruction is executed to permit interrupting with the user interruption and the NMI factor generated, the following operations are done.
  - The PS register precedes and is updated.
  - Execute an EIT processing routine (user interrupt or NMI).
  - After returning from EIT, the above instructions are executed and the PS register is updated to the same value as (1).

### ■ PC (Program Counter)



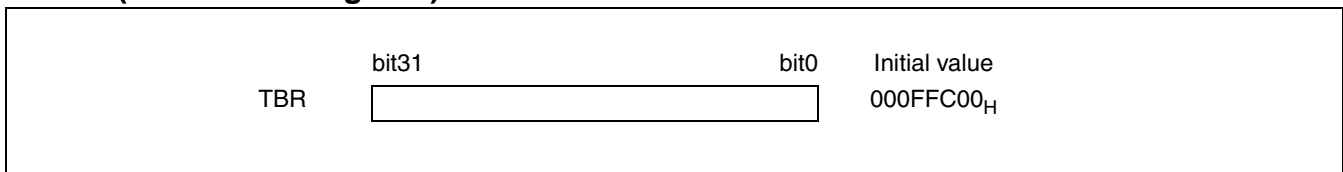
The address of the executed instruction is shown with the program counter.

Bit0 is set to "0" when the PC is updated during instruction execution. Bit0 can only go to "1" in the case when an odd-numbered address is specified as a branch destination address.

However, bit0 is ignored in this case also and instructions must be located at addresses that are a multiple of two.

The initial value by reset is irregular.

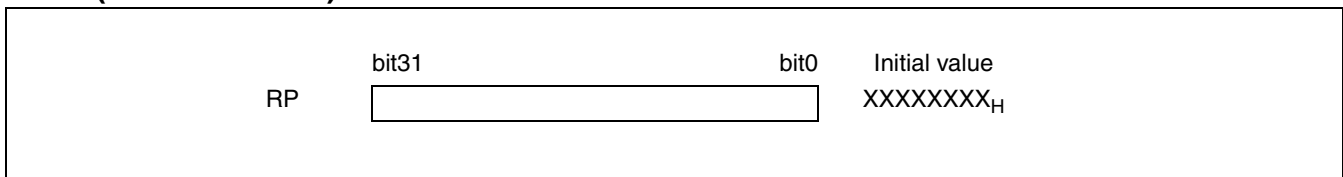
### ■ TBR (Table Base Register)



The table base register stores the start address of the vector table used in EIT processing.

The initial value by reset is 000FFC00<sub>H</sub>.

### ■ RP (Return Pointer)



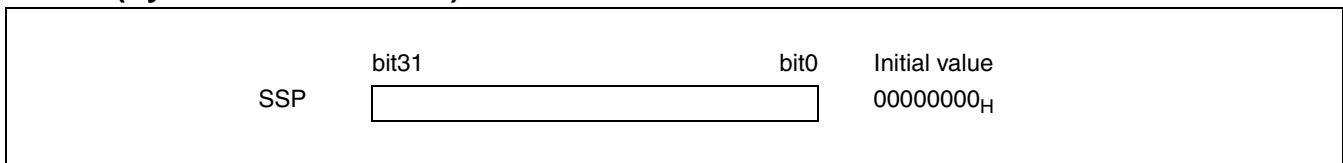
The address which returns from the sub routine is maintained with the return pointer.

The value of PC is forwarded to this RP at CALL instruction execution time.

The content of RP is forwarded to PC at RET instruction execution time.

The initial value by reset is irregular.

### ■ SSP (System Stack Pointer)



The SSP is the system stack pointer.

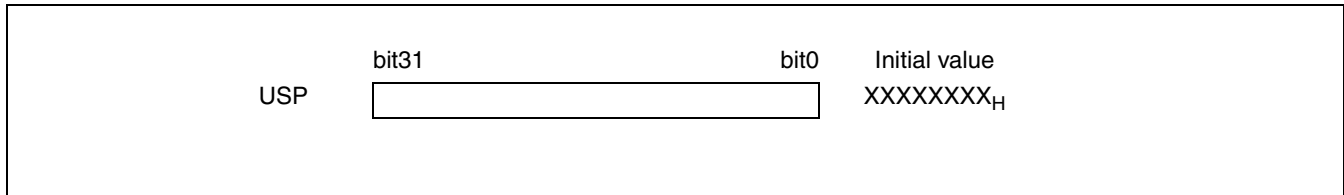
Functions as R15 when the S flag is "0".

The SSP can also be specified explicitly.

Also used as the stack pointer specifying the stack that saves the PS and PC when EIT occurs.

The initial value by reset is 00000000<sub>H</sub>.

## ■ USP (User Stack Pointer)



The USP is the user stack pointer.

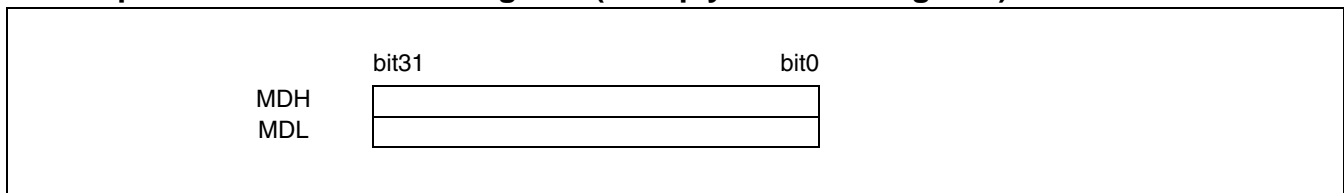
Functions as R15 when the S flag is "1".

The USP can also be specified explicitly.

The initial value by reset is irregular.

The use cannot be done in the RETI instruction.

## ■ Multiplication and Division Register (Multiply & Divide Register)



This register is the register for multiplication and division, and 32-bit lengths respectively.

The initial value by reset is irregular.

### Multiplication execution time

When performing 32-bit  $\times$  32-bit multiplication, 64-bit length calculation results are stored in the multiplication/division results storage register in the following format.

MDH: Higher 32 bits

MDL: Lower 32 bits

When 16 bits  $\times$  16 bits are multiplied, the result is stored as follows.

MDH: Indeterminate

MDL: 32 bits

### Division execution time

When beginning to calculate, the dividend is stored in MDL.

When division is performed using the DIV0S/DIV0U, DIV1, DIV2, DIV3, and DIV4S commands, the results are stored in MDL and MDH.

MDH: Surplus

MDL: Quotient

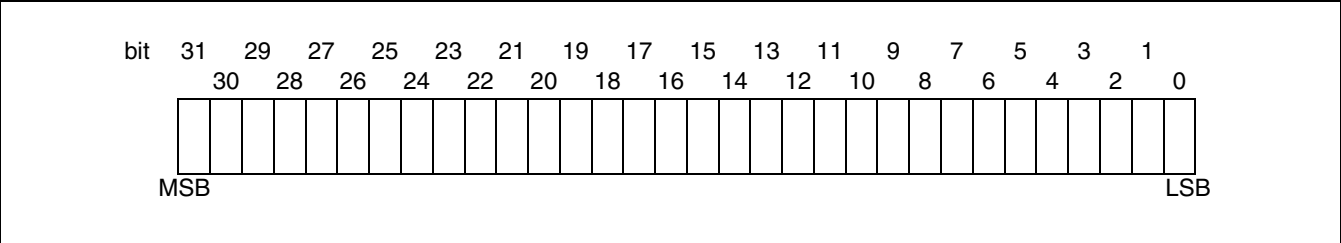
### 3.5 Data Structure

This section explains the bit ordering, byte ordering, and word alignment.

■ Bit Ordering

In the FR family, the little endian has been adopted as a bit ordering.

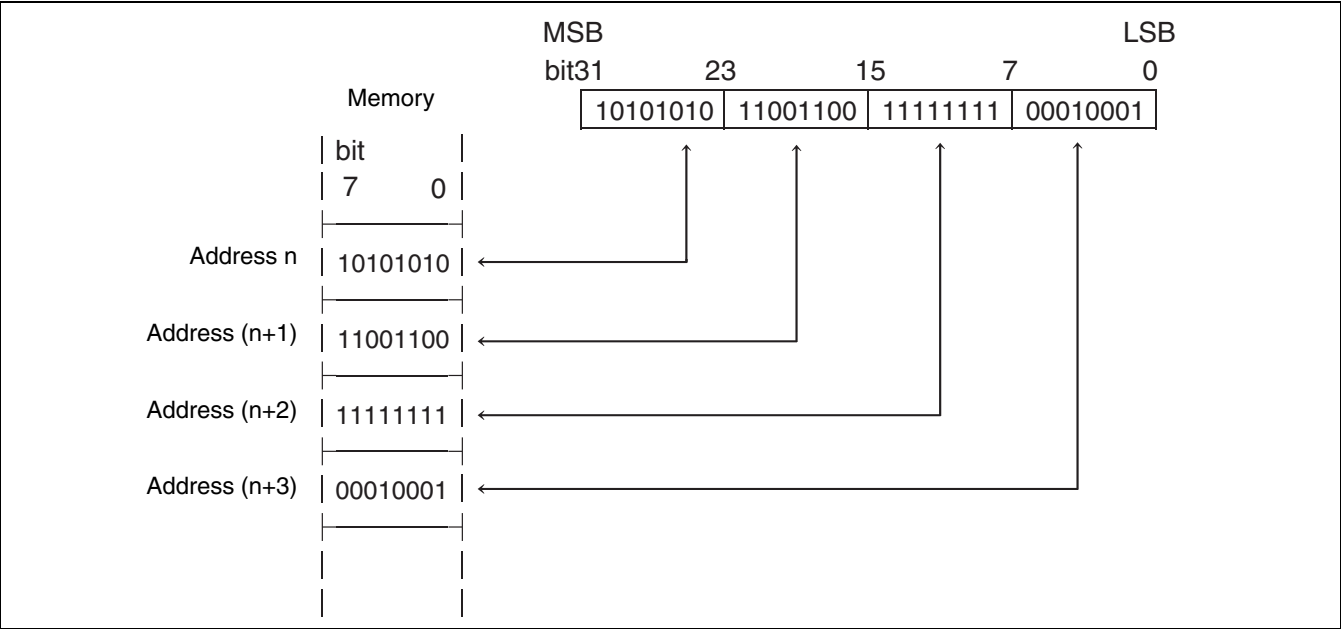
Figure 3.5-1 Bit Ordering



■ Byte Ordering

In the FR family, the big endian has been adopted as byte ordering.

Figure 3.5-2 Byte Ordering



## ■ Word Alignment

### ● Program access

It is necessary to arrange the program of the FR family in the address of the multiple of two.

Bit0 of PC is set to "0" when the PC is updated during instruction execution. The bit can only go to "1" in the case when an odd-numbered address is specified as a branch destination address.

However, bit0 is ignored in this case also and instructions must be located at addresses that are a multiple of two.

There is no odd-number address exception.

### ● Data access

In the FR family, when data access is performed, forced alignment is provided to addresses as follows in accordance with their width.

Word access : The address is a multiple of four (The lowest 2 bits are forcibly set to "00").

Half-word access : The address is a multiple of two (The lowest bit is forcibly set to "0").

Byte access : ----

When word or half-word data is accessed, "0" is forcibly set to some bits, which are the calculation results of the effective address. For example, in the @(R13, Ri) addressing mode, the pre-addition register is used for calculations as it is (even though the lowest bit is "1"), and the lower bits of the addition results will be masked. The register prior to the calculation is not masked.

[Example] LD@ (R13, R2), R0

	R13	00002222 <sub>H</sub>
	R2	00000003 <sub>H</sub>
+) )		
<hr/>		
Addition result		00002225 <sub>H</sub>
		↓ Compulsion mask of lower two bits
Address pin		00002224 <sub>H</sub>



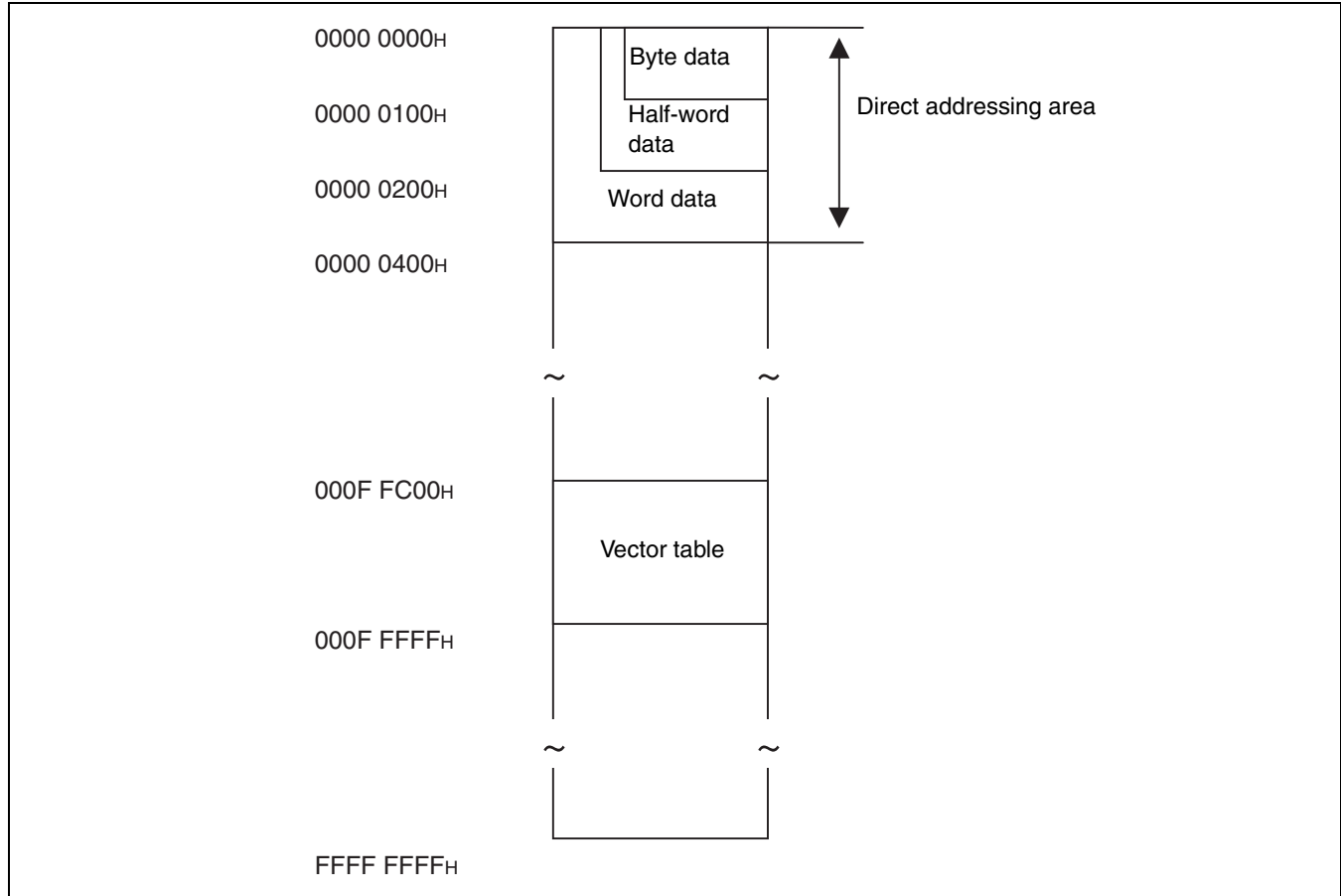
## 3.6 Memory Map

This section describes a memory map of the MB91265A series.

### ■ Memory Map

The address space is 32-bit linear.

Figure 3.6-1 Memory Map



### ■ Direct Addressing Area

The undermentioned area of the address space is a area for I/O. This area can directly specify the operand address in the instruction using the direct addressing.

The size of the address area of direct possible addressing is different in each data length.

- Byte data (8-bits) : 000<sub>H</sub> to 0FF<sub>H</sub>
- Half-word data (16-bits) : 000<sub>H</sub> to 1FF<sub>H</sub>
- Word data (32-bits) : 000<sub>H</sub> to 3FF<sub>H</sub>

### ■ Vector Table Initial Area

The area of 000FFC00<sub>H</sub> to 000FFFFF<sub>H</sub> is EIT vector table initial area.

The vector table used for EIT processing can be allocated to an arbitrary address by rewriting the TBR, but it is allocated to this address on initialization through reset.

## 3.7 Divergence Instructions

---

**In the FR family, whether the operations are with or without delay slots can be specified for the branch command.**

---

### ■ Operation with Delay Slot

#### ● Instruction

The instructions with the notation shown below perform a branch operation with a delay slot.

JMP:D	@Ri	CALL:D	label12	CALL:D	@Ri	RET:D
BRA:D	label9	BNO:D	label9	BEQ:D	label9	BNE:D label9
BC:D	label9	BNC:D	label9	BN:D	label9	BP:D label9
BV:D	label9	BNV:D	label9	BLT:D	label9	BGE:D label9
BLE:D	label9	BGT:D	label9	BLS:D	label9	BHI:D label9

#### ● Operation explanation

Operations with delay slots branch out after executing the command placed just after the branch command (called a "delay slot") before executing the branch destination command. As the instruction in the delay slot is executed prior to the branch, the apparent execution speed is one cycle. The NOP command must be placed as an alternative if an effective command cannot be inserted in the delay slot.

[Example]

```

;      Row of instruction
ADD    R1, R2  ;
BRA:D  LABEL  ; Divergence instructions
MOV    R2, R3  ; Delay slot ... Executed before branch.
...
LABEL:ST      R3,@R4 ; Branch destination

```

The command placed in the delay slot is executed regardless of whether the branch condition for the condition branch command will be met or not.

For delay branch commands, the execution order of the partial command seems to be reversed, but this applies only to PC update operations, and other operations (i.e. update and refer to register) are absolutely executed in the described order.

A concrete explanation is done as follows.

1. The Ri to be referred to for the JMP:D@Ri/CALL:D@Ri command will not be affected even if the command within the delay slot updates the Ri.

[Example]

```

LDI:32  #Label, R0
JMP:D   @R0      ; Branches out to Label
LDI:8    #0,     R0      ; Has no effect on branch destination address
...

```

2. The RP to be referred by the RET:D command will not be affected even if the command within the delay slot updates the RP.

[Example]

```
RET:D                ; Branches to the address in RP set previously.
MOV    R8,    RP    ; Has no effect on return operation.
...
```

3. Flags to be referred by the Bcc:D rel instruction are also not affected by the delay slot instruction.

[Example]

```
ADD    #1,    R0    ; Flag change
BC:D    Overflow    ; Branches based on execution result of above instruction.
ANDCCR #0                ; Do not refer to this flag update in the above-mentioned branch
                        ; instruction.
...
```

4. If the instruction in the delay slot for the CALL:D instruction refers RP, it reads the value after updating by CALL:D instruction.

[Example]

```
CALL:D Label        ; Updating RP and branching
MOV    RP,    R0    ; RP of an execution result in the above-mentioned
                        ; CALL:D is forwarded.
...
```

## ● Restrictions

### 1. Instruction that can be placed in the delay slot

Only instructions that satisfy the following conditions can be executed in the delay slot.

- 1-cycle command
- No branch instruction.
- Instruction whose operation is not affected even though the order is changed

The "1-cycle command" is a command with "1", "a", "b", "c", or "d" described in the cycle number column within the command list.

### 2. Step trace trap

Step trace trap is not generated between executing the branch command with the delay slot and the delay slot.

### 3. Interrupts and NMI

Interrupts and NMI cannot be received between execution of a branch instruction with a delay slot and the delay slot.

### 4. Undefined instruction exception

No undefined instruction exception occurs if the delay slot contains an undefined instruction. At this time, undefined instruction operates as NOP instruction.

## ■ Operation without Delay Slot

### ● Instruction

Instructions written as follows perform a branch operation without a delay slot:

JMP @Ri	CALL label12	CALL @Ri	RET
BRA label9	BNO label9	BEQ label9	BNE label9
BC label9	BNC label9	BN label9	BP label9
BV label9	BNV label9	BLT label9	BGE label9
BLE label9	BGT label9	BLS label9	BHI label9

### ● Operation explanation

When the delay slot is not used, instructions are executed in the sequence they are coded. The next instruction will not be executed prior to the branch.

[Example]

```

;      Row of instruction
      ADD    R1, R2    ;
      BRA    LABEL    ; Branch instruction (delay slot none)
      MOV    R2, R3    ; Not executed
      ...
      LABEL ST    R3, @R4 ; The divergence destination

```

Execution cycle number for branch commands without delay slots will be 2 cycles branched, or 1 cycle non-branched.

As the appropriate command cannot be inserted into the delay slot, the command code efficiency can be improved more than the branch command with delay slot described the NOP.

A balance between execution speed and code efficiency can be struck by selecting either the operation with the delay slot when effective commands can be set in the delay slot or the operation without the delay slot when effective commands cannot be set.

## 3.8 EIT (Exception, Interruption, and Trap)

---

EIT, which is the generic term for "Exception", "Interrupt", and "Trap" indicates that the program is suspended due to events generated while running the current program and another program is being executed.

The exception is an incident which occurs in relation to the context under execution.

Execution continues from the instruction that caused the exception.

The interruption is an incident which occurs without any relation to the context under execution. The event factor is hardware.

The trap is an incident which occurs in relation to the context under execution. There is something directed by the program like the system call. Execution continues from the instruction after the instruction that caused the trap.

---

### ■ Feature of EIT

- Multiple interrupt is supported to the interruption
- It is a level mask function (15 levels are available the user) to the interruption
- Trap instruction (INT)
- EIT (hardware/software) for emulator startup

### ■ EIT Factor

The following is used as a EIT factor.

- Reset
- User interruption (internal resource and external interruption)
- NMI
- Delayed interrupt
- Undefined instruction exception
- Trap instruction (INT)
- Trap instruction (INTE)
- Step trace trap
- Coprocessor absent trap
- Coprocessor error trap

### ■ Return from EIT

- The REIT instruction is used for the return from EIT.

## ■ Interrupt Level

Interrupt levels are 0 to 31 and are managed by five bits.

The allocation of each level is as follows.

**Table 3.8-1 Interrupt Level**

Level		Interrupt Factor	Notes
Binary	Decimal		
00000	0	(System reservation)	If the original value of ILM is between 16 and 31, the value of this range cannot be set in the ILM with program.
...	...	...	
...	...	...	
00011	3	(System reservation)	
00100	4	INTE instruction	
		Step trace trap	
00101	5	(System reservation)	
...	...	...	
...	...	...	
01110	14	(System reservation)	
01111	15	NMI (for user)	
10000	16	Interrupt	When ILM is set, it is a user interruption interdiction.
10001	17	Interrupt	
...	...	...	
...	...	...	
11110	30	Interrupt	
11111	31	-	When ICR is set, it is an interruption interdiction.

It is a level of 16 to 31 that the operation is possible.

Undefined command exceptions, coprocessor absent traps, coprocessor error traps, and INT commands are not affected for interruption levels. Moreover, ILM may not be changed.

## ■ I Flag

It is a flag which specifies the permission and interdiction of the interruption. Contained in bit4 of CCR in the PS register.

Value	Description
0	Interruption interdiction. Cleared to "0" when the INT instruction is executed. (However, the value saved on the stack is the value before the bit is cleared.)
1	Interruption permission. The mask processing of the interruption demand is controlled by the value which ILM maintains.

## ■ ILM

It is PS register (bit20 to bit16) which maintains the interrupt level mask value.

The interrupt request to be input to the CPU is accepted only when its interrupt level is stronger than the level indicated by this ILM.

The highest level is 0 (00000<sub>B</sub>) and the lowest level is 31 (11111<sub>B</sub>).

There is a limitation in the value which can be set from the program. When the original value is between 16 and 31, new values can only be set in the range 16 to 31. Executing an instruction that sets a value between 0 and 15 results in (specified value + 16) being transferred.

When former value is 0 to 15, any value of 0 to 31 can be set. Use the STILM instruction to set the ILM.

## ■ Level Mask to Interruption and NMI

When an NMI or interrupt request occurs, the interrupt level (see Table 3.8-1) corresponding to the interrupt factor is compared with the level mask set in the ILM. And, when the following condition consists, the mask is done, and the demand is not accepted.

Interrupt levels of factor  $\geq$  level mask value

## ■ ICR (Interrupt Control Register)

This register is located in the interrupt controller and specifies the level for each interrupt request. ICR is prepared corresponding to each of the interruption demand input. The mapping is done in the I/O space, and ICR is accessed by CPU through the bus.

### ● The ICR bit make-up

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	---11111 <sub>B</sub>
-	-	-	R	R/W	R/W	R/W	R/W	

[bit4] ICR4 is always "1".

[bit3 to bit0] ICR3 to ICR0

Lower four bits of the interrupt level for the corresponding interrupt factor. The read/write is possible.

ICR can set the value within the range of 16 to 31 together with bit4.

● ICR mapping

**Table 3.8-2 Interruption Factor, Interruption Control Register, and Interruption Vector**

Interrupt Factor	Interrupt Control Register		Corresponding Interruption Vector		
			Number		Address
			Hexadecimal	Decimal	
IRQ00	ICR00	00000440 <sub>H</sub>	10 <sub>H</sub>	16	TBR + 3BC <sub>H</sub>
IRQ01	ICR01	00000441 <sub>H</sub>	11 <sub>H</sub>	17	TBR + 3B8 <sub>H</sub>
IRQ02	ICR02	00000442 <sub>H</sub>	12 <sub>H</sub>	18	TBR + 3B4 <sub>H</sub>
...	...	...	...	...	...
...	...	...	...	...	...
IRQ45	ICR45	0000046D <sub>H</sub>	3D <sub>H</sub>	61	TBR + 308 <sub>H</sub>
IRQ46	ICR46	0000046E <sub>H</sub>	3E <sub>H</sub>	62	TBR + 304 <sub>H</sub>
IRQ47	ICR47	0000046F <sub>H</sub>	3F <sub>H</sub>	63	TBR + 300 <sub>H</sub>

TBR initial value: 000F FC00<sub>H</sub>

Reference: Please refer to "CHAPTER 5 INTERRUPT CONTROLLER".

■ **SSP (System Stack Pointer)**

SSP	<div> <div>bit31</div> <div>bit0</div> <div></div> </div>	Initial value 00000000 <sub>H</sub>
-----	---	--

The SSP is used as the pointer to the stack used to save and restore data when an EIT is accepted or a return operation occurs.

8 is deducted from the content during EIT processing, and 8 is added when returning from EIT in line with execution of the RETI command.

The initial value by reset is 00000000<sub>H</sub>.

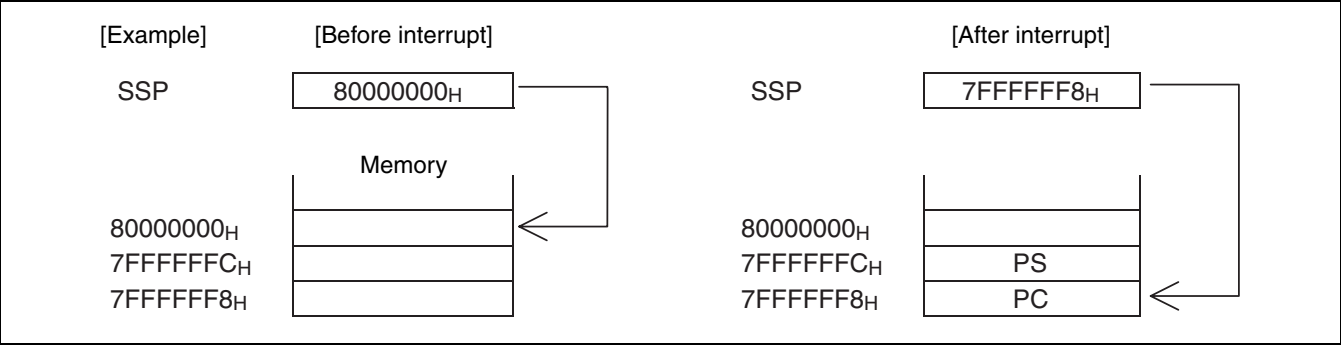
The SSP can also be used as general-purpose register R15 when the S flag in the CCR is "0".



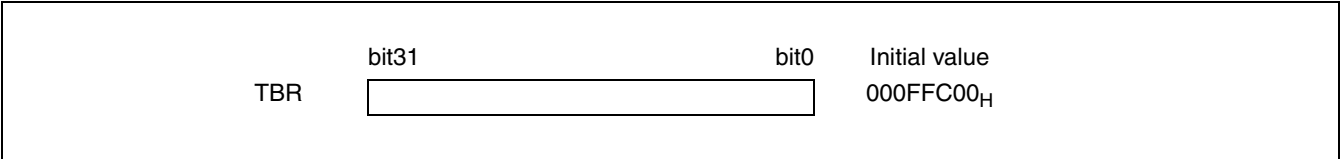
■ Interrupt Stack

The value in the PC or PS is saved to or restored from an area pointed to by the system stack pointer (SSP). After an interrupt, the PC is stored at the address contained in the SSP and PS is stored at the (SSP + 4) address.

Figure 3.8-1 Interrupt Stack



■ TBR (Table Base Register)



It is a register which shows the first address of the vector table for EIT.

The vector address is generated by adding the offset corresponding to each TBR and EIT factor.

The initial value by reset is 000FFC00\_H.

■ EIT Vector Table

The vector region for EIT is 1 KB region starting at address indicated by the TBR.

Each vector consists of four bytes and the relationship between the vector number and vector address is as follows.

$$\begin{aligned} \text{vctadr} &= \text{TBR} + \text{vctofs} \\ &= \text{TBR} + (3\text{FC}_\text{H} - 4 \times \text{vct}) \end{aligned}$$

vctadr : Vector address

Vctofs: Vector offset

vct: Vector number

The lower two bits of the addition result are always treated as "00<sub>B</sub>".

The region of 000FFC00\_H to 000FFFFFF\_H is an initial region of the vector table by reset.

A special function is partially allocated to the vector. Table 3.8-3 shows the vector table in architecture.

**Table 3.8-3 Vector Table (1 / 3)**

Interrupt Factor	Interrupt Number		Interrupt Level	Offset	Default Address of TBR
	Decimal	Hexadecimal			
Reset <sup>*1</sup>	0	00	-	3FC <sub>H</sub>	000FFFFC <sub>H</sub>
Mode vector <sup>*1</sup>	1	01	-	3F8 <sub>H</sub>	000FFFF8 <sub>H</sub>
Reserved for system	2	02	-	3F4 <sub>H</sub>	000FFFF4 <sub>H</sub>
Reserved for system	3	03	-	3F0 <sub>H</sub>	000FFFF0 <sub>H</sub>
Reserved for system	4	04	-	3EC <sub>H</sub>	000FFFE <sub>C</sub>
Reserved for system	5	05	-	3E8 <sub>H</sub>	000FFFE8 <sub>H</sub>
Reserved for system	6	06	-	3E4 <sub>H</sub>	000FFFE4 <sub>H</sub>
Coprocessor absent trap	7	07	-	3E0 <sub>H</sub>	000FFFE0 <sub>H</sub>
Coprocessor error trap	8	08	-	3DC <sub>H</sub>	000FFFD <sub>C</sub>
INTE instruction	9	09	-	3D8 <sub>H</sub>	000FFFD8 <sub>H</sub>
Reserved for system	10	0A	-	3D4 <sub>H</sub>	000FFFD4 <sub>H</sub>
Reserved for system	11	0B	-	3D0 <sub>H</sub>	000FFFD0 <sub>H</sub>
Step trace trap	12	0C	-	3CC <sub>H</sub>	000FFF <sub>CC</sub>
NMI demand (tool)	13	0D	-	3C8 <sub>H</sub>	000FFF <sub>C8</sub>
Undefined instruction exception	14	0E	-	3C4 <sub>H</sub>	000FFF <sub>C4</sub>
NMI demand	15	0F	15(F <sub>H</sub> ) fixed	3C0 <sub>H</sub>	000FFF <sub>C0</sub>
External interrupt 0	16	10	ICR00	3BC <sub>H</sub>	000FFF <sub>BC</sub>
External interrupt 1	17	11	ICR01	3B8 <sub>H</sub>	000FFF <sub>B8</sub>
External interrupt 2	18	12	ICR02	3B4 <sub>H</sub>	000FFF <sub>B4</sub>
External interrupt 3	19	13	ICR03	3B0 <sub>H</sub>	000FFF <sub>B0</sub>
External interrupt 4	20	14	ICR04	3AC <sub>H</sub>	000FFF <sub>AC</sub>
External interrupt 5	21	15	ICR05	3A8 <sub>H</sub>	000FFF <sub>A8</sub>
External interrupt 6/C-CAN wake up <sup>*2</sup>	22	16	ICR06	3A4 <sub>H</sub>	000FFF <sub>A4</sub>
External interrupt 7	23	17	ICR07	3A0 <sub>H</sub>	000FFF <sub>A0</sub>
Reload timer 0	24	18	ICR08	39C <sub>H</sub>	000FFF <sub>9C</sub>
Reload timer 1	25	19	ICR09	398 <sub>H</sub>	000FFF <sub>98</sub>
Reload timer 2	26	1A	ICR10	394 <sub>H</sub>	000FFF <sub>94</sub>
UART0 (receive)	27	1B	ICR11	390 <sub>H</sub>	000FFF <sub>90</sub>
UART0 (transmit)	28	1C	ICR12	38C <sub>H</sub>	000FFF <sub>8C</sub>
DTTI pin	29	1D	ICR13	388 <sub>H</sub>	000FFF <sub>88</sub>
DMAC0 (end and error)	30	1E	ICR14	384 <sub>H</sub>	000FFF <sub>84</sub>
DMAC1 (end and error)	31	1F	ICR15	380 <sub>H</sub>	000FFF <sub>80</sub>
DMAC2/DMAC3/DMAC4 (end and error)	32	20	ICR16	37C <sub>H</sub>	000FFF <sub>7C</sub>
UART1 (receive completed)	33	21	ICR17	378 <sub>H</sub>	000FFF <sub>78</sub>
UART1 (transmit completed)	34	22	ICR18	374 <sub>H</sub>	000FFF <sub>74</sub>
C-CAN 0 <sup>*2</sup>	35	23	ICR19	370 <sub>H</sub>	000FFF <sub>70</sub>

**Table 3.8-3 Vector Table (2 / 3)**

Interrupt Factor	Interrupt Number		Interrupt Level	Offset	Default Address of TBR
	Decimal	Hexadecimal			
Reserved for system	36	24	ICR20	36C <sub>H</sub>	000FFF6C <sub>H</sub>
Sum of product	37	25	ICR21	368 <sub>H</sub>	000FFF68 <sub>H</sub>
PPG0/PPG1	38	26	ICR22	364 <sub>H</sub>	000FFF64 <sub>H</sub>
PPG2/PPG3	39	27	ICR23	360 <sub>H</sub>	000FFF60 <sub>H</sub>
PPG4/PPG5/PPG6/PPG7	40	28	ICR24	35C <sub>H</sub>	000FFF5C <sub>H</sub>
Reserved for system	41	29	ICR25	358 <sub>H</sub>	000FFF58 <sub>H</sub>
Waveform (underflow) 0/1/2	42	2A	ICR26	354 <sub>H</sub>	000FFF54 <sub>H</sub>
Free-run timer 1 (compare clear)	43	2B	ICR27	350 <sub>H</sub>	000FFF50 <sub>H</sub>
Free-run timer 1 (0-detect)	44	2C	ICR28	34C <sub>H</sub>	000FFF4C <sub>H</sub>
Free-run timer 2 (compare clear)	45	2D	ICR29	348 <sub>H</sub>	000FFF48 <sub>H</sub>
Free-run timer 2 (0-detect)	46	2E	ICR30	344 <sub>H</sub>	000FFF44 <sub>H</sub>
Time-base timer overflow	47	2F	ICR31	340 <sub>H</sub>	000FFF40 <sub>H</sub>
Free-run timer 0 (compare clear)	48	30	ICR32	33C <sub>H</sub>	000FFF3C <sub>H</sub>
Free-run timer 0 (0-detect)	49	31	ICR33	338 <sub>H</sub>	000FFF38 <sub>H</sub>
Reserved for system	50	32	ICR34	334 <sub>H</sub>	000FFF34 <sub>H</sub>
A/D converter 1	51	33	ICR35	330 <sub>H</sub>	000FFF30 <sub>H</sub>
A/D converter 2	52	34	ICR36	32C <sub>H</sub>	000FFF2C <sub>H</sub>
PWC0 (measurement finished)	53	35	ICR37	328 <sub>H</sub>	000FFF28 <sub>H</sub>
Reserved for system	54	36	ICR38	324 <sub>H</sub>	000FFF24 <sub>H</sub>
PWC0 (overflow)	55	37	ICR39	320 <sub>H</sub>	000FFF20 <sub>H</sub>
Reserved for system	56	38	ICR40	31C <sub>H</sub>	000FFF1C <sub>H</sub>
ICU 0 (fetch)	57	39	ICR41	318 <sub>H</sub>	000FFF18 <sub>H</sub>
ICU 1 (fetch)	58	3A	ICR42	314 <sub>H</sub>	000FFF14 <sub>H</sub>
ICU 2/ICU 3 (fetch)	59	3B	ICR43	310 <sub>H</sub>	000FFF10 <sub>H</sub>
OCU0/OCU1 (match)	60	3C	ICR44	30C <sub>H</sub>	000FFF0C <sub>H</sub>
OCU2/OCU3 (match)	61	3D	ICR45	308 <sub>H</sub>	000FFF08 <sub>H</sub>
OCU4/OCU5 (match)	62	3E	ICR46	304 <sub>H</sub>	000FFF04 <sub>H</sub>
Delay interrupt trigger bit	63	3F	ICR47	300 <sub>H</sub>	000FFF00 <sub>H</sub>
Reserved for system (used for REALOS)	64	40	-	2FC <sub>H</sub>	000FFEFC <sub>H</sub>
Reserved for system (used for REALOS)	65	41	-	2F8 <sub>H</sub>	000FFE8 <sub>H</sub>
Reserved for system	66	42	-	2F4 <sub>H</sub>	000FFE4 <sub>H</sub>
Reserved for system	67	43	-	2F0 <sub>H</sub>	000FFE0 <sub>H</sub>
Reserved for system	68	44	-	2EC <sub>H</sub>	000FEEC <sub>H</sub>
Reserved for system	69	45	-	2E8 <sub>H</sub>	000FEE8 <sub>H</sub>
Reserved for system	70	46	-	2E4 <sub>H</sub>	000FEE4 <sub>H</sub>
Reserved for system	71	47	-	2E0 <sub>H</sub>	000FEE0 <sub>H</sub>
Reserved for system	72	48	-	2DC <sub>H</sub>	000FEDC <sub>H</sub>

**Table 3.8-3 Vector Table (3 / 3)**

Interrupt Factor	Interrupt Number		Interrupt Level	Offset	Default Address of TBR
	Decimal	Hexadecimal			
Reserved for system	73	49	-	2D8 <sub>H</sub>	000FFED8 <sub>H</sub>
Reserved for system	74	4A	-	2D4 <sub>H</sub>	000FFED4 <sub>H</sub>
Reserved for system	75	4B	-	2D0 <sub>H</sub>	000FFED0 <sub>H</sub>
Reserved for system	76	4C	-	2CC <sub>H</sub>	000FFECC <sub>H</sub>
Reserved for system	77	4D	-	2C8 <sub>H</sub>	000FFEC8 <sub>H</sub>
Reserved for system	78	4E	-	2C4 <sub>H</sub>	000FFEC4 <sub>H</sub>
Reserved for system	79	4F	-	2C0 <sub>H</sub>	000FFEC0 <sub>H</sub>
Used in INT instruction	80 to 255	50 to FF	-	2BC <sub>H</sub> to 000 <sub>H</sub>	000FFEBC <sub>H</sub> to 000FFC00 <sub>H</sub>

\*1: Even if the TBR value is modified, the fixed addresses 000FFFFC<sub>H</sub> and 000FFF8<sub>H</sub> are always used for the reset vector and mode vector.

\*2: Interrupt of C-CAN is function loaded in MB91F267NA/MB91267NA.

## ■ Multiple EIT Processing

When a number of EIT factors are simultaneously generated, the CPU repeats the operation of selecting and accepting one EIT factor, executing the EIT sequence, and then detecting EIT factors again.

When EIT factors are detected if there are no more EIT factors that can be accepted, the handler command for the last EIT factor accepted will be executed.

Accordingly, if more than one EIT factor occurs at the same time, the sequence for executing the handler for each EIT is determined by the following two elements:

- (1) Priority level of EIT factor acceptance
- (2) How other factors can be masked when one factor is accepted

## ■ Priority Level of EIT Factor

The priority for accepting EIT factors is the order for selecting factors executing EIT sequence that saves the PS and PC, updates the PC (on demand), and executes mask processing for other factors.

The handler of the factor previously accepted is not previously executed necessarily.

The priority of the EIT factor acceptance is shown in Table 3.8-4.

**Table 3.8-4 Priority of EIT Factor Acceptance and Mask to Other Factors**

Priority of acceptance	Factor	Mask to other factors
1	Reset	Other factors are annulled.
2	Undefined instruction exception	Cancellation
3	INTE instruction	ILM = 4 Other factors are annulled.
4	INT instruction	I flag = 0
5	Coprocessor absent trap Coprocessor error trap	-
6	User interrupt	ILM = level of accepted factor
7	NMI (for user)	ILM = 15
8	NMI (for emulator)	ILM = 4
9	Step trace trap	ILM = 4

Considering the mask processing for other EIT factors after an EIT factor is accepted, the sequence for executing the handlers for EIT factors that occur simultaneously is shown in Table 3.8-5.

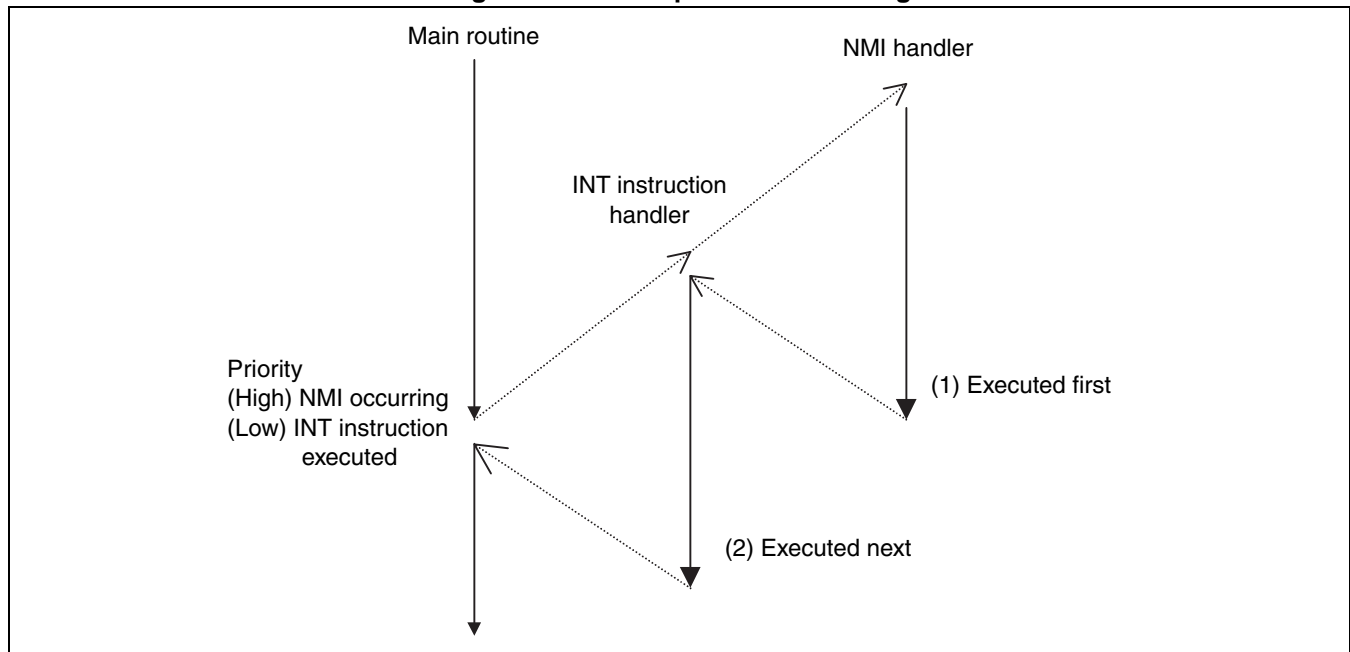
**Table 3.8-5 Execution Sequence of EIT Handler**

Execution sequence of handler	Factor
1	Reset *
2	Undefined instruction exception
3	INTE instruction *
4	Step trace trap
5	NMI (for user)
6	INT instruction
7	User interrupt
8	Coprocessor absent trap and coprocessor error trap

\*: Other factors are annulled.

[Example]

**Figure 3.8-2 Multiple EIT Processing**



## ■ EIT Operation

In the following explanation, the transfer source "PC" is address of the instruction at which each EIT factor was detected.

Similarly, the "address of the following instruction" has the following meaning depending on the instruction at which each EIT was detected.

- At LDI:  $32 \rightarrow PC + 6$
- At LDI: 20, COPOP, COPLD, COPST, COPSV  $\rightarrow PC + 4$
- At the other instructions:  $PC + 2$

### ● Operation of user interruption and NMI

When a user interrupt or user NMI interrupt request occurs, the following sequence is used to determine whether or not to accept the request.

[Right or wrong judgment of interruption demand acceptance]

- (1) The interruption levels of requests that are generated simultaneously are compared, and the one with the highest level (the smallest numeric value) will be selected.

For the level used for the comparison, the value held in the corresponding ICR is used for a maskable interrupt and the predefined constant is used for the NMI.

- (2) If a number of interruption requests with the same level are generated, the interruption request with the smallest interruption number will be selected.

- (3) When the interrupt level is greater than or equal to the level mask value, the interrupt request is masked and is not accepted.

To (4) at interrupt levels  $<$  level mask value.

- (4) When the selected interruption request is an interruption that can be masked, the interruption request will be masked and will not be accepted if the I flag is "0". To (5) if I flag is "1".

To (5) regardless of the I flag value when the selected interruption demand is NMI.

- (5) If the above conditions are satisfied, the interrupt request is accepted at the instruction processing boundary.

If a user interrupt or NMI request is accepted when an EIT request is detected, the CPU operation is as follows based on the interrupt number of the accepted interrupt request.

() in the [operation] shows the address which the register indicates.

[Operation]

- |   |                      |
|---|----------------------|
| (1) SSP - 4   | $\rightarrow$ SSP    |
| (2) PS  | $\rightarrow$ (SSP)  |
| (3) SSP - 4   | $\rightarrow$ SSP    |
| (4) Address of the following instruction                  | $\rightarrow$ (SSP)  |
| (5) Interrupt levels of accepted demand                   | $\rightarrow$ ILM    |
| (6) "0"   | $\rightarrow$ S flag |
| (7) (TBR + vector offset of accepted interruption demand) | $\rightarrow$ PC     |

Detection of any new EITs is performed after the interrupt sequence completes and before the initial instruction of the handler is executed. If an EIT that is able to be accepted is found at this time, the CPU changes to the EIT processing sequence.

### ● Operation of INT instruction

INT #u8

Branches to the interrupt handler at the vector indicated by u8.

[Operation]

- (1) SSP - 4 → SSP
- (2) PS → (SSP)
- (3) SSP - 4 → SSP
- (4) PC + 2 → (SSP)
- (5) "0" → I flag
- (6) "0" → S flag
- (7) (TBR + 3FC<sub>H</sub> - 4 × u8) → PC

### ● Operation of INTE instruction

INTE

Branches to the interrupt handler for the vector with vector number #9.

[Operation]

- (1) SSP - 4 → SSP
- (2) PS → (SSP)
- (3) SSP - 4 → SSP
- (4) PC + 2 → (SSP)
- (5) "00100" → ILM
- (6) "0" → S flag
- (7) (TBR + 3D8<sub>H</sub>) → PC

Do not use the INTE command during the INTE command and step trace trap processing routine.

Moreover, EIT is not generated while executing the step by INTE.

### ● Operation of step trace trap

If the T flag in the SCR in the PS is set to enable the step trace function, a trap occurs after each instruction and execution breaks.

[Condition of step trace trap detection]

- (1) T flag = 1
- (2) Not a delayed branch instruction.
- (3) Executing code other than an INTE instruction or step trace trap processing routine.
- (4) If the above conditions are satisfied, execution breaks at each instruction boundary.



[Operation]

- |  |          |
|--|----------|
| (1) SSP - 4                              | → SSP    |
| (2) PS                                   | → (SSP)  |
| (3) SSP - 4                              | → SSP    |
| (4) Address of the following instruction | → (SSP)  |
| (5) "00100"                              | → ILM    |
| (6) "0"                                  | → S flag |
| (7) (TBR + 3CC <sub>H</sub> )            | → PC     |

When step trace traps are enabled by setting the T flag, NMI for users and user interruption are disabled.

Moreover, EIT by the INTE instruction is not generated.

In the FR family, the trap is generated from the following instruction by which T flag is set.

● Operation of undefined instruction exception

An undefined instruction exception occurs if an undefined instruction is detected during instruction decoding.

[Detection condition of undefined instruction exception]

- (1) It is detected that it is undefined instruction at the decipherment of the instruction.
- (2) Located outside a delay slot. (Not located immediately after a delayed branch instruction.)
- (3) If the above conditions are satisfied, an undefined instruction exception is triggered and execution breaks.

[Operation]

- |                               |          |
|-------------------------------|----------|
| (1) SSP - 4                   | → SSP    |
| (2) PS                        | → (SSP)  |
| (3) SSP - 4                   | → SSP    |
| (4) PC                        | → (SSP)  |
| (5) "0"                       | → S flag |
| (6) (TBR + 3C4 <sub>H</sub> ) | → PC     |

The address saved as the PC is the address of the instruction at which the undefined instruction exception was detected.

● Coprocessor absent trap

When a coprocessor command using an unmounted coprocessor is executed, a coprocessor absent trap will be generated.

[Operation]

- |  |          |
|--|----------|
| (1) SSP - 4                              | → SSP    |
| (2) PS                                   | → (SSP)  |
| (3) SSP - 4                              | → SSP    |
| (4) Address of the following instruction | → (SSP)  |
| (5) "0"                                  | → S flag |
| (6) (TBR + 3E0 <sub>H</sub> )            | → PC     |

### ● Coprocessor error trap

If an error occurs when the coprocessor is used and then the coprocessor instruction that operates the coprocessor is executed, a coprocessor error trap will be generated.

[Operation]

- |  |          |
|--|----------|
| (1) SSP - 4                              | → SSP    |
| (2) PS                                   | → (SSP)  |
| (3) SSP - 4                              | → SSP    |
| (4) Address of the following instruction | → (SSP)  |
| (5) "0"                                  | → S flag |
| (6) (TBR + 3DC <sub>H</sub> )            | → PC     |

### ● Operation of RETI instruction

The RETI instruction is an instruction which returns from EIT processing routine.

[Operation]

- |             |       |
|-------------|-------|
| (1) (R15)   | → PC  |
| (2) R15 + 4 | → R15 |
| (3) (R15)   | → PS  |
| (4) R15 + 4 | → R15 |

The RETI instruction must be executed with the S flag set to "0".

## ■ Note

### ● Delay slot

In the delay slot of the branch instruction, there is a restriction concerning EIT.

Please refer to Section "3.7 Divergence Instructions" for details of the divergence instruction.

## 3.9 Operating Mode

In the operating mode of MB91265A series, there are a bus mode and an access mode. But, single chip mode is supported only.

### ■ Operating Mode

Bus mode

Single chip

#### ● Bus modes

Bus mode indicates the mode that controls the internal ROM operations and external access function operations. Bus mode is specified using the mode set up pins (MD2, MD1, and MD0).

Note: External bus mode is not supported on this model.

### ■ Bus Mode

Please refer to Section "3.2 Memory Map" for details of the bus mode.

#### ● Single chip mode

In this mode, internal I/O, internal RAM, and internal ROM are enabled, and access to all other areas is disabled. The external pins can be used by either the peripheral resources or general-purpose ports. The pin does not work as a bus pin.

Note : External bus mode is not supported on this model.

### ■ Mode Setting

The operating mode is set by the mode pins (MD2, MD1, and MD0) and mode data.

#### ● Mode pin

The MD2, MD1, and MD0 pins specify operation in relation to the mode vector and reset vector fetch.

Settings other than those listed in the table are prohibited.

Mode pins MD2, MD1, MD0	Mode name	Reset vector access area	Remark
0 0 0 <sub>B</sub>	Internal ROM mode vector	Internal	

Note: Settings other than MD2, MD1, MD0: 000<sub>B</sub> are prohibited.

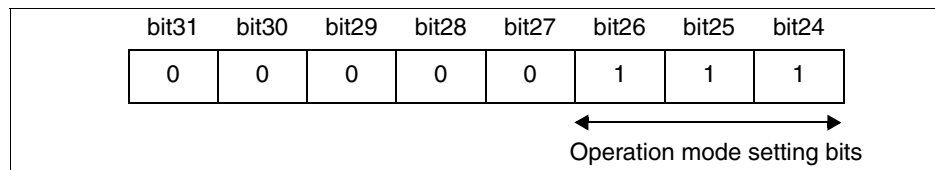
### ● Mode data

The data written to the internal mode register (MODR) by the mode vector fetch (see Section "3.10.3 Reset Sequence" in Section "3.10 Reset (Device Initialization)") is called the mode data.

After the mode register is set, the device operates in accordance with the operation mode set in the register.

The mode data is set by all types of reset. The mode data cannot be set by the user program.

<Detailed explanation of mode data>



[bit31 to bit24] Reserved bits

Always set to "00000111<sub>B</sub>". Operation is not guaranteed if a value other than "00000111<sub>B</sub>" is set.

Note : The mode data set in the mode vector must be located as byte data at 000FFFF8<sub>H</sub>.

As the MB91265A series uses big endian as byte endian, place in the most significant byte (bit31 to bit24) as shown below.

		bit 31	24	23	16	15	8	7	0
Incorrect	000FFFF8 <sub>H</sub>	XXXXXX		XXXXXX		XXXXXX		Mode Data	
Correct	000FFFF8 <sub>H</sub>	Mode Data		XXXXXX		XXXXXX		XXXXXX	
	000FFFC <sub>H</sub>	Reset Vector							

## 3.10 Reset (Device Initialization)

---

**This section describes the reset operation.**

---

### ■ Overview of Reset Operation

When reset factors are generated, the device suspends all programs and hardware operations and initializes the status. This state is called the reset state.

On removal of the reset factor, the device starts the program and hardware operation from its initialized state. The series of operations from the reset state to the start of operations is called the reset sequence.

### 3.10.1 Reset Level

---

The reset operation for this device is divided into two levels, each of which is triggered by different causes and performs different initialization. The following describes each reset level.

---

#### ■ Set Initialization Reset (INIT)

Reset of the strongest level to initialize all settings is called set initialization reset (INIT).

The main content initialized by set initialization reset (INIT) is as follows.

[Initialization part by set initialization reset (INIT)]

- Operation mode of device (setting of bus mode and width of external bus)
- All settings related to internal clock (clock source selection, PLL control, divide ratio setting)
- All settings concerning CS0 region of external bus
- All settings concerning state of other pin
- All parts initialized by operation initialization reset (RST)

Please refer to the explanation of each function for details.

Always use the  $\overline{\text{INIT}}$  pin to trigger a set initialization reset (INIT) after the power is turned on.

#### ■ Operation Initialization Reset (RST)

The normal reset level that initializes program operation is called an operation initialization reset (RST).

When a set initialization reset (INIT) is performed, the operation initialization reset (RST) is performed also.

The main content initialized by operation initialization reset (RST) is as follows.

[Initialization part by operation initialization reset (RST)]

- Program operation
- CPU and internal bus
- Register settings in peripheral circuits
- I/O port setting
- Operation mode of device (setting of bus mode and width of external bus)

Please refer to the explanation of each function for details.

## 3.10.2 Reset Factor

---

**This section describes each reset factor and the associated reset level.**

---

### ■ Reset Factor

Reset factors that were generated in the past can be identified by reading the reset source register (RSRR). (Refer to Section "3.11.6 Block Diagram of Clock Generation Control Unit" and Section "3.11.7 Explanation of Register Details for Clock Generation Control Unit" in the Section "3.11 Clock Generation Control" for details of the registers and flags referred to below.)

### ■ $\overline{\text{INIT}}$ Pin Input (Set Initialization Reset Pin)

The  $\overline{\text{INIT}}$  external pin acts as the set initialization reset pin.

A set initialization reset (INIT) request is generated while a low level input is applied to this pin.

Set initialization reset (INIT) demand is released by inputting the High level to this pin.

When set initialization reset (INIT) is generated at the request of this pin, the bit15: INIT bit within the reset source register (RSRR) will be set.

The set initialization reset (INIT) at the request of this pin is the strongest of all reset factors and will be handled in priority to all other inputs, operations, and statuses.

Always use the  $\overline{\text{INIT}}$  pin to trigger a set initialization reset (INIT) after turning on the power.

Immediately after the power is turned on, maintain low level input to the  $\overline{\text{INIT}}$  pin for the stabilization wait time requested by the oscillation circuit to acquire the oscillation stabilization wait time for the oscillation circuit. (When an INIT is triggered by the  $\overline{\text{INIT}}$  pin, the oscillation stabilization wait time is initialized to its minimum value.)

- Generation factor: Low level input to external  $\overline{\text{INIT}}$  pin
- Release factor: High level input to external  $\overline{\text{INIT}}$  pin
- Generation level: Set initialization reset (INIT)
- Correspondence flag: bit15 : INIT

### ■ STCR: SRST Bit Writing (Software Reset)

When "0" is written to the bit4: SRST bit within the standby control register (STCR), a software reset request will be generated.

Software reset request is operation initialization reset (RST) demand.

When the request is accepted and operation initialization reset (RST) is generated, the software reset request will be cancelled.

When operation initialization reset (RST) is generated by a software reset request, the bit11: SRST bit within the reset source register (RSRR) will be set.

Operation initialization reset (RST) through the software reset request is generated only after all bus accesses are stopped when the bit9: SYNCR bit within the time-base counter control register (TBCR) is set (Synchronous reset mode).

Thus, it may take a long time to generate an operation initialization reset (RST) depending on the usage status of the bus.

- Generation factor: Writing "0" to bit4: SRST bit of the standby control register (STCR).
- Release factor: Generation of operation initialization reset (RST)
- Generation level: Operation initialization reset (RST)
- Correspondence flag: bit11 : SRST

## ■ Watchdog Reset

The watchdog timer will be activated by writing to the watchdog timer control register (RSRR). Unless A5<sub>H</sub>/5A<sub>H</sub> is written to the time-base counter clear register (CTBR) within the cycle specified in bit9 and bit8: WT1 and WT0 bits in the RSRR, a watchdog reset request occurs.

Watchdog reset request is set initialization reset (INIT) demand. After the request is accepted, and when a set initialization reset (INIT) or operation initialization reset (RST) is generated, the watchdog reset request will be cancelled.

When a set initialization reset (INIT) is generated by a watchdog reset request, the bit13: WDOG bit within the reset source register (RSRR) will be set.

When a set initialization reset (INIT) is generated by a watchdog reset request, the setup for the oscillation stabilization wait time will not be initialized. Also, no oscillation stabilization wait time occurs if the main oscillation is not halted in main run or sub-run mode.

- Generation factor: Specified cycle elapsed on the watchdog timer
- Release factor: Generation of a set initialization reset (INIT) or operation initialization reset (RST)
- Generation level: Set initialization reset (INIT)
- Correspondence flag: bit13 : WDOG



### 3.10.3 Reset Sequence

---

The device begins the execution of the reset sequence by the disappearance of the reset factor. The operation of the reset sequence is different depending on the reset level. The content of the operation for the reset sequence at each reset level is explained.

---

#### ■ Set Initialization Reset (INIT) Release Sequence

On release of a set initialization reset (INIT) request, the device performs the following operations in the order.

However, if a watchdog reset occurs when the main oscillation is not halted during main run or sub-run mode, the oscillation stabilization wait time in (2) does not occur.

- (1) Releasing the set initialization reset (INIT) and transition to the oscillation stabilization wait state
- (2) The device remains in the operation initialization reset (RST) state during the oscillation stabilization wait time (set by bit3 and bit2: OS1, OS0 of STCR). Internal clock stopping
- (3) Operation initialization reset (RST) state and beginning of internal clock operation
- (4) Releases the operation initialization reset (RST) and changes to the normal operating state
- (5) Reading of mode vector from address 000FFFF8<sub>H</sub>
- (6) Writes the mode vector to the MODR (mode register)
- (7) Reading of reset vector from address 000FFFC<sub>H</sub>
- (8) Writing of the reset vector in PC (program counter)
- (9) Starting a program from the address contained in the PC (program counter)

#### ■ Operation Initialization Reset (RST) Release Sequence

This reset is triggered by a software reset.

On release of an operation initialization reset (RST) request, the device performs the following operations in the order.

- (1) Releases the operation initialization reset (RST) and changes to the normal operating state
- (2) Reading of mode vector from address 000FFFF8<sub>H</sub>
- (3) Writes the mode vector to the MODR (mode register)
- (4) Reading of reset vector from address 000FFFC<sub>H</sub>
- (5) Writing of the reset vector in PC (program counter)
- (6) Starting a program from the address contained in the PC (program counter)

### 3.10.4 Oscillation Stabilization Wait Time

Automatically transits to oscillation stabilization waiting status when the source oscillation of the device has been suspended or when returning from a status with such possibility. This function prevents the unstable oscillator output that occurs when the oscillation first starts from being used.

During the oscillation stabilization wait time, the internal and external clock provision is suspended, only built-in time-base counter operates, and pauses until the stabilization waiting time set by the standby control register (STCR) has expired.

Hereafter, details of oscillating the steady waiting operation are explained.

#### ■ Triggers for the Oscillation Stabilization Wait

The factor is shown below.

##### (1) When set initialization reset (INIT) is released

The device goes to the oscillation stabilization wait state immediately after a set initialization reset (INIT) is released due to the cause of the reset.

The device goes to the operation initialization reset (RST) state after the oscillation stabilization wait time elapses.

As the oscillation stabilization wait time is set to its minimum value by an  $\overline{\text{INIT}}$  pin initialization, no oscillation stabilization wait occurs. Ensure that the duration of the input to the  $\overline{\text{INIT}}$  pin is long enough to provide the required oscillation stabilization wait time.

##### (2) Recovery from stop mode

The device goes to the oscillation stabilization wait state immediately after stop mode is released.

If cancelled by requesting a set initialization reset (INIT), it transits to the set initialization reset (INIT) status, and then to the oscillation stabilization waiting status after the set initialization reset (INIT) is cancelled.

The device goes to the state corresponding to the factor that stop mode is released after the oscillation stabilization wait time elapses.

- When recovering due to input of a valid external interrupt request (including NMIs) →  
Transits to normal operation state
- When recovering due to a set initialization reset (INIT) request →  
Transits to set initialization reset (INIT) state

##### (3) Recovery from an abnormal state when the PLL is selected

While operations are performed using the PLL as the source clock, if any error (\* refer to the following) arises with the PLL control, it automatically transits to oscillation stabilization wait time to acquire the PLL lock time.

The device goes to the normal operation state after the oscillation stabilization wait time elapses.

\*: If the multiplier ratio is changed or the PLL operation enable bit is modified during PLL operation, etc.

## ■ Select Oscillation Stabilization Wait Time

The oscillation stabilization wait time is timed with built-in time-base counter.

When factors for oscillation stabilization waiting arise and it transits to the oscillation stabilization waiting status, built-in time-base counter begins measurement of the oscillation stabilization wait time after being initialized once.

4 types of oscillation stabilization wait time can be selected and set using the bit3 and bit2: OS1 and OS0 bits of the standby control register (STCR).

The setting that has been once selected will not be initialized by reset other than set initialization reset (INIT) using the external  $\overline{\text{INIT}}$  pin. Other set initialization resets (INIT) and operation initialization resets (RST) leave the existing oscillation stabilization wait time setting before reset.

The four available oscillation stabilization wait time settings are intended for use in the following situations.

- OS1, OS0 = 00<sub>B</sub> : No oscillation stabilization wait time  
(used when the PLL and oscillator do not halt in stop mode)
- OS1, OS0 = 01<sub>B</sub> : PLL lock wait time  
(used with an external clock input or when the oscillator does not halt in stop mode)
- OS1, OS0 = 10<sub>B</sub> : Oscillation stabilization wait time (medium)  
(used with an oscillator that is quick to stabilize such as a ceramic oscillator)
- OS1, OS0 = 11<sub>B</sub> : Oscillation stabilization wait time (long)  
(used with a standard crystal oscillator or similar)

Always use the  $\overline{\text{INIT}}$  pin to trigger a set initialization reset (INIT) after turning on the power.

In the following cases, maintain the low level input to the  $\overline{\text{INIT}}$  pin for the stabilization wait time required by the oscillation circuit to acquire the oscillation stabilization wait time of the oscillation circuit. (When an INIT is triggered by the  $\overline{\text{INIT}}$  pin, the oscillation stabilization wait time is initialized to its minimum value.)

- $\overline{\text{INIT}}$  pin input immediately after power on
- $\overline{\text{INIT}}$  pin input during STOP mode with the oscillation halted
- $\overline{\text{INIT}}$  pin input when the sub clock is selected as the clock source and the main oscillation is halted

Accordingly, input a "L" level to the  $\overline{\text{INIT}}$  pin for a period that satisfies the oscillation stabilization wait time for the main clock to allow the oscillation to stabilize.

### 3.10.5 Reset Operation Mode

---

There are two modes for operation initialization resets (RST), namely, normal (asynchronous) reset mode and synchronous reset mode, and which operation mode is to be used is set by the bit9: SYNCR bit of the time-base counter control register (TBCR).

This mode setting is initialized only by set initialization reset (INIT).

Set initialization reset (INIT) always does the reset action asynchronously.

Hereafter, each mode operation is explained.

---

#### ■ Normal Reset Operation

The operation whereby the device goes to the operation initialization reset (RST) state immediately after an operation initialization reset (RST) request occurs is called normal reset operation.

When a reset (RST) request is received in this mode, the device goes to the reset (RST) state immediately regardless of the current status of internal bus access.

Results of the bus access performed at the time of transition to each status cannot be guaranteed under this mode. However, the operation initialization reset (RST) request can be accepted reliably.

It will be normal reset mode when the bit9: SYNCR bit within the time-base counter control register (TBCR) is "0".

The initial value after a set initialization reset (INIT) is normal reset mode.

#### ■ Synchronous Reset Operation

The operation whereby the device goes to the operation initialization reset (RST) state only once all bus access halts after an operation initialization reset (RST) request occurs is called synchronous reset operation.

In this mode, the device does not change to the reset (RST) state while internal bus access is in progress even though a reset (RST) request may be present.

When the above request is accepted, a sleep request is issued to the internal bus. The device goes to the operation initialization reset (RST) state once each bus halts operation and goes to the sleep state.

As all bus accesses are stopped when transiting to each status under this mode, the results of all bus accesses can be guaranteed.

However, if bus access does not stop for some reason, no requests can be accepted during that time. However, even in this case, a set initialization reset (INIT) still occurs immediately.

The following cases can cause bus access not to halt.

- When ready requests (RDY) are continually input into the external extension bus interface, and bus wait is valid.

Note: Transfer of the DMA controller will be stopped on receiving each request, so transition to each status does not need to be delayed.

This will be the synchronous reset mode when the bit9: SYNCR bit within the time-base counter control register (TBCR) is "1".

The initial value after a set initialization reset (INIT) is to return to normal reset mode.

Note the restrictions that apply to bit 9: SYNCR in the TBCR (time-base counter control register) when using the synchronous mode software reset.

## 3.11 Clock Generation Control

---

**This section explains the clock generation control.**

---

### ■ Overview of Clock Generation Control

The internal operation clocks are generated as follows.

- Base clock generation: The base clock is generated from the source clock divided by two or by using the PLL oscillation.
- Generation of each internal clock: The base clock is divided to generate the operation clocks supplied to each block.

Hereafter, each clock generation and the control are explained.

Refer to Sections "3.11.6 Block Diagram of Clock Generation Control Unit" and "3.11.7 Explanation of Register Details for Clock Generation Control Unit" for details of the registers and flags referred to in the following sections.

### 3.11.1 Selection of the Source Clock

---

**This section describes how the source clock is selected.**

---

#### ■ Selection of the Source Clock

The source clock is the source oscillation generated in the internal oscillation circuit by connecting an oscillator to the X0 and X1 external oscillator pins.

All clock sources including the external bus clock are supplied from within the device.

The external oscillator pins and internal oscillation circuit can be switched at any time while the main clock is in operation.

- Main clock: Generated from the X0 and X1 pin inputs and intended for use as the high-speed clock.

The internal base clock can be selectively generated from the following source clocks.

- Main clock divided by two
- Main clock multiplied using the PLL

$\phi$  means the source clock divided-by-two or the base clock used to drive the PLL. Accordingly, the system base clock is the clock generated from the above internal base clock. Selection of the source clock is controlled by the clock source control register (CLKR) setting.

### 3.11.2 PLL Control

---

**Operation (oscillation) enable and disable and the multiplier ratio setting can be set for the PLL oscillation circuit for the main clock.**

**Each control is done by setting clock source control register (CLKR).**

**Hereafter, the content of each control is explained.**

---

#### ■ PLL Enabling Operation

The value of bit10: PLL1EN bit of the clock source control register (CLKR) enables or halts the main PLL oscillation.

The PLL1EN bit is initialized to "0" after a set initialization reset (INIT) to halt the main PLL oscillation. The output of the main PLL cannot be selected as the source clock while it is halted.

Once program operation has started, set the multiplier ratio of the main PLL to use as the clock source, enable operation, and then wait for the PLL lock wait time to elapse before switching the source clock. Using the time-base timer interrupt to time the PLL lock wait is recommended.

The PLL cannot be halted while the main PLL output is selected as the source clock.

Writing to the register is ignored. When you wish to stop the PLL such as when changing to stop mode, select the main clock divided by two as the source clock first before halting the PLL.

As the PLL stops automatically on changing to stop mode if bit0: OSCD1 bit of the standby control register (STCR) is set to stop the oscillation during stop mode, it does not need to be stopped beforehand. Afterwards, when returning from the stop mode, PLL automatically begins the oscillation operation. The PLL does not stop automatically if the oscillation is set to continue during stop mode. In this case, stop the operation before changing to stop mode if necessary.

#### ■ PLL Multiplication Rate

The multiplication rate for the main PLL is set up by the bit14 to bit12: PLL1S2, PLL1S1, and PLL1S0 bits of the clock source control register (CLKR).

All bits are initialized to "0" after a set initialization reset (INIT).

After program operation starts when changing the PLL multiplier ratio to a value different to its initial setting, always make the change before or at the same time as enabling PLL operation. After changing the multiplier ratio, wait for the lock wait time before switching the source clock. Using the time-base timer interrupt to time the PLL lock wait is recommended.

If you want to change the PLL multiplier ratio during operation, first change the source clock to something other than the PLL. After changing the multiplier ratio, wait for the lock wait time before switching the source clock, as in the case above.

The PLL multiplication rate setting can be changed while the PLL is in use, but in this case, it automatically transits into oscillation stabilization waiting status after the multiplication rate setting is rewritten, and program operation stops until the oscillation stabilization wait time that has been set has expired. The program operation does not stop when the clock source is switched besides PLL.

### 3.11.3 Oscillation Stabilization Wait and PLL Lock Wait Time

If the operation of the clock selected as the source clock is not stable, an oscillation stabilization wait time is required. (See Section "3.10.4 Oscillation Stabilization Wait Time".)

A wait time while the PLL locks is required after the PLL starts operating to allow the output to stabilize at the specified frequency.

#### ■ Wait Time after Power Supply is Turned on

An oscillation stabilization wait time for the main clock oscillation circuit is required first after the power is turned on.

Setting for oscillation stabilization wait time is initialized to the minimum value through input from the  $\overline{\text{INIT}}$  pin (set initialization reset pin), so this oscillation stabilization wait time will be acquired from the time for inputting the low level to the  $\overline{\text{INIT}}$  pin input.

As the PLL is still not enabled in this state, the lock wait time does not need to be considered in this case.

#### ■ Wait Time after Setting is Initialized

When a set initialization reset (INIT) is released, the device goes to the oscillation stabilization wait state. Here, the set oscillation stabilization wait time is internally generated.

As the setting time is initialized to its minimum value for the initial oscillation stabilization wait state after the  $\overline{\text{INIT}}$  pin input ends, this state ends quickly and the device changes to the operation initialization reset (RST) state. Also, no oscillation stabilization wait time occurs after a watchdog reset if the main oscillation is not halted in main run and sub-run modes.

Under such statuses, no operation of any PLL is enabled, so the lock wait time does not need to be considered at this stage.

#### ■ Wait Time after Enabling PLL Operation

If you intend to enable the PLL from the halted state after program operation starts, the output of the PLL cannot be used until the lock wait time has elapsed.

If main PLL is not selected as the source clock, program execution can continue while waiting for the PLL to lock. Using the time-base timer interrupt to time the PLL lock wait is recommended.

#### ■ Wait Time after Changing the PLL Multiplier Ratio

After program operation starts if you want to change the multiplier ratio for the PLL while it is running, the output of the PLL cannot be used until the lock wait time has elapsed.

If main PLL is not selected as the source clock, program execution can continue while waiting for the PLL to lock. Using the time-base timer interrupt to time the PLL lock wait is recommended.



### ■ Wait Time after Recovering from Stop Mode

After program operation starts, the oscillation stabilization wait time set by the program is generated internally after recovering from stop mode.

If the device is set to halt the oscillation circuit for the clock selected as the source clock during stop mode, the longer of the oscillation stabilization wait time for the oscillation circuit and the lock wait time for the PLL must be used as the wait time. Always set the longer of the two oscillation stabilization wait times before changing to stop mode.

If the device is set not to halt the oscillation circuit for the clock selected as the source clock during stop mode, the PLL is not halted automatically. Accordingly, no oscillation stabilization wait time is required unless you halt the PLL.

It is recommended that you set the minimum value of the oscillation stabilization wait time before changing to stop mode.

### 3.11.4 Clock Distribution

---

The operation clocks for each function are generated from the base clock which is generated from the source clock respectively.

There are a total of three different internal operation clocks, CPU clock, peripheral clock, and external bus clock, and the divide ratio can be set independently for each clock.

The each internal operation clock is explained as follows.

---

#### ■ CPU Clock (CLKB)

It is a clock used for CPU, an internal memory, and an internal bus.

The circuit which uses this clock is as follows.

- CPU
- Internal RAM, internal ROM
- Bit search module
- I-bus, D-bus, F-bus, X-bus
- DMA controller
- On chip Debug Support Unit (DSU)

Do not set a combination of multiplier ratio and divide ratio that results in the upper-limit frequency being exceeded.

#### ■ Clock in Surrounding (CLKP)

This is the clock used by the peripheral resources and the peripheral bus.

The circuit which uses this clock is as follows.

- Peripheral (surrounding) bus
- Clock controller (bus interface part only)
- Interrupt controller
- I/O port
- Peripheral resources such as the external interrupt inputs, UART, and 16-bit timer.

Do not set a combination of multiplier ratio and divide ratio that results in the upper-limit frequency being exceeded.

#### ■ External Bus Clock (CLKT)

It is a clock used for the external extended bus interface.

The circuit which uses this clock is as follows.

- External expansion bus interface
- External clock output

Do not set a combination of multiplier ratio and divide ratio that results in the upper-limit frequency being exceeded.

---

Notes:

- External bus mode is not supported on this model.
  - The processing performance of CPU is influenced from the setting of flash memory wait register (FLWC). Adjust the setting of this register to the best value and use it. Refer to Section "19.2.2 Flash Memory Wait Register (FLWC)".
-

### 3.11.5 Clock Divider

---

**The base clock divide ratio can be set independently for each internal operation clock. The best operation frequency for each circuit can be set by this function.**

---

#### ■ Clock Divider

The division rate is set up using basic clock dividing frequency set registers 0 (DIVR0) and 1 (DIVR1). There are 4 setting bits that support each clock in each register, and (register set up value + 1) will be the division rate for the base clock of that clock. The duty ratio is always 50% even if an odd-numbered divide ratio is set.

If the setting is modified, the new divide ratio applies from the next rising edge on the clock signal.

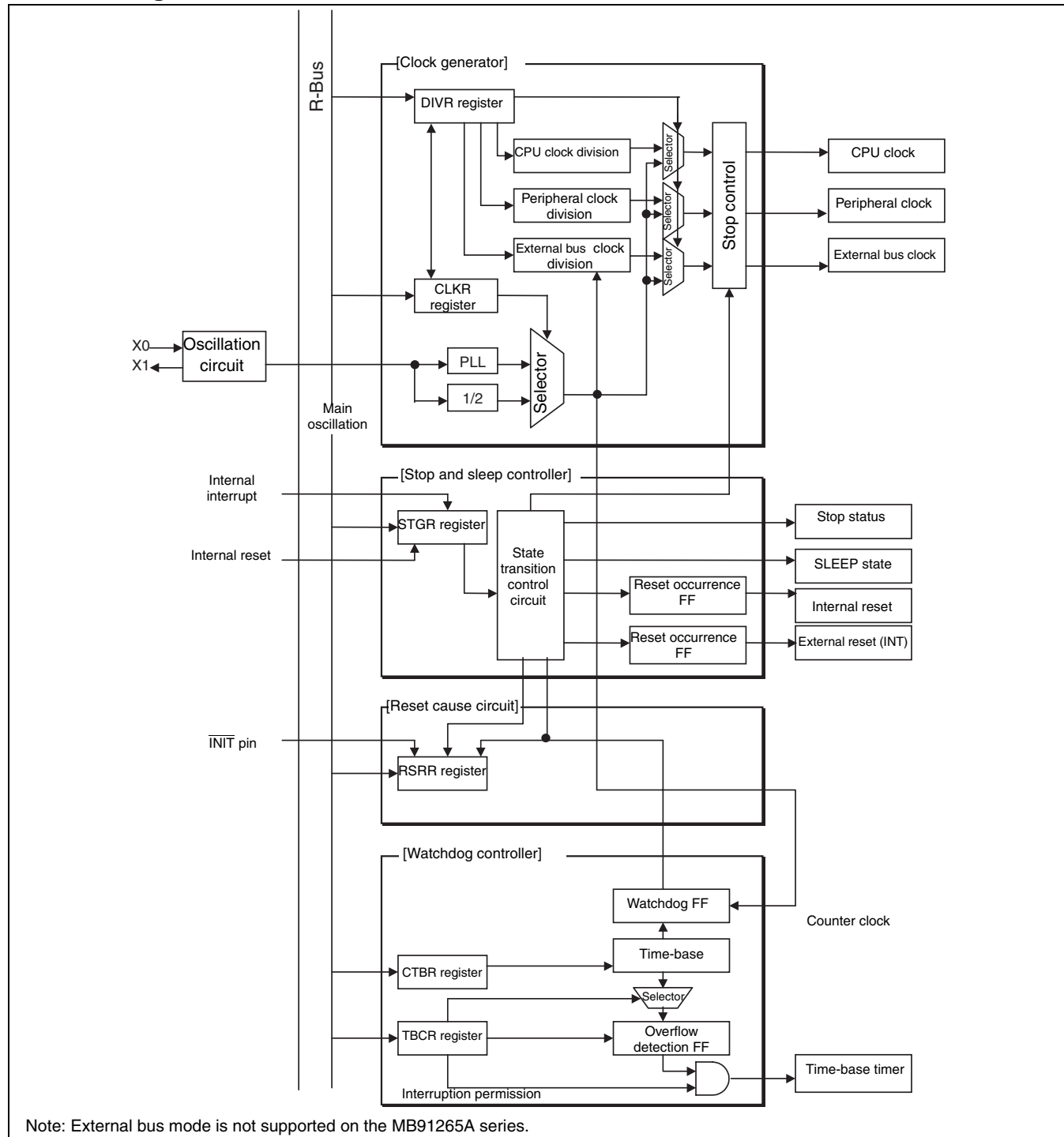
The divide ratio setting is not initialized by an operation initialization reset (RST) and the setting prior to the reset remains. The setting is only initialized by a set initialization reset (INIT). If changing the source clock from its initial setting to a higher speed, always set the divide ratio beforehand.

Operation is not guaranteed if the combination of the source clock selection, main PLL multiplier ratio setting, and divide ratio setting results in the upper-limit frequency being exceeded. Great care must be taken (in particular not to adopt the wrong order with modification settings for the source clock selection).

### 3.11.6 Block Diagram of Clock Generation Control Unit

The block diagram of clock generation control unit is shown as follows. Refer to Section "3.11.7 Explanation of Register Details for Clock Generation Control Unit" for detailed explanations of the register within the figure.

#### ■ Block Diagram of Clock Generation Control Unit



### 3.11.7 Explanation of Register Details for Clock Generation Control Unit

This section describes the register of the clock generation control unit.

#### ■ RSRR: Reset Source Register/Watchdog Timer Control Register

	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
Address: 000480 <sub>H</sub>	INIT	-	WDOG	-	SRST	-	WT1	WT0
	R	R	R	R	R	R	R/W	R/W
Initial value ( $\overline{\text{INIT}}$ pin)	1	0	0	0	0	0	0	0
Initial value (INIT)	*	*	*	×	×	*	0	0
Initial value (RST)	×	×	×	*	*	×	0	0

\*: Vary depending on the source.  
 ×: Not initialized

This register retains reset factors that were generated just beforehand, performs cycle setting and initiation control of the watchdog timer.

After reading, the maintained reset factor is cleared when this register is read. If a number of resets are generated before reading, the reset factor flags accumulate, and a number of the flags will be set.

Writing to this register starts the watchdog timer. The watchdog timer keeps working until reset (RST) is generated after that.

[bit15] INIT (INITialize reset occurred)

Indicates whether a reset triggered by the  $\overline{\text{INIT}}$  pin input (INIT) has occurred.

0	No INIT has occurred due to an $\overline{\text{INIT}}$ pin input.
1	INIT has occurred due to an $\overline{\text{INIT}}$ pin input.

- Initialized to "0" after a read.
- Read-only. Writing has no effect on the bit values.

[bit14] (reserved bit)

[bit13] WDOG (WatchDOG reset occurred)

Indicates whether a reset triggered by the watchdog timer (INIT) has occurred.

0	No INIT has occurred due to the watchdog timer.
1	INIT has occurred due to the watchdog timer.

- Initialized to "0" after a read and after a reset triggered by the  $\overline{\text{INIT}}$  pin input (INIT).
- Read-only. Writing has no effect on the bit values.

[bit12] (reserved bit)

**[bit11] SRST (Software ReSeT occurred)**

Indicates whether reset (RST) by writing the SRST bit (software reset) of the STCR register is generated or not.

Note the restrictions that apply to bit9:SYNCR in the TBCR (time-base counter control register) when using the synchronous mode software reset.

0	No RST has occurred due to a software reset.
1	RST has occurred due to a software reset.

- Initialized to "0" after a read and after a reset triggered by the  $\overline{\text{INIT}}$  pin input (INIT).
- Read-only. Writing has no effect on the bit values.

**[bit10] (reserved bit)****[bit9, bit8] WT1, WT0 (Watchdog interval Time select)**

Sets the period of the watchdog timer.

The period of the watchdog timer is selected from the following four settings based on the value written to these bits.

WT1	WT0	Minimum period for writing to CTBR to prevent a watchdog reset from occurring	Time from writing last 5A <sub>H</sub> to CTBR until a watchdog reset occurs
0	0	$\phi \times 2^{16}$ (Initial value)	$\phi \times 2^{16}$ to $\phi \times 2^{17}$
0	1	$\phi \times 2^{18}$	$\phi \times 2^{18}$ to $\phi \times 2^{19}$
1	0	$\phi \times 2^{20}$	$\phi \times 2^{20}$ to $\phi \times 2^{21}$
1	1	$\phi \times 2^{22}$	$\phi \times 2^{22}$ to $\phi \times 2^{23}$

( $\phi$  is the period of the internal base clock.)

- Initialized to "00<sub>B</sub>" by reset (RST).
- Read is enabled while write is valid only once after reset (RST), thereafter write will be invalid.

## ■ STCR: Standby Control Register

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
Address: 000481 <sub>H</sub>	STOP	SLEEP	HIZ	SRST	OS1	OS0	-	OSCD1
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value ( $\overline{\text{INIT}}$ pin)	0	0	1	1	0	0	1	1
Initial value (INIT)	0	0	1	1	×	×	1	1
Initial value (RST)	0	0	×	1	×	×	×	×
R/W: Readable/Writable								
X: Not initialized								

It is a register which controls the operation mode of the device.

Transits to two standby modes, namely stop and sleep, controls the pins under the stop mode and carries out oscillation stop control, sets up oscillation stabilization wait time, and issues software resets.

Note the restrictions that apply to bit9:SYNCR in the TBCR (time-base counter control register) when using the synchronous mode software reset.

### Note:

To place the device in standby mode, use the synchronous standby mode (set with bit8: SYNCS bit of the time-base counter control register (TBCR)) and be sure to use the following sequence:

/\* Writing STCR \*/

```
ldi    #_STCR, R0          ; STCR register (0481H)
ldi    #Val_of_Stby, r1    ; Val_of_Stby is the write data to STCR.
stb    r1, @r0             ; Writing to STCR
```

/\* Writing STBR \*/

```
ldi    #_CTBR, r2          ; CTBR register (0483H)
ldi    #0xA5, r1           ; Clear command (1)
stb    r1, @r2             ; Writing A5 to CTBR
ldi    #0xA5, r1           ; Clear command (2)
stb    r1, @r2             ; Writing A5 to CTBR
```

/\* Clearing the time base counter in here \*/

```
ldub   @r0, r1             ; Reading STCR
```

/\* Starting synchronous standby transition \*/

```
ldub   @r0, r1             ; Reading dummy STCR
nop                               ; NOP × 5 for timing adjustment
nop
nop
nop
nop
```

The following describes the functions of each bit in the standby control register (STCR).

**[bit7] STOP (STOP mode)**

Changes the device to stop mode. When 1 is written to both bit6: SLEEP bit and this bit, this bit is given priority and the device transits to the stop mode.

0	Does not change to stop mode (Initial value).
1	Changes to stop mode.

- Initialized to "0" by a reset (RST) and by stop recovery factor.
- Read and write are possible.

**[bit6] SLEEP (SLEEP mode)**

The transition to sleep mode is directed. When 1 is written to both bit7: STOP bit and this bit, bit7: STOP bit has precedence, and the device transits to the stop mode.

0	Does not change to sleep mode (Initial value).
1	Changes to sleep mode.

- Initialized to "0" by a reset (RST) and by sleep recovery factor.
- Read and write are possible.

**[bit5] HIZ (HIZ mode)**

The state of the pin at the stop mode is controlled.

0	The state of the pin before shifting to the stop mode is maintained.
1	Pin output is put into the state of high impedance in the stop mode (Initial value).

- Initialized to "0" by reset (INIT).
- Read and write are possible.

**[bit4] SRST (Software ReSeT)**

Invokes a software reset (RST).

0	Generates a software reset.
1	Does not generate a software reset (Initial value).

- Initialized to 1 by reset (RST).
- Read and write are possible. The value read is always "1".



**[bit3, bit2] OS1, OS0 (Oscillation Stabilization time select)**

Sets the oscillation stabilization wait time to use after a reset (INIT) or recovery from stop mode.

The length of the oscillation stabilization wait time is selected from the following four settings based on the value written to these bits.

OS1	OS0	Oscillation stabilization wait time	When main oscillation 4MHz
0	0	$\phi \times 2^1$ (Initial value)	1.0 $\mu$ s
0	1	$\phi \times 2^{11}$	1.0 ms
1	0	$\phi \times 2^{16}$	32.7 ms
1	1	$\phi \times 2^{22}$	2.0 s

( $\phi$  is the period of the internal base clock and is twice the period of the main oscillation.)

- Initialized to "00<sub>B</sub>" by a reset triggered by the  $\overline{\text{INIT}}$  pin input (INIT).
- Read and write are possible.

**[bit1] (reserved bit)**

Reserved bit. Always write "1" to this bit on this model.

**[bit0] OSCD1 (Oscillation Disable mode for XIN1)**

Controls whether the oscillation halts for the main oscillation input (X0 and X1) during stop mode.

0	The main oscillation does not halt during stop mode.
1	The main oscillation halts during stop mode (Initial value).

- Initialized to "1" by reset (INIT).
- Read and write are possible.

## ■ TBCR: Time-base Counter Control Register

	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
Address: 000482 <sub>H</sub>	TBIF	TBIE	TBC2	TBC1	TBC0	-	SYNCR	SYNCS
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value (INIT)	0	0	×	×	×	×	×	×
Initial value (RST)	0	0	×	×	×	×	×	×
R/W: Readable/Writable								
X: Not initialized								

It is a register which controls the time-base timer interruption etc.

Enables time-base timer interruption, selects interruption interval time, and also sets up option functions for reset operations.

The following describes the functions of each bit in the time-base counter control register (TBCR).

### [bit15] TBIF (Time-Base timer Interrupt Flag)

The interrupt flag for the time-base timer.

Indicates that the time-base counter has expired the set interval time (bit13 to bit11: TBC2 to TBC0 bits).

While interruption generation is enabled (TBIE=1) by the bit14: TBIE bit, when this bit is "1", a time-base timer interruption request is generated.

Clear factor	Writing of 0 by instruction
Set factor	After specified interval time has elapsed (detection of a falling edge on the output of the time-base counter)

- Initialized to 0 by reset (RST).
- Read and write are possible. However, only 0 can be written to this bit, and writing 1 does not change the bit value.

Moreover, the read value in the read modification write system instruction always becomes 1.

### [bit14] TBIE (Time-Base timer Interrupt Enable)

It is a time-base timer interruption demand output permission bit.

Controls output of interrupt requests when the interval time set for the time-base counter elapses. While this bit is "1", bit15: TBIF bit will be "1", and a time-base timer interruption request is generated.

0	Disable output of time-base timer interrupt requests. (Initial value)
1	Enable output of time-base timer interrupt requests.

- Initialized to "0" by reset (RST).
- Read and write are possible.

**[bit13 to bit11] TBC2,TBC1,TBC0 (Time-Base timer Counting time select)**

The interval time of the time-base counter used with the time-base timer is set.

The interval time is selected from the following eight settings based on the value written to these bits.

TBC2	TBC1	TBC0	Timer interval time	For a 4MHz source oscillation and a x8 PLL multiplier
0	0	0	$\phi \times 2^{11}$	64 [ $\mu$ s]
0	0	1	$\phi \times 2^{12}$	128 [ $\mu$ s]
0	1	0	$\phi \times 2^{13}$	256 [ $\mu$ s]
0	1	1	$\phi \times 2^{22}$	131 [ms]
1	0	0	$\phi \times 2^{23}$	262 [ms]
1	0	1	$\phi \times 2^{24}$	524 [ms]
1	1	0	$\phi \times 2^{25}$	1048 [ms]
1	1	1	$\phi \times 2^{26}$	2097 [ms]

( $\phi$  is the period of the internal base clock and is the output period of the main PLL.)

- The initial value is irregular. Be sure to set a value before enabling the interrupt.
- Read and write are possible.

**[bit10] (reserved bit)**

Reserved bit. The value when read is undefined. Writing has no effect on the operation.

**[bit9] SYNCR (SYNChronous Reset enable)**

It is a synchronous reset operation permission bit.

This bit specifies whether normal reset operation or synchronous reset operation is executed when an operation initialization reset (RST) request or hardware standby request occurs. Normal reset operation performs a reset (RST) immediately or the transition to hardware standby. Synchronous reset operation performs an operation initialization reset (RST) after all bus accesses have stopped or the transition to hardware standby.

0	Normal reset operation (Initial value)
1	Synchronous reset operation

- Initialized to "0" by reset (INIT).
- Read and write are possible.

**Note:**

When using the synchronous mode software reset, the following two conditions must be satisfied before setting the SRST bit in the STCR (standby control register) to "0".

- Set interrupt enable flag (I-Flag) to interrupts disabled (I-Flag=0)
- NMI not used

**[bit8] SYNCS (SYNChronous Standby enable)**

It is a synchronous standby operation permission bit.

Always set to "1" when using standby modes (sleep or stop mode).

0	Normal standby operation (Initial value)
1	Synchronous standby operation

- Initialized to "0" by reset (INIT).
- Read and write are possible.

**■ CTBR: Time-base Counter Clear Register**

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
Address: 000483 <sub>H</sub>	D7	D6	D5	D4	D3	D2	D1	D0
	W	W	W	W	W	W	W	W
Initial value (INIT)	×	×	×	×	×	×	×	×
Initial value (RST)	×	×	×	×	×	×	×	×
W: Write only								
X: Not initialized								

It is a register to initialize the time-base counter.

Successively writing {A5<sub>H</sub>} and then {5A<sub>H</sub>} to this register clears all bits of the time-base counter to "0" immediately after the {5A<sub>H</sub>} value is written. Although there is no limit on the length of time between writing {A5<sub>H</sub>} and {5A<sub>H</sub>}, if you write a value other than {5A<sub>H</sub>} after writing {A5<sub>H</sub>}, the counter is not cleared the next time you write {5A<sub>H</sub>} unless you first write {A5<sub>H</sub>} again.

As this counter is cleared automatically when the CPU is not operating, such as during DMA transfer or stop or sleep mode, the watchdog reset is automatically delayed in these cases.

The reading value of this register is irregular.

**Note:**

When this register is used to clear the time-base counter, the oscillation stabilization wait time, watchdog timer period, and time-base timer period change temporarily.

## ■ CLKR: Clock Source Control Register

	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
Address: 00000484 <sub>H</sub>	-	PLL1S2	PLL1S1	PLL1S0	-	PLL1EN	CLKS1	CLKS0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value (INIT)	0	0	0	0	0	0	0	0
Initial value (RST)	×	×	×	×	×	×	×	×
R/W: Readable/Writable								
X: Not initialized								

This register is used to select the clock source to use for the internal base clock and to control the main PLL.

This register selects the clock source. The register is also used to enable the main PLL and set the multiplier ratio.

[bit15] (reserved bit)

Reserved bit. Always write "0" to this bit on this model.

[bit14 to bit12] PLL1S2, PLL1S1, PLL1S0 (PLL1 ratio Select 2 to 0)

The main PLL multiplier ratio selection bits.

The main PLL multiplier ratio can be selected from the following combinations.

Modifying these bits while the main PLL is selected as the clock source is prohibited.

Do not specify a setting that will result in the upper-limit frequency for the device being exceeded.

PLL1S2	PLL1S1	PLL1S0	Main PLL multiply-by rate	When main oscillation 4MHz
0	0	0	× 1 (equal)	$\phi = 250$ ns (at 4 MHz)
0	0	1	× 2 (2 multiplication)	$\phi = 125$ ns (at 8 MHz)
0	1	0	× 3 (3 multiplication)	$\phi = 83.33$ ns (at 12 MHz)
0	1	1	× 4 (4 multiplication)	$\phi = 62.50$ ns (at 16 MHz)
1	0	0	× 5 (5 multiplication)	$\phi = 50.00$ ns (at 20 MHz)
1	0	1	× 6 (6 multiplication)	$\phi = 41.67$ ns (at 24 MHz)
1	1	0	× 7 (7 multiplication)	$\phi = 35.71$ ns (at 28 MHz)
1	1	1	× 8 (8 multiplication)	$\phi = 31.25$ ns (at 32 MHz)

( $\phi$  is the period of the internal base clock.)

- Initialized to "000<sub>B</sub>" by reset (INIT).
- Read and write are possible.

[bit11] (reserved bit)

Reserved bit. Always write "0" to this bit on this model.

**[bit10] PLL1EN (PLL1 ENable)**

The operation enable bit for the main PLL.

Modifying this bit while the main PLL is selected as the clock source is prohibited.

Selecting the main PLL as the clock source while this bit is "0" is prohibited.

See the bit9 and bit8: CLKS1 and CLKS0 bits settings.

If bit0: OSCD1 of STCR is "1", the main PLL stops during stop mode even if these bits are "1".

After the device returns from the stop mode, the main PLL is enabled again.

0	Main PLL stopped (Initial value)
1	Main PLL enabled

- Initialized to "0" by reset (INIT).
- Read and write are possible.

**[bit9, bit8] CLKS1,CLKS0 (CLocK source Select)**

Sets the clock source to use.

The clock source is selected from the following three settings based on the value written to these bits.

CLKS1	CLKS0	Clock source setting
0	0	Source oscillation input from X0/X1 divided by two (Initial value)
0	1	Setting disabled
1	0	Main PLL
1	1	Setting disabled

- Initialized to "00<sub>B</sub>" by reset (INIT).
- Read and write are possible.

**Note:**

Changing the value of bit8: CLKS0 when bit9: CLKS1 is "1" is prohibited.

[Combinations able to be modified]
"00 <sub>B</sub> " → "10 <sub>B</sub> "
"10 <sub>B</sub> " → "00 <sub>B</sub> "

- Setting other than one of the above combinations is prohibited.

## ■ DIVR0: Basic Clock Dividing Frequency Set Register 0

	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
Address: 00000486 <sub>H</sub>	B3	B2	B1	B0	P3	P2	P1	P0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value (INIT)	0	0	0	0	0	0	1	1
Initial value (RST)	×	×	×	×	×	×	×	×
R/W: Readable/Writable								
X: Not initialized								

This register controls the ratios for dividing the base clock to generate each internal clock.

Under this register, the division rate between the CPU and the clock (CLKB) of the internal bus, and peripheral circuit and peripheral bus clock (CLKP) will be set.

Operation is not guaranteed if the combination of the source clock selection, main PLL multiplier ratio setting, and divide ratio setting results in the upper-limit frequency being exceeded. Please take great care with this point. Also take care not to make a mistake in the sequence when changing the source clock selection.

When settings for this register are modified, after the set up, the division rate after modification from the next clock rate will be valid.

[bit15 to bit12] B3,B2,B1,B0 (clkB divide select 3 to 0)

These are a clock dividing frequency ratio set bits of the CPU clock (CLKB).

Sets the clock divide ratio for the clock used for the CPU, internal memory, and internal bus (CLKB).

The data written to these bits selects the division rate of the clock to the base clock (clock frequency) for the CPU and internal bus from the 16 types shown in the following table.

Do not set a divide ratio that will result in the upper-limit frequency for the device being exceeded.

B3	B2	B1	B0	Clock division ratio	Clock frequency: For a 4MHz source oscillation and a x8 main PLL multiplier
0	0	0	0	$\phi$	32 MHz (Initial value)
0	0	0	1	$\phi \times 2$ (divided by 2)	16 MHz
0	0	1	0	$\phi \times 3$ (divided by 3)	10.7 MHz
0	0	1	1	$\phi \times 4$ (divided by 4)	8 MHz
0	1	0	0	$\phi \times 5$ (divided by 5)	6.4 MHz
0	1	0	1	$\phi \times 6$ (divided by 6)	5.33 MHz
0	1	1	0	$\phi \times 7$ (divided by 7)	4.57 MHz
0	1	1	1	$\phi \times 8$ (divided by 8)	4 MHz
...	...	...	...	...	...
1	1	1	1	$\phi \times 16$ (divided by 16)	2 MHz

( $\phi$  is the period of the internal base clock.)

- Initialized to "0000<sub>B</sub>" by reset (INIT).
- Read and write are possible.

[bit11 to bit8] P3,P2,P1,P0 (clkP divide select 3 to 0)

These are a clock dividing frequency ratio set bits of the clock in the surrounding (CLKP) .

Sets the clock divide ratio for the clock used for the peripheral circuits and peripheral bus (CLKP).

The value written to these bits selects the division rate of the clock to the base clock (clock frequency) for peripheral circuits and peripheral buses from the 16 types shown in the following table.

Do not set a divide ratio that will result in the upper-limit frequency for the device being exceeded.

P3	P2	P1	P0	Clock division ratio	Clock frequency: For a 4MHz source oscillation and a $\times 8$ main PLL multiplier
0	0	0	0	$\phi$	32 MHz
0	0	0	1	$\phi \times 2$ (divided by 2)	16 MHz
0	0	1	0	$\phi \times 3$ (divided by 3)	10.7 MHz
0	0	1	1	$\phi \times 4$ (divided by 4)	8 MHz (Initial value)
0	1	0	0	$\phi \times 5$ (divided by 5)	6.4 MHz
0	1	0	1	$\phi \times 6$ (divided by 6)	5.33 MHz
0	1	1	0	$\phi \times 7$ (divided by 7)	4.57 MHz
0	1	1	1	$\phi \times 8$ (divided by 8)	4 MHz
...	...	...	...	...	...
1	1	1	1	$\phi \times 16$ (divided by 16)	2 MHz

( $\phi$  is the period of the system base clock.)

- Initialized to "0011<sub>B</sub>" by reset (INIT).
- Read and write are possible.

### ■ DIVR1:Basic Clock Dividing Frequency Set Register 1

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
Address: 000487 <sub>H</sub>	T3	T2	T1	T0	-	-	-	-
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial value (INIT)	0	0	0	0	0	0	0	0
Initial value (RST)	×	×	×	×	×	×	×	×
R/W: Readable/Writable								
X: Not initialized								

This register controls the ratios for dividing the base clock to generate each internal clock.

This register sets the divide ratio for the clock used by the external bus interface (CLKT).

Operation is not guaranteed if the combination of the source clock selection, main PLL multiplier ratio setting, and divide ratio setting results in the upper-limit frequency being exceeded. Please take great care with this point. Also take care not to make a mistake in the sequence when changing the source clock selection.

When settings for this register are modified, after the set up, the division rate after modification from the next clock rate will be valid.



[bit7 to bit4] T3,T2,T1,T0 (clkT divide select 3 to 0)

These are a clock dividing frequency ratio set bits of the external bus clock (CLKT).

Sets the clock divide ratio for the clock used by the external bus interface (CLKT).

The value written to these bits selects the division rate of the clock to the base clock (clock frequency) for external extension bus interfaces from the 16 types shown in the following table.

Do not set a divide ratio that will result in the upper-limit frequency for the device being exceeded.

T3	T2	T1	T0	Clock division ratio	Clock frequency: For a 4MHz source oscillation and a $\times 8$ main PLL multiplier
0	0	0	0	$\phi$	32 MHz (Initial value)
0	0	0	1	$\phi \times 2$ (divided by 2)	16 MHz
0	0	1	0	$\phi \times 3$ (divided by 3)	10.7 MHz
0	0	1	1	$\phi \times 4$ (divided by 4)	8 MHz
0	1	0	0	$\phi \times 5$ (divided by 5)	6.4 MHz
0	1	0	1	$\phi \times 6$ (divided by 6)	5.33 MHz
0	1	1	0	$\phi \times 7$ (divided by 7)	4.57 MHz
0	1	1	1	$\phi \times 8$ (divided by 8)	4 MHz
...	...	...	...	...	...
1	1	1	1	$\phi \times 16$ (divided by 16)	2 MHz

( $\phi$  is the period of the system base clock.)

- Initialized to "0000<sub>B</sub>" by reset (INIT).
- Read and write are possible.

Setting "1111<sub>B</sub>" (divided by 16) is recommended if not using the external bus interface.

[bit3 to bit0] (reserved bit)

- Initialized to "0000<sub>B</sub>" by reset (INIT).
- Always write "0000<sub>B</sub>" to these bits.

Note: External bus mode is not supported on this model.

### 3.11.8 Peripheral Circuit Functions in the Clock Controller

---

**This section describes the peripheral circuit functions in the clock controller.**

---

#### ■ Time-base Counter

The clock controller includes a 26-bit time-base counter which runs on the internal base clock.

In addition to measuring the oscillation stabilization wait time (see Section "3.10.4 Oscillation Stabilization Wait Time"), the time-base counter is used for the following purposes.

- Watchdog timer

The watchdog timer is used to detect system runaway and measures the bit output of the time-base counter.

- Time-base timer

Uses the output of the time-base counter to generate interval interrupts.

Hereafter, these functions are explained.

#### ● Watchdog timer

The watchdog timer uses the output of the time-base counter to detect program runaway.

If postponement of the watchdog reset is not generated between the intervals that have been set, due to a program runaway or such like, the set initialization reset (INIT) request is generated as a watchdog reset.

[Starting the watchdog timer and setting the period]

The watchdog timer is activated by writing to the 1st reset source register/watchdog timer control register (RSRR) after reset (RST). In this case, the interval for the watchdog timer is set by the bit9 and bit8: WT1 and WT0 bits. For the interval setting, only the time that has been set through this first writing will be valid, and all other writings after that will be ignored.

[Postponing generation of the watchdog reset]

Once the watchdog timer has started, the program must periodically write {A5<sub>H</sub>} and {5A<sub>H</sub>} (in that order) to the time-base counter clear register (CTBR). The flag for the watchdog reset generation is initialized by this operation.

[Generation of a watchdog reset]

The flag for generating watchdog resets is set by the falling edge of the time-base counter output for the interval that has been set. If the flag is set when the 2nd falling edge is detected, a set initialization reset (INIT) request is generated as the watchdog reset.

[Stopping watchdog timer]

Once the watchdog timer is activated, the watchdog timer cannot be stopped until an operation initialization reset (RST) is generated.

Under the following status in which an operation initialization reset (RST) is generated, the watchdog timer is stopped and does not function until activated by a re-program operation.

- State of operation initialization reset (RST)
- State of set initialization reset (INIT)
- State of oscillation stabilization wait reset (RST)

[Temporarily halting the watchdog timer (automatically postpone generation)]

The watchdog timer initializes the flag to once generate a watchdog reset when the program operation of the CPU is stopped and postpones generation of a watchdog reset. The stop of the program operation concretely shows the following operations.

- Sleep state
- Stop state
- Oscillation stabilization wait RUN state
- During DMA transfer to the D-bus (data bus)
- During a break when using the emulator-debugger

When the time-base counter is cleared, the flag for generating watchdog resets is simultaneously initialized, and generation of a watchdog reset will be postponed.

### ● Time-base timer

The time-base timer uses the output of the time-base counter to generate interval interrupts.

The timer is suitable for measuring relatively long times, up to  $\{\text{base clock} \times 2^{27}\}$  cycles such as for the main PLL lock wait time and the oscillation stabilization wait time for the sub clock.

The time-base timer interruption request is generated when the falling edge of the output for the time-base counter that supports the set interval is detected.

[Setting of time-base timer at startup and intervals]

The time-base timer sets the interval using the bit13, bit12, bit11: TBC2, TBC1, and TBC0 bits of the time-base counter control register (TBCR).

As the falling edge of the output for the time-base counter that supports the set interval is always detected, after the interval is set, firstly clear the bit15: TBIF bit, and then enable interruption request output by setting "1" as the bit14: TBIE bit.

Before changing the interval time, disable the interrupt request output by setting "0" as the bit14: TBIE bit.

As the time-base counter always counts without being influenced by these settings, clear the time-base counter before enabling interruption in order to get accurate interval interruption times. If this is not done, an interrupt request may be generated immediately after interrupts are enabled.

[Clearness of time-base counter by program]

Writing  $\{A5_H\}$  and then  $\{5A_H\}$  to the time-base counter clear register (CTBR) clears all bits of the time-base counter to "0" immediately after writing  $\{5A_H\}$ . Although there is no limit on the length of time between writing  $\{A5_H\}$  and  $\{5A_H\}$ , if you write a value other than  $\{5A_H\}$  after writing  $\{A5_H\}$ , the counter is not cleared the next time you write  $\{5A_H\}$  unless you first write  $\{A5_H\}$  again.

The flag for generating watchdog resets is simultaneously initialized when this time-base counter is cleared, and generation of watchdog resets will be postponed.

[Clearness of time-base counter by state of device]

The time-base counter is cleared all bits to "0" simultaneously when the device goes to the following states.

- Stop state
- State of set initialization reset (INIT)
- Hardware standby state

In particular, under the stop status, the time-base counter is used to measure oscillation stabilization wait times, so an interval interruption of the time-base timer may be generated unintentionally. Thus, before setting the stop mode, disable time-base timer interruption, and try not to use the time-base timer.

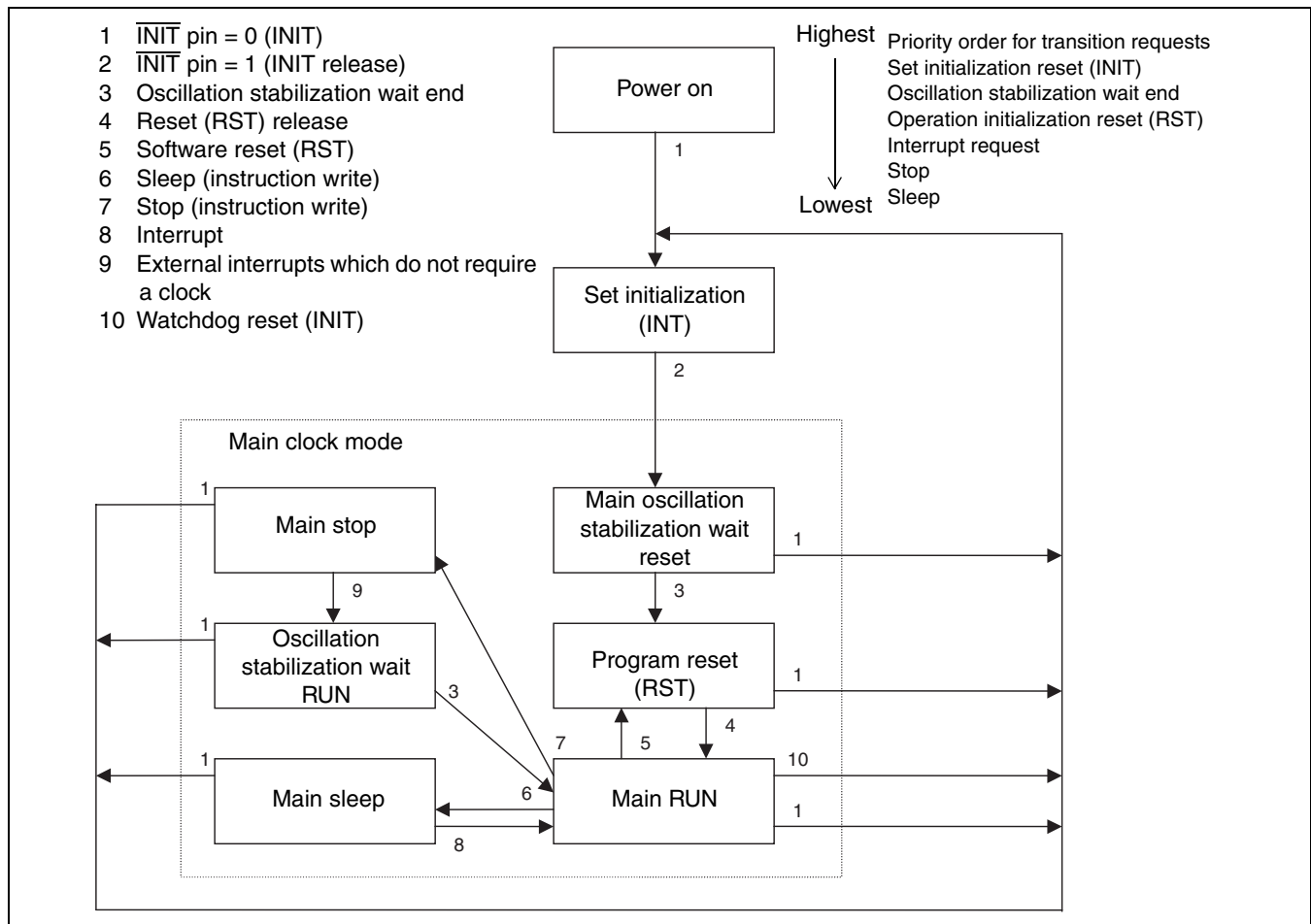
For statuses other than that, time-base timer interruption is automatically disabled as operation initialization reset (RST) is generated.

## 3.12 Device State Control

This section describes the various device states and how these are controlled.

### ■ State of Device and Each Transition

The figure below shows the state transitions.



The following has the device operation states.

#### ● State of RUN (normal operation)

This is the program execution state.

All internal clocks are supplied, and all the circuits are operable.

Only the bus clock is stopped for the 16-bit peripheral bus while access is not carried out.

Each status transition request is accepted, but when synchronous reset mode is selected, the status transition operation for normal reset mode cases and for some requests is different. See "■ Synchronous Reset Operation" in Section "3.10.5 Reset Operation Mode" for details.

### ● Sleep state

The program halts in this state. The device is set to this state by program operation.

Only the program execution of CPU stops, and the circuit in the surrounding is operable. All internal memory and internal and external buses halt unless requested by the DMA controller.

The device recovers from this state when a valid interrupt request occurs and goes to the RUN state (normal operation).

Transits to the set initialization reset (INIT) status by generating a set initialization reset (INIT) request.

An operation initialization reset (RST) request causes the device to go to the operation initialization reset (RST) state.

### ● Stop state

It is a stopped state of the device. The device is set to this state by program operation.

All internal circuits stop. All internal clocks halt. The oscillation circuit and main PLL can also be set to halt in this state.

Moreover, an external pin can be made uniform high impedance by setting. (A part of pin is excluded)

The device changes to the oscillation stabilization wait RUN state when certain interrupt requests occur (interrupt requests able to be generated while the clock is halted).

Transits to the set initialization reset (INIT) status by generating a set initialization reset (INIT) request.

The device changes to the oscillation stabilization wait reset (RST) state when an operation initialization reset (RST) request occurs.

### ● Oscillation stabilization wait RUN state

It is a stopped state of the device. The device changes to this state after recovering from stop mode.

All internal circuits except the clock generation control unit (time-base counter and device status control unit) are stopped. All internal clocks halt, but the oscillation circuit and the main PLL (if enabled) continue to operate.

The high impedance control of an external pin in the state of the stop, etc. is released.

The device goes to the RUN state (normal operation) after the specified oscillation stabilization wait time elapses.

Transits to the set initialization reset (INIT) status by generating a set initialization reset (INIT) request.

The device changes to the oscillation stabilization wait reset (RST) state when an operation initialization reset (RST) request occurs.

### ● Oscillation stabilization wait reset (RST) state

It is a stopped state of the device. The device goes to this state after recovering from stop mode or a set initialization reset (INIT) status.

All internal circuits except the clock generation control unit (time-base counter and device status control unit) are stopped. All internal clocks halt, but the oscillation circuit and the main PLL (if enabled) continue to operate.

The high impedance control of an external pin in the state of the stop, etc. is released.

Operation initialization reset (RST) is output to an internal circuit.

The device goes to the oscillation stabilization wait reset (RST) state after the specified oscillation stabilization wait time elapses.

Transits to the set initialization reset (INIT) status by generating a set initialization reset (INIT) request.

### ● State of operation initialization reset (RST)

The program is being initialized. Transits by receiving the operation initialization reset (RST) request or ending the oscillation stabilization wait reset (RST) status.

The program execution of CPU stops, and the program counter is initialized. The circuit in the surrounding is initialized excluding part. All internal clocks, the oscillation circuit, and the main PLL (if enabled) operate.

Operation initialization reset (RST) is output to an internal circuit.

Transits to the RUN status (normal operation) by diminishing the operation initialization reset (RST) request, and operation initialization reset sequence is executed. After recovering from the set initialization reset (INIT) state, the set initialization reset sequence is executed.

Transits to the set initialization reset (INIT) status by generating a set initialization reset (INIT) request.

### ● State of set initialization reset (INIT)

All settings are being initialized. The device changes to this state on receiving a set initialization reset (INIT) request.

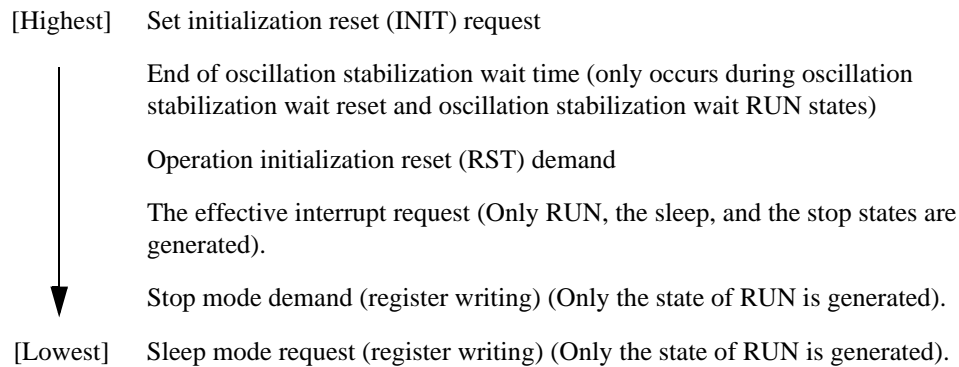
The program execution of CPU stops, and the program counter is initialized. All the circuits in the surrounding are initialized. The oscillation circuit operates, but the main PLL is halted. All internal clocks are halted while the external  $\overline{\text{INIT}}$  pin remains at the low level, but continue to operate at other times.

Outputs a set initialization reset (INIT) and operation initialization reset (RST) to internal circuits.

This status is cancelled by diminishing the set initialization reset (INIT) request, and transits to the oscillation stabilization wait reset (RST) status. Next, the device goes to the operation initialization reset (RST) state and the set initialization reset sequence is executed.

### ● Priority level of each state transition demand

In any state, each state transition demand conforms to the following priority levels. However, as certain requests only occur in specific states, these are only meaningful in those states.



## ■ Low-power Consumption Mode

The following describes the low-power consumption modes available on this device and how to use them. This device has the following low-power consumption modes.

- Sleep mode  
The device is set to sleep mode by writing to a register.
- Stop mode  
The device is set to stop mode by writing to a register.

The following describes each mode.

### ● Sleep mode

Writing "1" to bit6: SLEEP bit of standby control register (STCR) sets sleep mode and changes the device to the sleep state. The device remains in the sleep state until something happens that causes the device to recover from the sleep state.

See also "● Sleep state" of "■ State of Device and Each Transition" in the Section "3.12 Device State Control" for details about sleep state.

#### [Transition to sleep mode]

If synchronous standby mode (set using bit8: SYNCS bit of the time-base counter control register (TBCR)) is used when setting the device to sleep mode, always use the following sequence.

/\* Writing STCR \*/

```
ldi      #_STCR, R0          ; STCR register (0481H)
ldi      #_Val_of_Stby, r1    ; Val_of_Stby is the write data to STCR.
stb      r1, @r0             ; Writing to STCR
```

/\* Writing STBR \*/

```
ldi      #_CTBR, r2          ; CTBR register (0483H)
ldi      #0xA5, r1           ; Clear command (1)
stb      r1, @r2             ; Writing A5 to CTBR
ldi      #0xA5, r1           ; Clear command (2)
stb      r1, @r2             ; Writing A5 to CTBR
```

/\* Clearing the time base counter in here \*/

```
ldub     @r0, r1             ; Reading STCR
```

/\* Starting synchronous standby transition \*/

```
ldub     @r0, r1             ; Reading dummy STCR
nop                                     ; NOP × 5 for timing adjustment
nop
nop
nop
nop
```

If "1" is written to both this bit and bit7: STOP bit of the standby control register (STCR), bit7: STOP bit has precedence and the device goes to the stop state.

#### [Circuits that halt during sleep state]

- Program execution by the CPU



The following circuits operate if DMA transfer occurs.

- Bit search module
- Various internal memory
- Internal and external buses

Note: External bus mode is not supported on this model.

[Circuits that do not halt during sleep state]

- Oscillation circuit
- Main PLL (if enabled)
- Clock generation control unit
- Interrupt controller
- Peripheral circuit
- Watch timer
- Main oscillation stabilization wait timer
- DMA controller
- On chip Debug Support Unit (DSU)

[Events that recover the device from sleep state]

- Generation of a valid interrupt request

The device recovers from sleep mode when an interrupt request occurs that is not disabled interrupt (1111<sub>B</sub>) in the ICR register and changes to the RUN state (normal operation). In this case, you should set the I flag in the CPU's PS register to "1" to enable interrupt acceptance and cause the interrupt handler to be executed on recovering from sleep mode.

The device does not recover from sleep mode if an interrupt request occurs that is disabled interrupt (1111<sub>B</sub>) in the ICR register.

- Generation of a set initialization reset (INIT) request

The device always goes to the set initialization reset (INIT) state unconditionally when a set initialization reset (INIT) request occurs.

- Generation of an operation initialization reset (RST) request

The device always goes to the operation initialization reset (RST) state unconditionally when an operation initialization reset (RST) request occurs.

Note: See "● Priority level of each state transition demand" of "■ State of Device and Each Transition" in the Section "3.12 Device State Control" for details of the priority order for the different types of trigger.

[Synchronous standby operation]

Synchronous standby operation is enabled if bit8: SYNCS bit of the time-base counter control register (TBCR) is set to "1". Transition to the sleep state is not caused only by a write to the SLEEP bit. Then, transition to the sleep state occurs by reading the STCR register.

To enter the sleep mode, be sure to use the sequence in [Transition to sleep mode].

## ● Stop mode

Writing "1" to bit7: STOP bit of the standby control register (STCR) sets stop mode and changes the device to the stop state. The device remains in the stop state until something happens that causes the device to recover from the stop state.

See also "● Stop state" of "■ State of Device and Each Transition" in the Section "3.12 Device State Control" for details about the stop state.

### [Transition to stop mode]

If synchronous standby mode (set using bit8: SYNCs bit of the time-base counter control register (TBCR)) is used when setting the device to stop mode, always use the following sequence.

/\* Writing STCR \*/

```
ldi    #_STCR, R0          ; STCR register (0481H)
ldi    #Val_of_Stby, r1     ; Val_of_Stby is the write data to STCR.
stb    r1, @r0             ; Writing to STCR
```

/\* Writing CTBR \*/

```
ldi    #_CTBR, r2          ; CTBR register (0483H)
ldi    #0xA5, r1           ; Clear command (1)
stb    r1, @r2             ; Writing A5 to CTBR
ldi    #0xA5, r1           ; Clear command (2)
stb    r1, @r2             ; Writing A5 to CTBR
```

/\* Clearing the time base counter in here \*/

```
ldub    @r0, r1            ; Reading STCR
```

/\* Starting synchronous standby transition \*/

```
ldub    @r0, r1            ; Reading dummy STCR
nop                                           ; NOP × 5 for timing adjustment
nop
nop
nop
nop
```

If "1" is written to both this bit and bit6: SLEEP bit of the standby control register (STCR), bit7: STOP bit has precedence and the device goes to the stop state.

### [Circuits that halt during stop state]

All circuits halt except the following.

### [Circuits that do not halt during stop state]

- Oscillation circuits that are not set to halt

The oscillation circuit for the main clock does not halt during the stop state if bit0: OSCD1 bit in standby control register (STCR) is set to "0".

- Main PLL if enabled and if connected to an oscillation circuit that is not set to halt

The PLL for the main clock does not halt during the stop state if bit0: OSCD1 bit in standby control register (STCR) is set to "0" and bit10: PLL1EN bit of the clock source control register (CLKR) is set to "1".

### [Pin high-impedance control during the stop state]

Pin outputs go to the high-impedance state during the stop state if bit5: HIZ bit of the standby control register (STCR) is set to "1". The pins to which this control applies are listed in "APPENDIX C Pin States in Each CPU State".

If bit5: HIZ bit of the standby control register (STCR) is set to "0" during the stop state, pin outputs maintain the values they had prior to the device changing to the stop state. For details, refer to "APPENDIX C Pin Status in Each CPU State".

### [Events that recover the device from stop mode]

- Generation of certain valid interrupt requests (clock is not required).

Only some external interrupt and the  $\overline{\text{NMI}}$  input pin are valid.

The device recovers from stop mode when an interrupt request occurs that is not disabled interrupt (1111<sub>B</sub>) in the ICR register and changes to the oscillation stabilization wait RUN state. In this case, you should set the I flag in the CPU's PS register to "1" to enable acceptance of interrupts and cause the interrupt handler to be executed on recovering from stop mode.

The device does not recover from stop mode if an interrupt request occurs that is disabled interrupt (1111<sub>B</sub>) in the ICR register.

- Generation of a set initialization reset (INIT) request

The device always goes to the set initialization reset (INIT) state unconditionally when a set initialization reset (INIT) request occurs.

- Generation of an operation initialization reset (RST) request

The device always goes to the operation initialization reset (RST) state unconditionally when an operation initialization reset (RST) request occurs.

Note: See "● Priority level of each state transition demand" of "■ State of Device and Each Transition" in the Section "3.12 Device State Control" for details of the priority order for the different types of trigger.

### [Clock source selection in stop mode]

In self-generated oscillation mode, always set the clock source to the main clock divided by two before setting stop mode. See the "Clock Generation Control", particularly the "3. PLL Control" section for details.

The same restrictions as in normal operation apply to the setting of the divide ratio.

### [Synchronous standby operation]

Synchronous standby operation is enabled if bit8 (SYNCS bit) of the time-base counter control register (TBCR) is set to "1". In this case, simply writing to the STOP bit will not change the device to the stop state. Then, transition to the stop state occurs by reading the STCR register.

When using stop mode, always use the sequence described in [Transition to stop mode].

# **CHAPTER 4**

---

## ***I/O PORT***

**This chapter explains the overview of the I/O ports and the configuration and functions of registers.**

- 4.1 Basic Block Diagram of the I/O Port
- 4.2 Registers of the I/O Port
- 4.3 Analog Input Port

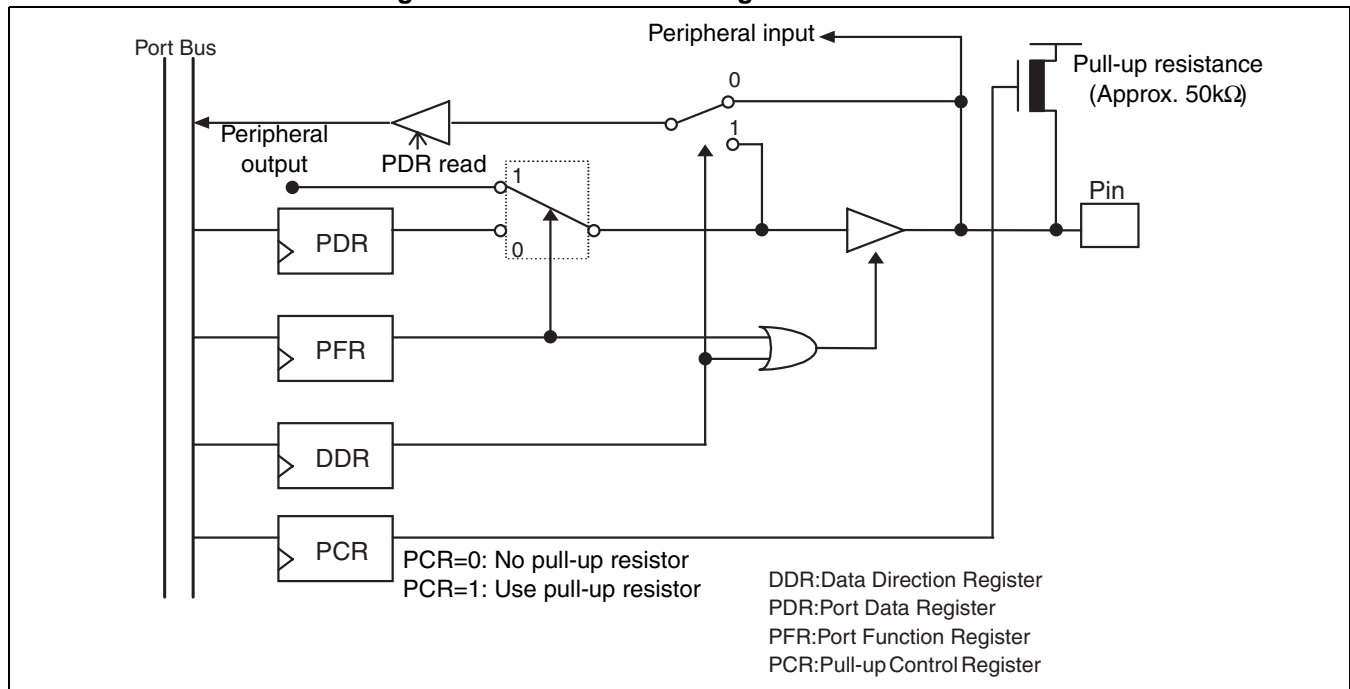
## 4.1 Basic Block Diagram of the I/O Port

MB91265A series is able to use all pins that are not used for I/O by the associated peripheral or the external bus interface as I/O ports.

### ■ Basic Block Diagram of the I/O Port

The figure below shows the basic configuration of a port.

Figure 4.1-1 Basic Block Diagram of the I/O Port



## 4.1.1 I/O Port with Pull-up Resistor

---

The I/O ports with pull-up resistors consist of the following:

- Port data register (PDR)
  - Port direction register (DDR)
  - Port function register (PFR)
  - Pull-up control register (PCR)
- 

### ■ I/O Port Mode

- Port input mode (PFR = 0 and DDR = 0)

PDR read: Reads the levels of the associated external pins.

PDR write: Writes a setting value to the PDR.

- Port output mode (PFR = 0 and DDR = 1)

PDR read: Reads the PDR value.

PDR write: The PDR values are output to the associated external pins.

- Peripheral output mode (PFR = 1 and DDR = x)

PDR read: Reads the value of the associated peripheral output.

PDR write: Writes a setting value to the PDR.

---

Notes:

- Use byte access to access the ports.
  - The setting in the pull-up control register has precedence during stop mode (HIZ = 0).
  - The setting in the pull-up control register is ignored during stop mode (HIZ = 1).
  - No registers are provided for selecting between general-purpose port inputs and peripheral inputs.  
The value input from the external pin is always passed to both the general-purpose port and peripheral circuit.  
Similarly, when a port is set as an output by the DDR, the output value is always passed to both the general-purpose port and peripheral circuit.  
If using a port as a peripheral input, set as an input in the DDR and enable the corresponding peripheral input signal.
-

## 4.2 Registers of the I/O Port

This section explains the registers of the I/O port.

### ■ Port Data Registers (PDR: PDR0 to PDR5, PDRG)

#### PDR0

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00000000 <sub>H</sub>	P07	P06	P05	P04	P03	P02	P01	P00	XXXXXXXX <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

#### PDR1

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00000001 <sub>H</sub>	P17	P16	P15	P14	P13	P12	P11	P10	XXXXXXXX <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

#### PDR2

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00000002 <sub>H</sub>	P27	P26	P25	P24	P23	P22	P21	P20	XXXXXXXX <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

#### PDR3

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00000003 <sub>H</sub>	P37	P36	P35	P34	P33	P32	P31	P30	XXXXXXXX <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

#### PDR4

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00000004 <sub>H</sub>	–	P46	P45	P44	P43	P42	P41	P40	XXXXXXXX <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

#### PDR5

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00000005 <sub>H</sub>	P57	P56	P55	P54	P53	P52	P51	P50	XXXXXXXX <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

#### PDRG

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00000010 <sub>H</sub>	–	–	–	–	–	–	PG1	–	-----X <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable

PDR0 to PDR5, PDRG registers are the I/O data registers for the I/O ports. The corresponding DDR0 to DDR5, DDRG, PFR0 to PFR5, and PFRG registers control input and output.

## ■ Data Direction Registers (DDR: DDR0 to DDR5, DD RG)

### DDR0

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00000400 <sub>H</sub>	P07	P06	P05	P04	P03	P02	P01	P00	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

### DDR1

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00000401 <sub>H</sub>	P17	P16	P15	P14	P13	P12	P11	P10	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

### DDR2

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00000402 <sub>H</sub>	P27	P26	P25	P24	P23	P22	P21	P20	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

### DDR3

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00000403 <sub>H</sub>	P37	P36	P35	P34	P33	P32	P31	P30	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

### DDR4

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00000404 <sub>H</sub>	–	P46	P45	P44	P43	P42	P41	P40	-0000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

### DDR5

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00000405 <sub>H</sub>	P57	P56	P55	P54	P53	P52	P51	P50	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

### DD RG

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00000410 <sub>H</sub>	–	–	–	–	–	–	PG1	–	-----0 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable

The DDR0 to DDR5, DD RG registers control the input/output direction of the corresponding I/O port independently for each bit.

At PFR=0      DDR=0: Port input

DDR=1: Port output

At PFR=1      DDR=0: Peripheral input

DDR=1: Peripheral output



## ■ Pull-up Control Registers (PCR: PCR0 to PCR4, PCRG)

### PCR0

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00000600 <sub>H</sub>	P07	P06	P05	P04	P03	P02	P01	P00	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

### PCR1

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00000601 <sub>H</sub>	P17	P16	P15	P14	P13	P12	P11	P10	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

### PCR2

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00000602 <sub>H</sub>	P27	P26	P25	P24	P23	P22	P21	P20	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

### PCR3

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00000603 <sub>H</sub>	P37	P36	—	—	—	—	—	—	00----- <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

### PCR4

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00000604 <sub>H</sub>	—	—	—	—	P43	P42	P41	P40	----0000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

### PCRG

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00000610 <sub>H</sub>	—	—	—	—	—	—	PG1	—	-----0- <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable

PCR0 to PCR4 and PCRG registers control the pull-up resistors for the corresponding I/O ports.

PCR=0: No pull-up resistor

PCR=1: Use pull-up resistor

## ■ Port Function Registers (PFR: PFR0, PFR1, PTFR0)

### PFR0

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00000420 <sub>H</sub>	TOT2E	TOT1E	—	—	—	—	—	—	00----- <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

### PFR1

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00000421 <sub>H</sub>	TX0E*	—	SCK1E	—	SOT1	SCK0E	—	SOT0E	0-0-00-0 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

### PTFR0

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00000433 <sub>H</sub>	PPG7E	PPG6E	PPG5E	PPG4E	PPG3E	PPG2E	PPG1E	PPG0E	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable

The PFR0, PFR1, PTFR0 registers control the corresponding peripheral outputs independently for each bit. Always write "0" to the unused bits in the PFR registers.

\*: TX0E (TX0 output enable bit of C-CAN) of bit7 in the PFR1 register is set only in MB91F267NA/MB91267NA. For other model, the bit is always set to "0".

### Note:

For PPG6 of Port 17 and TX0 output in MB91F267NA/MB91267NA, PPG6 output has precedence when PPG6E = 1 and TX0E = 1.

The following table lists the initial value and function of each PFR register.

Register Name	Bit	Bit Name	Set value	Function
PFR0	7	TOT2E	0	General-purpose port [Initial value]
			1	Reload timer 2 output
	6	TOT1E	0	General-purpose port [Initial value]
			1	Reload timer 1 output
PFR1	7	TX0E*	0	General-purpose port [Initial value]
			1	C-CAN data output
	5	SCK1E	0	General-purpose port [Initial value]
			1	UART1 clock output
	3	SOT1	0	General-purpose port [Initial value]
			1	UART1 data output
	2	SCK0E	0	General-purpose port [Initial value]
			1	UART0 clock output
	0	SOT0E	0	General-purpose port [Initial value]
			1	UART0 data output

\*: TX0E (TX0 output enable bit of C-CAN) of bit7 in the PFR1 register is set in MB91F267NA/MB91267NA.

For other model, the bit is always set to "0".

Register Name	Bit	Bit Name	Set value	Function
PTFR0	7	PPG7E	0	General-purpose port [Initial value]
			1	PPG timer 7 output
	6	PPG6E	0	General-purpose port [Initial value]
			1	PPG timer 6 output
	5	PPG5E	0	General-purpose port [Initial value]
			1	PPG timer 5 output
	4	PPG4E	0	General-purpose port [Initial value]
			1	PPG timer 4 output
	3	PPG3E	0	General-purpose port [Initial value]
			1	PPG timer 3 output
	2	PPG2E	0	General-purpose port [Initial value]
			1	PPG timer 2 output
	1	PPG1E	0	General-purpose port [Initial value]
			1	PPG timer 1 output
	0	PPG0E	0	General-purpose port [Initial value]
			1	PPG timer 0 output

---

**Note:**

For PPG6 of P17 and TX0 output in MB91F267NA/MB91267NA, PPG6 output has precedence when PPG6E =1 and TX0E =1.

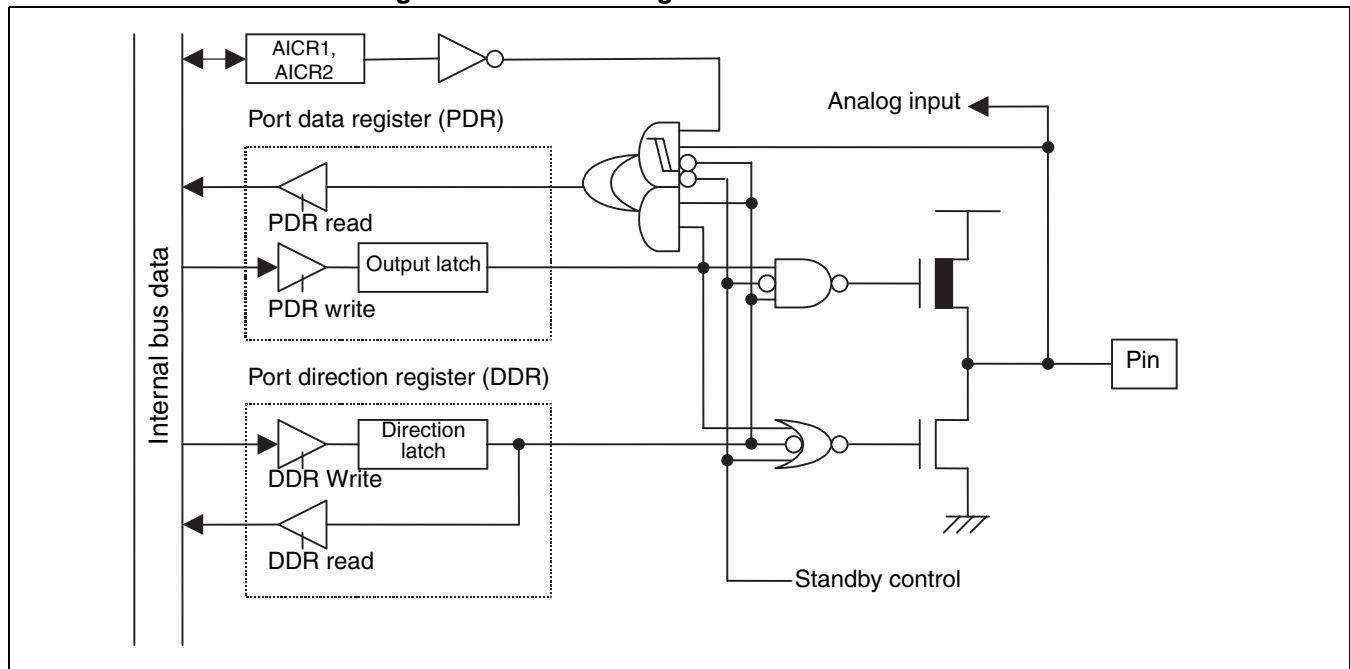
---

## 4.3 Analog Input Port

This section describes the block diagram of the A/D converter and the register in use.

### ■ Block Diagram of 8/10-bit A/D Converter Pin

Figure 4.3-1 Block Diagram of AN0 to AN10 Pins



#### Notes:

- Set the DDR register bits corresponding to the pins to use as input ports to "0" and apply pull-up resistance to the external pins. Also set the bits corresponding to the AICR register to "0".
- Set the AICR register bit corresponding to the pin to be used as the analog input pin to "1". In this case, the value read from the PDR register is the value of the PDR register.

## ■ Analog Input Control Registers (AICR: AICR1, AICR2)

### AICR1

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00000086 <sub>H</sub>	—	—	—	—	AN3E	AN2E	AN1E	AN0E	----0000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

### AICR2

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0000008E <sub>H</sub>	—	AN10E	AN9E	AN8E	AN7E	AN6E	AN5E	AN4E	-0000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable

AICR register controls each I/O port pin as follows.

AICR=0: Port input mode

AICR=1: Analog input mode

This bit becomes "0" after a reset.



# **CHAPTER 5**

---

# ***INTERRUPT CONTROLLER***

**This chapter explains the overview of the interrupt controller, the configuration and functions of registers, and interrupt controller operation.**

- 5.1 Overview
- 5.2 Interrupt Controller Registers
- 5.3 Block Diagram
- 5.4 Register Details Explanation
- 5.5 Operation of the Interrupt Controller



## 5.1 Overview

---

**The interrupt controller manages interrupt reception and arbitration processing.**

---

### ■ Hardware Configuration of the Interrupt Controller

Interrupt controller consists of the following register and circuit.

- ICR register
- Interrupt priority judgment circuit
- Interrupt level and interrupt number (vector) generation unit
- Unit for generating hold request cancel requests

### ■ Major Functions

The main functions of this module are as follows.

- Detection of NMI requests and interrupt requests
- Judgement of priorities (based on interrupt level and number)
- Pass the interrupt level for the interrupt selected by judgement result (to the CPU)
- Pass the interrupt number for the interrupt selected by judgement result (to the CPU)
- Send a stop mode recovery notification (to the CPU) if an NMI or interrupt with a level other than "1111<sub>B</sub>" occurs.
- Generates the hold request cancel request to the DMAC.

## 5.2 Interrupt Controller Registers

Figure 5.2-1 shows the interrupt controller registers.

### ■ Interrupt Controller Registers

Figure 5.2-1 Interrupt Controller Registers

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	
000440 <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR00
000441 <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR01
000442 <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR02
000443 <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR03
000444 <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR04
000445 <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR05
000446 <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR06
000447 <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR07
000448 <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR08
000449 <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR09
00044A <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR10
00044B <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR11
00044C <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR12
00044D <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR13
00044E <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR14
00044F <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR15
000450 <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR16
000451 <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR17
000452 <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR18
000453 <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR19
000454 <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR20
000455 <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR21
000456 <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR22
000457 <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR23
000458 <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR24
000459 <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR25
00045A <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR26
00045B <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR27
00045C <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR28
00045D <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR29
00045E <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR30
00045F <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR31
				R	R/W	R/W	R/W	R/W	

(Continued)

## CHAPTER 5 INTERRUPT CONTROLLER

(Continued)

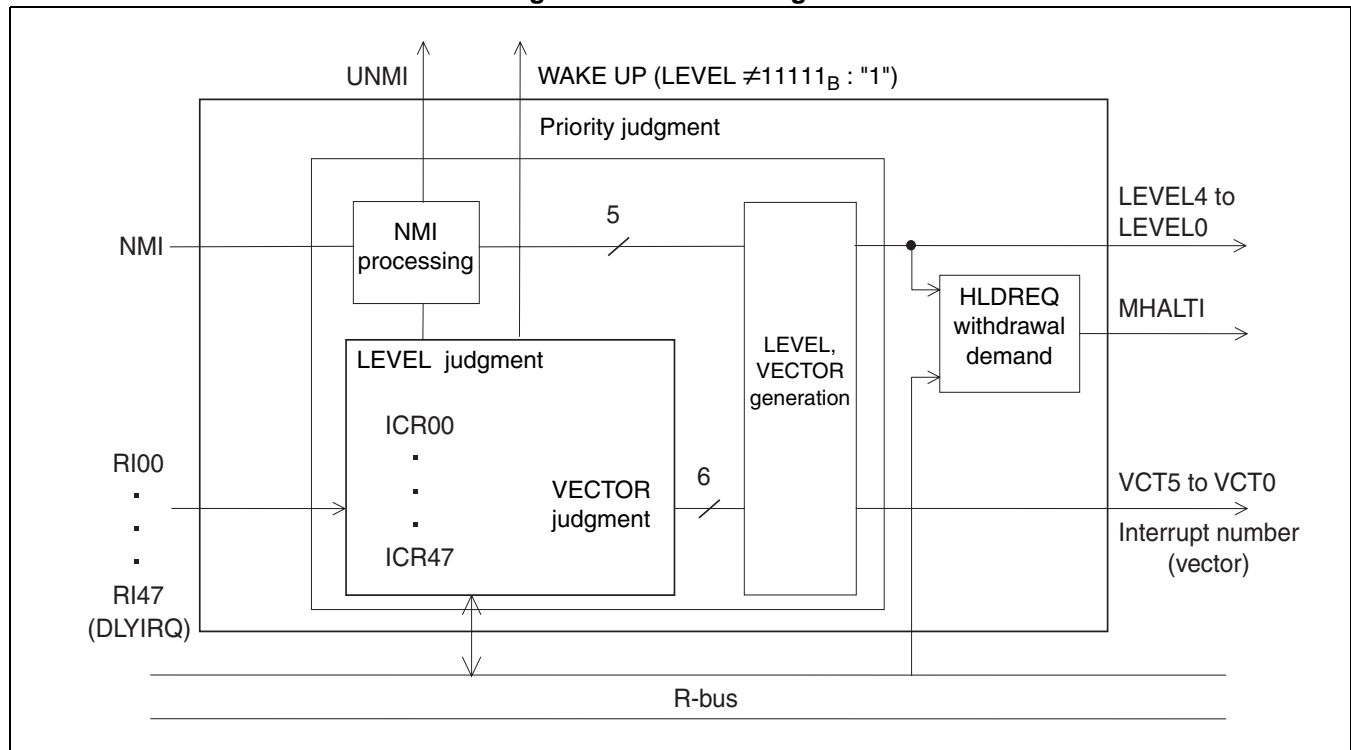
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	
000460 <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR32
000461 <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR33
000462 <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR34
000463 <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR35
000464 <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR36
000465 <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR37
000466 <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR38
000467 <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR39
000468 <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR40
000469 <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR41
00046A <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR42
00046B <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR43
00046C <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR44
00046D <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR45
00046E <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR46
00046F <sub>H</sub>	-	-	-	ICR4	ICR3	ICR2	ICR1	ICR0	ICR47
				R	R/W	R/W	R/W	R/W	
000045 <sub>H</sub>	MHALTI	-	-	LVL4	LVL3	LVL2	LVL1	LVL0	HRCL
	R/W			R	R/W	R/W	R/W	R/W	

## 5.3 Block Diagram

Figure 5.3-1 shows the block diagram of the interrupt controller.

### ■ Block Diagram of the Interrupt Controller

Figure 5.3-1 Block Diagram



## 5.4 Register Details Explanation

This section explains the registers used by the interrupt controller in detail.

### ■ Interrupt Control Register (ICR)

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
–	–	–	ICR4	ICR3	ICR2	ICR1	ICR0	---1111 <sub>B</sub>
			R	R/W	R/W	R/W	R/W	

R/W: Readable/Writable  
 R: Read only  
 –: Undefined bit

The interrupt control register. One register is provided for each interrupt input to set the interrupt level for the corresponding interrupt request.


[bit4 to bit0] ICR4 to ICR0

The interrupt level setting bits specify the interrupt level for the corresponding interrupt request. An interrupt request is masked in the CPU if the interrupt level set in this register is greater than or equal to the level mask value set in the ILM register of CPU.

Initialized to "1111<sub>B</sub>" by reset.

Table 5.4-1 lists the correspondence between the interrupt level and the available interrupt level setting bits.

**Table 5.4-1 Correspondence between Interrupt Levels and Available Interrupt Level Setting Bit Settings**

ICR4	ICR3	ICR2	ICR1	ICR0	Interrupt level	
0	0	0	0	0	0	System reserved
0	1	1	1	0	14	
0	1	1	1	1	15	NMI
1	0	0	0	0	16	Maximum enabled level setting (High)  (Low)
1	0	0	0	1	17	
1	0	0	1	0	18	
1	0	0	1	1	19	
1	0	1	0	0	20	
1	0	1	0	1	21	
1	0	1	1	0	22	
1	0	1	1	1	23	
1	1	0	0	0	24	
1	1	0	0	1	25	
1	1	0	1	0	26	
1	1	0	1	1	27	
1	1	1	0	0	28	
1	1	1	0	1	29	
1	1	1	1	0	30	
1	1	1	1	1	31	Interrupt disabled

ICR4 is fixed at "1". Writing "0" is not permitted.

## ■ Hold Request Cancel Level Register (HRCL)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
000045 <sub>H</sub>	MHALTI	—	—	LVL4	LVL3	LVL2	LVL1	LVL0	0--11111 <sub>B</sub>
	R/W			R	R/W	R/W	R/W	R/W	

R/W: Readable/Writable  
 R: Read only  
 —: Undefined bit

The level setting register used to generate requests to cancel a hold request.

### [bit7] MHALTI

The MHALTI bit causes DMA transfers to be halted by an NMI request. The bit is set to "1" by an NMI request and is cleared by writing "0". Always clear this bit at the end of the NMI routine in the same way as for standard interrupt routines.

### [bit4 to bit0] LVL4 to LVL0

These set the interrupt level for generating a request to the bus master to cancel a hold request.

If an interrupt request with a higher-priority level than the interrupt level set in this register occurs, a request to cancel the hold request is passed to the bus master.

The LVL4 bit is fixed at "1". Writing "0" is not permitted.

## 5.5 Operation of the Interrupt Controller

---

This section explains the operation of the interrupt controller.

- Determining the priority
  - NMI
  - Hold request cancel request
  - Recovery from a standby mode (stop or sleep)
- 

### ■ Determining the Priority

This module selects the highest priority interrupt amongst any interrupt factors that occur simultaneously and outputs the interrupt level and interrupt number for the interrupt factor to the CPU.

The criteria for determining the priority of interrupt factor are as follows.

- (1) NMI
- (2) Interrupt factor that satisfy the following conditions
  - Interrupt level is other than 31 (31 is interrupt disabled)
  - Interrupt factor with lowest interrupt level value
  - Of these, the interrupt factor with the lowest interrupt number

If no interrupt factor is selected by the above criteria, 31 ("11111<sub>B</sub>") is output as the interrupt level. The interrupt number in this case is undefined.

Table 5.5-1 lists the relationship among the interrupt factor, interrupt number, and interrupt level.

**Table 5.5-1 Relationship among Interrupt Factors, Interrupt Numbers, and Interrupt Levels (1 / 3)**

Interrupt factor	Interrupt number		Interrupt level	Offset	TBR default address
	Decimal	Hexadecimal			
Reset	0	00	-	3FC <sub>H</sub>	000FFFFC <sub>H</sub>
Mode vector	1	01	-	3F8 <sub>H</sub>	000FFFF8 <sub>H</sub>
System reserved	2	02	-	3F4 <sub>H</sub>	000FFFF4 <sub>H</sub>
System reserved	3	03	-	3F0 <sub>H</sub>	000FFFF0 <sub>H</sub>
System reserved	4	04	-	3EC <sub>H</sub>	000FFFE <sub>C</sub>
System reserved	5	05	-	3E8 <sub>H</sub>	000FFFE8 <sub>H</sub>
System reserved	6	06	-	3E4 <sub>H</sub>	000FFFE4 <sub>H</sub>
Coprocessor absent trap	7	07	-	3E0 <sub>H</sub>	000FFFE0 <sub>H</sub>
Coprocessor error trap	8	08	-	3DC <sub>H</sub>	000FFFD <sub>C</sub>
INTE instruction	9	09	-	3D8 <sub>H</sub>	000FFFD8 <sub>H</sub>
System reserved	10	0A	-	3D4 <sub>H</sub>	000FFFD4 <sub>H</sub>
System reserved	11	0B	-	3D0 <sub>H</sub>	000FFFD0 <sub>H</sub>
Step trace trap	12	0C	-	3CC <sub>H</sub>	000FFFC <sub>C</sub>
NMI demand (tool)	13	0D	-	3C8 <sub>H</sub>	000FFFC8 <sub>H</sub>
Undefined instruction exception	14	0E	-	3C4 <sub>H</sub>	000FFFC4 <sub>H</sub>
NMI demand	15	0F	15(F <sub>H</sub> ) fixed	3C0 <sub>H</sub>	000FFFC0 <sub>H</sub>
External interrupt 0	16	10	ICR00	3BC <sub>H</sub>	000FFFB <sub>C</sub>
External interrupt 1	17	11	ICR01	3B8 <sub>H</sub>	000FFFB8 <sub>H</sub>
External interrupt 2	18	12	ICR02	3B4 <sub>H</sub>	000FFFB4 <sub>H</sub>
External interrupt 3	19	13	ICR03	3B0 <sub>H</sub>	000FFFB0 <sub>H</sub>
External interrupt 4	20	14	ICR04	3AC <sub>H</sub>	000FFFA <sub>C</sub>
External interrupt 5	21	15	ICR05	3A8 <sub>H</sub>	000FFFA8 <sub>H</sub>
External interrupt 6/C-CAN wake up*	22	16	ICR06	3A4 <sub>H</sub>	000FFFA4 <sub>H</sub>
External interrupt 7	23	17	ICR07	3A0 <sub>H</sub>	000FFFA0 <sub>H</sub>
Reload timer 0	24	18	ICR08	39C <sub>H</sub>	000FFF9 <sub>C</sub>
Reload timer 1	25	19	ICR09	398 <sub>H</sub>	000FFF98 <sub>H</sub>
Reload timer 2	26	1A	ICR10	394 <sub>H</sub>	000FFF94 <sub>H</sub>
UART0 (receive completed)	27	1B	ICR11	390 <sub>H</sub>	000FFF90 <sub>H</sub>
UART0 (transmit completed)	28	1C	ICR12	38C <sub>H</sub>	000FFF8 <sub>C</sub>

\*: Interrupt of C-CAN is function loaded in MB91F267NA/MB91267NA.



**Table 5.5-1 Relationship among Interrupt Factors, Interrupt Numbers, and Interrupt Levels (2 / 3)**

Interrupt factor	Interrupt number		Interrupt level	Offset	TBR default address
	Decimal	Hexadecimal			
DTTI pin	29	1D	ICR13	388 <sub>H</sub>	000FFF88 <sub>H</sub>
DMAC0 (end and error)	30	1E	ICR14	384 <sub>H</sub>	000FFF84 <sub>H</sub>
DMAC1 (end and error)	31	1F	ICR15	380 <sub>H</sub>	000FFF80 <sub>H</sub>
DMAC2/DMAC3/DMAC4 (end and error)	32	20	ICR16	37C <sub>H</sub>	000FFF7C <sub>H</sub>
UART1 (receive completed)	33	21	ICR17	378 <sub>H</sub>	000FFF78 <sub>H</sub>
UART1 (transmit completed)	34	22	ICR18	374 <sub>H</sub>	000FFF74 <sub>H</sub>
C-CAN0*	35	23	ICR19	370 <sub>H</sub>	000FFF70 <sub>H</sub>
System reserved	36	24	ICR20	36C <sub>H</sub>	000FFF6C <sub>H</sub>
Sum of product	37	25	ICR21	368 <sub>H</sub>	000FFF68 <sub>H</sub>
PPG0/PPG1	38	26	ICR22	364 <sub>H</sub>	000FFF64 <sub>H</sub>
PPG2/PPG3	39	27	ICR23	360 <sub>H</sub>	000FFF60 <sub>H</sub>
PPG4/PPG5/PPG6/PPG7	40	28	ICR24	35C <sub>H</sub>	000FFF5C <sub>H</sub>
System reserved	41	29	ICR25	358 <sub>H</sub>	000FFF58 <sub>H</sub>
Waveform (underflow) 0/1/2	42	2A	ICR26	354 <sub>H</sub>	000FFF54 <sub>H</sub>
Free-run timer 1 (compare clear)	43	2B	ICR27	350 <sub>H</sub>	000FFF50 <sub>H</sub>
Free-run timer 1 (0-detect)	44	2C	ICR28	34C <sub>H</sub>	000FFF4C <sub>H</sub>
Free-run timer 2 (compare clear)	45	2D	ICR29	348 <sub>H</sub>	000FFF48 <sub>H</sub>
Free-run timer 2 (0-detect)	46	2E	ICR30	344 <sub>H</sub>	000FFF44 <sub>H</sub>
Time-base timer overflow	47	2F	ICR31	340 <sub>H</sub>	000FFF40 <sub>H</sub>
Free-run timer 0 (compare clear)	48	30	ICR32	33C <sub>H</sub>	000FFF3C <sub>H</sub>
Free-run timer 0 (0-detect)	49	31	ICR33	338 <sub>H</sub>	000FFF38 <sub>H</sub>
System reserved	50	32	ICR34	334 <sub>H</sub>	000FFF34 <sub>H</sub>
A/D converter 1	51	33	ICR35	330 <sub>H</sub>	000FFF30 <sub>H</sub>
A/D converter 2	52	34	ICR36	32C <sub>H</sub>	000FFF2C <sub>H</sub>
PWC0 (measurement finished)	53	35	ICR37	328 <sub>H</sub>	000FFF28 <sub>H</sub>
System reserved	54	36	ICR38	324 <sub>H</sub>	000FFF24 <sub>H</sub>
PWC0 (overflow)	55	37	ICR39	320 <sub>H</sub>	000FFF20 <sub>H</sub>
System reserved	56	38	ICR40	31C <sub>H</sub>	000FFF1C <sub>H</sub>
ICU 0 (fetch)	57	39	ICR41	318 <sub>H</sub>	000FFF18 <sub>H</sub>

\*: Interrupt of C-CAN is function loaded in MB91F267NA/MB91267NA.

**Table 5.5-1 Relationship among Interrupt Factors, Interrupt Numbers, and Interrupt Levels (3 / 3)**

Interrupt factor	Interrupt number		Interrupt level	Offset	TBR default address
	Decimal	Hexadecimal			
ICU 1 (fetch)	58	3A	ICR42	314 <sub>H</sub>	000FFF14 <sub>H</sub>
ICU 2/ICU 3 (fetch)	59	3B	ICR43	310 <sub>H</sub>	000FFF10 <sub>H</sub>
OCU0/OCU1 (match)	60	3C	ICR44	30C <sub>H</sub>	000FFF0C <sub>H</sub>
OCU2/OCU3 (match)	61	3D	ICR45	308 <sub>H</sub>	000FFF08 <sub>H</sub>
OCU4/OCU5 (match)	62	3E	ICR46	304 <sub>H</sub>	000FFF04 <sub>H</sub>
Delay interrupt trigger bit	63	3F	ICR47	300 <sub>H</sub>	000FFF00 <sub>H</sub>
System reserved (used for REALOS)	64	40	-	2FC <sub>H</sub>	000FFEFC <sub>H</sub>
System reserved (used for REALOS)	65	41	-	2F8 <sub>H</sub>	000FFE8 <sub>H</sub>
System reserved	66	42	-	2F4 <sub>H</sub>	000FFE4 <sub>H</sub>
System reserved	67	43	-	2F0 <sub>H</sub>	000FFE0 <sub>H</sub>
System reserved	68	44	-	2EC <sub>H</sub>	000FFEEC <sub>H</sub>
System reserved	69	45	-	2E8 <sub>H</sub>	000FFEE8 <sub>H</sub>
System reserved	70	46	-	2E4 <sub>H</sub>	000FFEE4 <sub>H</sub>
System reserved	71	47	-	2E0 <sub>H</sub>	000FFEE0 <sub>H</sub>
System reserved	72	48	-	2DC <sub>H</sub>	000FFEDC <sub>H</sub>
System reserved	73	49	-	2D8 <sub>H</sub>	000FFED8 <sub>H</sub>
System reserved	74	4A	-	2D4 <sub>H</sub>	000FFED4 <sub>H</sub>
System reserved	75	4B	-	2D0 <sub>H</sub>	000FFED0 <sub>H</sub>
System reserved	76	4C	-	2CC <sub>H</sub>	000FFECC <sub>H</sub>
System reserved	77	4D	-	2C8 <sub>H</sub>	000FFEC8 <sub>H</sub>
System reserved	78	4E	-	2C4 <sub>H</sub>	000FFEC4 <sub>H</sub>
System reserved	79	4F	-	2C0 <sub>H</sub>	000FFEC0 <sub>H</sub>
Used in INT instruction	80 to 255	50 to FF	-	2BC <sub>H</sub> to 000 <sub>H</sub>	000FFEBC <sub>H</sub> to 000FFC00 <sub>H</sub>

## ■ NMI (Non Maskable Interrupt)

The NMI has the highest priority of all the interrupt factors handled by this module.

Accordingly, the NMI is always selected if it occurs at the same time as another interrupt factor.

- The following information is passed to the CPU when an NMI occurs.

Interrupt level : 15 ("01111<sub>B</sub>")

Interrupt number : 15 ("0001111<sub>B</sub>")

- NMI detection

The external interrupts/NMI module sets and detects NMIs. In this module, an NMI request only generates the interrupt level, interrupt number, and MHALTI.

- Halt on DMA transfer by NMI

When an NMI request occurs, the MHALTI bit in the HRCL register goes to "1" to halt DMA transfer. To release the halt on DMA transfer, clear the MHALTI bit to "0" at the end of the NMI routine.

## ■ Hold Request Cancel Request

If you want to process high priority interrupts during a CPU hold (during DMA transfer), the module that generated the hold request needs to cancel the request. Set the interrupt level at which a request to cancel is to be generated in the HRCL register.

- Generation criteria

If an interrupt factor with a higher-priority level than the level set in the HRCL register occurs, a request to cancel the hold request is issued to the DMAC.

If interrupt level in HRCL register < level of interrupt after the priority judgement, then do generate cancel request.

If interrupt level in HRCL register ≤ level of interrupt after the priority judgement, then do not generate cancel request.

The cancel request remains active until the interrupt factor that generated the cancel request is cleared and therefore no DMA transfer occurs during this time. Always clear the associated interrupt factor.

As the MHALTI bit in the HRCL register goes to "1" when an NMI is used, the cancel request is active.

- Possible levels

The values able to be set in the HRCL register are "10000<sub>B</sub>" to "11111<sub>B</sub>", the same as the ICR.

If "11111<sub>B</sub>" is set, a cancel request is generated for all interrupt levels. If "10000<sub>B</sub>" is set, a cancel request is only generated for an NMI.

Table 5.5-2 shows the interrupt level settings for generating a request to cancel a hold request.

**Table 5.5-2 Interrupt Level Settings that Generate a Hold Request Cancel Request**

HRCL register	Interrupt levels that generate a cancel request
16	NMI only
17	NMI, interrupt level 16
18	NMI, interrupt levels 16, 17
31	NMI, interrupt levels 16 to 30 [Initial value]

Once reset, DMA transfer is halted for all interrupt levels. As this means that no DMA transfer will be performed when an interrupt occurs, set the required value in the HRCL register.

### ■ Recovery from a Standby Mode (Stop or Sleep)

The function for using an interrupt request to recover from stop mode is performed by this module.

If one or more interrupt requests from a peripheral including NMI (with interrupt level other than "11111<sub>B</sub>") occur, a request to recover from stop mode is sent to the clock controller.

As the priority judgement unit restarts operation once the clock supply starts after recovery from stop mode, the CPU is able to execute instructions until the priority judgement unit produces a result.

The same operation occurs when recovering from sleep mode.

Access to the registers in this module remains possible even in sleep mode.

---

#### Notes:

- The device also recovers from stop mode when an NMI request occurs. However, please apply a valid input level to the  $\overline{\text{NMI}}$  pin in stop mode.
  - Set the interrupt level for interrupt factors that you do not want to cause the device to recover from stop or sleep mode to "11111<sub>B</sub>" in the corresponding peripheral control register.
-

## ■ Example of Using the Function to Generate a Request to Cancel a Hold Request (HRCR)

If you want the CPU to perform high-priority processing during DMA transfer, you need to cancel the hold state by requesting the DMA to cancel its hold request. This uses an interrupt to cause the DMA to cancel its hold request and to give priority to CPU operation.

### ● Control register

(1) HRCL (Hold request cancel level setting register) : this module

If an interrupt with a higher-priority level than the interrupt level set in this register occurs, a request to cancel the hold request is passed to the DMA. Sets the level to use as the criterion.

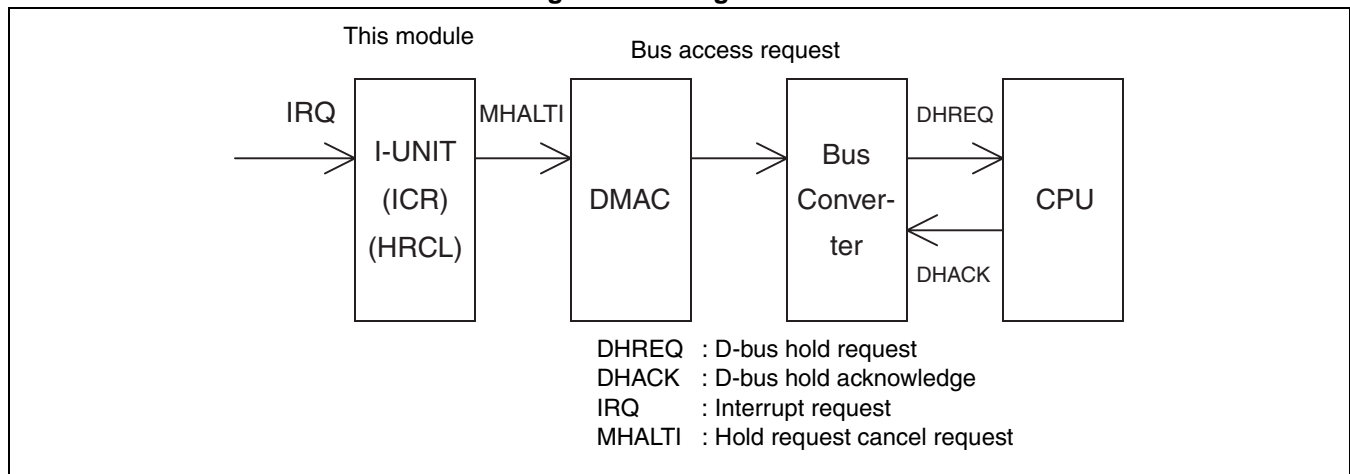
(2) ICR: this module

Set an interrupt level with a higher priority than the level set in the HRCL register in the ICRs of the interrupt factors you want to use.

### ● Hardware configuration

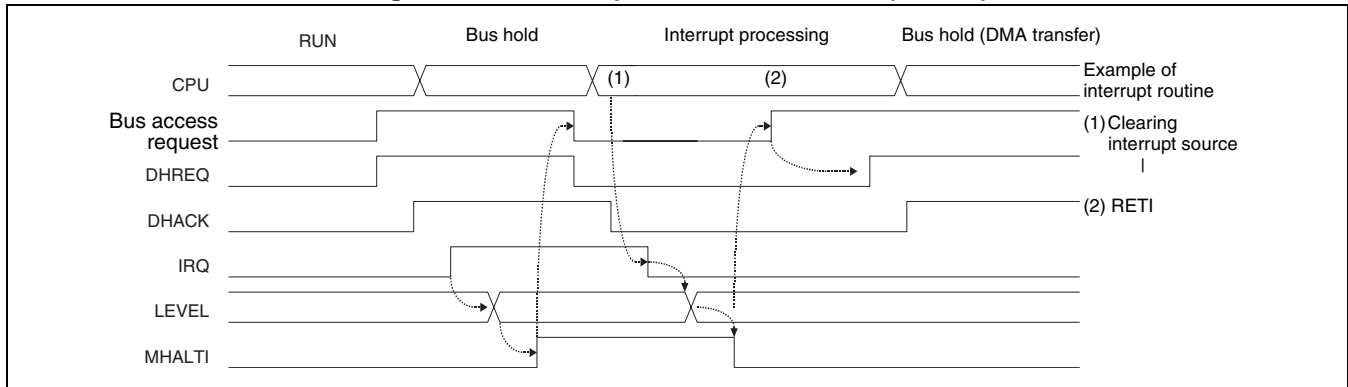
The signal flow is shown below.

**Figure 5.5-1 Signal Flow**



● Sequence

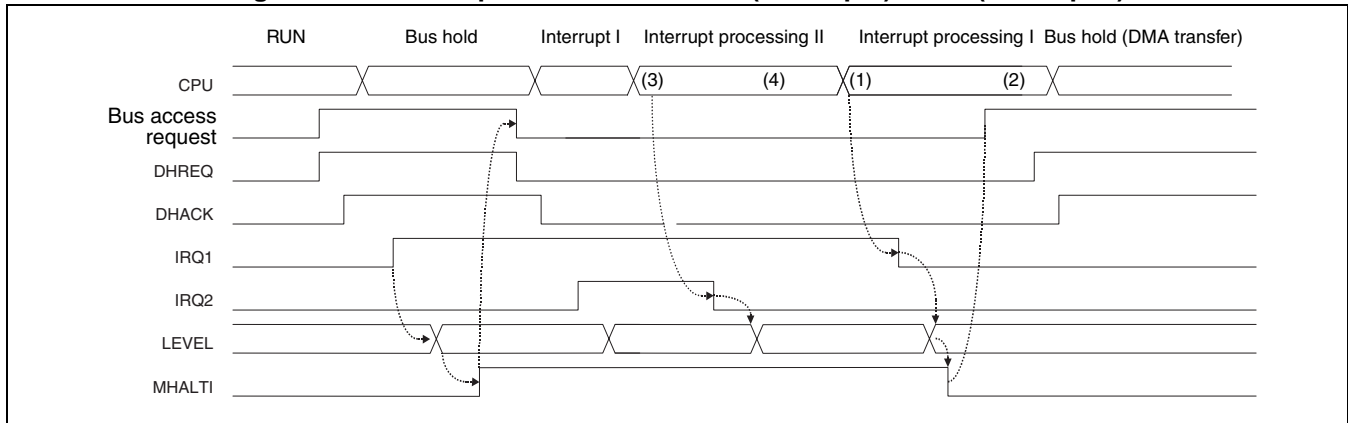
**Figure 5.5-2 Interrupt Level  $HRCL < ICR$  (LEVEL)**



When an interrupt request occurs and the interrupt level changes, the MHALTI signal to the DMA goes active if the new level has a higher priority than the level set in the HRCL register. This causes the DMA to halt access requests and the CPU to recover from the hold state and start processing the interrupt.

The diagram below shows the case when multiple interrupts occur.

**Figure 5.5-3 Interrupt Level  $HRCL < ICR$  (Interrupt I)  $< ICR$  (Interrupt II)**



Example of interrupt routine

(1), (3): Clearing interrupt source

(2), (4): RETI

The above example shows the case when a higher priority interrupt occurs during execution of interrupt routine I.

DHREQ becomes "L" level while the interrupt with an interrupt level higher than the interrupt level set in the HRCL register is present.

Note:

Take note of the relationship between the interrupt levels set in the HRCL register and ICRs.



# **CHAPTER 6**

---

## ***EXTERNAL INTERRUPT AND NMI CONTROLLER***

This chapter describes the overview of the external interrupt and NMI controller, the configuration and functions of registers, and operation of the external interrupt and NMI controller.

### 6.1 External Interrupt and NMI Controller



## 6.1 External Interrupt and NMI Controller

The external interrupt controller is a block that controls external interrupt requests input to NMI and INT0 to INT7.

The external interrupt input can be selected from H level, L level, rising edge, or falling edge as the level of a request to be detected.

### ■ Register List

#### EIRR0

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
000040 <sub>H</sub>	ER7	ER6	ER5	ER4	ER3	ER2	ER1	ER0	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

#### ENIR0

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
000041 <sub>H</sub>	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

#### ELVR0

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
000042 <sub>H</sub>	LB7	LA7	LB6	LA6	LB5	LA5	LB4	LA4	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

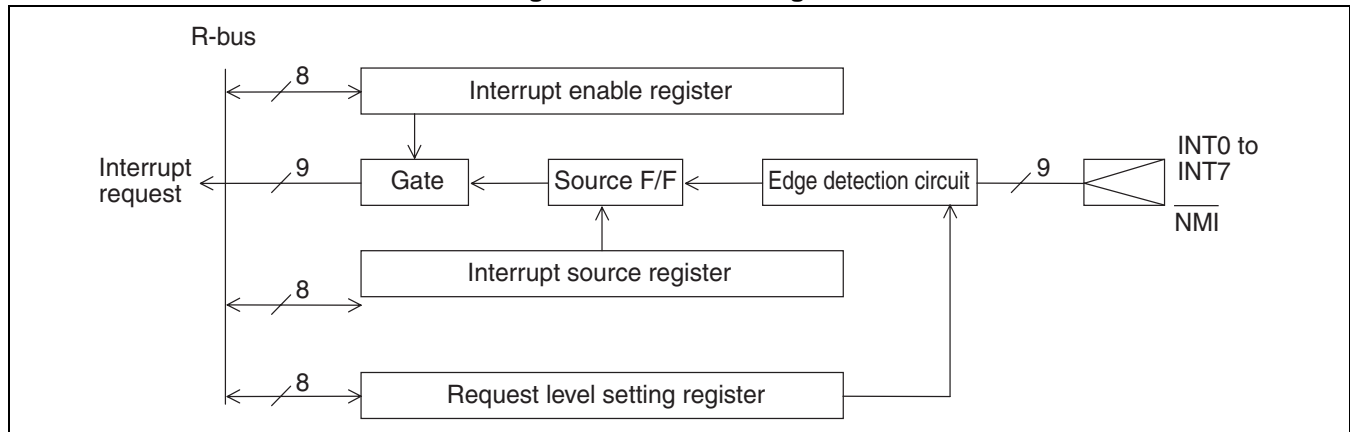
#### ELVR0

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
000043 <sub>H</sub>	LB3	LA3	LB2	LA2	LB1	LA1	LB0	LA0	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable

## ■ Block Diagram

Figure 6.1-1 Block Diagram



## ■ Detailed Explanation of Register

### ● Interrupt enable register (ENIR: ENable Interrupt request Register)

ENIR0

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
000041 <sub>H</sub>	EN7	EN6	EN5	EN4	EN3	EN2	EN1	EN0	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable

The ENIR performs mask control for external interrupt request output. Output for an interrupt request is enabled based on the bit in this register to which "1" has been written (INT0 enable is controlled by EN0), after which the request is output to the interrupt controller. The pin corresponding to the bit to which "0" is written holds the interrupt source but does not generate a request to the interrupt controller.

No enable bit exists for NMI.

### ● External interrupt source register (EIRR: External Interrupt Request Register)

EIRR0

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
000040 <sub>H</sub>	ER7	ER6	ER5	ER4	ER3	ER2	ER1	ER0	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable

The EIRR register, when it is read, indicates that a corresponding external interrupt request exists. When it is written to, the contents of the flip-flop that indicates this request are cleared.

If "1" is read from the EIRR register, an external interrupt request exists at the pin corresponding to this bit.

Write "0" to this register to clear the request flip-flop of the corresponding bit.

Writing "1" to this has no effect.

For a read by a read modify write (RMW) type instructions, "1" is read.

Flag for NMI cannot be accessed by the user.

● External interrupt request level setting register (ELVR: External LeVel Register)

## ELVR0

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
000042 <sub>H</sub>	LB7	LA7	LB6	LA6	LB5	LA5	LB4	LA4	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

## ELVR0

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
000043 <sub>H</sub>	LB3	LA3	LB2	LA2	LB1	LA1	LB0	LA0	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable

The ELVR selects how a request is detected. Two bits are assigned to each of INT0 to INT7, which results in the following setting. Even though the bits of the EIRR are cleared while the request input is a level, the pertinent bits are set again as long as the input is active level.

**Table 6.1-1 Assignment of ELVR**

LBx	LAx	Operation
0	0	"L" level indicates the existence of a request.
0	1	"H" level indicates the existence of a request.
1	0	A rising edge indicates the existence of a request.
1	1	A falling edge indicates the existence of a request.

The detection level of NMI is always falling edge.

Also, when NMI is used for returning from the stop state, "L" level is detected.

Note:

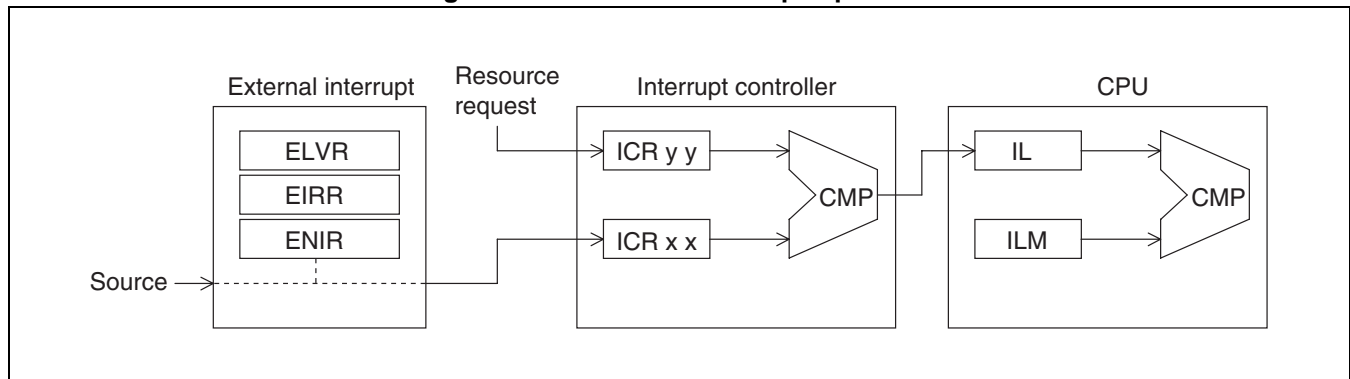
As changing the external interrupt request level may cause an interrupt source internally, always clear the external interrupt request register (EIRR) after changing the external interrupt request level. To clear the external interrupt request register, first read the external interrupt request level setting register and then write to the EIRR register to clear it.

## ■ Explanation of Operation

### ● Operation of external interrupt

If, after a request level and an enable register are defined, a request defined in the ELVR register is input to the corresponding pin, this module generates an interrupt request signal to the interrupt controller. For simultaneous interrupt requests from resources, the interrupt controller determines the interrupt request with the highest priority and generates an interrupt for it.

**Figure 6.1-2 External Interrupt Operation**



### ● Return from standby

Be sure to disable the channel that is not used before entering to standby.

### ● Operating procedure for an external interrupt

Set up a register located inside the external interrupt controller as follows:

1. Set the general-purpose I/O port to be shared with pin for the external interrupt input to input port.
2. Disable the target bit in the interrupt enable register (ENIR).
3. Set the target bit in the external interrupt request level setting register (ELVR).
4. Read the external interrupt request level setting register (ELVR).
5. Clear the target bit in the external interrupt source register (EIRR).
6. Enable the target bit in the interrupt enable register (ENIR).

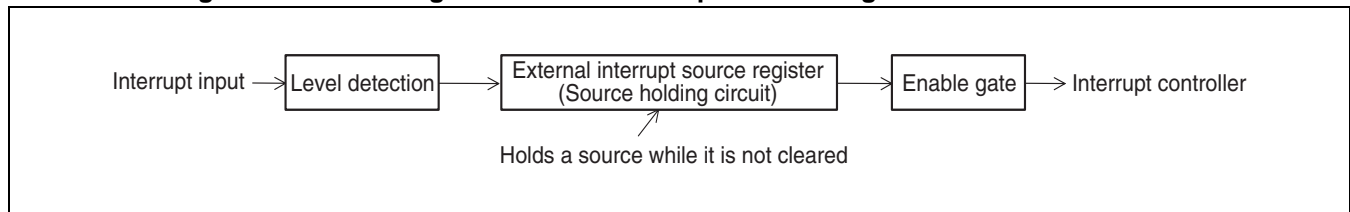
(Simultaneous writing of 16-bit data is supported for steps 5) and 6)).

Before setting a register in this module, you must disable the enable register. In addition, before enabling the enable register, you must clear the interrupt source register. This procedure is required to prevent an interrupt source from occurring by mistake while a register is being set or an interrupt is enabled.

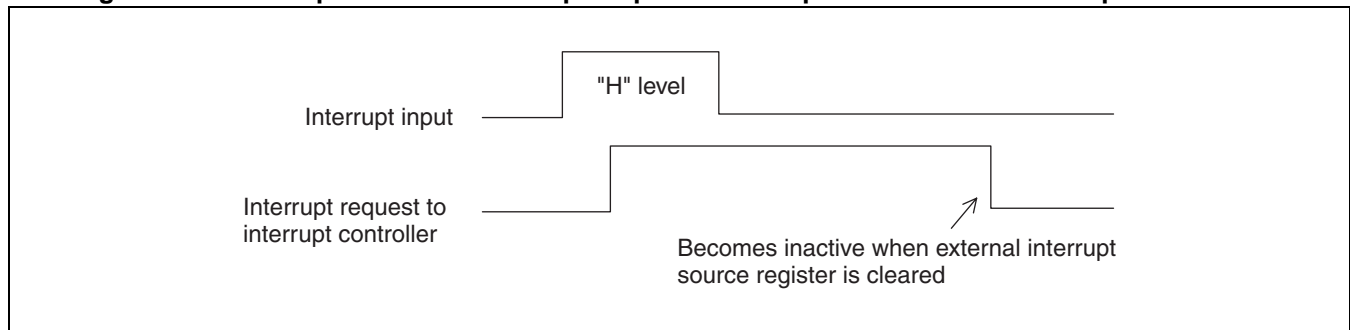
### ● External interrupt request level

- If the request level is in an edge request, a pulse width of at least three machine cycles (peripheral clock machine cycles) is required to detect an edge.
- When the request input level is set to level request, the pulse width must be at least three machine cycles in duration. Also, while the interrupt input pin remains at the active level, the interrupt request to the interrupt controller will remain present even if you clear the external interrupt source register.
- If the request input level is a level setting and request input arrives from outside and is then cancelled, the request to the interrupt controller remains active because a source holding circuit exists internally. The external interrupt source register must be cleared to cancel a request to the interrupt controller.

**Figure 6.1-3 Clearing the External Interrupt Source Register when a Level is Set**

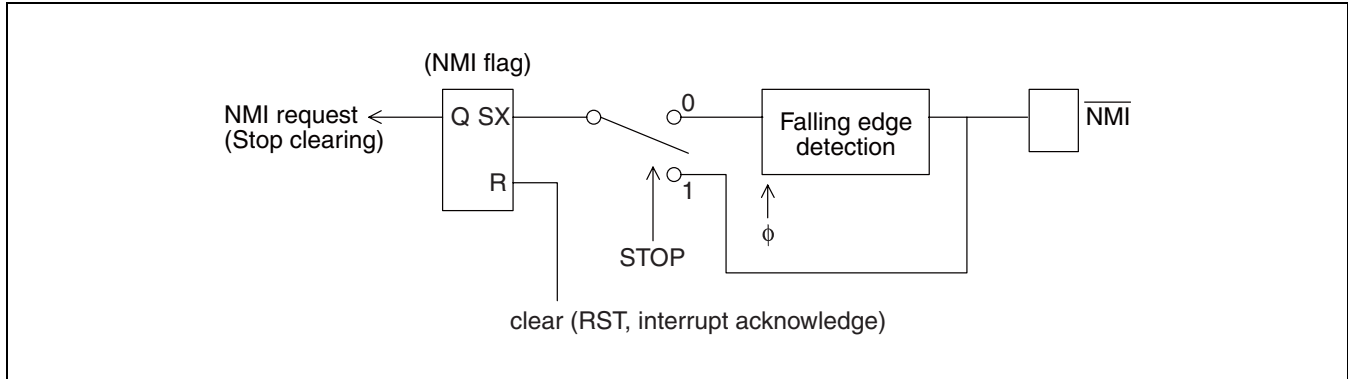


**Figure 6.1-4 Interrupt Source and Interrupt Request to Interrupt Controller when Interrupts are Enabled**



### ● NMI

- An NMI has priority over all other user interrupts, and cannot be masked. As an exception, if an NMI is activated without setting ILM before it is used, the CPU does not accept the NMI request but only detects the NMI source. The NMI source is then held until ILM is set to the level that allows the NMI to be accepted. For this reason, before using an NMI, be sure to set ILM to 16 or more after a reset. In addition, because the internal source flag of the NMI cannot be accessed by CPU, the  $\overline{\text{NMI}}$  pin holds "H" level after a reset.
- An NMI is accepted under the following conditions:
  - Normal : Falling edge
  - STOP mode : "L" level
- An NMI can be used to clear stop mode, inputting the "L" level in the stop state clears the stop state and causes the oscillation stabilization wait time to start. The NMI request detector has an NMI flag that is set for an NMI request and is cleared only if an interrupt for the NMI itself is accepted or a reset occurs. Note that this bit is not readable or writable.

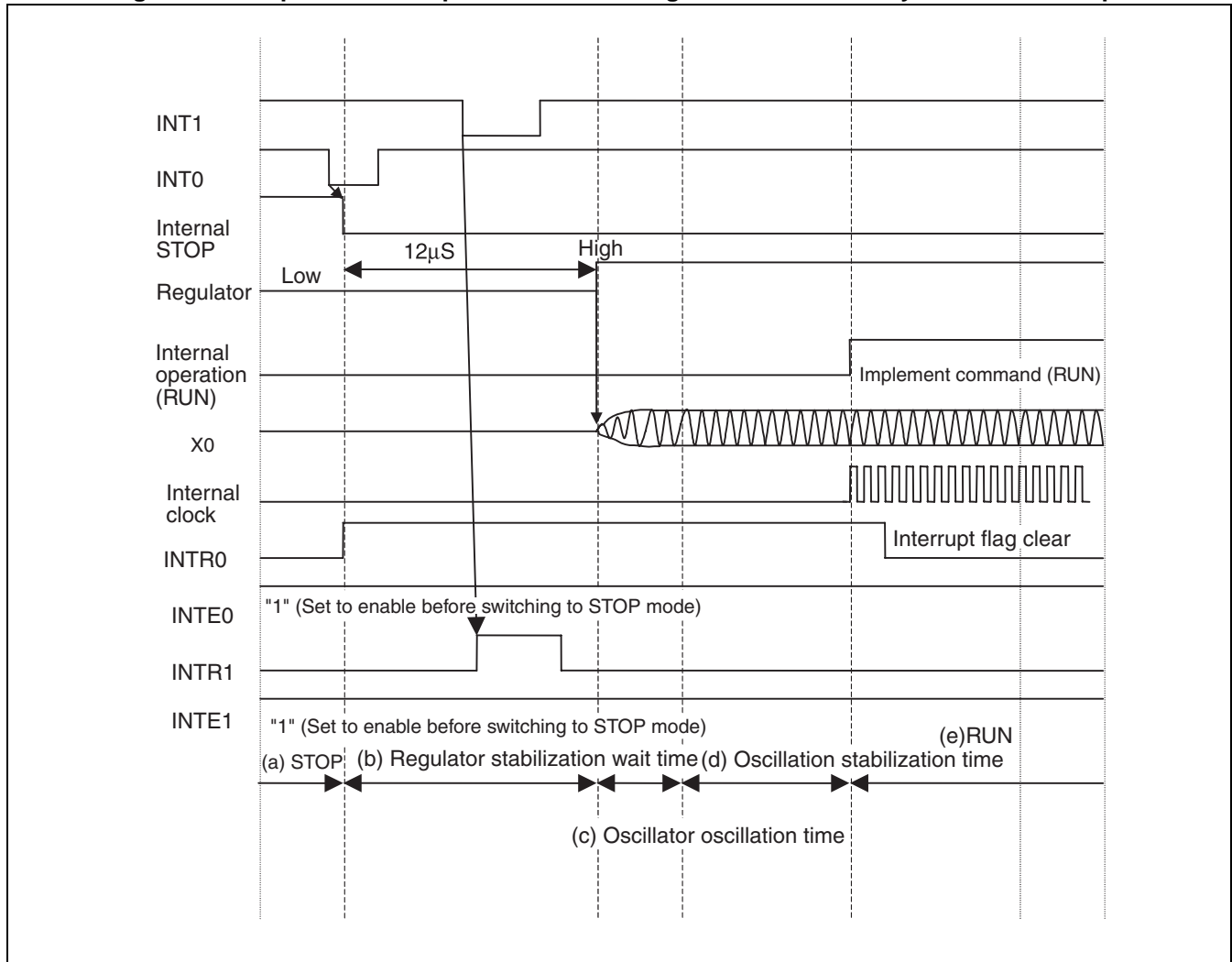
**Figure 6.1-5 NMI Request Detection**

### ■ Precautions when Returning from STOP State Using External Interrupt

The external interrupt signal that is initially input to the INT pin in a STOP state is input asynchronously, allowing the device to return from the STOP state. Note, however, that there are periods from the release of the STOP state till the end of the oscillation stabilization wait time, in such periods the input of other external interrupt signals cannot be identified (period of  $b + c + d$  in Figure 6.1-6). This is because the external input signal after the release of STOP mode is synchronized with the internal clock; consequently, the corresponding interrupt source cannot be retained while the clock is still unstable.

Therefore, input an external interrupt signal after the oscillation stabilization wait time has elapsed, when inputting an external interrupt after the release of STOP mode.

**Figure 6.1-6 Operational Sequence for Returning from STOP State by External Interrupt**



## ■ Operation when Recovering from STOP Mode

The operation when recovery from STOP mode is triggered by an external interrupt from an operating circuit is as follows.

### ● Processing prior to entering STOP mode

External interrupt input

When you wish to recover from STOP mode, the external interrupt signal is passed to the input signal asynchronously. The mechanism for falling the internal STOP signal is triggered as soon as the interrupt signal becomes active. At the same time, the input is switched in the external interrupt circuit to synchronize it with any other level interrupt inputs.

### ● Regulator stabilization wait time

The mechanism for switching from the regulator used during STOP mode to the regulator used during RUN mode is invoked when the internal STOP signal is fallen. As misoperation may occur if internal operation starts before the voltage output of the RUN mode regulator has stabilized, a delay time is used to wait for the internal output voltage to stabilize. The clock remains halted during this time.

### ● Oscillator startup time

The clock oscillation starts after the regulator stabilization delay time elapses. The startup time of the oscillator depends on the type of oscillator used.

### ● Oscillation stabilization wait time

After the oscillator startup time elapses, the device waits for the oscillation stabilization wait time which is generated internally. This time is specified by the OS1 and OS0 bits in the standby control register. After the oscillation stabilization wait time elapses, the internal clock supply starts and execution of the interrupt handler for the external interrupt starts. At the same time the external interrupts other than the return source from STOP mode become able to be received.





# **CHAPTER 7**

---

# ***REALOS-RELATED HARDWARE***

The REALOS-related hardware is used by the real-time OS. Accordingly, these functions cannot be used by user programs if using REALOS.

This chapter explains the overview of the delayed interrupt module and bit search module, the configuration and functions of registers, and the operation of the delayed interrupt module and bit search module.

7.1 Delayed Interrupt Module

7.2 Bit Search Module

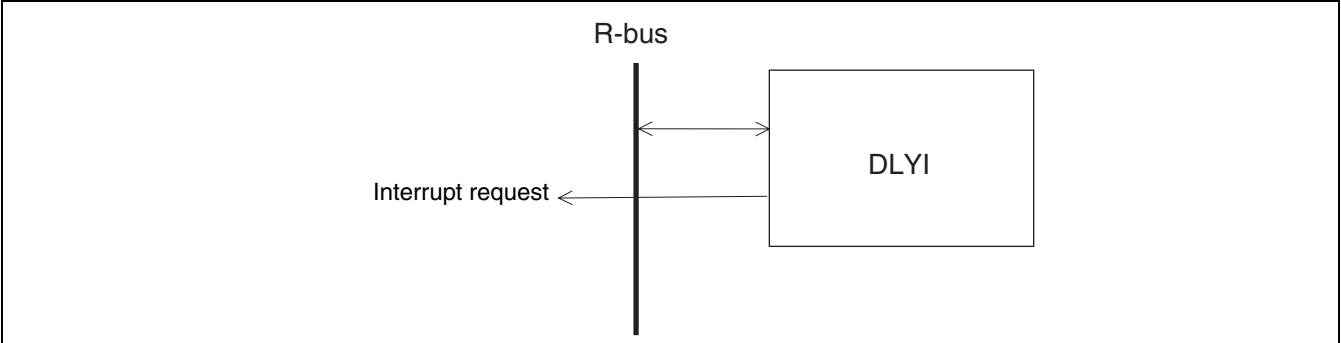
# 7.1 Delayed Interrupt Module

The delayed interrupt module is used to generate the interrupt for task switching. An interrupt request to the CPU can be generated and cleared by software using this module.

## ■ Delayed Interrupt Module Register

DICR								
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
000044 <sub>H</sub>	–	–	–	–	–	–	–	DLYI
								R/W
R/W: Readable/Writable								
–: Undefined bit								

## ■ Block Diagram of Delayed Interrupt Module



## ■ Register Details Explanation

### ● DICR (Delayed Interrupt Control Register)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
000044 <sub>H</sub>	—	—	—	—	—	—	—	DLYI	-----0 <sub>B</sub>
								R/W	
R/W: Readable/Writable									
—: Undefined bit									

This register controls the delayed interrupt.

[bit0] DLYI

DLYI	Description
0	Delayed interrupt source cleared or no request present [Initial value]
1	Delayed interrupt source occurred

This bit controls generation and clearing of the interrupt source.

## ■ Operation Explanation

The delayed interrupt is used to generate the interrupt for task switching. An interrupt request to the CPU can be generated and cleared by software using this function.

### ● Interrupt number

The delayed interrupt is assigned to the interrupt source corresponding to the highest interrupt number.

On this model, the delayed interrupt has interrupt number 63 (3F<sub>H</sub>).

### ● DLYI Bit of DICR

Writing "1" to this bit generates a delayed interrupt source. Similarly, writing "0" to this bit clears the delayed interrupt source.

This bit functions like a standard interrupt source flag and should be cleared in the interrupt routine at the same time as performing task switching.

## 7.2 Bit Search Module

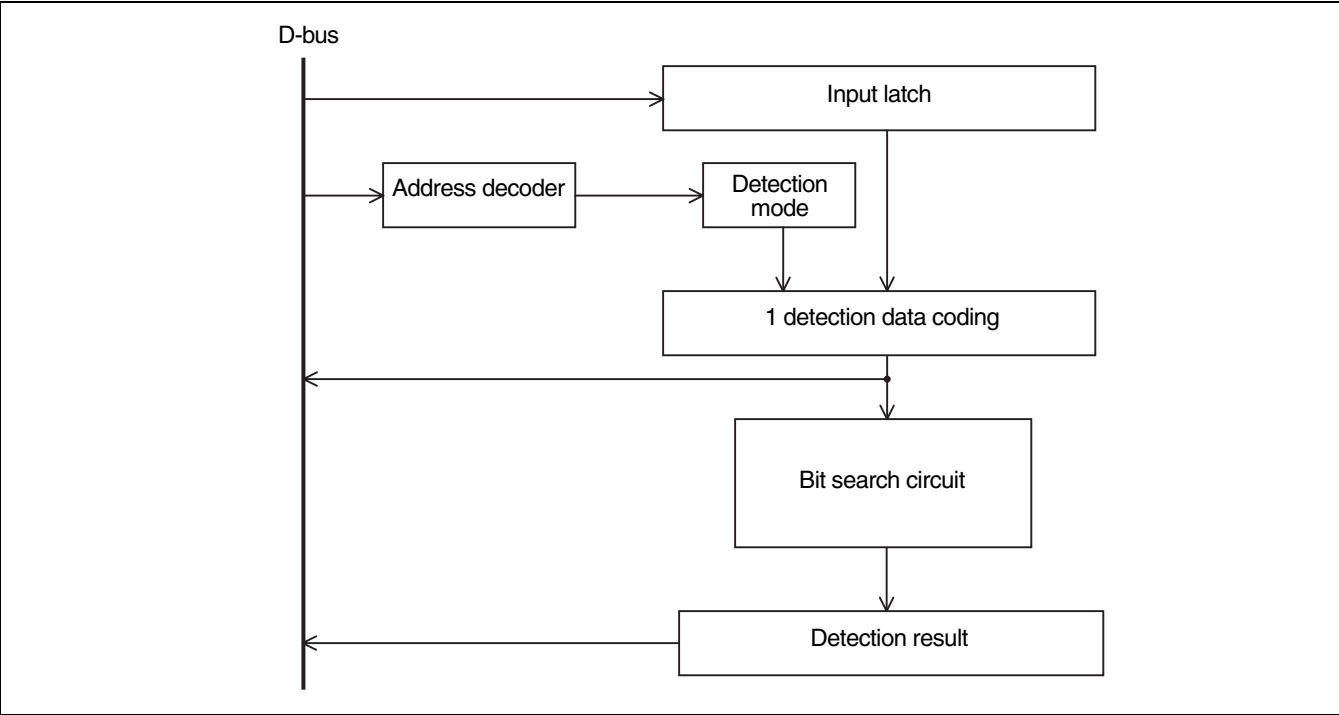
Bit search module searches for "0", "1", or change point in the data written to the input register and returns the detected bit position.

### Bit Search Module Registers

	bit31	bit0
Address: 0003F0 <sub>H</sub>	BSD0	0-detect data register
Address: 0003F4 <sub>H</sub>	BSD1	1-detect data register
Address: 0003F8 <sub>H</sub>	BSDC	Change point detection data register
Address: 0003FC <sub>H</sub>	BSRR	Detection result register

### Block Diagram of Bit Search Module

Figure 7.2-1 Block Diagram



## ■ Register Details Explanation

### ● 0-detect data register (BSD0)

Address	bit31	bit0
0003F0 <sub>H</sub>		
Attribute → Write only		
Initial value →	XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX	

Detects "0" in the value written.

The initial value by reset is undefined. The read value is undefined.

Use a 32-bit data transfer instruction to transfer the data.

(Do not use 8-bit or 16-bit data transfer instructions.)

### ● 1-detect data register (BSD1)

Address	bit31	bit0
0003F4 <sub>H</sub>		
Attribute → Readable/Writable		
Initial value →	XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX	

Use a 32-bit data transfer instruction to transfer the data.

(Do not use 8-bit or 16-bit data transfer instructions.)

- Writing

Detects "1" in the written value.

- Reading

Reads the data to enable the internal state of the bit search module to be saved. This is used to save and restore the original state when the bit search module is used by an interrupt handler or similar.

Saving and restoring can be performed using only the 1-detect data register even if data is written to the 0-detect or change point detection data registers.

The initial value by reset is undefined.

### ● Change point detection data register (BSDC)

Address	bit31	bit0
0003F8 <sub>H</sub>		
Attribute → Write only		
Initial value →	XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX	

Detects the change point in the written value.

The initial value by reset is undefined.

The read value is undefined.

Use a 32-bit data transfer instruction to transfer the data.

(Do not use 8-bit or 16-bit data transfer instructions.)

● Detection result register (BSRR)

Reads the result of the 0-detect, 1-detect, or change point detect operation.  
Which result is read from this register is determined by which data register was written to most recently.

Address	bit31	bit0
0003FC <sub>H</sub>		
Attribute	→ Read only	
Initial value	→ XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX	

■ Operation Explanation

● 0 detection

The module scans the value written to the 0-detect data register from the MSB to the LSB and returns the position of the first "0" detection.  
The result is obtained by reading the detection result register.  
Table 7.2-1 shows the relationship between the returned value and the detected position.  
If no "0" exists (if the value is "FFFFFFFF<sub>H</sub>"), "32" is returned as the search result.

[Execution example]

Write data		Read data (decimal)
111111111111111111000000000000 <sub>B</sub> (FFFFF000 <sub>H</sub> )	→	20
11111000010010011110000010101010 <sub>B</sub> (F849E0AA <sub>H</sub> )	→	5
100000000000001010101010101010 <sub>B</sub> (8002AAAA <sub>H</sub> )	→	1
111111111111111111111111111111 <sub>B</sub> (FFFFFFFF <sub>H</sub> )	→	32

● 1 detection

The module scans the value written to the 1-detect data register from the MSB to the LSB and returns the position of the first "1" detection.  
The result is obtained by reading the detection result register.  
Table 7.2-1 shows the relationship between the returned value and the detected position.  
If no "1" exists (if the value is "00000000<sub>H</sub>"), "32" is returned as the search result.

[Execution example]

Write data		Read data (decimal)
001000000000000000000000000000 <sub>B</sub> (20000000 <sub>H</sub> )	→	2
00000001001000110100010101100111 <sub>B</sub> (01234567 <sub>H</sub> )	→	7
000000000000001111111111111111 <sub>B</sub> (0003FFFF <sub>H</sub> )	→	14
0000000000000000000000000000001 <sub>B</sub> (00000001 <sub>H</sub> )	→	31
000000000000000000000000000000 <sub>B</sub> (00000000 <sub>H</sub> )	→	32

### ● Change point detection

The data written to the change point detection data register is scanned from bit30 to the LSB and compared with the MSB. The position of the first bit with a value different to the MSB is returned.

The result is obtained by reading the detection result register.

Table 7.2-1 shows the returned value and detected position.

If no change point exists, "32" is returned.

The change point detection function never returns a result of "0".

[Execution example]

Write data		Read data (decimal)
00100000000000000000000000000000 <sub>B</sub> (20000000 <sub>H</sub> )	→	2
00000001001000110100010101100111 <sub>B</sub> (01234567 <sub>H</sub> )	→	7
00000000000000111111111111111111 <sub>B</sub> (0003FFFF <sub>H</sub> )	→	14
00000000000000000000000000000001 <sub>B</sub> (00000001 <sub>H</sub> )	→	31
00000000000000000000000000000000 <sub>B</sub> (00000000 <sub>H</sub> )	→	32
1111111111111111111110000000000000 <sub>B</sub> (FFFFFF00 <sub>H</sub> )	→	20
11111000010010011110000010101010 <sub>B</sub> (F849E0AA <sub>H</sub> )	→	5
10000000000000101010101010101010 <sub>B</sub> (8002AAAA <sub>H</sub> )	→	1
11111111111111111111111111111111 <sub>B</sub> (FFFFFFFF <sub>H</sub> )	→	32

**Table 7.2-1 Bit Position and Return Value (Decimal)**

Detected bit location	Return value	Detected bit location	Return value	Detected bit location	Return value	Detected bit location	Return value
31	0	23	8	15	16	7	24
30	1	22	9	14	17	6	25
29	2	21	10	13	18	5	26
28	3	20	11	12	19	4	27
27	4	19	12	11	20	3	28
26	5	18	13	10	21	2	29
25	6	17	14	9	22	1	30
24	7	16	15	8	23	0	31
						Not exist	32



## ■ Backup/Restore Processing

When it is necessary to backup and restore the internal state of the bit search module such as when using the bit search module in an interrupt handler, always use the following procedure.

- (1) Read the 1-detect data register and save the value. (backup)
- (2) Use the bit search module.
- (3) Write the data saved in step (1) to the 1-detect data register (restore).

This ensures that the value returned when the detection result register is next read will be based on the value written to the bit search module prior to step (1). This procedure will correctly restore the result, even if the data register written to previously was the 0-detect or change point detect data register.

# **CHAPTER 8**

---

## ***RELOAD TIMER***

**This chapter describes the overview of the reload timer, the configuration and functions of registers, and the reload timer operation.**

- 8.1 Overview
- 8.2 Block Diagram
- 8.3 Reload Timer Register
- 8.4 Operation of Reload Timer
- 8.5 Notes on Using the 16-bit Reload Timer

## 8.1 Overview

The 16-bit reload timer consists of a 16-bit down-counter, 16-bit reload register, internal count, clock generation prescaler, and control register.

The clock source can be selected from three internal clocks, namely, CPU clock, peripheral clock, and external bus clock (machine clock divided by 2, 8, 32, 64, or 128) or the external trigger.

MB91265A series has three internal timer channels.

No output is performed to the reload timer 0 pin.

### ■ Reload Timer Registers

#### Control Status Registers (TMCSR0 to TMCSR3)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
ch.0: 00004E <sub>H</sub>	—	—	—	CSL2	CSL1	CSL0	MOD2	MOD1	---00000 <sub>B</sub>
ch.1: 000056 <sub>H</sub>	—	—	—	R/W	R/W	R/W	R/W	R/W	
ch.2: 00005E <sub>H</sub>									
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
	MOD0	—	OUTL	RELD	INTE	UF	CNTE	TRG	00000000 <sub>B</sub>
	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	

#### 16-bit timer register (TMR0 to TMR3)

	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
ch.0: 00004A <sub>H</sub>									XXXXXXXX <sub>B</sub>
ch.1: 000052 <sub>H</sub>	R	R	R	R	R	R	R	R	
ch.2: 00005A <sub>H</sub>									
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
		—							XXXXXXXX <sub>B</sub>
	R	R	R	R	R	R	R	R	

#### 16-bit reload register (TMRLR0 to TMRLR3)

	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
ch.0: 000048 <sub>H</sub>									XXXXXXXX <sub>B</sub>
ch.1: 000050 <sub>H</sub>	W	W	W	W	W	W	W	W	
ch.2: 000058 <sub>H</sub>									
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
		—							XXXXXXXX <sub>B</sub>
	W	W	W	W	W	W	W	W	

R/W: Readable/Writable

R: Read only

W: Write only

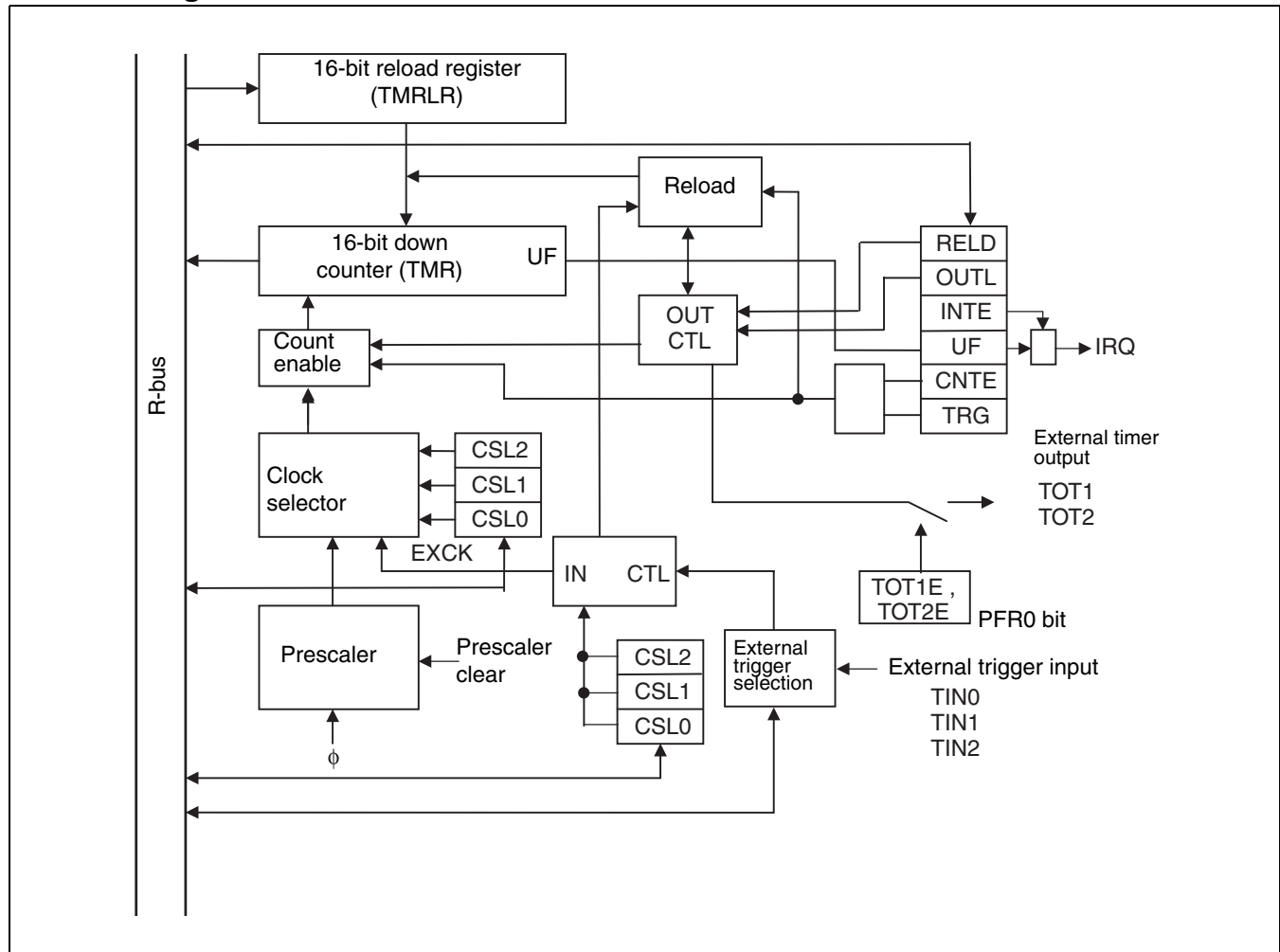
X: Undefined value

—: Undefined

## 8.2 Block Diagram

**This section shows the block diagram of the reload timer.**

### ■ Block Diagram of the Reload Timer



## 8.3 Reload Timer Register

This section describes the configuration and functions of registers used by the reload timer.

### ■ Control Status Register (TMCSR: TMCSR0 to TMCSR2)

Controls the 16-bit timer operation mode and interrupts.

Only modify the bits other than UF, CNTE, and TRG when CNTE = 0.

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
ch.0: 00004E <sub>H</sub>	–	–	–	CSL2	CSL1	CSL0	MOD2	MOD1	---00000 <sub>B</sub>
ch.1: 000056 <sub>H</sub>	–	–	–	R/W	R/W	R/W	R/W	R/W	
ch.2: 00005E <sub>H</sub>									
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
	MOD0	–	OUTL	RELD	INTE	UF	CNTE	TRG	00000000 <sub>B</sub>
	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable  
 R: Read only  
 –: Undefined

[bit12 to bit10] CSL2 to CSL0 (Count source select)

These are the count source selection bits. Either the external trigger or an internal clock can be set as the count source. The available count sources are as follows.

CSL2	CSL1	CSL0	Count Source ( $\phi$ : Machine clock)	$\phi=32\text{MHz}$	$\phi=16\text{MHz}$
0	0	0	Internal clock $\phi/2^1$ [Initial value]	62.5ns	125ns
0	0	1	Internal clock $\phi/2^3$	250ns	0.5 $\mu\text{s}$
0	1	0	Internal clock $\phi/2^5$	1.0 $\mu\text{s}$	2.0 $\mu\text{s}$
0	1	1	External trigger	-	-
1	0	0	Setting disabled	-	-
1	0	1	Internal clock $\phi/2^6$	2.0 $\mu\text{s}$	4.0 $\mu\text{s}$
1	1	0	Internal clock $\phi/2^7$	4.0 $\mu\text{s}$	8.0 $\mu\text{s}$
1	1	1	ch.1 timer output (only ch.2 can be set)	ch.1	ch.1

The MOD1 and MOD0 bits specify the active edge on which to count when the external trigger is set as the count source.

The minimum pulse width able to be used with the external trigger is  $2 \times T$  (T: machine clock cycle). CSL2, CSL1, CSL0 = 111<sub>B</sub> connects ch.1 + ch.2 in cascade. In this case, only the ch.2 register can be set. The ch.1 register is setting disabled.

**[bit9 to bit7] MOD2 to MOD0 (Mode)**

These bits select the operating mode. The function is different depending upon whether the count source is set to "internal clock" or "external trigger".

- In internal clock mode: Sets the reload trigger
- In external trigger mode: Sets the active edge on which to count

Always set MOD2 to "0".

**[Reload trigger setting when an internal clock is selected]**

When an internal clock is selected as the count source, the contents of the reload register are loaded and the count continues when the active edge specified by bits MOD2 to MOD 0 is input.

MOD2	MOD1	MOD0	Active edge
0	0	0	Software trigger [Initial value]
0	0	1	External trigger (rising edge)
0	1	0	External trigger (falling edge)
0	1	1	External trigger (both edges)
1	X	X	Setting disabled

**[Active edge setting when the external trigger is selected]**

If the external clock is selected as the count source, the counter counts on input of the active edge (specified by the MOD2 to MOD0 bits) to the external trigger.

MOD2	MOD1	MOD0	Active edge
X	0	0	-
X	0	1	External trigger (rising edge)
X	1	0	External trigger (falling edge)
X	1	1	External trigger (both edges)
X	X	X	Setting disabled

If the external trigger is selected, the reload is performed by a software trigger or when an underflow occurs.

**[bit6] Undefined bit.**

Reading always returns "0".

**[bit5] OUTL**

This bit sets the level of the external timer output. The output level is opposite when this bit is set to "0" and when set to "1".

**[bit4] RELD**

The reload enable bit. Setting "1" sets reload mode. In this case, the contents of the reload register are loaded to the counter and the counting continues when the counter value underflows from "0000<sub>H</sub>" to "FFFF<sub>H</sub>". Setting "0" sets one-shot mode. In this case, the counting halts when the counter value underflows from "0000<sub>H</sub>" to "FFFF<sub>H</sub>".

TOxE	OUTL	RELD	Output waveform
0	X	X	Output disabled
1	0	0	"H" square waveform during count
1	1	0	"L" square waveform during count
1	0	1	"L" toggle output when count starts
1	1	1	"H" toggle output when count starts

TOxE indicates the TO1E and TO2E in the PFR0 (Port Function Register).

Note: No output is performed to the reload timer 0 pin.

#### [bit3] INTE

The interrupt request enable bit. If this bit is set to "1", an interrupt request is generated when the UF bit is "1". No interrupt request is generated if this bit is set to "0".

#### [bit2] UF

The timer interrupt request flag. This bit is set to "1" when the counter underflows from "0000<sub>H</sub>" to "FFFF<sub>H</sub>". Writing "0" to this bit clears it.

Writing "1" to this bit has no meaning.

Read-modify-write instructions always read the bit as "1".

#### [bit1] CNTE

The count enable bit for the timer. Writing "1" to this bit sets the timer to wait for a start trigger. Writing "0" halts the count.

#### [bit0] TRG

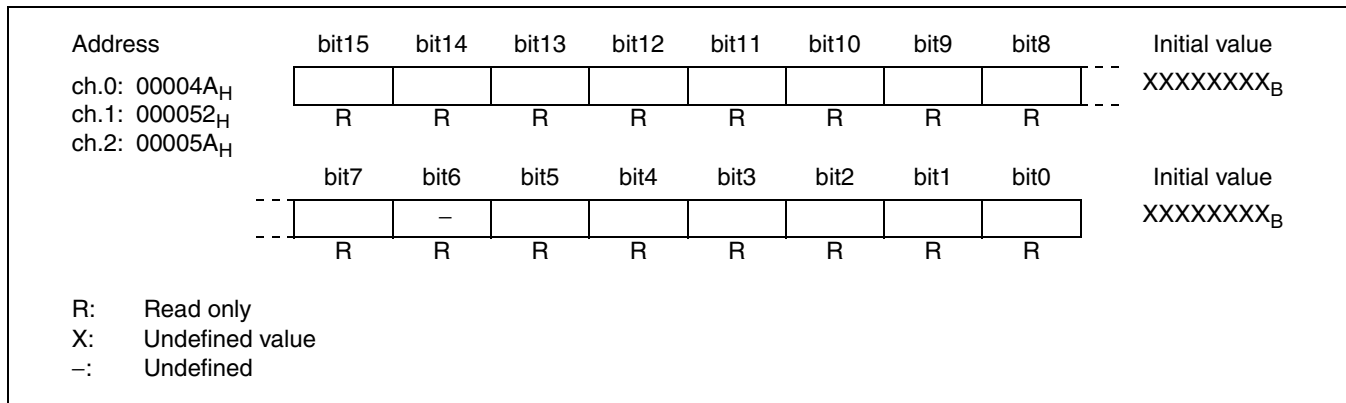
The software trigger bit. Writing "1" to this bit generates a software trigger which loads the contents of the reload register to the counter and starts the counting.

Writing "1" to this bit has no meaning. Reading value is always "0".

The trigger input in this register is only meaningful when CNTE=1. The trigger has no effect if CNTE=0.

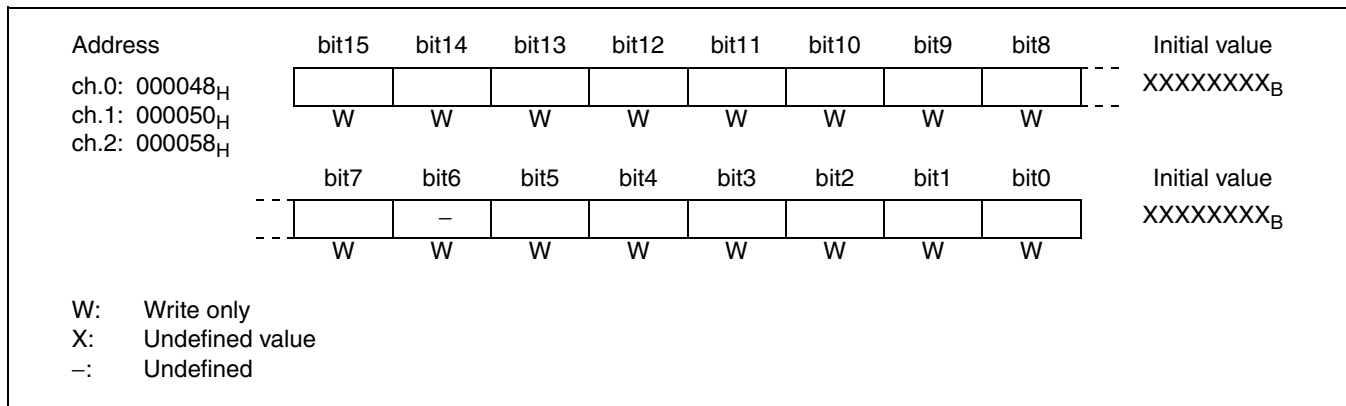
## ■ TMR Register (16-bit Timer Register)

The count value of the 16-bit timer can be read from this register. The initial value is undefined. Always use a 16-bit data transfer instruction to read this register.



## ■ TMRLR Register (16-bit Reload Register)

This register stores the initial value of the count. The initial value is undefined. Always use a 16-bit data transfer instruction to read this register.





## 8.4 Operation of Reload Timer

This section explains the internal clock operation and underflow operation of the reload timer.

### ■ Internal Clock Operation

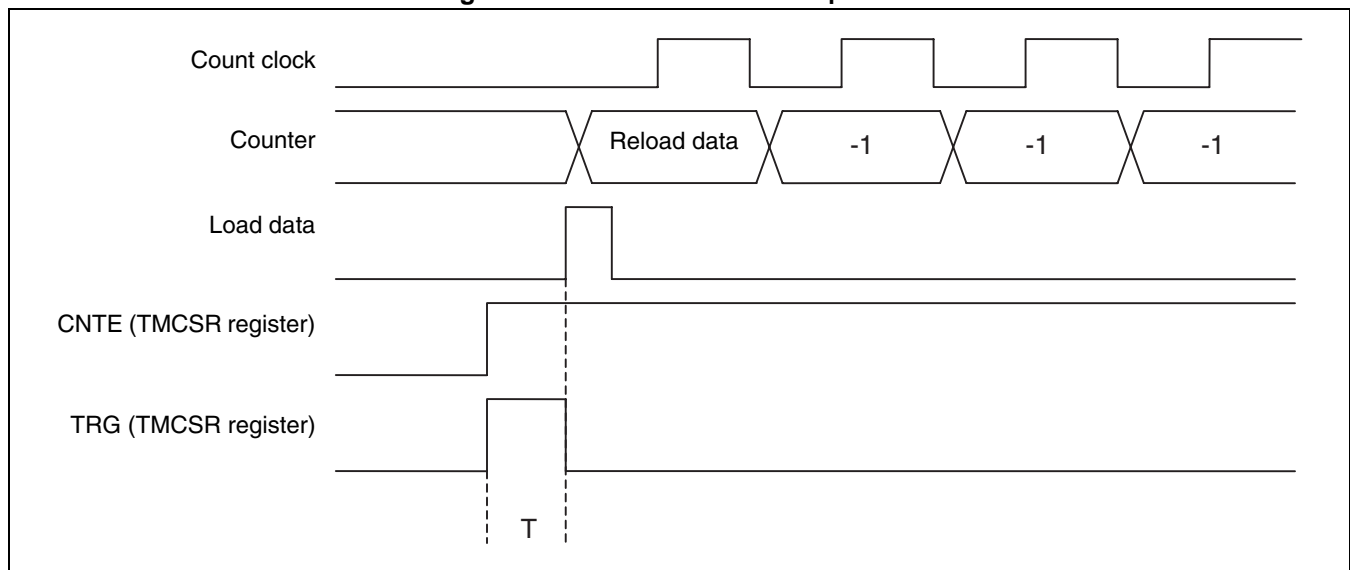
When the timer is driven by the divided internal clock, the clock source can be selected from the machine clock divided by 2, 8, 32, 64, or 128.

If you wish to enable and start the count at the same time, write "1" to both the CNTE bit and TRG bit in the control status register. The TRG bit trigger input always functions regardless of the operation mode, provided the timer is enabled (CNTE = 1).

Figure 8.4-1 shows counter start and counter operation.

The time between input of a count start trigger and the data in the reload register being loaded to the counter is T (T: machine cycle of peripheral clock).

**Figure 8.4-1 Counter Start and Operation**



## ■ Underflow Operation

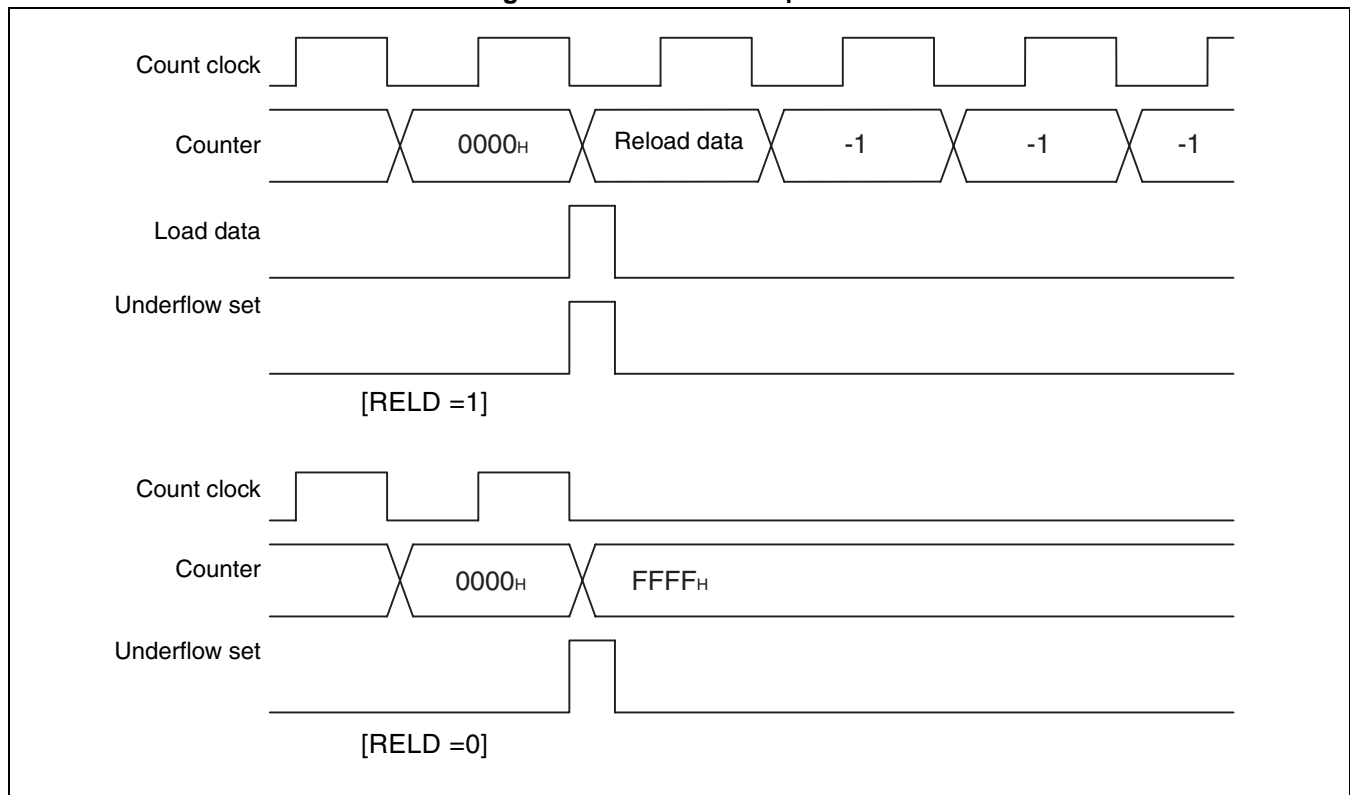
An underflow on this timer is defined as when the value of counter goes from "0000<sub>H</sub>" to "FFFF<sub>H</sub>". Accordingly, an underflow occurs after (reload register setting value + 1) counts.

If the RELD bit in the control status register is "1" when the underflow occurs, the contents of the reload register are loaded to the counter and the counting continues. If the RELD bit is "0", the counter halts at "FFFF<sub>H</sub>".

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

The figure below shows the underflow operation.

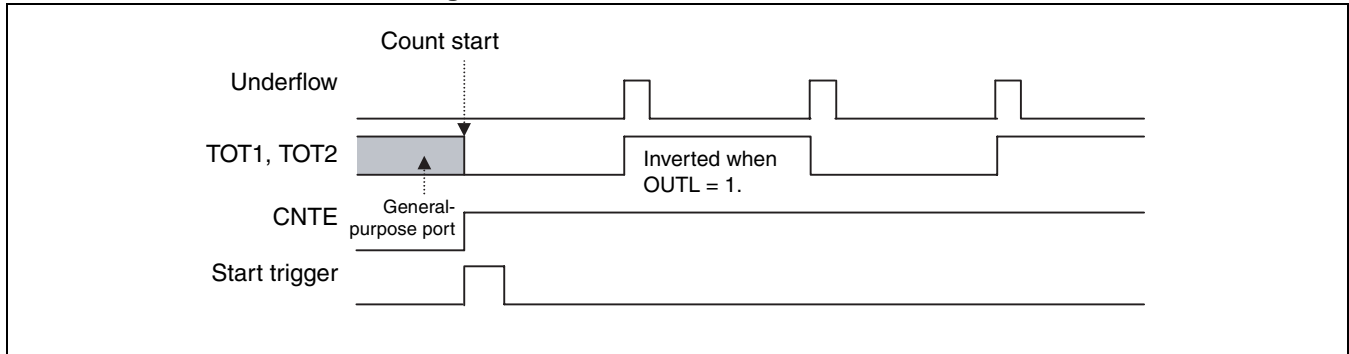
**Figure 8.4-2 Underflow Operation**



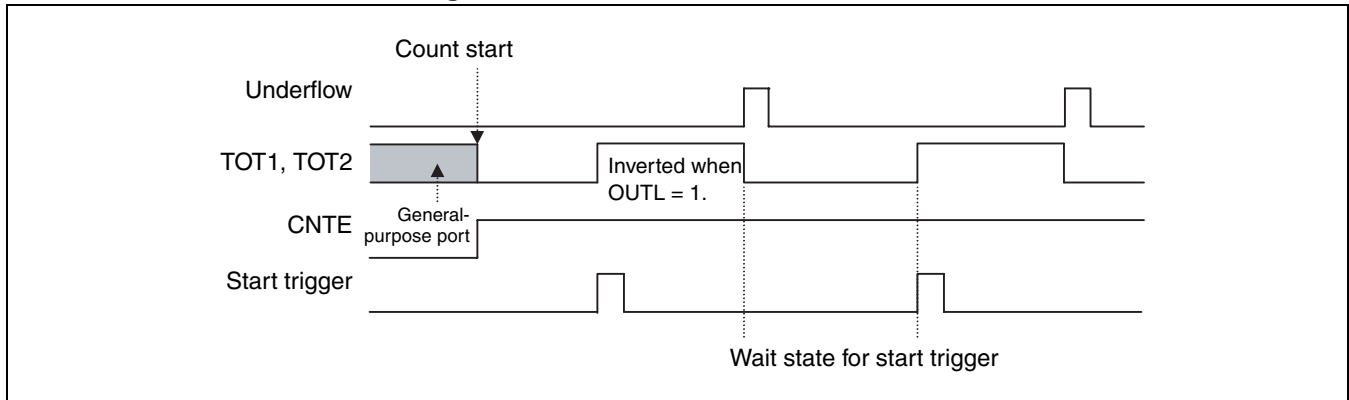
## ■ Output Pin Function

In reload mode, the TOT1, TOT2 output pins provide the toggle output inverted by the underflow operation. In one-shot mode, the pins function as pulse outputs that indicate when the count is in progress. The output polarity is set by the OUTL bit in the register. When OUTL = 0, the initial value of the toggle output is "0" and the one-shot pulse output outputs "1" while the count is in progress. The output waveform is inverted when OUTL = 1.

**Figure 8.4-3 Output Pin Function [RELD=1,OUTL=0]**



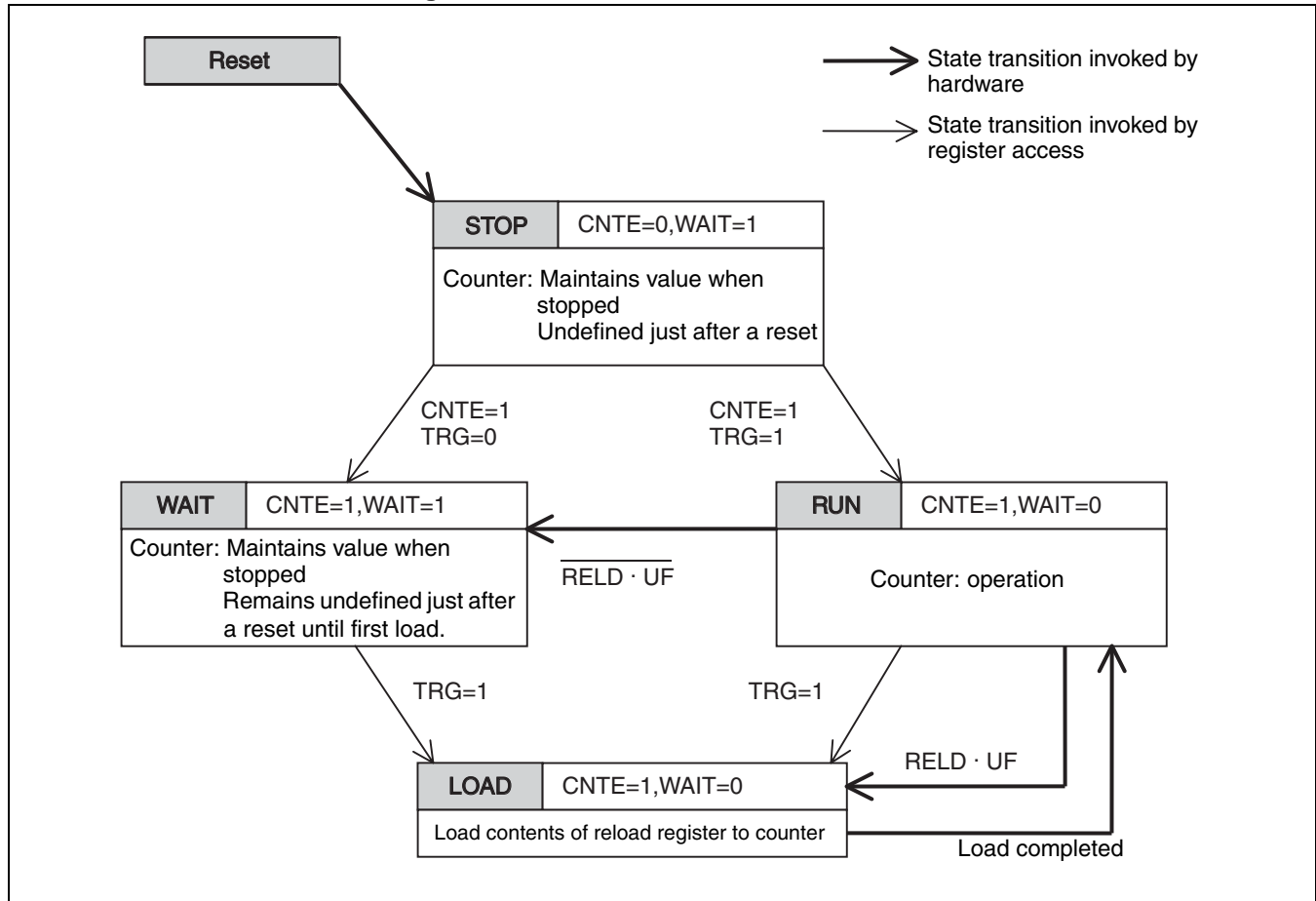
**Figure 8.4-4 Output Pin Function [RELD=0,OUTL=0]**



## ■ Counter Operation States

The counter state is determined by the CNTE bit in the control status register and by the internal WAIT signal. The states that can be set include the stop state, when CNTE=0 and WAIT=1 (STOP state); the startup trigger wait state, when CNTE=1 and WAIT=1 (WAIT state); and the operation state, when CNTE=1 and WAIT=0 (RUN state). Figure 8.4-5 shows the transitions between states.

**Figure 8.4-5 Status Transitions of Counter**



## 8.5 Notes on Using the 16-bit Reload Timer

---

**This section explains notes on using the reload timer.**

---

### ■ Notes

- Operation of the internal prescaler is enabled when a trigger (software trigger or external trigger) occurs while bit1 of the control status register (timer enable: CNTE) is set to "1".
- If the interrupt request flag is set and cleared at the same time, the flag set operation has precedence and the clear operation is ignored.
- If writing to the 16-bit reload register occurs at the same time as a reload timing, the old data and the new data are loaded to the counter at next reload timing.
- If a 16-bit timer register load occurs at the same time as a count, the load (reload) operation has precedence.

# **CHAPTER 9**

---

# ***TIMING GENERATOR***

**This chapter explains the overview of the timing generator, the configuration and functions of registers, and operation of the timing generator.**

## **9.1 Timing Generator**

## 9.1 Timing Generator

The timing generator has a function to activate the delay synchronizing multiple PPG timers between timers.

### ■ Register List

#### Control Register: TTCR

Address	bit31	bit30	bit29	bit28	bit27	bit26	bit25	bit24	Initial value
000144 <sub>H</sub>	TRG60	TRG40	TRG20	TRG00	CS1	CS0	MONI	STR	11110000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

#### Test Register: TSTPR (writing is disabled, read value has no meaning)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
000147 <sub>H</sub>	—	—	—	—	—	—	—	—	00000000 <sub>B</sub>
	R	R	R	R	R	R	R	R	

#### Compare Register 0: COMP0

Address	bit31	bit30	bit29	bit28	bit27	bit26	bit25	bit24	Initial value
000148 <sub>H</sub>	D7	D6	D5	D4	D3	D2	D1	D0	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

#### Compare Register 2: COMP2

Address	bit23	bit22	bit21	bit20	bit19	bit18	bit17	bit16	Initial value
000149 <sub>H</sub>	D7	D6	D5	D4	D3	D2	D1	D0	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

#### Compare Register 4: COMP4

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
00014A <sub>H</sub>	D7	D6	D5	D4	D3	D2	D1	D0	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

#### Compare Register 6: COMP6

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00014B <sub>H</sub>	D7	D6	D5	D4	D3	D2	D1	D0	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

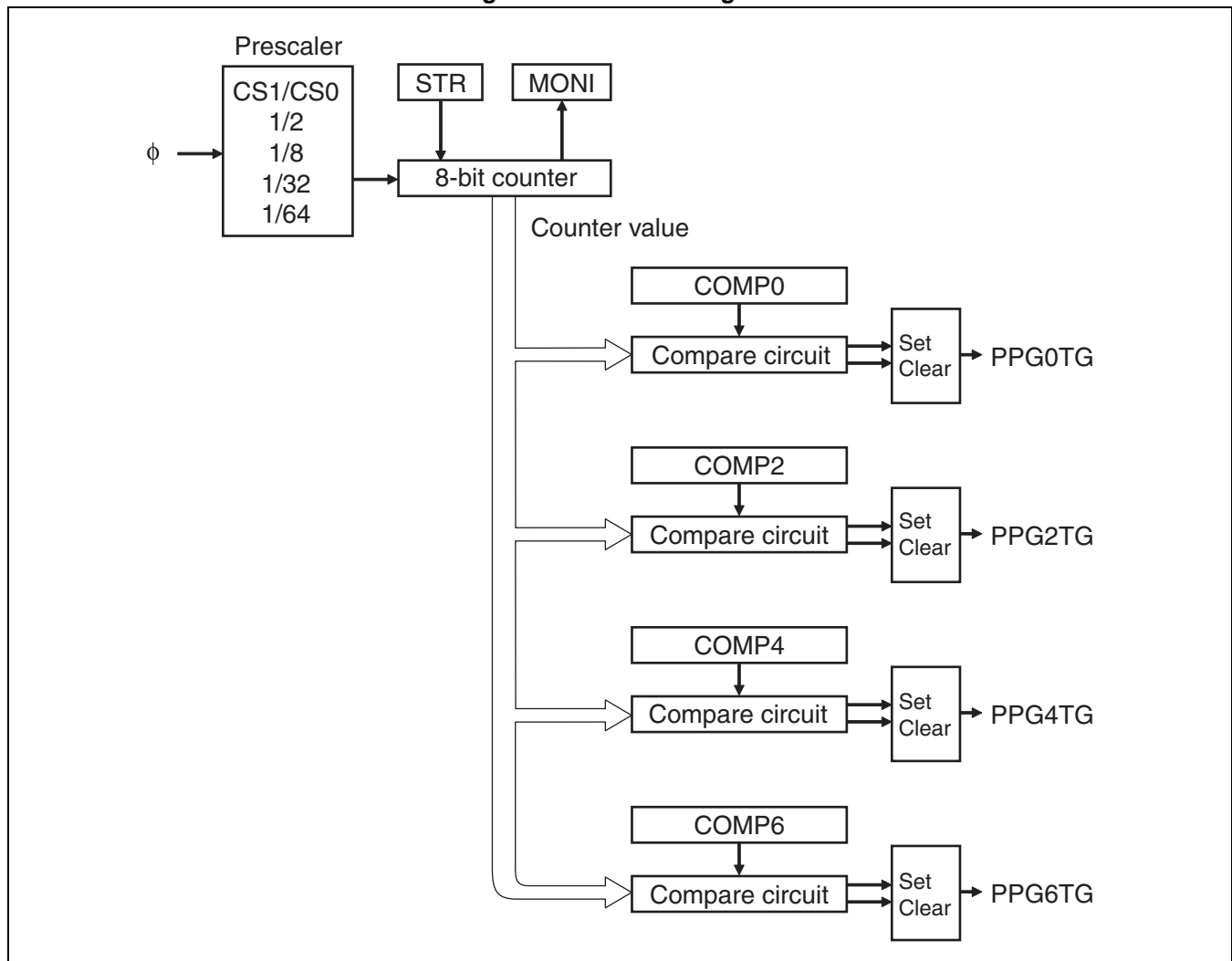
R/W: Readable/Writable

R: Read only

—: Undefined

## ■ Block Diagram of Timing Generator

Figure 9.1-1 Block Diagram





## ■ Registers of Timing Generator

### ● TTCR (Control Register)

Address	bit31	bit30	bit29	bit28	bit27	bit26	bit25	bit24	Initial value
000144 <sub>H</sub>	TRG60	TRG40	TRG20	TRG00	CS1	CS0	MONI	STR	11110000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
R/W: Readable/Writable									

Timing generator control register.

[bit31 to bit28] TRG60/TRG40/TRG20/TRG00 (PPG Trigger Clear bit): PPG trigger clear bits

Writing "0" to these bits clears the PPG start trigger to be output. The correspondence with trigger of the bits is shown below.

TRG00: PPG0TG

TRG20: PPG2TG

TRG40: PPG4TG

TRG60: PPG6TG

Read values of this register are always "1".

[bit27, bit26] CS1/CS0 (Count Select bit): Count clock selection bits

The operation clock of the 8-bit counter is selected as follows:

CS1	CS0	Clock source
0	0	Machine clock / 2 (62.5ns @32MHz)
0	1	Machine clock / 8 (250ns @32MHz)
1	0	Machine clock / 32 (1μs @32MHz)
1	1	Machine clock / 64 (2μs @32MHz)

[bit25] MONI (MONITER bit): 8-bit counter operating monitor bit

The operation of the 8-bit counter is selected as follows:

0	Stopping counter
1	Operation counter

Writing value has no meaning.

[bit24] STR (START bit): 8-bit counter operation enable bit

The operation of the 8-bit counter is selected as follows:

0	Has no meaning
1	Start counter operation

Writing "0" has no meaning. Read value is always "0".

## Compare Register: COMP0, COMP2, COMP4, COMP6

Address	bit31	bit30	bit29	bit28	bit27	bit26	bit25	bit24	Initial value
000148 <sub>H</sub>	D7	D6	D5	D4	D3	D2	D1	D0	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Address	bit23	bit22	bit21	bit20	bit19	bit18	bit17	bit16	Initial value
000149 <sub>H</sub>	D7	D6	D5	D4	D3	D2	D1	D0	00000000 <sub>B</sub>
	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	Initial value
00014A <sub>H</sub>	D7	D6	D5	D4	D3	D2	D1	D0	00000000 <sub>B</sub>
	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	Initial value
00014B <sub>H</sub>	D7	D6	D5	D4	D3	D2	D1	D0	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable

When the 8-bit counter matches with the value of this register, the PPG start signal is set. Do not rewrite this register during the count operation.

When the value of this register is "00000000<sub>B</sub>", the PPG start signal is not set.

## ■ Operation Overview of Timing Generator

### ● Operation of prescaler

This operation sets clock that is the count clock for the 8-bit counter divided by the machine clock.

### ● 8-bit counter

The 8-bit counter starts counting by the count clock from the prescaler due to the STR bit. The 8-bit counter starts counting up and stops the counting with overflow. Starting counter during counting is ignored.

"1" is read to the MONI bit while the 8-bit counter is counting. When stopped, "0" is read.

The count value of the 8-bit counter is input to comparators.

**Figure 9.1-2 Operation/stop Timing of 8-bit Counter**

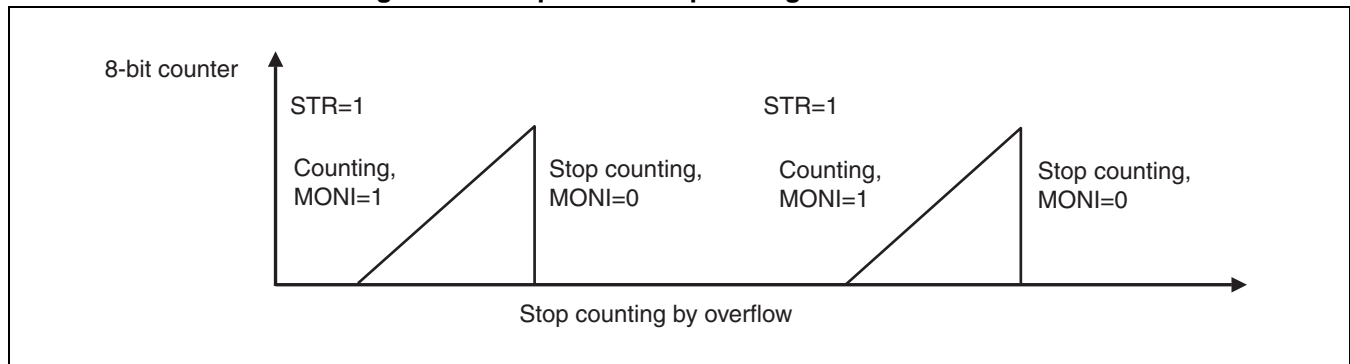
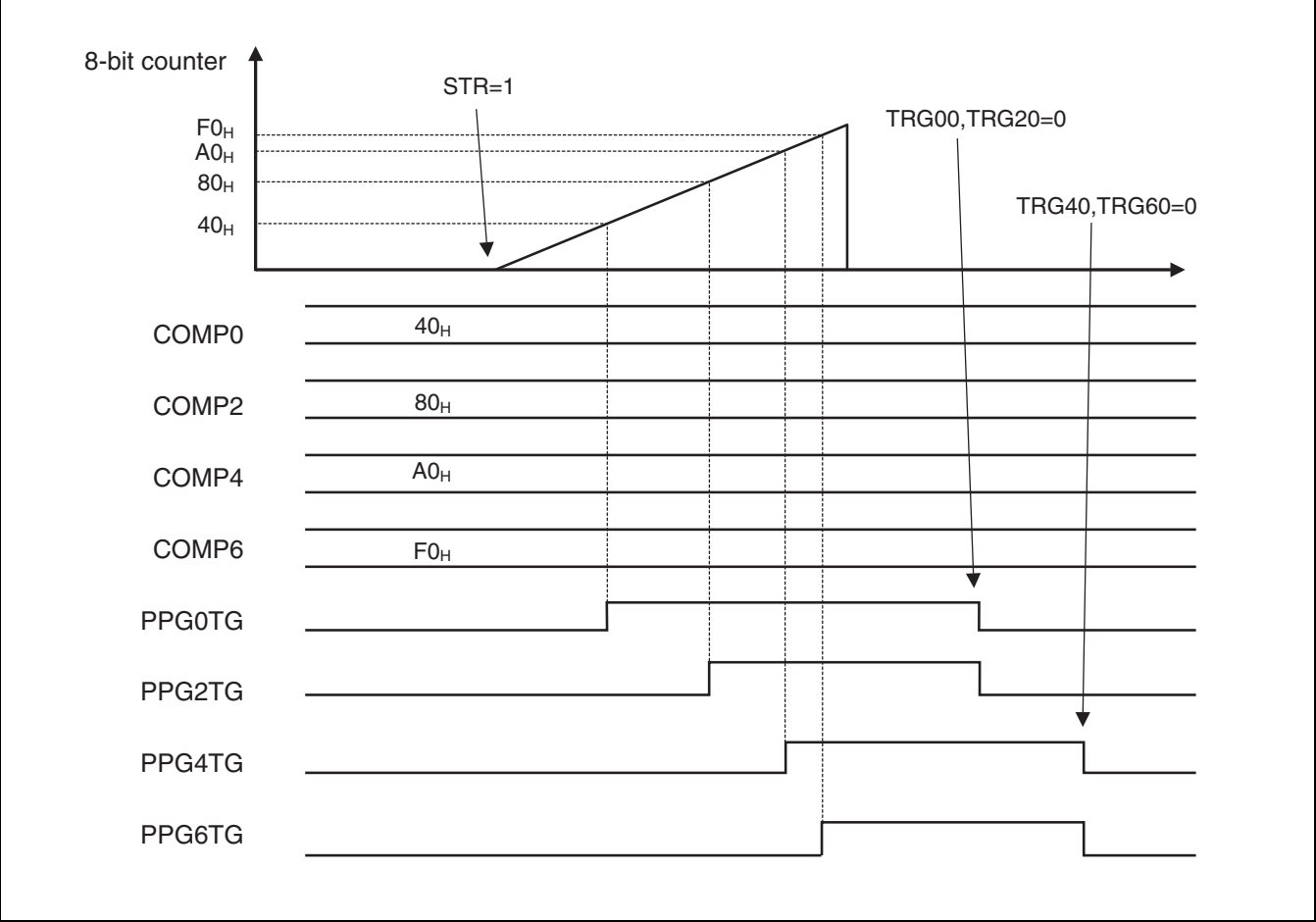


Figure 9.1-3 Trigger Timing



# **CHAPTER 10**

---

## **PPG**

**This chapter explains the overview of the PPG, the configuration and functions of registers, and the PPG operation.**

- 10.1 Overview
- 10.2 Block Diagram
- 10.3 Register of PPG
- 10.4 Operation Explanation

## 10.1 Overview

---

The PPG is an 8-bit reload timer module that can be used as a PPG output to output pulses controlled by the timer operation.

The hardware consists of  $8 \times 8$ -bit down counters,  $16 \times 8$ -bit reload registers, control register,  $8 \times$  external pulse outputs, and  $8 \times$  interrupt output.

MB91265A series has 8 channels for 8-bit PPG and 4 channels for 16-bit PPG.

---

### ■ Functions of the PPG

- 8-bit PPG output independent operation mode  
Can operate as an independent PPG output.
- 16-bit PPG output operation mode  
1 channel 16-bit PPG output can be operated.
- 8 + 8-bit PPG output operation mode  
With setting the  $ch.(n + 1)$  output as the  $ch.(n)$  clock input, the 8-bit PPG output in any cycle can be operated ( $n=0, 2, 4, 6$ ).
- 16 + 16-bit PPG output operation mode  
This mode sets the 16-bit prescaler output,  $ch.(n + 3) + ch.(n + 2)$  as a clock input for the 16-bit PPG,  $ch.(n + 1) + ch.(n)$  ( $n=0, 4$ ).
- PPG output operation  
Outputs a pulse waveform with arbitrary period and duty ratio.  
Can also be used in conjunction with an external circuit to form a D/A converter.
- Output inversion function  
The PPG output value can be inverted.

## ■ Registers of the PPG Timer

### PPG start register (TRG)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
000130 <sub>H</sub>	–	–	–	–	–	–	–	–	00000000 <sub>B</sub>
	–	–	–	–	–	–	–	–	
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
	PEN7	PEN6	PEN5	PEN4	PEN3	PEN2	PEN1	PEN0	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

### Output inversion register (REVC)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
000134 <sub>H</sub>	–	–	–	–	–	–	–	–	00000000 <sub>B</sub>
	–	–	–	–	–	–	–	–	
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
	REV7	REV6	REV5	REV4	REV3	REV2	REV1	REV0	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

### GATE function control register (GATEC)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
000133 <sub>H</sub>	–	–	–	–	–	–	STGR	EDGE	XXXXXX00 <sub>B</sub>
	–	–	–	–	–	–	R/W	R/W	

### PPG0 to PPG7 operation mode control register (PPGC0 to PPGC7)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
ch.0: 000108 <sub>H</sub>	PIEn	PUFn	INTMn	PCS1	PCS0	MD1*	MD0*	TTRGn	00000000 <sub>B</sub>
ch.1: 000109 <sub>H</sub>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

ch.2: 00010A<sub>H</sub>ch.3: 00010B<sub>H</sub>ch.4: 000114<sub>H</sub>ch.5: 000115<sub>H</sub>ch.6: 000116<sub>H</sub>ch.7: 000117<sub>H</sub>

n=0 to 7

R/W: Readable/Writable

X: Undefined value

–: Undefined

\*: MD1 and MD0 only exist for even-numbered channels and not for odd-numbered channels.

The initial value for odd-numbered channels is undefined. Writing has no effect.

● Reload register: 8-bit PPG mode

Reload register H (PRLH0 to PRLH7)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
ch.0: 000100 <sub>H</sub>									XXXXXXXX <sub>B</sub>
ch.1: 000102 <sub>H</sub>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
ch.2: 000104 <sub>H</sub>									
ch.3: 000106 <sub>H</sub>									
ch.4: 00010C <sub>H</sub>									
ch.5: 00010E <sub>H</sub>									
ch.6: 000110 <sub>H</sub>									
ch.7: 000112 <sub>H</sub>									

Reload register L (PRL0 to PRL7)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
ch.0: 000101 <sub>H</sub>									XXXXXXXX <sub>B</sub>
ch.1: 000103 <sub>H</sub>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
ch.2: 000105 <sub>H</sub>									
ch.3: 000107 <sub>H</sub>									
ch.4: 00010D <sub>H</sub>									
ch.5: 00010F <sub>H</sub>									
ch.6: 000111 <sub>H</sub>									
ch.7: 000113 <sub>H</sub>									

R/W: Readable/Writable

X: Undefined

● Reload register: 16-bit PPG mode

Reload register H (PRLH0, PRLH2, PRLH4, PRLH6)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
ch.0: 000100 <sub>H</sub>									XXXXXXXX <sub>B</sub>
ch.2: 000104 <sub>H</sub>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
ch.4: 00010C <sub>H</sub>									
ch.6: 000110 <sub>H</sub>									
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
									XXXXXXXX <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Reload Registers L (PRL0, PRL2, PRL4, PRL6)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
ch.0: 000102 <sub>H</sub>									XXXXXXXX <sub>B</sub>
ch.2: 000106 <sub>H</sub>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
ch.4: 00010E <sub>H</sub>									
ch.6: 000112 <sub>H</sub>									
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
									XXXXXXXX <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

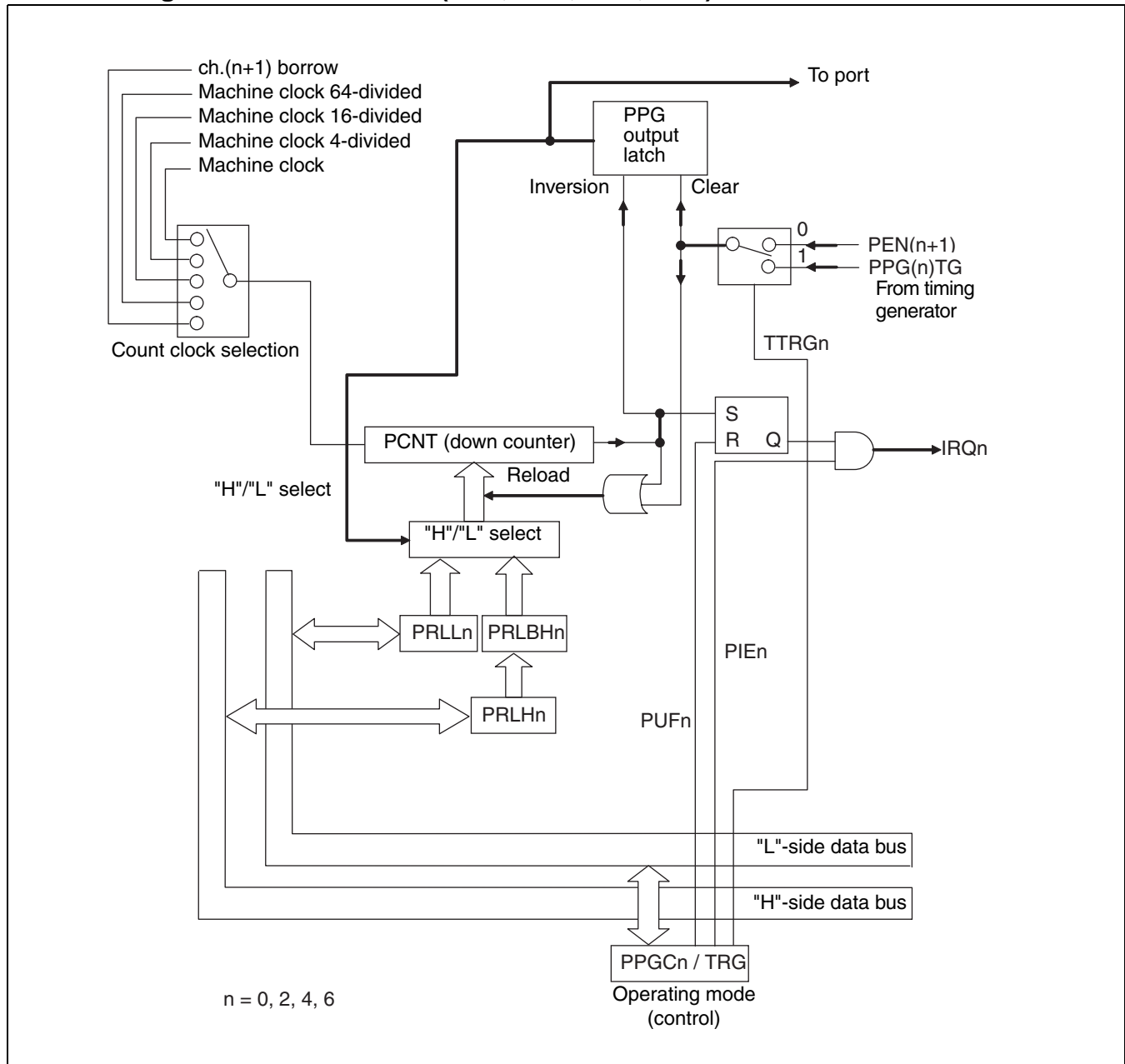
R/W: Readable/Writable

X: Undefined

## 10.2 Block Diagram

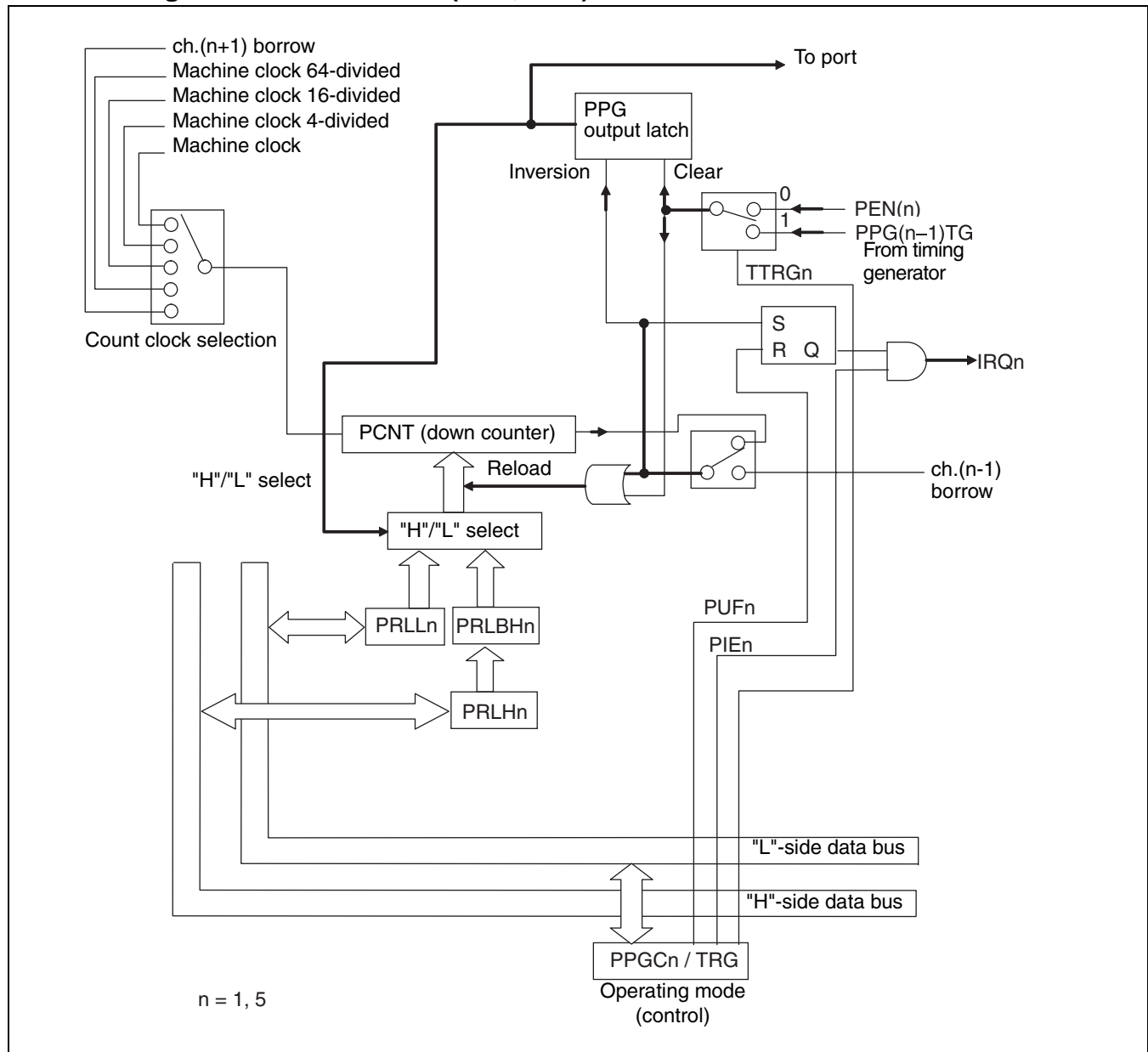
**This section shows the block diagram of the PPG.**

### ■ Block Diagram of the 8-bit PPG (ch.0, ch.2, ch.4, ch.6)

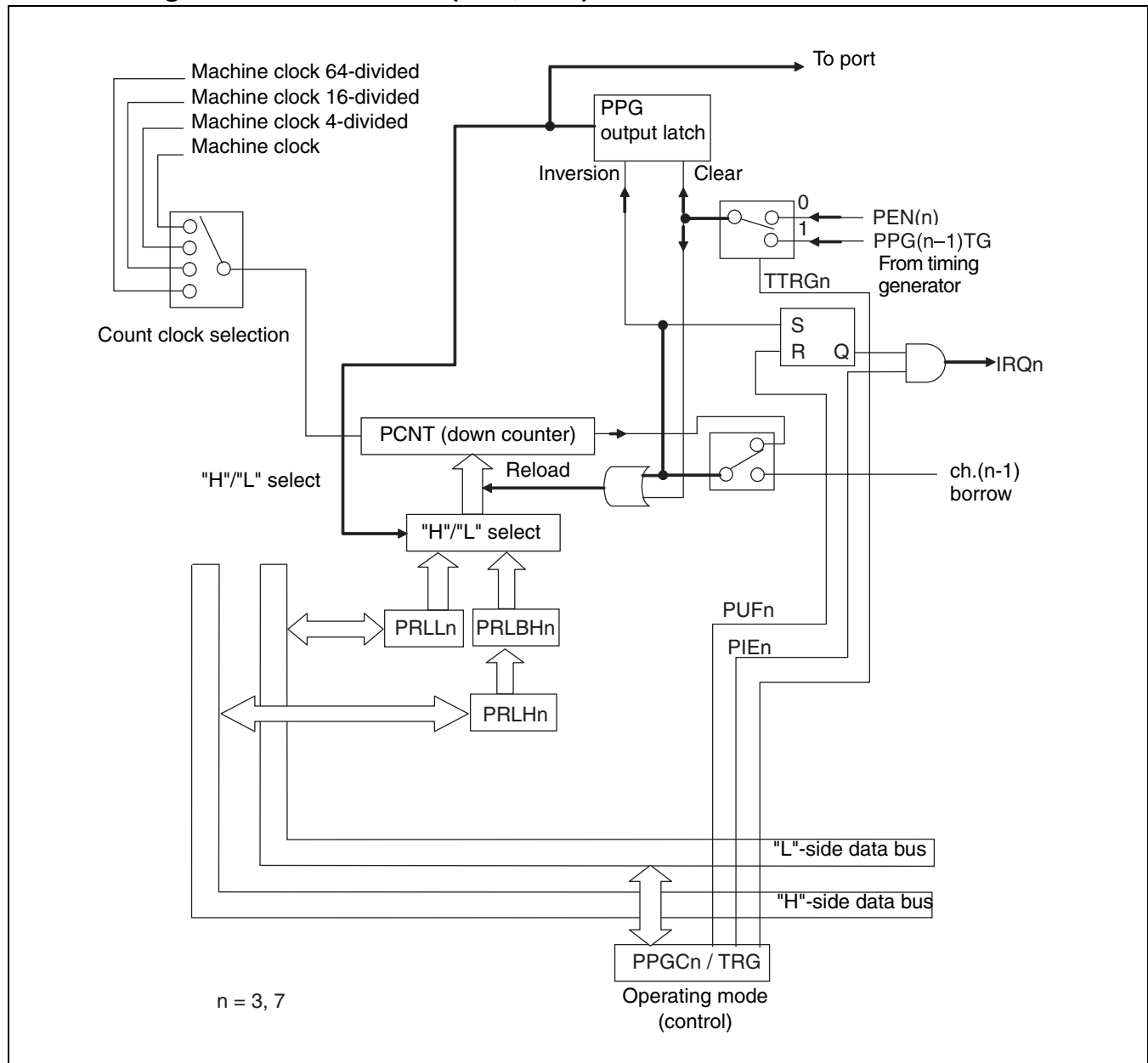




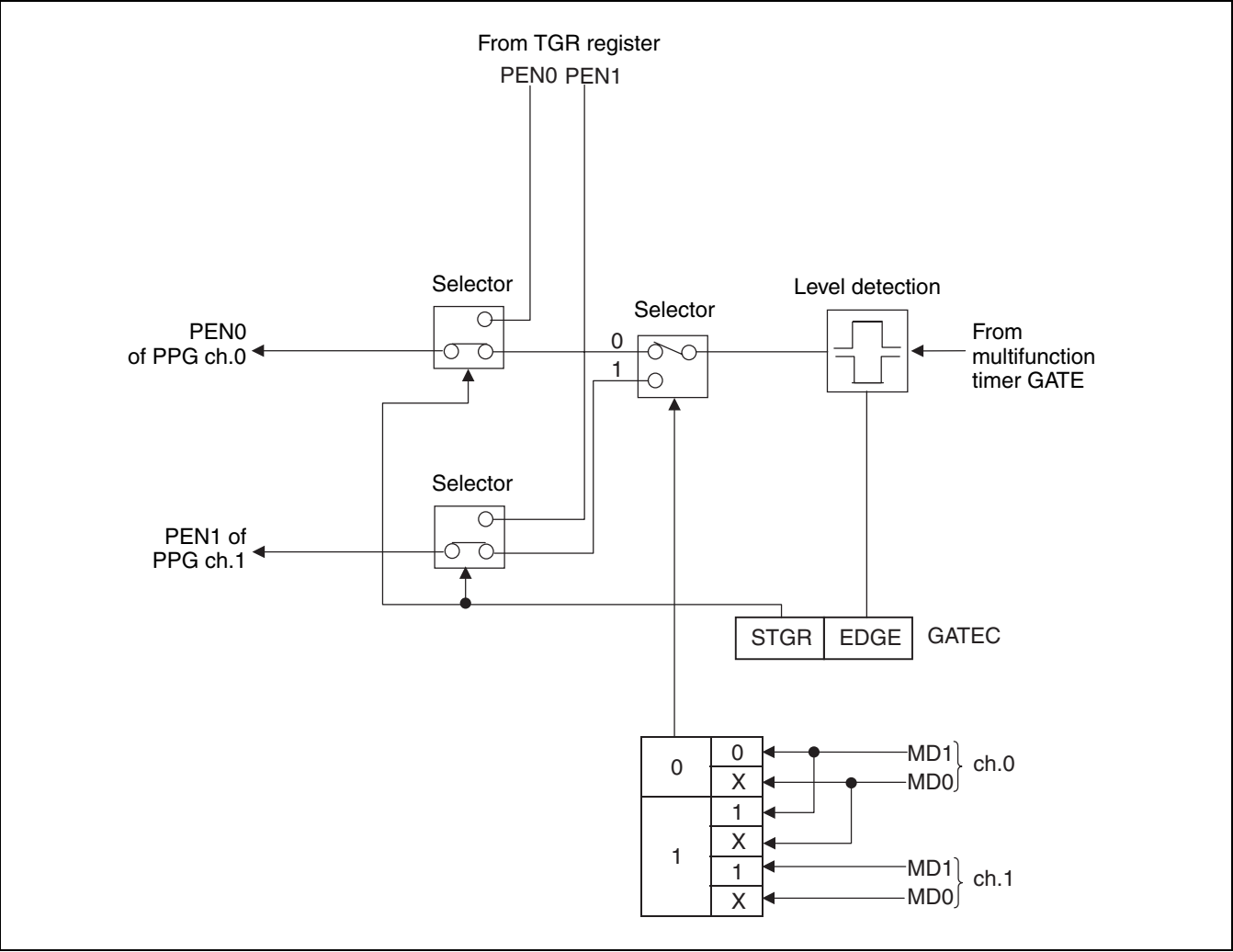
# ■ Block Diagram of the 8-bit PPG (ch.1, ch.5)



### ■ Block Diagram of the 8-bit PPG (ch.3, ch.7)



■ Block Diagram of Gate Function



## 10.3 Register of PPG

This section describes the PPG registers.

### ■ PPGCn Register (PPGn Operation Mode Control Register) n=0,1,2,3,4,5,6,7

PPG0 to PPG7 operation mode control register (PPGCn) n=0 to 7

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
ch.0: 000108 <sub>H</sub>	PIEn	PUFn	INTMn	PCS1	PCS0	MD1	MD0	TTRGn	XXXXXXXX <sub>B</sub>
ch.1: 000109 <sub>H</sub>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
ch.2: 00010A <sub>H</sub>									
ch.3: 00010B <sub>H</sub>									
ch.4: 000114 <sub>H</sub>									
ch.5: 000115 <sub>H</sub>									
ch.6: 000116 <sub>H</sub>									
ch.7: 000117 <sub>H</sub>									

R/W: Readable/Writable

X: Undefined

[bit7] PIEn (Ppg Interrupt Enable) : PPG interrupt enable bit

This bit controls to enable a PPG interrupt as described below.

0	Interrupt is disabled.
1	Interrupt is enabled.

- If this bit is set to "1", an interrupt request is generated when the PUFn is set to "1".
- If this bit is set to "0", no interrupt request is generated.
- Initialized to "0" by reset.
- Reading and writing are permitted.

[bit6] PUFn (Ppg Underflow Flag) : PPG counter underflow bit

The PPG counter underflow bit is controlled as follows.

0	PPG counter underflow has not been detected.
1	PPG counter underflow has been detected.

- In 8-bit PPG 2 channel mode and 8-bit prescaler + 8-bit PPG mode, the bit is set to "1" when the ch.0 count value underflows from "00<sub>H</sub>" to "FF<sub>H</sub>".
- In 16-bit PPG 1 channel mode, the bit is set to "1" when the ch.1/ch.0 count value underflows from "0000<sub>H</sub>" to "FFFF<sub>H</sub>".
- Writing "0" clears the bit to "0".
- Writing "1" to this bit is meaningless.
- When this bit is read to a read-modify-write instruction, "1" is always read.
- Initialized to "0" by reset.
- Reading and writing are permitted.

**[bit5] INTMn (Interrupt Mode) : Interrupt mode bit**

Detection of the PUFn bit can be restricted to the PRLBHn underflow timing only.

0	PUFn is set to "1" at an underflow.
1	PUFn is set to "1" at an underflow only from PRLBHn.

- Initialized to "0" by reset.
- Reading and writing are permitted.
- If this bit is set to "1", an interrupt is enabled at one cycle output of PPG waveform.
- Do not rewrite this bit when the interrupt is allowed.

**[bit4, bit3] PCS1, PCS0 (Ppg Count Select) : Count clock select bits**

These bits are used to select the down counter operating clock as shown below.

PCS1	PCS0	Operating mode
0	0	Machine clock (62.5 ns machine clock at 16MHz)
0	1	Machine clock /4 (250 ns machine clock at 16MHz)
1	0	Machine clock /6 (1 $\mu$ s machine clock at 16MHz)
1	1	Machine clock /64 (4 $\mu$ s machine clock at 16MHz)

- Initialized to "00<sub>B</sub>" by reset.
- Reading and writing are permitted.

**[bit2, bit1] MD1, MD0 (ppg count MoDe) : Operation mode select bits**

These bits are used to select the PPG timer operation mode as shown below.

MD1	MD0	Operating mode
0	0	8-bit PPG 2 channels independent mode
0	1	8-bit prescaler + 8-bit PPG mode
1	0	16-bit PPG mode
1	1	16-bit prescaler + 16-bit PPG mode

- Initialized to "00<sub>B</sub>" by reset.
- Reading and writing are permitted.
- These bits exist only in even-numbered channels.

**[bit0] TTRGn (Timing TRGer) : Timing trigger selection bit**

The PPG can only be started by a start signal from the timing generator.

0	Started by TRG register or multifunction timer
1	Only started by timing generator

- Initialized to "00<sub>B</sub>" by reset.
- Reading and writing are permitted.

## ■ PRL/PRLH Register (Reload Register: PRL0 to PRL7/PRLH0 to PRLH7)

### Reload register H (PRLH0 to PRLH7)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
ch.0: 000100 <sub>H</sub>									XXXXXXXX <sub>B</sub>
ch.1: 000102 <sub>H</sub>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
ch.2: 000104 <sub>H</sub>									
ch.3: 000106 <sub>H</sub>									
ch.4: 00010C <sub>H</sub>									
ch.5: 00010E <sub>H</sub>									
ch.6: 000110 <sub>H</sub>									
ch.7: 000112 <sub>H</sub>									

### Reload register L (PRL0 to PRL7)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
ch.0: 000101 <sub>H</sub>									XXXXXXXX <sub>B</sub>
ch.1: 000103 <sub>H</sub>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
ch.2: 000105 <sub>H</sub>									
ch.3: 000107 <sub>H</sub>									
ch.4: 00010D <sub>H</sub>									
ch.5: 00010F <sub>H</sub>									
ch.6: 000111 <sub>H</sub>									
ch.7: 000113 <sub>H</sub>									

R/W: Readable/Writable  
X: Undefined

These registers hold reload values for a down counter PCNT. Each register has own role as shown below.

Register Name	Function
PRL	Maintains reload value of "L"-side
PRLH	Maintains reload value of "H"-side

Reading and writing are permitted.

#### Note:

In the 8-bit prescaler + 8-bit PPG mode and 16-bit prescaler + 16-bit PPG mode, setting the same value to the PRL and PRLH in the prescaler side is recommended because the PPG waveform may be different for each cycle when the different value is set to the PRL and PRLH in the prescaler side.

## ■ PPG Start Register (TRG)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
000130 <sub>H</sub>	–	–	–	–	–	–	–	–	00000000 <sub>B</sub>
	–	–	–	–	–	–	–	–	
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
	PEN7	PEN6	PEN5	PEN4	PEN3	PEN2	PEN1	PEN0	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable  
–: Undefined

[bit7 to bit0] PEN7 to PEN0 (Ppg ENable) : PPG Operation Enable Bits

These bits are used to select the PPG operation start and the operation mode as shown below.

PEN7 to PEN0	Operation state
0	Operation stopped ("L" level output held)
1	PPG operation enabled

- Initialized to "0" by reset.
- Reading and writing are permitted.

## ■ Output Inversion Register (REVC)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
000134 <sub>H</sub>	–	–	–	–	–	–	–	–	00000000 <sub>B</sub>
	–	–	–	–	–	–	–	–	
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
	REV7	REV6	REV5	REV4	REV3	REV2	REV1	REV0	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable  
–: Undefined

[bit7 to bit0] REV7 to REV0: Output inversion bits

The PPG output values including the initial levels are inverted.

REV7 to REV0	Output level
0	Normal
1	Inverted

- Initialized to "0" by reset.
- Reading and writing are permitted.
- Since these bits invert the PPG outputs simply, they also invert the initial levels.  
The relationship between the reload register "L" and "H" is reversed as well.

## ■ GATE Function Control Register (GATEC)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
000133 <sub>H</sub>	—	—	—	—	—	—	STGR	EDGE	XXXXXX00 <sub>B</sub>
	—	—	—	—	—	—	R/W	R/W	

R/W: Readable/Writable  
 X: Undefined value  
 —: Undefined

### [bit1] STGR: Gate function select bit

Whether to start operation using the TRG register or using the start signal from the multifunction timer is selected as follows.

STGR	Operating mode
0	Start by TRG register
1	Start by start signal from multifunction timer

- Initialized to "0" by reset.
- Reading and writing are permitted.

### [bit0] EDGE: Start valid edge select bit

This bit is used to select the start valid edge from the multifunction timer as shown below.

EDGE	Operating mode
0	Start on rising edge → stop on falling edge <sup>*1</sup>
1	Start on falling edge → stop on rising edge <sup>*2</sup>

- Initialized to "0" by reset.
- Reading and writing are permitted.

\*1 : Starts at "H" period.

\*2 : Starts at "L" period.



## 10.4 Operation Explanation



The PPG consists of eight 8-bit PPG units and can be used in four different modes. In addition to independent mode, these consist of the 8-bit prescaler + 8-bit PPG mode, 16-bit PPG 1 channel mode, and 16-bit prescaler + 16-bit PPG mode in which PPGs are linked together.

### ■ Operation Explanation

Each of 8-bit length PPG units has two 8-bit-length reload registers for the "L" and "H" sides (PRL, PRLH). The "L"-side and "H"-side values written to this register are alternately reloaded to the 8-bit down-counter (PCNT) which counts down on each cycle of the count clock. The value of the pin output (PPG) is toggled each time a counter borrow occurs to trigger another reload. With this operation, the pin output (PPG) becomes a pulse output, which has "L"/"H" width corresponding to the value of reload register.

The operation starts/restarts when the bit of the register is written.

The relationship between the reload operation and the pulse output is shown below.

Reload operation	Pin output change
PRLH → PCNT	PPGn [0 → 1] 
PRL → PCNT	PPGn [1 → 0] 

n=0 to 7

When bit7 (PIEn) of the PPGCn register is "1", an interrupt request is output when the counter goes from "00<sub>H</sub>" to "FF<sub>H</sub>" causing a borrow (or a borrow from "0000<sub>H</sub>" to "FFFF<sub>H</sub>" in 16-bit PPG mode).

### ● Operation modes

There are four operation modes: independent mode, 8-bit prescaler + 8-bit PPG mode, 16-bit PPG 1 channel mode, and 16-bit prescaler + 16-bit PPG mode.

- In the independent mode, a channel can operate as 8-bit PPG independently. The PPG output of ch.(n) is connected to PPGn pin (n = 0 to 7).
- The 8-bit prescaler + 8-bit PPG mode makes 1 channel operate as an 8-bit prescaler, counts its borrow output, and then allows the 8-bit PPG waveform in any cycle to be output. For example, the prescaler output of ch.1 is connected to the PPG1 pin; the PPG output of ch.0 is connected to the PPG0 pin.
- In the 16-bit PPG 1 channel mode, two channels are combined, and the combined channel operates as 16-bit PPG. For example, if ch.0 and ch.1 are combined, 16-bit PPG outputs are connected to both PPG0 pin and PPG1 pin.

### ● PPG output operation

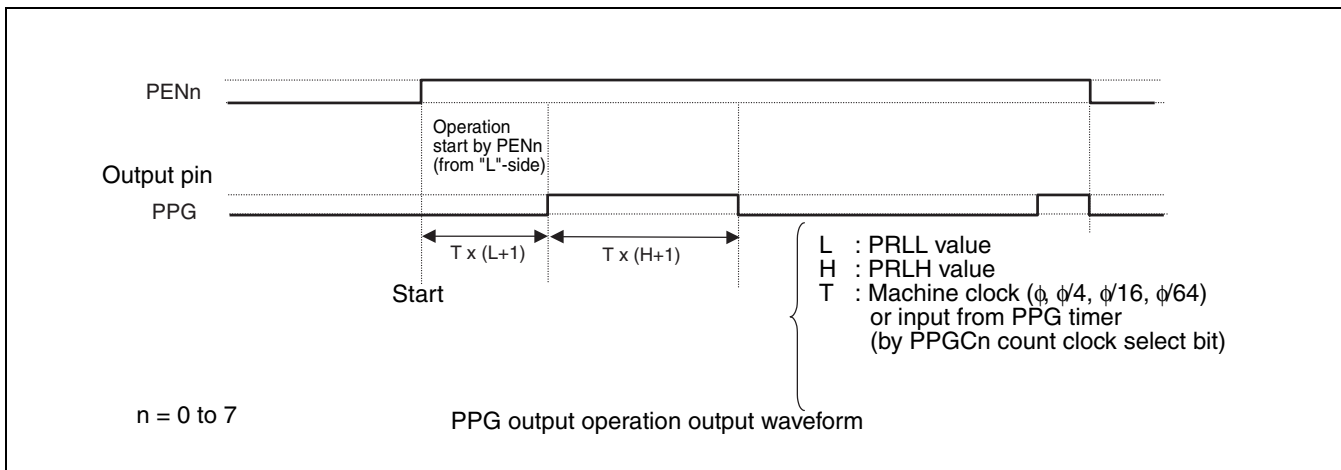
The PPG is activated and starts counting when the bit corresponding to each channel in the TRG register (PPG start register) is set to "1". After operation starts, the count operation is stopped when each channel bit of TRG register writes "0". After having stopped, the pulse output holds "L" level.

Do not set the PPG channel as the operating state, with the prescaler channel as the stopped state, in the 8-bit prescaler + 8-bit PPG mode and the 16-bit prescaler + 16-bit PPG mode.

In 16-bit PPG mode, use the PENn bits for each channel in the TRG register to simultaneously start and stop operation ( $n = 0$  to 7).

PPG output operation is explained below.

In PPG operation, the pulse wave with any frequency/duty ratio (the ratio between "H" level period and "L" level period in pulse wave) is output continuously. If the pulse wave output is started, PPG will not stop it before operation stop is set.



### ● Relationship between reload value and pulse width

The pulse width to be output is the value that multiplies the cycle of the count clock by the value in the reload register plus 1. Note that the pulse width will be one cycle of the count clock when the reload register value is set to "00<sub>H</sub>" at operating the 8-bit PPG and when the reload register value is set to "0000<sub>H</sub>" at operating the 16-bit PPG. Note that the pulse width will be 256 cycles of the count clock when the reload register value is set to "FF<sub>H</sub>" at operating the 8-bit PPG and the pulse width will be 65536 cycles of the count clock when the reload register value is set to "FFFF<sub>H</sub>" at operating the 16-bit PPG.

The equations for calculating the pulse width are shown below:

$$\begin{aligned} PI &= T \times (L + 1) \\ Ph &= T \times (H + 1) \end{aligned} \quad \left\{ \begin{array}{l} L : \text{PRLH value} \\ H : \text{PRLH value} \\ T : \text{Period of input clock} \\ Ph : \text{"H" pulse width} \\ Pl : \text{"L" pulse width} \end{array} \right.$$

● Count clock selection

The count clock to be used for this block operation uses the input for the peripheral clock and time-base counter, and can be selected from one of the following four types of count clock inputs.

The count clock operates as shown below.

PPGC0 to PPGC7 register		Count clock operation
PCS1	PCS0	
0	0	Count clock is counted for peripheral clock
0	1	Count clock is counted for 4 cycles of peripheral clock
1	0	Count clock is counted for 16 cycles of peripheral clock
1	1	Count clock is counted for 64 cycles of peripheral clock

Note that, in 8-bit prescaler + 8-bit PPG mode and 16-bit prescaler + 16-bit PPG mode, the period of the initial count may vary if the PPG is started when the prescaler side is running and the PPG side is halted.

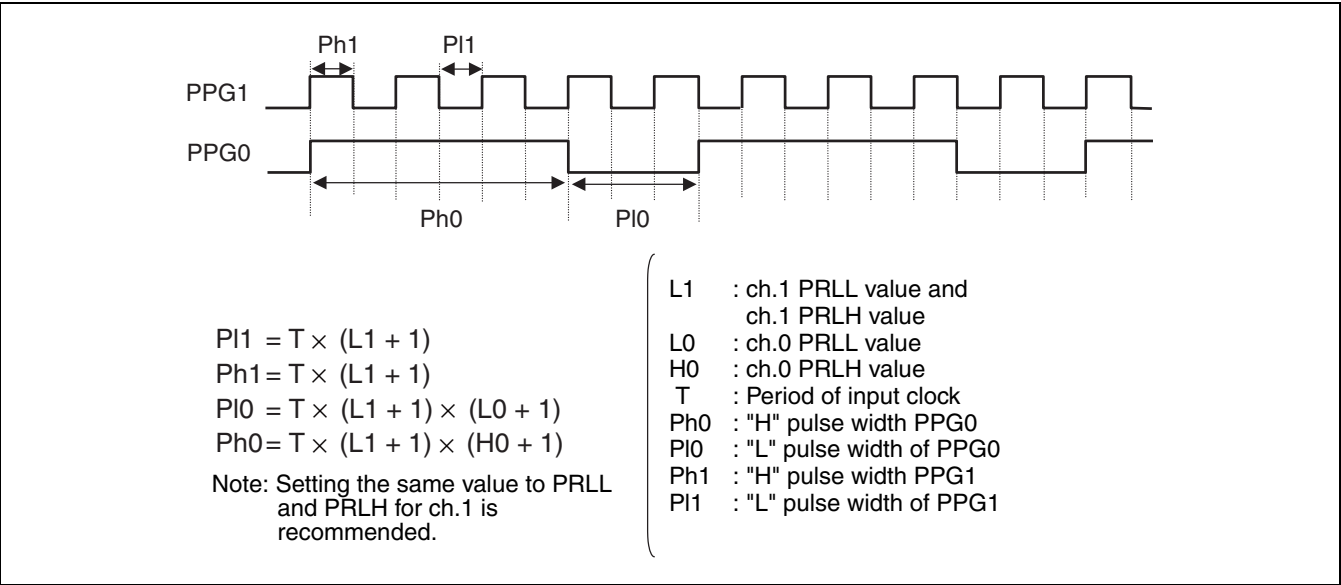
● Control of pulse pin output

The pulse output generated by operating this module can be output from external pins (PPG0 to PPG7).

As both PPG(m) and PPG(m + 1) output the same waveform in 16-bit PPG mode, the same output can be obtained whichever of these is enabled as an external output pin (m = 0, 2, 4, 6).

In the 8-bit prescaler + 8-bit PPG mode and the 16-bit prescaler + 16-bit PPG mode, the 8-bit prescaler toggle waveform is output on the prescaler side, and the 8-bit PPG waveform is output on the PPG side. The following shows an example of the output waveform in this mode.

Figure 10.4-1 8 + 8 PPG Output Operation Output Waveform



### ● Interrupt

The interrupt on this module becomes active when a reload value is counted out and a borrow occurs.

However, the interrupt only goes to active if the INTMn bit is "1" when an underflow (borrow) occurs for PRLBHn. The interrupt occurs when H width pulse ends.

In the 8-bit PPG mode and the 8-bit prescaler +8-bit PPG mode, an interrupt request is performed by the relevant counter borrow. However, in 16-bit PPG mode and 16-bit prescaler +16-bit PPG mode, PUF(m) and PUF(m+1) are concurrently set by the borrow of the 16-bit counter. For this reason, it is recommended that either PIE (m) or PIE (m + 1) is enabled in order to unify the interrupt sources. Similarly, it is recommended that you write to PUF(m) and PUF(m + 1) simultaneously when clearing the interrupt source (m = 0, 2, 4, 6).

### ● GATE function

By using the multifunction timer signal, PPG can be: started-stopped.

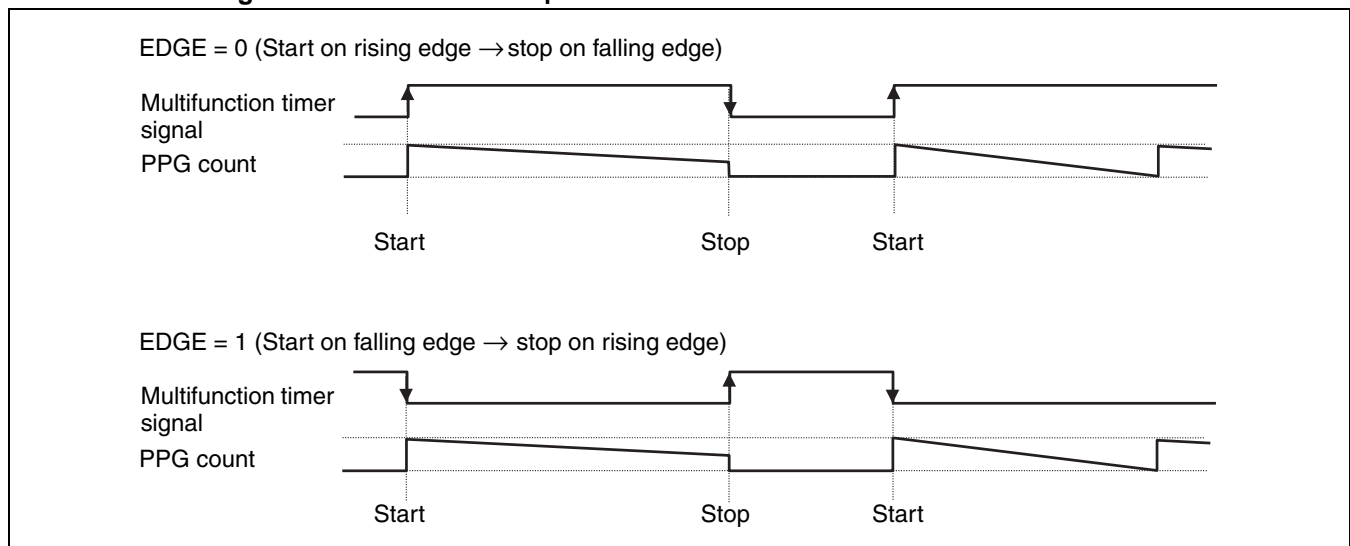
- In the 8-bit PPG mode and the 8-bit prescaler + 8-bit PPG mode, this function can activate the PPG ch.0.
- In the 16-bit PPG mode and the 16-bit prescaler + 16-bit PPG mode, this function can activate the PPG ch.0, ch.1.

The change of each mode is decided by the setting of MD1, MD0 bit of the PPG operation mode control register.

- When PPG ch.0 : MD1, MD0 = 0, X, PPG ch.0 is started (8-bit PPG)
- When PPG ch.0 : MD1, MD0 = 1, X, PPG ch.0, ch.1 are started (16-bit PPG)

EDGE bit and the multifunction timer signal can control the period when PPG activation is valid.

**Figure 10.4-2 PPG Count Operation Based on EDGE Bit and Multifunction Timer**



### ● Initial values for hardware

Each hardware of this block is initialized by a reset as shown below.

- < Register > PPGC(n) → 0000000XB
- < Pulse output > PPG(n) → "L"
- < Interruption request > IRQ(n) → "L" (n = 0 to 7)

Any hardware other than those above is not initialized.

## ● PPG combinations

ch.0: PPGC		ch.2: PPGC		ch.0	ch.1	ch.2	ch.3
MD1	MD0	MD1	MD0				
0	0	0	0	8-bit PPG	8-bit PPG	8-bit PPG	8-bit PPG
0	0	0	1	8-bit PPG	8-bit PPG	8-bit PPG ←	8-bit prescaler
0	0	1	0	8-bit PPG	8-bit PPG	16-bit PPG	
0	0	1	1	Setting disabled			
0	1	0	0	8-bit PPG ←	8-bit prescaler	8-bit PPG	8-bit PPG
0	1	0	1	8-bit PPG ←	8-bit prescaler	8-bit PPG ←	8-bit prescaler
0	1	1	0	8-bit PPG ←	8-bit prescaler	16-bit PPG	
0	1	1	1	Setting disabled			
1	0	0	0	16-bit PPG		8-bit PPG	8-bit PPG
1	0	0	1	16-bit PPG		8-bit PPG	8-bit prescaler
1	0	1	0	16-bit PPG		16-bit PPG	
1	0	1	1	Setting disabled			
1	1	0	0				
1	1	0	1				
1	1	1	0				
1	1	1	1	16-bit PPG ←		16-bit prescaler	

The operations for ch.(4, 5, 6, 7) can be combined in the same way as for ch.(0, 1, 2, 3).

Replace as shown below.

$$\left\{ \begin{array}{l} \text{ch.0} = \text{ch.4} \\ \text{ch.1} = \text{ch.5} \\ \text{ch.2} = \text{ch.6} \\ \text{ch.3} = \text{ch.7} \end{array} \right.$$

# **CHAPTER 11**

---

## ***PWC***

### ***(PULSE WIDTH COUNT)***

**This chapter explains the overview of the pulse width counter (PWC), the register configuration and functions and the counter operation.**

- 11.1 Overview
- 11.2 PWC Block Diagram
- 11.3 PWC Register
- 11.4 Operation Explanation

# 11.1 Overview

The function of this module is to measure pulse widths on the input signal.

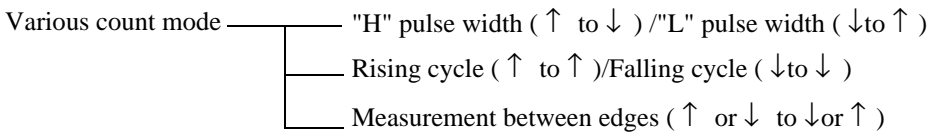
## ■ Functions of the PWC

The hardware consists of one 16-bit up-counter, one input pulse divider and divide ratio control register, one measurement input pin, and one 16-bit control register. These are used to implement the following functions.

### ● Pulse width measurement function

Measure the time between any events of external pulse inputs.

Internal clock to be used as the criterion can be selected from one of the following types (machine clock 4/16/32-divided).



The 8-bit input divider divides the input pulse by  $2^n$  (n = 1, 2, 3, 4, 5, 6, 7, or 8) and can measure the period of the input pulse.

Enable to generate an interrupt request when the measurement is completed.

Enable to select either the one-time-only measurement or the continuous measurement.

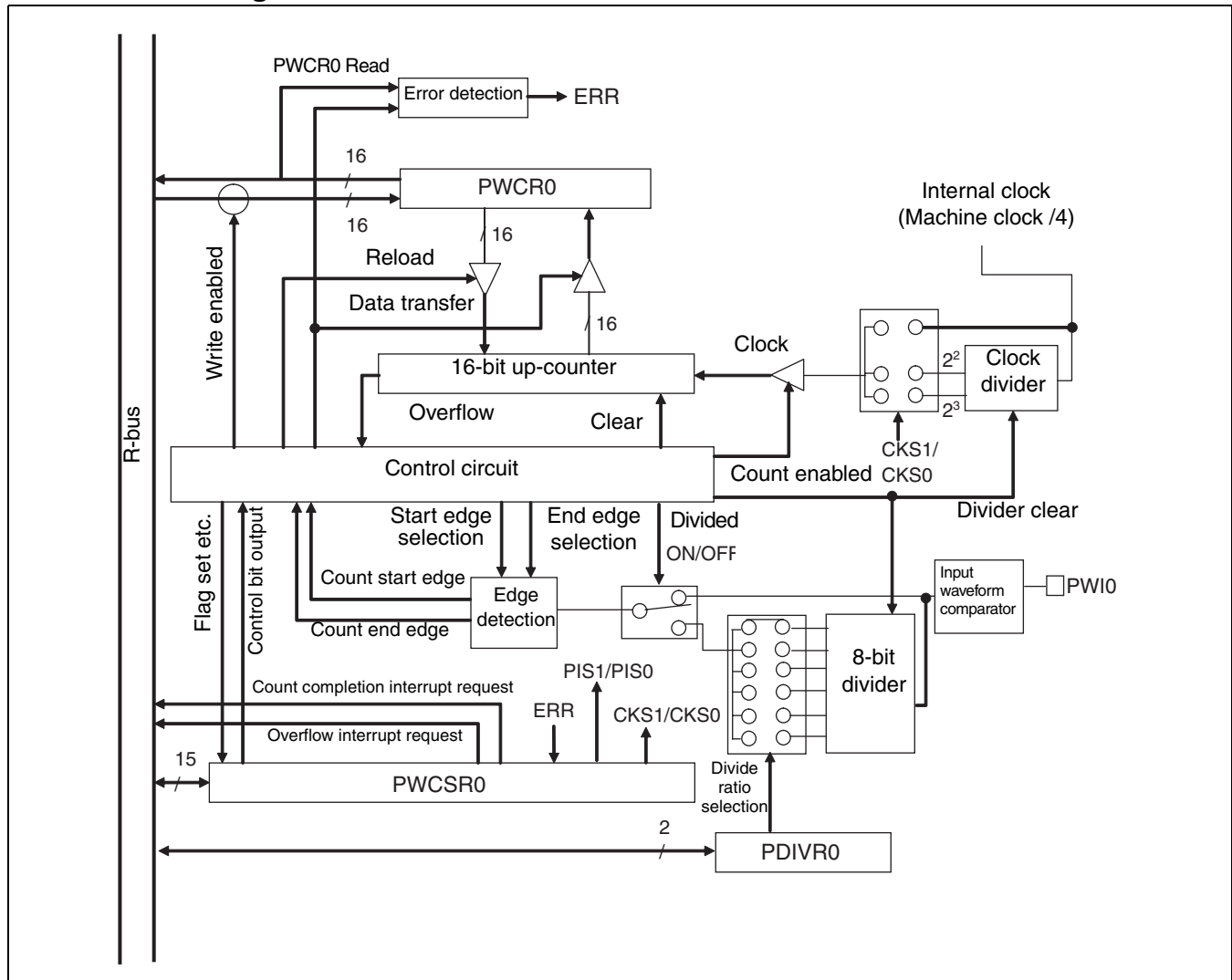
## ■ Registers of the Pulse Width Counter

Address		bit15 to bit8	bit7 to bit0	
	0000E9 <sub>H</sub>		PDIVR0	Ratio of dividing frequency control register 0
0000E0 <sub>H</sub>	0000E1 <sub>H</sub>	PWCSR0		PWC Control/status Register 0
0000E2 <sub>H</sub>	0000E3 <sub>H</sub>	PWCR0		PWC data buffer register 0

## 11.2 PWC Block Diagram

This section shows the block diagram of the PWC.

### ■ PWC Block Diagram





## 11.3 PWC Register

This section describes the PWC registers.

### ■ PWC Control/Status Register (PWCSR: PWCSR0)

PWCSR0 (Upper)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
0000E0 <sub>H</sub>	STRT	STOP	EDIR	EDIE	OVIR	OVIE	ERR	—	00000000 <sub>B</sub>
	R/W	R/W	R	R/W	R/W	R/W	R	R/W	

PWCSR0 (Lower)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0000E1 <sub>H</sub>	CKS1	CKS0	PIS1	PIS0	SC	MOD2	MOD1	MOD0	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable

R: Read only

[bit15] STRT: Counter start bit

[bit14] STOP: Counter stop bit

These bits start, restart, and stop the 16-bit up-counter. Reading the bit indicates the counter operating state. The following shows the bit functions.

**Table 11.3-1 Function at Writing (Operation Control)**

STRT	STOP	Operation control function
0	0	No function/no effect on the operation.
0	1	Start/restart a counter (enable count) Note: Clear bit command can be used
1	0	Stop a counter operation forcibly (disable count) Note: Clear bit command can be used
1	1	No function/no effect on the operation.

**Table 11.3-2 Function at Reading (Operating Status Indication)**

STRT	STOP	Indicate the operating state
0	0	Counter halted (not started or count finished) [Initial value]
1	1	Count operating (on measuring)

- Be initialized to "00<sub>B</sub>" when resetting.
- Reading and writing are permitted. However, there are different meanings in writing and in reading as shown above.
- If a read-modify-write instruction is issued, "11<sub>B</sub>" is read regardless of the operation.
- A bit manipulation instruction (such as a bit clear) corresponding to each bit can be used for writing to the STRT and STOP bits to start/stop the counter. However, the bit manipulation instructions cannot be used for reading an operating status. (Note that the status will be in operation surely if it is read.)

#### [bit13] EDIR: Measurement end interrupt request flag

The flag indicates that the measurement is completed in the pulse width count mode. A measurement end interrupt request is generated if this bit is set when the measurement end interrupt request is enabled (bit12: EDIE=1).

Set factor	Set when the pulse width count ends (when measurement result saved in PWCR)
Clear factor	Cleared when PWCR (measurement result) is read

- Be initialized to "0" when resetting.
- This bit is read-only. Writing does not change the bit value.

#### [bit12] EDIE: Measurement end interrupt request enable bit

This bit controls the measurement end interrupt request in the pulse width count mode as described below.

0	Disable output of measurement end interrupt requests (No interrupt is generated when EDIR is set.) [Initial value]
1	Enable output of measurement end interrupt requests (Generate interrupt when EDIR is set.)

- Be initialized to "0" when resetting.
- Reading and writing are permitted.

#### [bit11] OVIR: Counter overflow interrupt request flag

This flag indicates that the 16-bit up-counter overflowed from "FFFF<sub>H</sub>" to "0000<sub>H</sub>" in all modes. If counter overflow interrupt requests are enabled (bit10:OVIE = 1), a counter overflow interrupt request is generated when this bit is set.

Set factor	Set when a counter overflow occurs (from "FFFF <sub>H</sub> " to "0000 <sub>H</sub> ")
Clear factor	Cleared by writing "0".

- Be initialized to "0" when resetting.
- Reading and writing are permitted. However, only "0" can be written on it. Writing "1" does not change the bit value.
- The read value by a read-modify-write instruction is always "1", regardless of the bit value.

**[bit10] OVIE: Interrupt request enable bit for the counter overflow**

This bit controls counter overflow interrupt requests as follows.

0	Disable output of overflow interrupt requests (No interrupt is generated when OVIR is set.) [Initial value]
1	Enable output of overflow interrupt requests (Generate interrupt when OVIR is set.)

- Be initialized to "0" when resetting.
- Reading and writing are permitted.

**[bit9] ERR: Error flag**

The flag indicates that the next measuring is completed before reading the measurement result in PWCR in the continuous measurement mode of the pulse width count mode. At this point, the PWCR is updated to a new measurement result, the previous measurement result will be lost. The measurement is continued regardless of this bit value.

Set factor	Set if the previous result has not been read when overwritten by the next result.
Clear factor	Cleared when PWCR (measurement result) is read.

- Be initialized to "0" when resetting.
- This bit is read-only. Writing does not change the bit value.

**[bit8] (Reserved)**

This is a reserved bit. Reading returns "0".

- Be sure to write "0".

**[bit7, bit6] CKS1, CKS0: Clock select bits**

Select an internal count as listed in the table below.

CKS1	CKS0	Count clock selection
0	0	Machine clock divided by 4 [Initial value]
0	1	Machine clock divided by 16
1	0	Machine clock divided by 32
1	1	Setting disabled

- Be initialized to "00<sub>B</sub>" when resetting.
- Reading and writing are permitted. Do not set "11<sub>B</sub>".

Note: These bits must not be updated after starting. These bits must be written before starting or after stopped.

[bit5, bit4] PIS1,PIS0: Pulse width count input pin select bits

These bits select the input pin for pulse width measurement.

PIS1	PIS0	Input clock selection
0	0	(The PWI0 pin is selected) [Initial value]
0	1	Setting disabled
1	0	Setting disabled
1	1	Setting disabled

- Be initialized to "00<sub>B</sub>" when resetting.
- Reading and writing are enabled. However, setting any value other than "00<sub>B</sub>" is not permitted.

Note: These bits must not be updated after starting. These bits must be written before starting or after stopped.

[bit3] SC: Measurement mode (single/continuous) select bit

Select a measurement mode as shown below.

SC	Count mode selection	Pulse width count mode
0	Single measurement mode [Initial value]	Stop after one measurement
1	Continuous measurement mode	Continuous measurement: buffer register enabled

- Be initialized to "0" when resetting.
- Reading and writing are permitted.

Note: This bit must not be updated after starting. This bit must be written before starting or after stopped.

[bit2 to bit0] MOD2 to MOD0: Operation mode/measurement edge select bits

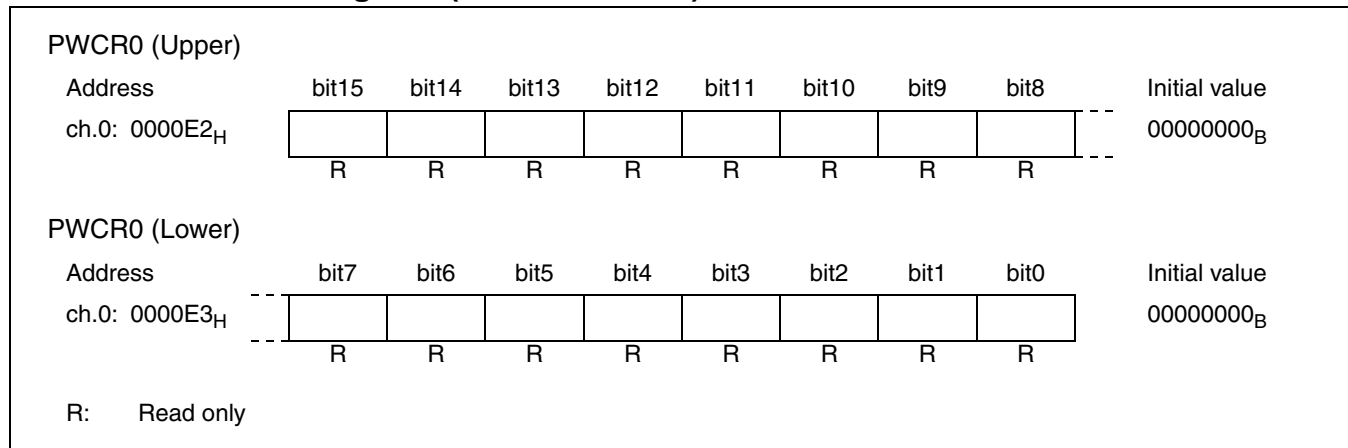
The operation mode and which edges to measure the width are selected as follows.

MOD2	MOD1	MOD0	Operation mode/measurement edge selection
0	0	0	Edge-to-edge pulse width count mode ( $\uparrow$ or $\downarrow$ to $\downarrow$ or $\uparrow$ ) [Initial value]
0	0	1	Division cycle measurement mode (input divider enabled)
0	1	0	Rising edge to rising edge period measurement mode ( $\uparrow$ to $\uparrow$ )
0	1	1	"H" pulse width measurement mode ( $\uparrow$ to $\downarrow$ )
1	0	0	"L" pulse width measurement mode ( $\downarrow$ to $\uparrow$ )
1	0	1	Falling edge-to-falling edge period measurement mode ( $\downarrow$ to $\downarrow$ )
1	1	0	Setting disabled
1	1	1	

- Be initialized to "000<sub>B</sub>" when resetting.
- Reading and writing are permitted.

Note: These bits must not be updated after starting. These bits must be written before starting or after stopped.

## ■ PWC Data Buffer Register (PWCR: PWCR0)



### ● Pulse width measurement mode

In the continuous measurement mode (PWCSR bit3: SC = 1), this register becomes a buffer register which holds the previous measurement result. In this case, only read operation is possible, but write operation does not affect the register value.

In the single measurement mode (PWCSR bit3: SC = 0), this register becomes an interface to access the up-counter directly. Also in this case, only read operation is possible, but write operation does not affect the register value. Read operation is always possible and can get the count value while counting is in progress. The measurement result is saved after the measurement ends.

Note: Always use half-word or word transfer instructions to access this register.

- Be initialized to "0000<sub>H</sub>" when resetting.
- Only reading is permitted.

### ■ Ratio of Dividing Frequency Control Register (PDIVR: PDIVR0)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
Ch.0: 0000E9 <sub>H</sub>	—	—	—	—	—	DIV2	DIV1	DIV0	XXXXX000 <sub>B</sub>
	—	—	—	—	—	R/W	R/W	R/W	

R/W: Readable/Writable  
 X: Undefined value  
 —: Undefined

This register is used in the division cycle measurement mode (PWCSR bit2 to bit0 : MOD2 to MOD0 = 001<sub>B</sub>), and invalid in other modes.

In the division cycle measurement mode, divide the input pulse in the count pin by the divided-by rate set in this register and measure one cycle width after divided. Select the divide-by rate as shown below.

DIV2	DIV1	DIV0	Division ratio select
0	0	0	$2^1 = \text{divided by } 2$ [Initial value]
0	0	1	$2^2 = \text{divided by } 4$
0	1	0	$2^3 = \text{divided by } 8$
0	1	1	$2^4 = \text{divided by } 16$
1	0	0	$2^5 = \text{divided by } 32$
1	0	1	$2^6 = \text{divided by } 64$
1	1	0	$2^7 = \text{divided by } 128$
1	1	1	$2^8 = \text{divided by } 256$

- Be initialized to "000<sub>B</sub>" when resetting.
- Reading and writing are permitted.

Note: These bits must not be updated after starting. These bits must be written before starting or after stopped.

## 11.4 Operation Explanation

PWC has the measurement input pin and 8-bit input division. PWC has also the pulse width count function. Three types of count clocks can be selected. The following describes the basic functions/operations of the pulse width count function.

### ■ Pulse Width Measurement Function

Enable to measure the time/cycle between any events of input pulse by the counter.

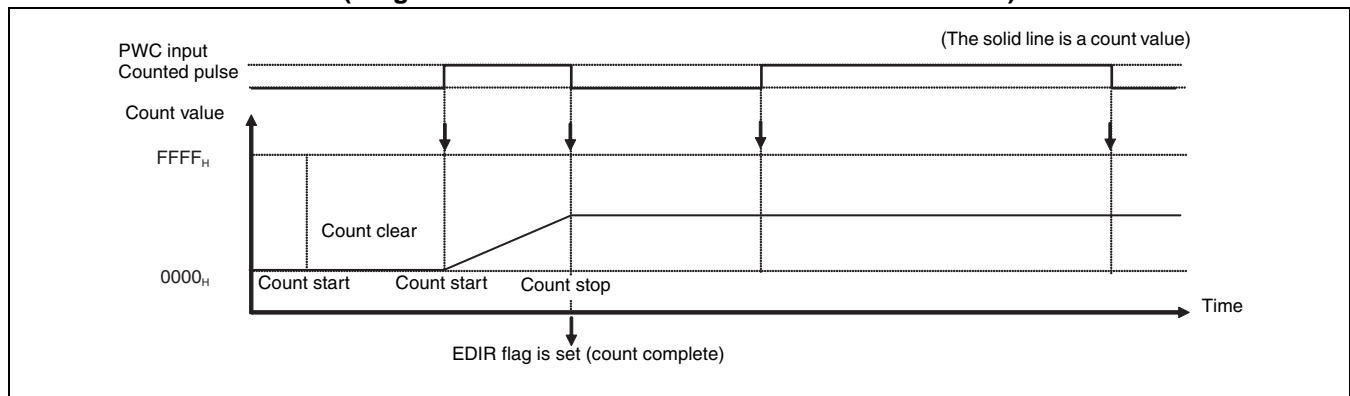
After starting, counting will not be performed until the defined measurement start edge is input. Start incremental counting after the counter which detects a start edge is cleared to 0000<sub>H</sub>. Stop the counting when a stop edge is detected. The value counted in this period is saved to the register as the pulse width.

An interrupt request can be generated when the measurement count completes or overflow occurs.

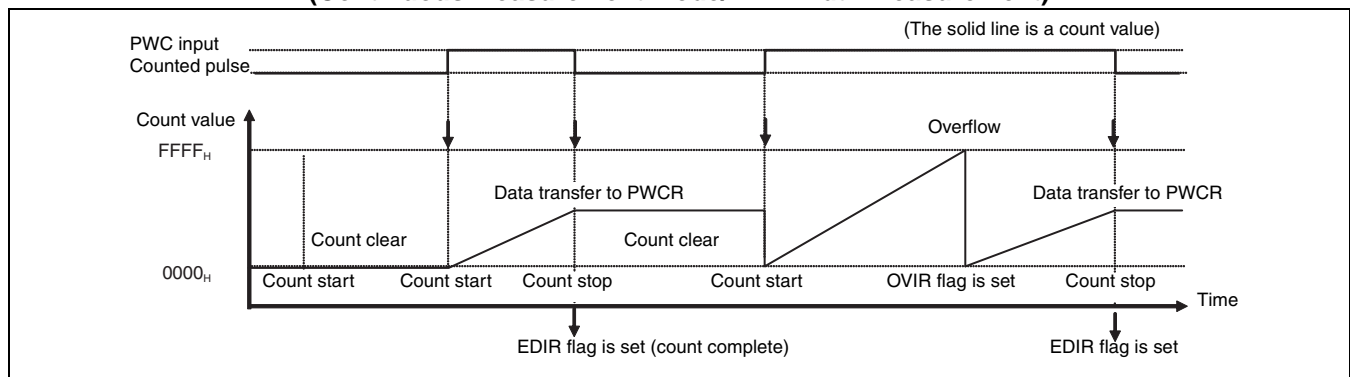
Operates as follows depending on the measurement mode after the measurement.

- In single measurement mode: Stop operation.
- In continuous measurement mode: After transferred a counter value to the buffer register, stop the counting until the measurement start edge is input again.

**Figure 11.4-1 Pulse Width Measurement Operation  
(Single Measurement Mode/ "H" Width Measurement)**



**Figure 11.4-2 Pulse Width Measurement Operation  
(Continuous Measurement Mode/ "H" Width Measurement)**





## ■ Count Clock Selection

The count clock of counter can select three types of internal clock sources by setting of PWCSR register (bit7, bit6 : CKS1, CKS0).

Selectable count clocks are as shown below.

PWCSR	Selected internal count clock
CKS1, CKS0	
00 <sub>B</sub>	Machine clock divided by 4 [Initial value]
01 <sub>B</sub>	Machine clock divided by 16
10 <sub>B</sub>	Machine clock divided by 32

The clock generated by dividing the machine clock by 4 is selected for the initial value after a reset.

Note: Always select the count clock before starting the counter.

## ■ Select Operation Mode

The PWCSR settings are used to select the operation mode and measurement mode.

- Operation mode setting: Bit2 to bit0 of PWCSR register (MOD2 to MOD0)  
(used to set parameters including the pulse width measurement mode and which edges to measure)
- Measurement mode setting: Bit3 of PWCSR (SC)  
(selects single or continuous measurement)

The table below lists the operation mode selected by each combination of mode setting bit values.

Operating mode			SC	MOD2	MOD1	MOD0
Pulse width count	↑ or ↓ to ↑ or ↓ Measurement between all edges	Single measurement: Buffer invalidity	0	0	0	0
		Continuous measurement: Buffer effective	1	0	0	0
	Divided period measurement (divided by 1 to 256)	Single measurement: Buffer invalidity	0	0	0	1
		Continuous measurement: Buffer effective	1	0	0	1
	↑ to ↑ Rising edge to rising edge period count	Single measurement: Buffer invalidity	0	0	1	0
		Continuous measurement: Buffer effective	1	0	1	0
	↑ to ↓ "H" pulse width measurement	Single measurement: Buffer invalidity	0	0	1	1
		Continuous measurement: Buffer effective	1	0	1	1
	↓to ↑ "L" pulse width measurement	Single measurement: Buffer invalidity	0	1	0	0
		Continuous measurement: Buffer effective	1	1	0	0
	↓to ↓ Falling edge-to-falling edge period measurement	Single measurement: Buffer invalidity	0	1	0	1
		Continuous measurement: Buffer effective	1	1	0	1
Setting disabled			0	1	1	0
			1	1	1	0
			0	1	1	1
			1	1	1	1

All edge-to-edge measurement to single measurement mode is selected for the initial value after a reset.

Note: Always set the activation mode before starting the counter.

## ■ Starting and Stopping of the Pulse Width Count

Each operation is started, restarted, and forcibly stopped using bit15 and bit14 (STRT and STOP) of PWCSR.

STRT bit starts/restarts a pulse width count, and STOP bit stops forcibly it. They can work when "0" is written to their bit. But they cannot work unless the value written to both bits is exclusive. Be sure to write the following combination when an instruction other than bit manipulation instructions (by the bigger unit than byte) is written.

Function	STRT	STOP
Starting and restarting of the pulse width count	0	1
Forcibly stopping of the pulse width count	1	0

When a bit manipulation instruction (bit clear instruction) is used, these combinations are written by the hardware automatically without having to take them into consideration.

### ● Operation after startup

The following describes the operation after starting a pulse width count mode.

- Pulse width count mode: The count will not be performed until the measurement start edge is input. After the measurement start edge is detected, 16-bit up- counter is cleared to "0000<sub>H</sub>", and the counting is started.

### ● Restarting

A restart is to start a pulse width count (write "0" to the STRT bit) during operation after the count has been started. By restarting, the following operation occurs.

- Has no effect on operation while waiting for the count start edge. If the measurement is in progress, stops the counting, and then waits for the measurement start edge again. At this time, if a restart occurs simultaneously with detecting a measurement end edge, the measurement end interrupt flag (EDIR) is set, and the measurement result is transferred to PWCR in continuous measurement mode.

### ● Stopping

As the count operation stops automatically when the count completes or the counter overflows in single count mode, no particular consideration is required in this case. In other modes, or to stop operation before stopping automatically, the operation must be stopped forcibly.

### ● Check of operation state

The above STRT bit and STOP bit work as an operating status indication bit during reading. The value displayed means the following description.

STRT	STOP	Operating State
0	0	Counter stopped (except in waiting for a measurement start edge) It is not active or shows that the measurement ended.
1	1	Count operating or waiting for a measurement start edge

Note: The same value is read from either the STRT bit or the STOP bit. If a read-modify-write instruction (such as bit-processing instruction) is issued, "11<sub>B</sub>" is always read from these bits. Do not read by using such instruction.

## ■ Counter Clear

The 16-bit up-counter is cleared to "0000<sub>H</sub>" in the following cases.

- At a reset
- Start the count when a measurement start edge is detected in the pulse width count mode

## ■ Details of Pulse Width Count Operation

### ● Single measurement and continuous measurement

Pulse width counting has one mode for performing a single count and another mode for performing continuous counting. PWCSR SC bit is used to select each mode (see "■ Selects operation mode"). The differences between these modes are as described below.

- Single count mode: The counter stops counting when the first count end edge is input, sets the measurement end interrupt flag (EDIR) of PWCSR, and performs no further counting. However, if a restart occurs simultaneously, the operation waits for a measurement start.
- Continuous count mode: When the count end edge is input, the operation stops the counter count, sets the measurement end interrupt flag (EDIR) in PWCSR, and stops the count until the count start edge is input again. When the measurement start edge is input again, the operation clears the counter to "0000<sub>H</sub>", and then starts the measurement. When the measurement ends, the measurement result of the counter is transferred to PWCR.

Note: Only select or modify the count mode while the counter is halted.

### ● Count result data

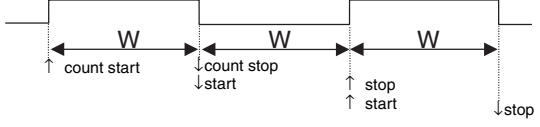
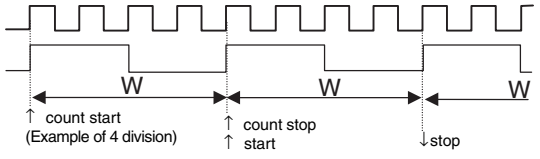
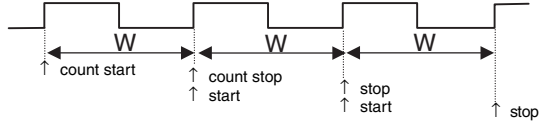
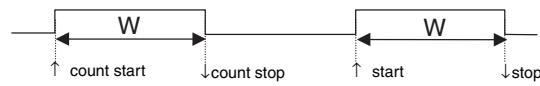
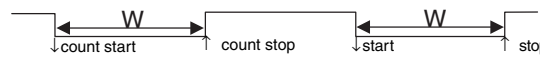
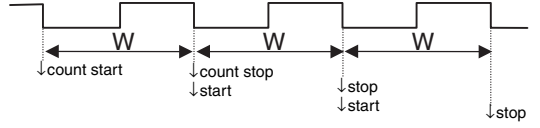
The differences between the single measurement mode and the continuous measurement mode are in handling of a measurement result and a counter value and the PWCR function. The following describes the differences of the measurement results in both modes.

- Single measurement mode: A count value in measuring can be obtained when PWCR is read during operation. A measurement result can be obtained when PWCR is read after the measurement ends.
- Continuous measurement mode: A measurement result in counter is transferred to PWCR when the measurement ends. Reading the PWCR returns the previous count result as the PWCR continues to store the previous result during counting. A count value in measuring cannot be read.

If the next measurement is completed before reading the measurement result in continuous measurement mode, the previous measurement result is cleared by the new measurement result. In this case, the error flag (ERR) in the PWCSR is set. An error flag (ERR) is cleared automatically when PWCR is read.

● Measurement mode and count operation

The measurement mode can be selected from six types depending on the place where the input pulse is measured. The cycle measurement mode is also prepared to arbitrarily divide the input pulse for high-precision measurement of higher frequency pulse width. The following describes each mode.

Count mode	MOD2	MOD1	MOD0	Measurement details (W: pulse width to be measured)
Edge-to-edge pulse width count	0	0	0	 <p>The width between the edges continuously input is measured. Count (measurement) beginning: When edge is detected Count (measurement) end: When edge is detected</p>
Measurement at cycle of dividing frequency	0	0	1	 <p>The input pulse is divided only for the divide-by rate selected by the division ratio setting register PDIVR and the cycle is measured. Count (measurement) start: When the first rising edge is detected after being started Count (measurement) end: When one cycle ends after it is divided</p>
Rising edge to rising edge period count	0	1	0	 <p>Measures the period between rising edges. Count (measurement) start: Detection of a rising edge Count (measurement) end: Detection of a rising edge</p>
"H" pulse width measurement	0	1	1	 <p>The width at "H" period is measured. Count (measurement) start: Detection of a rising edge Count (measurement) end: At falling edge detection</p>
"L" pulse width measurement	1	0	0	 <p>The width at "L" period is measured. Count (measurement) start: At falling edge detection Count (measurement) end: Detection of a rising edge</p>
Falling edge to falling edge period count	1	0	1	 <p>Measure a cycle between falling edges. Count (measurement) start: At falling edge detection Count (measurement) end: At falling edge detection</p>

In all modes, the counter does not operate (count) in the time between the count operation being enabled and input of the count start edge. The counter is cleared to "0000<sub>H</sub>" when the count start edge is input and the counter continues to count up one each count clock until the count end edge is input.

When the measurement end edge is input, the following operations are executed.

- Measurement ending flag (EDIR) in PWCSR is set.
- Stop the counter count operation (Except in case that a restart occurs simultaneously).
- In continuous measurement mode: a counter value (count result) is transferred to PWCR, stops the counting until the next measurement start edge is input.
- In single measurement mode: End the measurement (Except in case that a restart occurs simultaneously).

In continuous count mode, the count end edge is also the count start edge for the next count when counting the period or edge-to-edge pulse width.

#### ● Minimum input pulse width

The following restrictions apply to the pulse that can be input to the pulse width count input pin (PWI0).

- Minimum input width: Machine cycle  $\times$  4 or longer (0.25 $\mu$ s or longer for a 16MHz machine clock)

If the pulse whose width is smaller than the above pulse width is input, the operation will not be guaranteed.

#### ● Calculation method of pulse width/cycle

After the measurement ends, the pulse width/cycle to be counted can be calculated from the data of the measurement result obtained in PWCR.

	$T_W$ : Pulse width/cycle to be measured [ $\mu$ s ]
	$n$ : Measurement result data in PWCR
$T_W = n \times t / D_{IV}$ [ $\mu$ s ]	$t$ : Period of count clock [ $\mu$ s ]
	$D_{IV}$ : Division ratio selected by the division ratio register PDIVR (Substitute "1" in a mode other than the division cycle measurement mode.)

#### ● Pulse width/cycle measurement range

The measurable range of the pulse width/cycle depends on the combination of a count clock and a divide-by rate of the input divider.

The table below shows an example of measurement range for the case when the machine clock (referred to as  $\phi$  below) = 16MHz.

Divide ratio	DIV2, DIV1, DIV0	In CKS1, CKS0 = 00 <sub>B</sub> ( $\phi/4$ )	In CKS1, CKS0 = 01 <sub>B</sub> ( $\phi/16$ )	In CKS1, CKS0 = 10 <sub>B</sub> ( $\phi/32$ )
Not divided	-	0.25 $\mu$ s to 16.4ms [250ns]	0.25 $\mu$ s to 65.5ms [1.0 $\mu$ s]	0.25 $\mu$ s to 131ms [2.0 $\mu$ s]
Divided by 2	000 <sub>B</sub>	0.25 $\mu$ s to 8.19ms [125ns]	0.25 $\mu$ s to 32.8ms [0.5 $\mu$ s]	0.25 $\mu$ s to 65.5ms [1.0 $\mu$ s]
Divided by 4	001 <sub>B</sub>	0.25 $\mu$ s to 4.10ms [62.5ns]	0.25 $\mu$ s to 16.4ms [250ns]	0.25 $\mu$ s to 32.8ms [0.5 $\mu$ s]
Divided by 8	010 <sub>B</sub>	0.25 $\mu$ s to 2.05ms [31.25ns]	0.25 $\mu$ s to 8.19ms [125ns]	0.25 $\mu$ s to 16.4ms [250ns]
Divided by 16	011 <sub>B</sub>	0.25 $\mu$ s to 1.02ms [15.6ns]	0.25 $\mu$ s to 4.10ms [62.5ns]	0.25 $\mu$ s to 8.19ms [125ns]
Divided by 64	100 <sub>B</sub>	0.25 $\mu$ s to 256 $\mu$ s [3.91ns]	0.25 $\mu$ s to 1.024ms [15.6ns]	0.25 $\mu$ s to 2.05ms [31.25ns]
Divided by 256	101 <sub>B</sub>	0.25 $\mu$ s to 64.0 $\mu$ s [0.98ns]	0.25 $\mu$ s to 256 $\mu$ s [3.91ns]	0.25 $\mu$ s to 512 $\mu$ s [7.81ns]
-	the others	Setting disabled		

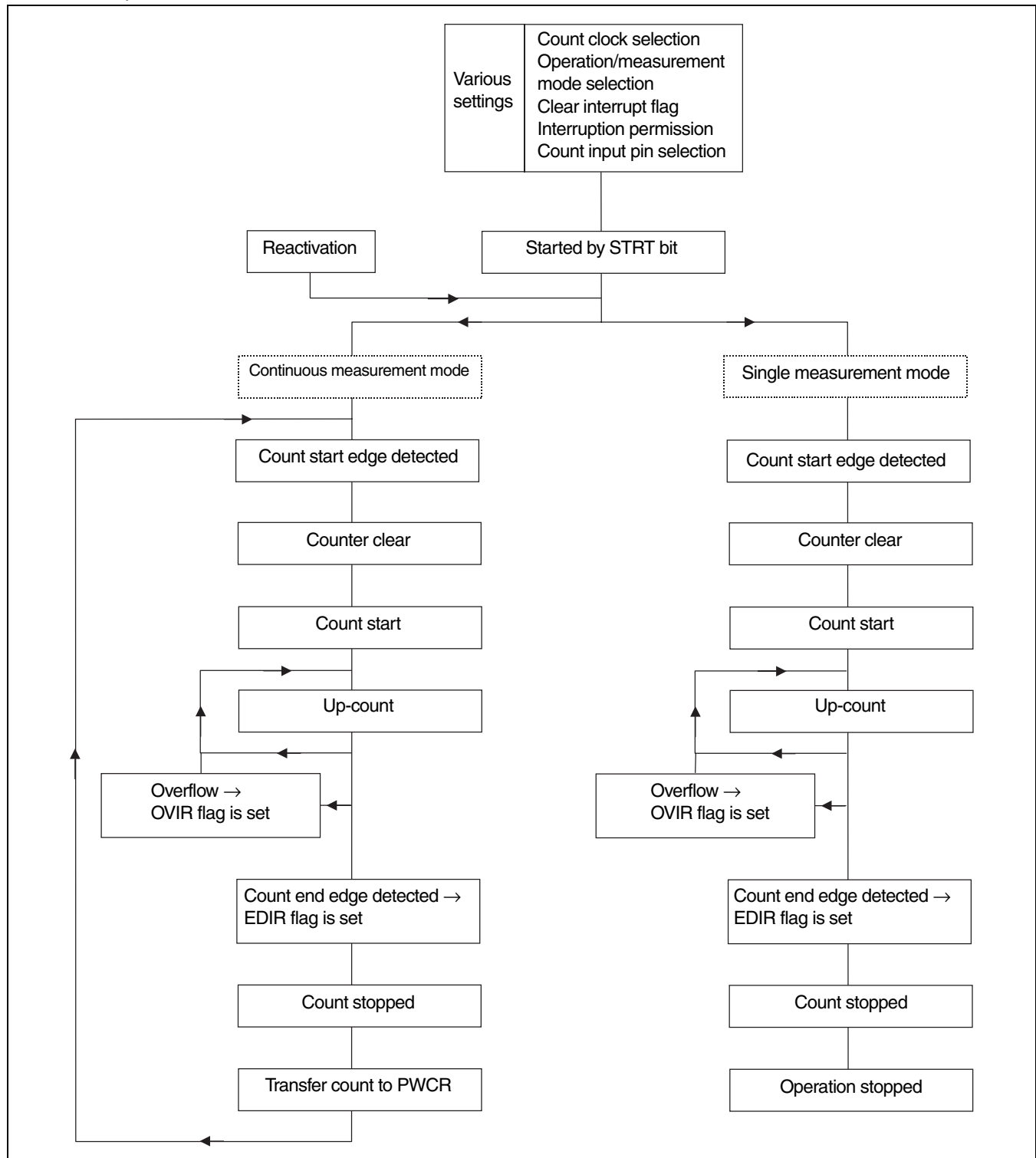
Notes: • When machine clock  $\phi = 16\text{MHz}$   
 • The value in [ ] indicates the resolution per bit.

#### ● Interrupt request generation

In the pulse width count mode, the following two interrupt requests can be generated.

- Interrupt request due to counter overflow  
 When an up-count causes the counter to overflow during measurement, the overflow flag is set and (if the overflow interrupt request is enabled) an interrupt request is generated.
- Interrupt request due to measurement completion  
 When the measurement termination edge is detected, the measurement end interrupt flag (EDIR) in the PWCSR is set. If the measurement end interrupt request is permitted, the interrupt request occurs.  
 A measurement end interrupt flag (EDIR) is cleared automatically when a measurement result PWCR is read.

- Operation Flowchart for Pulse Width Count





## ■ Notes

### ● Note on register rewriting

The following bit of the PWCSR register must not be rewritten during an operation. Only change the setting before starting or after stopping operation.

[bit7, bit6] CKS1,CKS0 : Clock selection bits

[bit5, bit4] PIS1,PIS0 : Pulse width count input pin select bits

[bit3] SC : Measurement mode (single/continuous) select bit

[bit2 to bit0] MOD2 to MOD0 : Operation mode / count edge selection bits

The PDIVR register must not be rewritten during an operation. Only change the setting before starting or after stopping operation.

### ● STRT bit and STOP bit of the PWCSR register

Note that both bits have different meanings in writing and in reading. (See "■ PWC Control/Status Register (PWCSR: PWCSR0)")

Also, the read value by a read-modify-write instruction is always "11<sub>B</sub>", regardless of the bit value. For this reason, note that the bit-processing instructions cannot be used for reading an operation status (the status must be in operation if it is read).

A bit manipulation instruction (such as a bit clear instruction) corresponding to each bit can be used for writing to the STRT and STOP bits to start/stop the counter.

### ● Counter clear

In the pulse width count mode, because a measurement start edge clears a counter, the data which exists in counter before starting becomes invalidated.

### ● Minimum input pulse width

The following restrictions apply to the pulse that can be input to the pulse width count input pin.

- Minimum input width: Machine cycle  $\times$  4 ( $\geq 250\text{ns}$  when machine cycle is  $62.5\text{ns}$ )
- Maximum input frequency: Clock generated by dividing the machine clock by 4 ( $\leq 4\text{MHz}$  when machine cycle is  $16\text{MHz}$ )

If the pulse which has smaller width/higher frequency than the above is input, the operation will not be guaranteed. If there may be such noise on the input signal, remove the noise through a filter outside the chip, and then input the pulse.

### ● Divided period measurement mode

Note that the pulse width obtained from the measured result calculation is the average value because the input pulses are divided in the division cycle measurement mode among the pulse width measurement modes.

### ● Clock selection bits

Setting "11<sub>B</sub>" to the (bit7, bit6): CKS1, CKS0: clock select bits of the PWCSR register is prohibited.

### ● Reserved bit

The (bit8) of PWCSR register is a reserved bit. Always write "0" to this bit.

● Restarting during operation

The following may occur if operation is restarted after the count has started.

- In the pulse width single count mode, if starting a measurement end edge is simultaneously  
The operation is restarted and waits for a measurement start edge, but a measurement end interrupt flag (EDIR) is set.
- In the pulse width continuous measurement mode, if starting a measurement end edge is simultaneously  
The operation is restarted and waits for a measurement start edge, but the measurement end interrupt flag (EDIR) is set. The measurement result at that point is transferred to the PWCR.

As described above, consider flag operation and take measures such as controlling interrupts if restarting during operation.



# **CHAPTER 12**

---

## ***MULTIFUNCTION TIMER***

**This chapter explains the overview of the multifunction timer, the configuration and functions of registers, and operation of the multifunction timer.**

- 12.1 Overview
- 12.2 Block Diagram
- 12.3 Pins of the Multifunction Timer
- 12.4 Multifunction Timer Register
- 12.5 Multifunction Timer Interrupt
- 12.6 Operation of the Multifunction Timer
- 12.7 Notes on Using the Multifunction Timers
- 12.8 Example Program for Multifunction Timer

## 12.1 Overview

---

**Multifunction timer consists of three 16-bit free-run timer, six 16-bit output compares, four 16-bit input captures, four channels of 8/16-bit PPG timer, one waveform generator, and two A/D activation compares. If this waveform generator is used, 6 different waveforms can be output from the 16-bit free-run timer, an input pulse width and an external clock cycle can be measured.**

---

### ■ Structure of the Multifunction Timer

#### ● 16-bit free-run timer (× 3)

- The 16-bit free-run timer consists of 16-bit up/down counters, control registers, 16-bit compare clear registers (with buffer registers), and prescalers.
- Nine different counter operating clocks are available ( $\phi$ ,  $\phi/2$ ,  $\phi/4$ ,  $\phi/8$ ,  $\phi/16$ ,  $\phi/32$ ,  $\phi/64$ ,  $\phi/128$ , and  $\phi/256$ ). ( $\phi$ : Machine clock)
- A compare clear interrupt is generated when a compare clear register compares and matches 16-bit free-run timer. A 0-detection interrupt is generated while the 16-bit free-run timer detects a count value "0".
- A compare clear register has selectable buffer registers. (Data written to this buffer register is transferred to a compare clear register.) When the 16-bit free-run timer is stopped and data is written to the buffer, the transfer is performed immediately. When the timer value "0" is detected during the 16-bit free-run timer operation, data is transferred from the buffer.
- When a compare match with a reset, a software clear, or a compare clear register occurs in the up count mode, the counter value is reset to "0000<sub>H</sub>".
- This counter output value can be used as a multifunction timer output compare and an input capture clock count.
- When a 0-detection or a compare match occurs, A/D can be activated.

#### ● 16-bit output compare (× 6)

- The 16-bit output compare consists of six 16-bit compare registers (with selectable buffer registers), compare output latches, and compare control registers. An interrupt is generated and the output level is inverted when a match occurs between the value of the selected 16-bit free-run timer and the compare register.
- Six compare registers can be operated independently. An output pin and an interrupt flag are assigned to each compare register.
- Two compare registers can be paired to control an output pin. The output pin can be reversed with using two compare registers together.
- The initial value of each output pin can be set.
- An interrupt can be generated when the output compare register matches the 16-bit free-run timer.

### ● 16-bit input capture (× 4)

- The input capture consists of four independent external input pins, and capture registers and capture control registers associated to these pins. Detection of an edge on the input signal from the external pin causes the value of the selected 16-bit free-run timer to be copied to the capture register and an interrupt to be generated.
- The trigger edge for the external input signal can be selected from the three types: Rising edge, falling edge, and both edges. Also there are registers that indicate whether the trigger edge is a rising edge or a falling edge.
- Four input capture can be used independently.
- An interrupt can be generated when a valid edge of an external input signal is detected.

### ● 8/16-bit PPG timer (× 4)

The PPG timer 0 is used to provide the PPG signal for the waveform generator. See the "CHAPTER 10 PPG" for details of PPG timer 0.

### ● Waveform generator

- The waveform generator consists of three 16-bit dead timer registers, three timer control registers, and one 16-bit waveform control register.
- The waveform generator can generate real-time output, 16-bit PPG waveform output, non-overlap 3-phase waveform output (for inverter control), and DC chopper waveform output.
- The non-overlap waveform output can be generated on the basis of the dead time of the 16-bit dead timer (dead time timer function).
- The non-overlap waveform output can be generated when the real-time output is activated in 2 channel mode (dead time timer function).
- When a real-time output compare match is detected, GATE signal is generated to start or stop the PPG timer operation (GATE function).
- When a real-time output compare match is detected, the 16-bit dead timer becomes active. With generating the GATE signal for controlling the PPG operation, the PPG timer 0 can be started or stopped easily (GATE function).
- DTTI pins can be used to control stopping forcibly.
- DTTI registers can be used to control stopping forcibly.

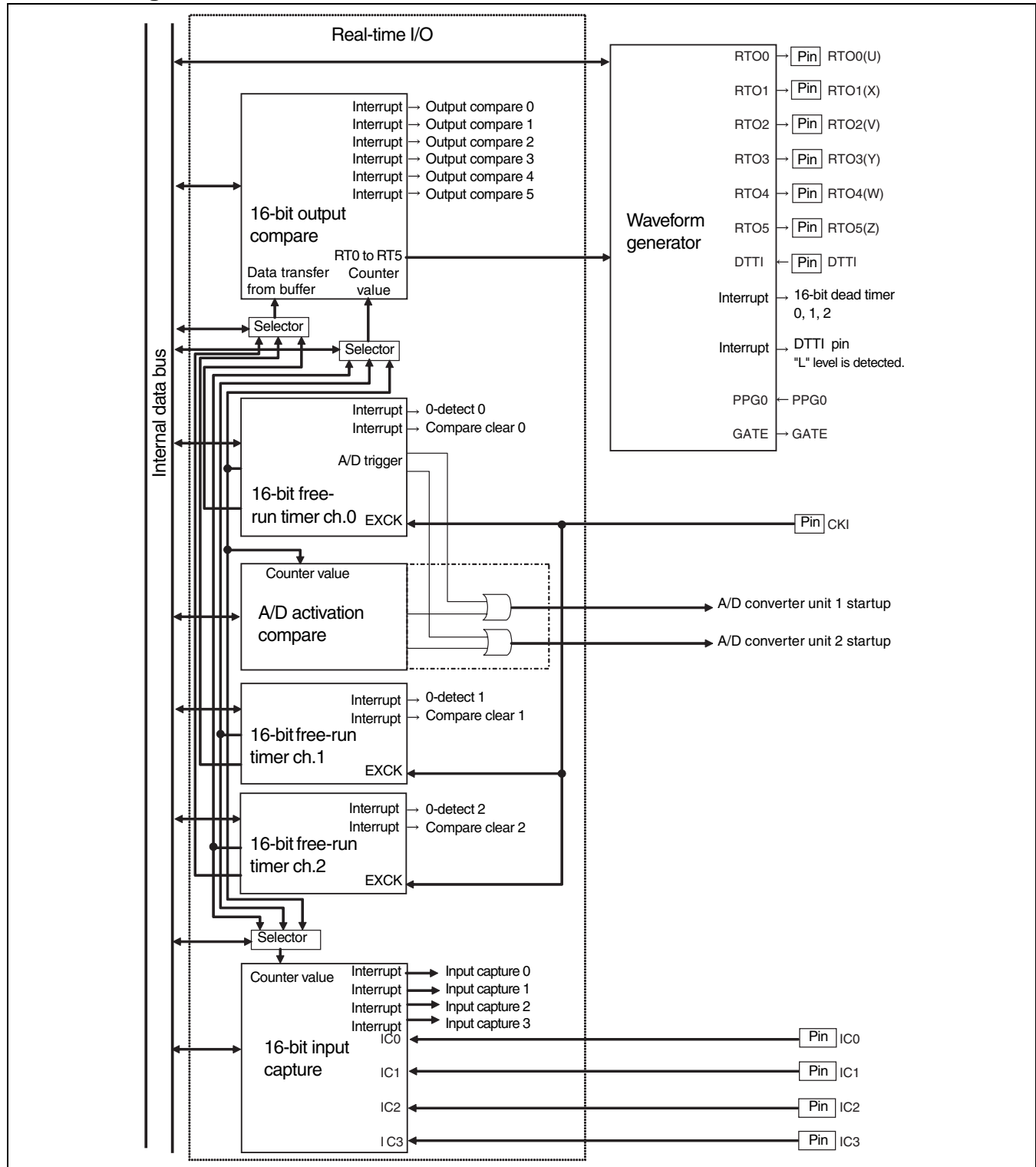
### ● A/D activation compare (× 2)

- The A/D can be triggered when a match occurs between the value of the ch.0 16-bit free-run timer and the compare register. (ch.1 and ch.2 of the 16-bit free-run timer cannot be used to start the A/D.)
- Compare 1 can activate A/D1.
- Compare 2 can activate A/D2.
- You can specify that activation only occurs when a match is detected on an up-count of the 16-bit free-run timer.
- You can specify that activation only occurs when a match is detected on a down-count of the 16-bit free-run timer.
- You can specify that activation occurs when a match is detected on both an up and down count of the 16-bit free-run timer.

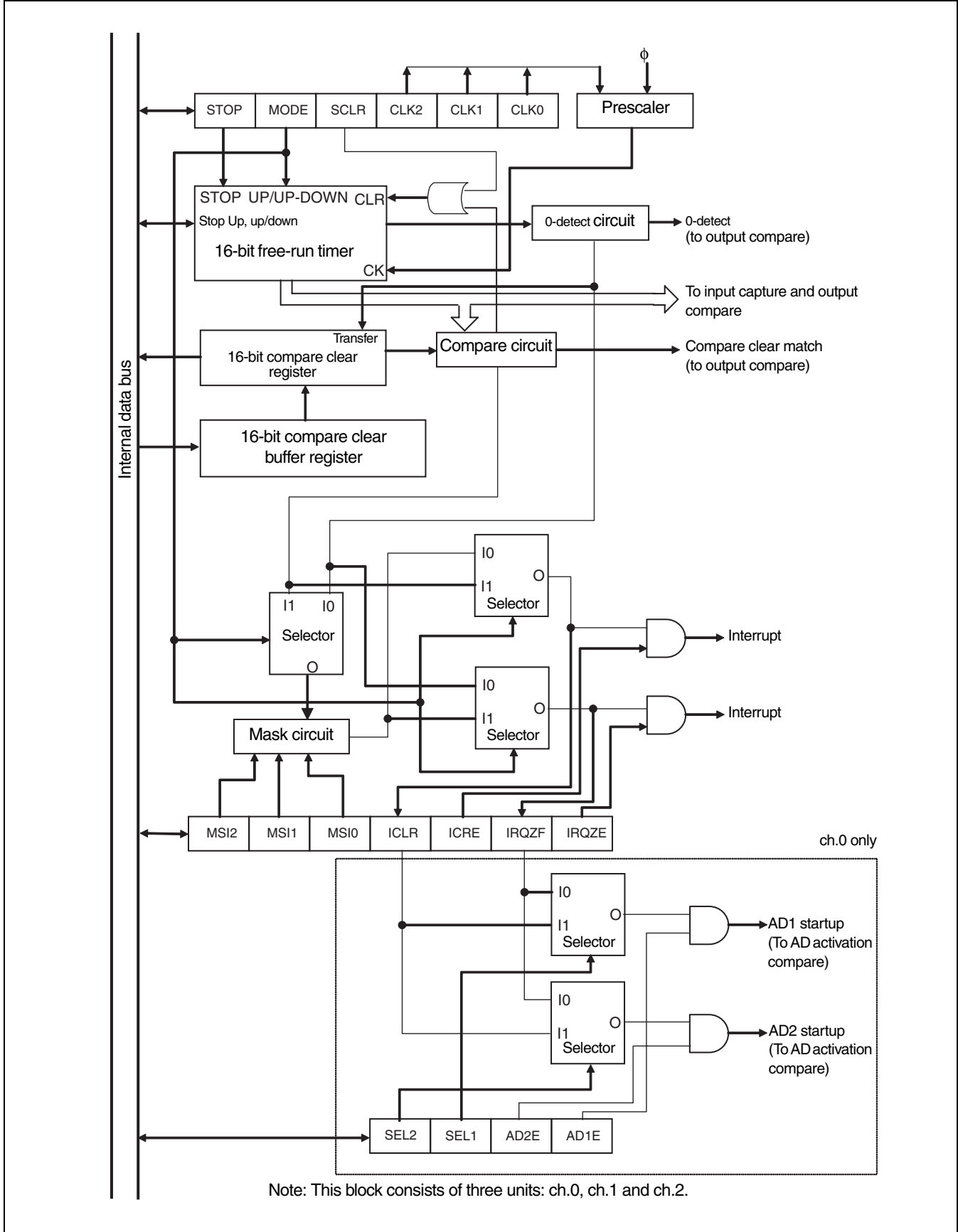
## 12.2 Block Diagram

This section shows the block diagram of the multifunction timer.

### ■ Block Diagram of the Multifunction Timer

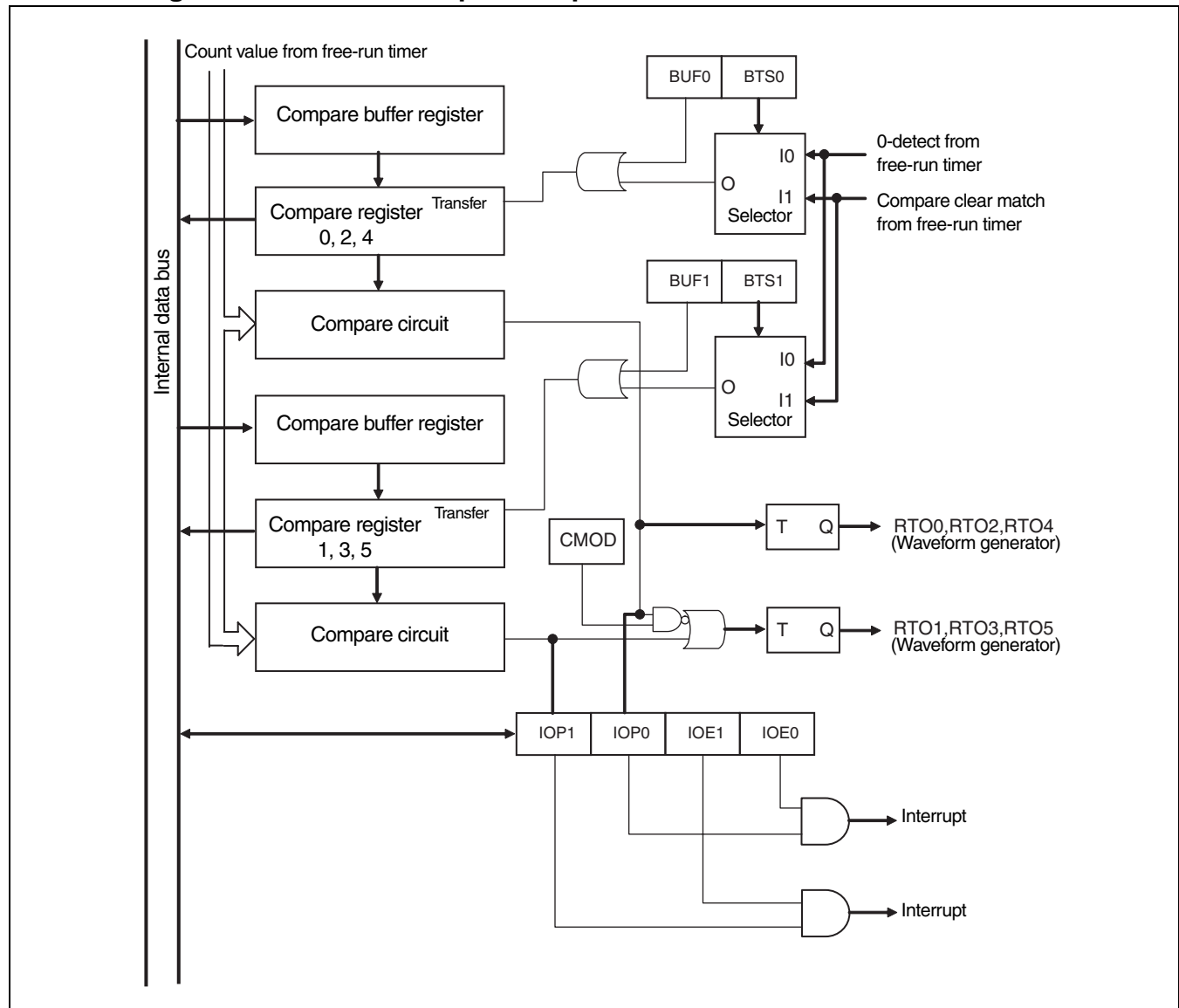


■ Block Diagram of 16-bit Free-run Timer

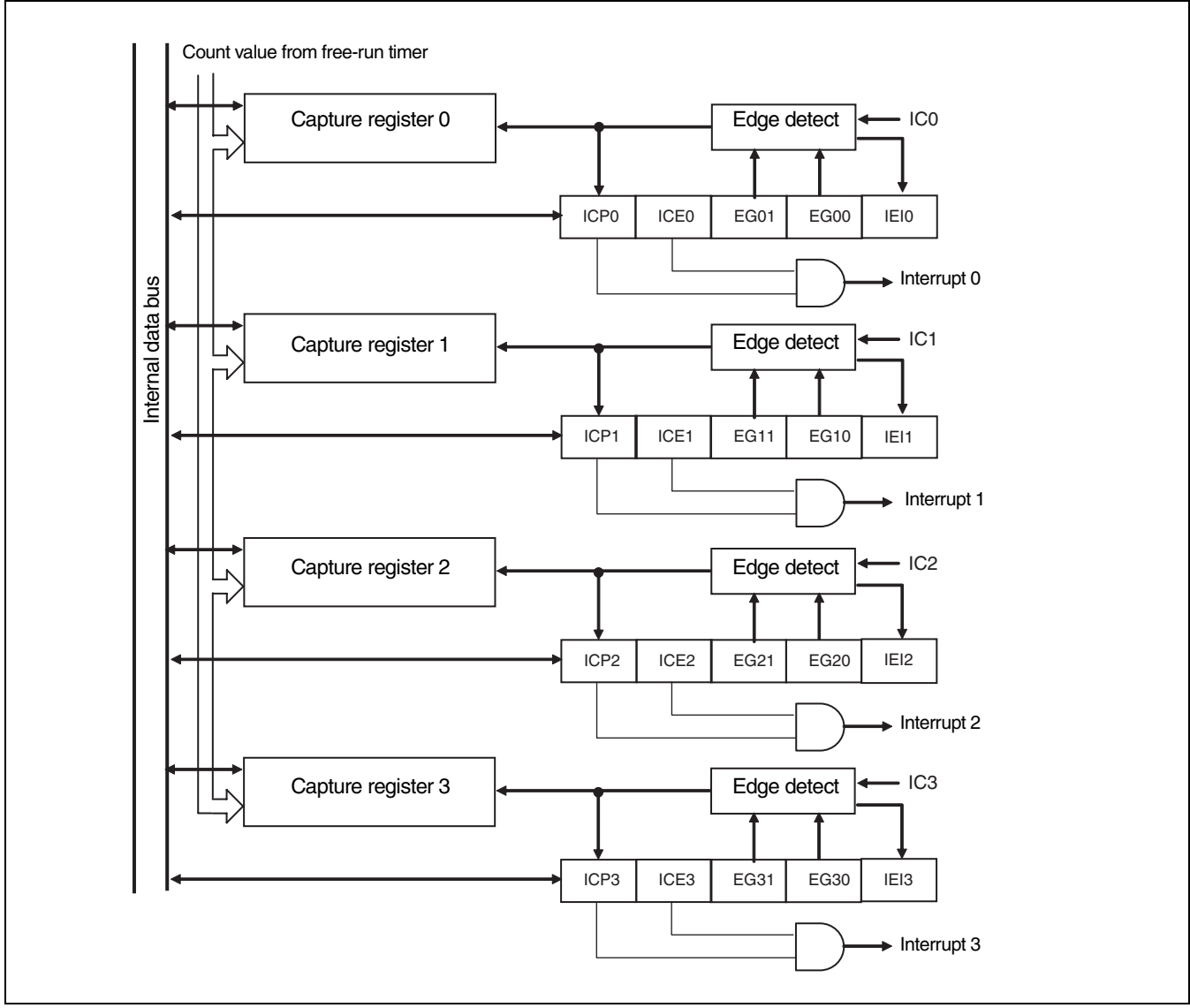




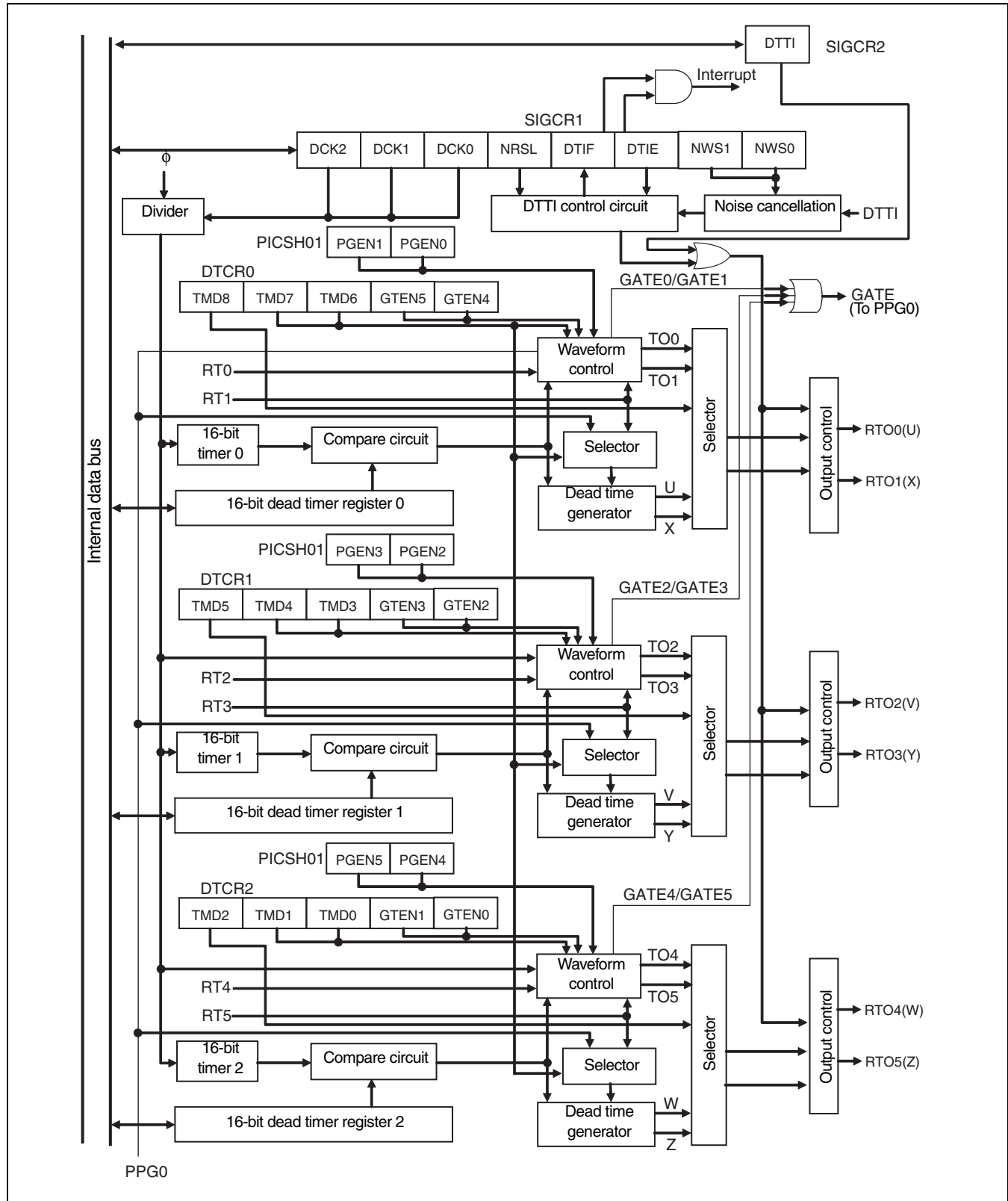
## ■ Block Diagram of the 16-bit Output Compare



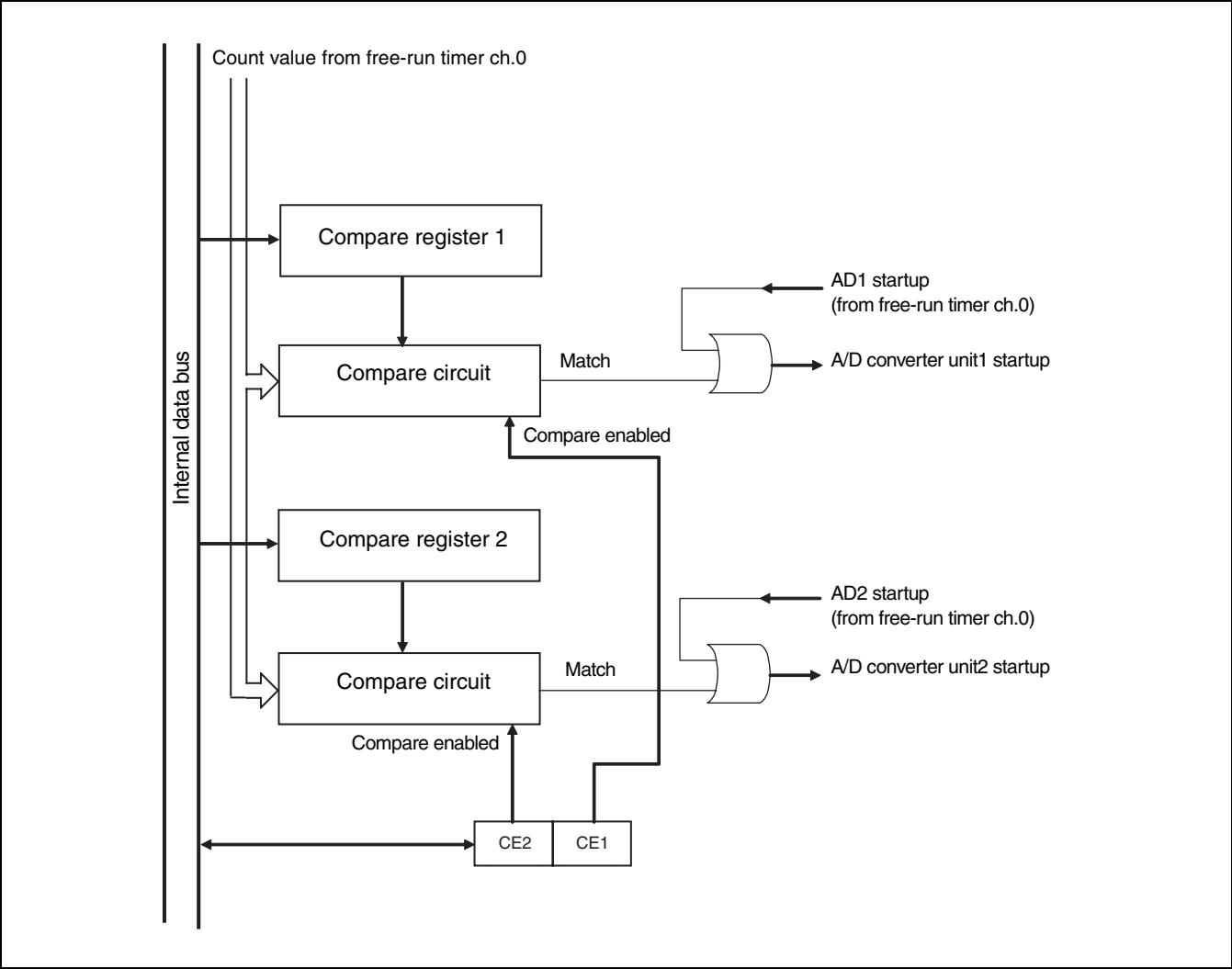
■ Block Diagram of the 16-bit Input Capture



# ■ Block Diagram of the Waveform Generator

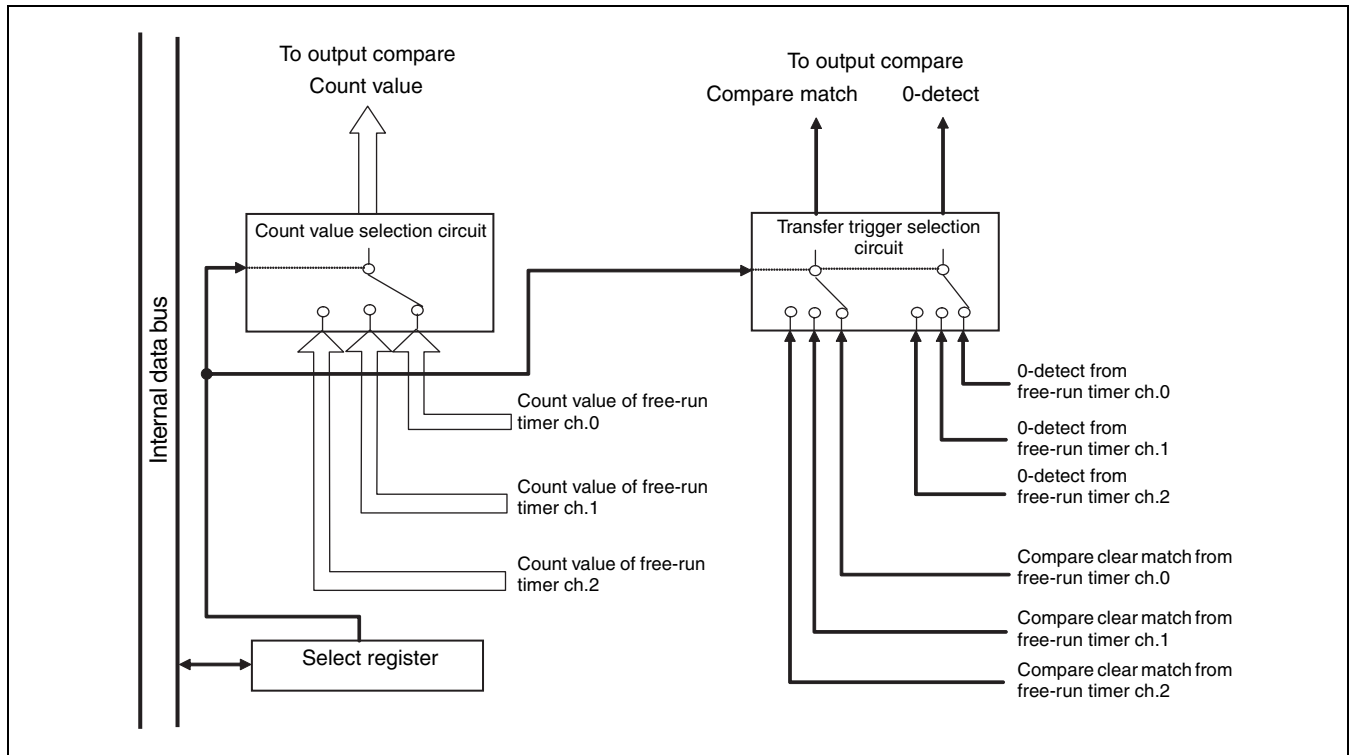


■ Block Diagram of the A/D Activation Compare

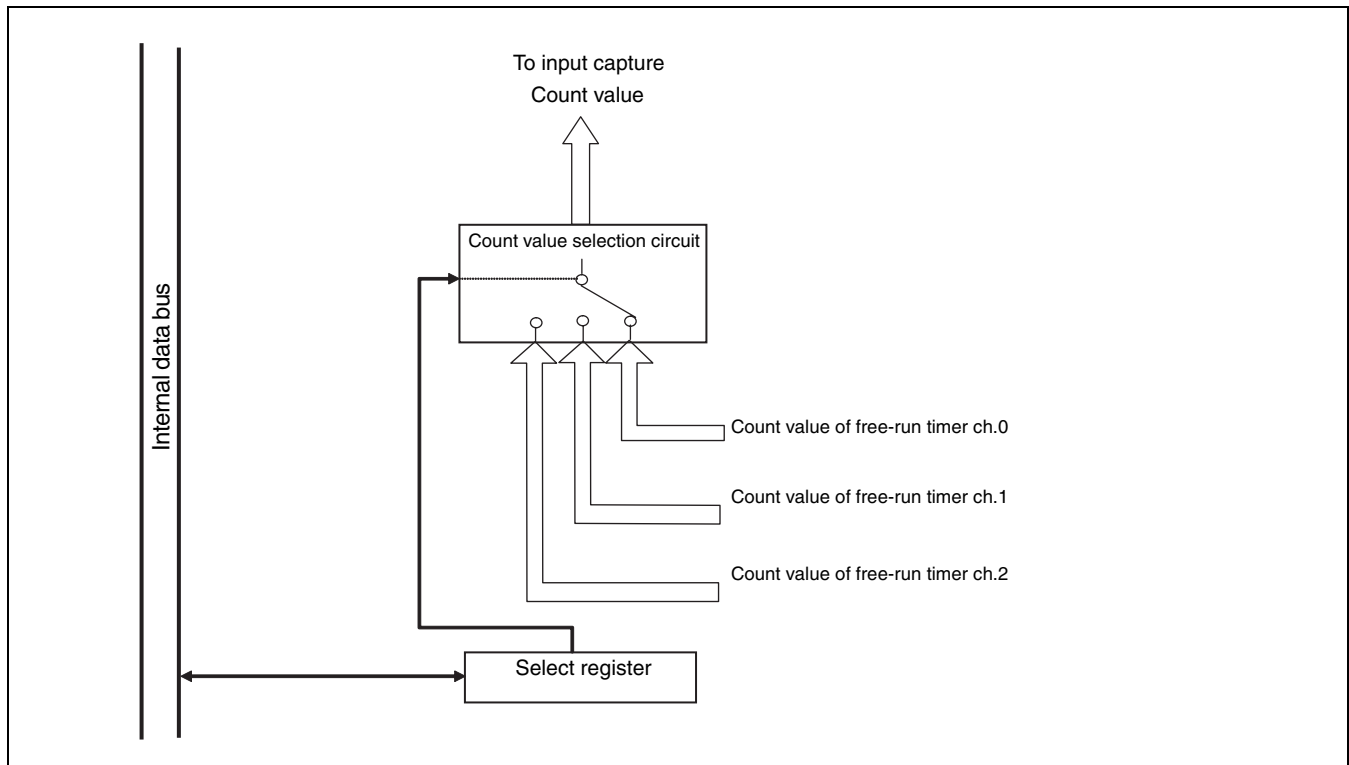


## ■ Block Diagram of the Free-run Timer Selector

- For output compare (An instance of this block exists for each output compare channel.)



- For input capture (An instance of this block exists for each input capture channel.)



## 12.3 Pins of the Multifunction Timer

This section describes the pins of the multifunction timer.

### ■ Pins of the Multifunction Timer

Table 12.3-1 Pins of the Multifunction Timer

Pin name	Pin function	I/O form	Pull-up option	Standby control	Pin setting
P23/DTTI	Port I/O 2, DTTI input	CMOS output, CMOS hysteresis input	Selectable	Yes	Set a pin as input port (DDR2: bit3=0)
P24/CKI	Port I/O 2, External clock				Set a pin as input port (DDR2: bit4=0)
P25/IC0	Port I/O 2, Input capture 0				Set a pin as input port (DDR2: bit5=0)
P26/IC1	Port I/O 2, Input capture 1				Set a pin as input port (DDR2: bit6=0)
P20/IC2/ADTG1	Port I/O 2, input capture 2, AD external trigger input 1				Set a pin as input port (DDR2: bit0=0)
P21/IC3/ADTG2	Port I/O 2, input capture 3, AD external trigger input 2				Set a pin as input port (DDR2: bit1=0)
RTO0 (U)	Port I/O 3, RTO0		None		Set RTO0 output. (OCSH1: OTE0=1)
RTO1 (X)	Port I/O 3, RTO1				Set RTO1 output. (OCSH1: OTE1=1)
RTO2 (V)	Port I/O 3, RTO2				Set RTO2 output. (OCSH3: OTE0=1)
RTO3 (Y)	Port I/O 3, RTO3				Set RTO3 output. (OCSH3: OTE1=1)
RTO4 (W)	Port I/O 3, RTO4				Set RTO4 output. (OCSH5: OTE0=1)
RTO5 (Z)	Port I/O 3, RTO5				Set RTO5 output. (OCSH5: OTE1=1)

DDR<sub>x</sub>: Port direction register

OCSH<sub>x</sub>: Compare control register

## 12.4 Multifunction Timer Register

This section describes the multifunction timer registers.

### ■ 16-bit Free-run Timer Register

Compare clear buffer register, compare clear register (Upper)

CPCLRBH0 to CPCLRBH2/CPCLRHH0 to CPCLRHH2

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
0000A4 <sub>H</sub>	CL15	CL14	CL13	CL12	CL11	CL10	CL09	CL08	11111111 <sub>B</sub>
000154 <sub>H</sub>	W	W	W	W	W	W	W	W	← CPCLRBH write
00015C <sub>H</sub>	R	R	R	R	R	R	R	R	← CPCLRHH read

Compare clear buffer register, compare clear register (Lower)

CPCLRBL0 to CPCLRBL2/CPCLRL0 to CPCLRL2

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
	CL07	CL06	CL05	CL04	CL03	CL02	CL01	CL00	11111111 <sub>B</sub>
	W	W	W	W	W	W	W	W	← CPCLRBL write
	R	R	R	R	R	R	R	R	← CPCLRL read

Timer data register (Upper)

TCDTH0 to TCDTH2

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
0000A6 <sub>H</sub>	T15	T14	T13	T12	T11	T10	T09	T08	00000000 <sub>B</sub>
000156 <sub>H</sub>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
00015E <sub>H</sub>									

Timer data register (Lower)

TCDTL0 to TCDTL2

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
	T07	T06	T05	T04	T03	T02	T01	T00	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

(Continued)

(Continued)

## Timer status control register (Upper)

TCCSH0 to TCCSH2

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
0000A8 <sub>H</sub>	ECKE	IRQZF	IRQZE	MSI2	MSI1	MSI0	ICLR	ICRE	00000000 <sub>B</sub>
000158 <sub>H</sub>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
000160 <sub>H</sub>									

## Timer status control register (Lower)

TCCSL0 to TCCSL2

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0000A9 <sub>H</sub>	BFE	STOP	MODE	SCLR	CLK3	CLK2	CLK1	CLK0	01000000 <sub>B</sub>
000159 <sub>H</sub>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
000161 <sub>H</sub>									

## A/D trigger control register

ADTRGC

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0000AB <sub>H</sub>	–	–	–	–	SEL2	SEL1	AD2E	AD1E	0XXX0000 <sub>B</sub>
	–	–	–	–	R/W	R/W	R/W	R/W	

R/W: Readable/Writable

R: Read only

W: Write only

X: Undefined value

–: Undefined



## ■ 16-bit Output Compare Register

Output compare buffer register, output compare register (Upper)

OCCPBH0 to OCCPBH5/OCCPH0 to OCCPH5

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
000090 <sub>H</sub>	OP15	OP14	OP13	OP12	OP11	OP10	OP09	OP08	00000000 <sub>B</sub>
000092 <sub>H</sub>	W	W	W	W	W	W	W	W	← OCCPBH write
000094 <sub>H</sub>	R	R	R	R	R	R	R	R	← OCCPH read
000096 <sub>H</sub>									
000098 <sub>H</sub>									
00009A <sub>H</sub>									

Output compare buffer register, output compare register (Lower)

OCCPBL0 to OCCPBL5/OCCPL0 to OCCPL5

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
000091 <sub>H</sub>	OP07	OP06	OP05	OP04	OP03	OP02	OP01	OP00	00000000 <sub>B</sub>
000093 <sub>H</sub>	W	W	W	W	W	W	W	W	← OCCPBL write
000095 <sub>H</sub>	R	R	R	R	R	R	R	R	← OCCPL read
000097 <sub>H</sub>									
000099 <sub>H</sub>									
00009B <sub>H</sub>									

Compare control register 1, 3, 5 (Upper)

OCSH1, OCSH3, OCSH5

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
00009C <sub>H</sub>	–	BTS1	BTS0	CMOD	OTE1	OTE0	OTD1	OTD0	X1100000 <sub>B</sub>
00009E <sub>H</sub>	–	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0000A0 <sub>H</sub>									

Compare control register 0, 2, 4 (Lower)

OCSL0, OCSL2, OCSL4

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00009D <sub>H</sub>	IOP1	IOP0	IOE1	IOE0	BUF1	BUF0	CST1	CST0	00000000 <sub>B</sub>
00009F <sub>H</sub>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0000A1 <sub>H</sub>									

Compare mode control register

OCMOD

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
0000A2 <sub>H</sub>	–	–	MOD15	MOD14	MOD13	MOD12	MOD11	MOD10	XX000000 <sub>B</sub>
	–	–	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable

R: Read only

W: Write only

X: Undefined value

–: Undefined

## ■ 16-bit Input Capture Register

### Input capture data register (Upper)

IPCPH0 to IPCPH3

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
0000AC <sub>H</sub>	CP15	CP14	CP13	CP12	CP11	CP10	CP09	CP08	XXXXXXXX <sub>B</sub>
0000AE <sub>H</sub>	R	R	R	R	R	R	R	R	
0000B0 <sub>H</sub>									
0000B2 <sub>H</sub>									

### Input capture data register (Lower)

IPCPL0 to IPCPL3

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0000AD <sub>H</sub>	CP07	CP06	CP05	CP04	CP03	CP02	CP01	CP00	XXXXXXXX <sub>B</sub>
0000AF <sub>H</sub>	R	R	R	R	R	R	R	R	
0000B1 <sub>H</sub>									
0000B3 <sub>H</sub>									

### Input capture status control register (ch.2, ch.3) (Upper)

ICSH23

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
0000B6 <sub>H</sub>	–	–	–	–	–	–	IEI3	IEI2	XXXXXX00 <sub>B</sub>
	–	–	–	–	–	–	R	R	

### Input capture status control register (ch.2, ch.3) (Lower)

ICSL23

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0000B7 <sub>H</sub>	ICP3	ICP2	ICE3	ICE2	EG31	EG30	EG21	EG20	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

### PPG output control/input capture status control register (ch.0, ch.1) (Upper)

PICSH01

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
0000B4 <sub>H</sub>	PGEN5	PGEN4	PGEN3	PGEN2	PGEN1	PGEN0	IEI1	IEI0	00000000 <sub>B</sub>
	W	W	W	W	W	W	R	R	

### Input capture status control register (ch.0, ch.1) (Lower)

PICSL01

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0000B5 <sub>H</sub>	ICP1	ICP0	ICE1	ICE0	EG11	EG10	EG01	EG00	00000000 <sub>B</sub>
	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

X: Undefined value

–: Undefined

## ■ Waveform Generator Register

### 16-bit dead timer register (Upper)

TMRRH0 to TMRRH2

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
0000BC <sub>H</sub>	TR15	TR14	TR13	TR12	TR11	TR10	TR09	TR08	XXXXXXXX <sub>B</sub>
0000BE <sub>H</sub>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0000C0 <sub>H</sub>									

### 16-bit dead timer register (Lower)

TMRRL0 to TMRRL2

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0000BD <sub>H</sub>	TR07	TR06	TR05	TR04	TR03	TR02	TR01	TR00	XXXXXXXX <sub>B</sub>
0000BF <sub>H</sub>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0000C1 <sub>H</sub>									

### 16-bit dead timer control register 0

DTCR0

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
0000C4 <sub>H</sub>	DMOD0	GTEN1	GTEN0	TMIF0	TMIE0	TMD2	TMD1	TMD0	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

### 16-bit dead timer control register 1

DTCR1

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0000C5 <sub>H</sub>	DMOD1	GTEN3	GTEN2	TMIF1	TMIE1	TMD5	TMD4	TMD3	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

### 16-bit dead timer control register 2

DTCR2

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
0000C6 <sub>H</sub>	DMOD2	GTEN5	GTEN4	TMIF2	TMIE2	TMD8	TMD7	TMD6	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

### Waveform control register 1

SIGCR1

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0000C9 <sub>H</sub>	DTIE	DTIF	NRSL	DCK2	DCK1	DCK0	NWS1	NWS0	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

### Waveform control register 2

SIGCR2

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0000CB <sub>H</sub>	—	—	—	—	—	—	—	DTTI	XXXXXXXX1 <sub>B</sub>
	—	—	—	—	—	—	—	R/W	

R/W: Readable/Writable

X: Undefined value

—: Undefined

## ■ A/D Activation Compare Register

Compare register 2 (Upper)

ADCOMP1, ADCOMP2

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
ch.1: 0000CE <sub>H</sub>	CMP15	CMP14	CMP13	CMP12	CMP11	CMP10	CMP09	CMP08	00000000 <sub>B</sub>
ch.2: 0000D0 <sub>H</sub>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Compare register 2 (Lower)

ADCOMP1, ADCOMP2

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
ch.1: 0000CF <sub>H</sub>	CMP07	CMP06	CMP05	CMP04	CMP03	CMP02	CMP01	CMP00	00000000 <sub>B</sub>
ch.2: 0000D1 <sub>H</sub>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Control register 1

ADCOMPC1

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0000D3 <sub>H</sub>	–	–	–	–	–	CE2	CE1	–	XXXXX00X <sub>B</sub>
	–	–	–	–	–	R/W	R/W	–	

Control register 2

ADCOMPC2

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
0000D2 <sub>H</sub>	–	–	SEL21	SEL20	SEL11	SEL10	–	–	XX0000XX <sub>B</sub>
	–	–	R/W	R/W	R/W	R/W	–	–	

R/W: Readable/Writable

X: Undefined value

–: Undefined

## ■ Free-run Timer Selector Register

FSR2

Address	bit23	bit22	bit21	bit20	bit19	bit18	bit17	bit16	Initial value
000169 <sub>H</sub>	ICU31	ICU30	ICU21	ICU20	ICU11	ICU10	ICU01	ICU00	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

FSR1

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00016A <sub>H</sub>	–	–	–	–	OCU51	OCU50	OCU41	OCU40	XXXX0000 <sub>B</sub>
	–	–	–	–	R/W	R/W	R/W	R/W	

FSR0

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00016B <sub>H</sub>	OCU31	OCU30	OCU21	OCU20	OCU11	OCU10	OCU01	OCU00	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable

X: Undefined value

–: Undefined

## 12.4.1 Compare Clear Buffer Register (CPCLRBH0 to CPCLRBH2, CPCLRBL0 to CPCLRBL2) / Compare Clear Register (CPCLRH0 to CPCLRH2, CPCLRL0 to CPCLRL2)

The compare clear buffer register (CPCLRBH, CPCLRBL) is a 16-bit buffer register which exists in the compare clear register (CPCLRH, CPCLRL). Both the register (CPCLRBH, CPCLRBL) and the register (CPCLRH, CPCLRL) exist in the same address.

### ■ Compare Clear Buffer Register (CPCLRBH0 to CPCLRBH2, CPCLRBL0 to CPCLRBL2)

Compare clear buffer register (Upper)

CPCLRBH0 to CPCLRBH2

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
0000A4 <sub>H</sub>	CL15	CL14	CL13	CL12	CL11	CL10	CL09	CL08	11111111 <sub>B</sub>
000154 <sub>H</sub>	W	W	W	W	W	W	W	W	
00015C <sub>H</sub>									

Compare clear buffer register (Lower)

CPCLRBL0 to CPCLRBL2

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
	CL07	CL06	CL05	CL04	CL03	CL02	CL01	CL00	11111111 <sub>B</sub>
	W	W	W	W	W	W	W	W	

W: Write only

The compare clear buffer register is a buffer register which exists in the same address of the compare clear register (CPCLRH, CPCLRL). When the buffer function is disabled (the lower of the timer state control register (TCCSL), BFE: bit7 = 0), or the free-run timer is stopped, the value of the compare clear buffer register is transferred to the compare clear register immediately. If the buffer function is enabled, the value is transferred to the compare clear register when the count value 0 of the 16-bit free-run timer is detected.

To access this register, use a half-word or word access instruction.

## ■ Compare Clear Register (CPCLR<sub>H0</sub> to CPCLR<sub>H2</sub>, CPCLR<sub>L0</sub> to CPCLR<sub>L2</sub>)

Compare clear register (Upper)

CPCLR<sub>H0</sub> to CPCLR<sub>H2</sub>

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
0000A4 <sub>H</sub>	CL15	CL14	CL13	CL12	CL11	CL10	CL09	CL08	11111111 <sub>B</sub>
000154 <sub>H</sub>	R	R	R	R	R	R	R	R	
00015C <sub>H</sub>									

Compare clear register (Lower)

CPCLR<sub>L0</sub> to CPCLR<sub>L2</sub>

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
	CL07	CL06	CL05	CL04	CL03	CL02	CL01	CL00	11111111 <sub>B</sub>
	R	R	R	R	R	R	R	R	

R: Read only

The compare clear register is used to compare with the count value of the 16-bit free-run timer. In up-count mode, the 16-bit free-run timer is reset to "0000<sub>H</sub>" when the 16-bit free-run timer count value matches the value in this register. In up/down count mode, the 16-bit free-run timer changes its mode from the up counting to the down counting when this register matches the count value of the 16-bit free-run timer; or changes from the down counting to up counting when a 0-detection occurs.

To access this register, use a half-word or word access instruction.

# 12.4.2 Timer Data Register (TCDTH0 to TCDTH2, TCDTL0 to TCDTL2)

The timer data register (TCDTH, TCDTL) is used to read the count value of the 16-bit free-run timer.

## ■ Timer Data Register (TCDTH0 to TCDTH2, TCDTL0 to TCDTL2)

Timer data register (Upper)									
TCDTH0 to TCDTH2									
Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
0000A6 <sub>H</sub>	T15	T14	T13	T12	T11	T10	T09	T08	00000000 <sub>B</sub>
000156 <sub>H</sub>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
00015E <sub>H</sub>									
Timer data register (Lower)									
TCDTL0 to TCDTL2									
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
	T07	T06	T05	T04	T03	T02	T01	T00	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
R/W: Readable/Writable									

The timer data register is used to read the count value of the 16-bit free-run timer. The count value is cleared to "0000<sub>H</sub>" immediately when a reset occurs. Writing the value to this register can be set the timer value. However, write a value while the timer is stopped (the lower of the timer state control register (TCCSL), STOP: bit6 = 1). To access the timer data register, use a half-word or word access instruction.

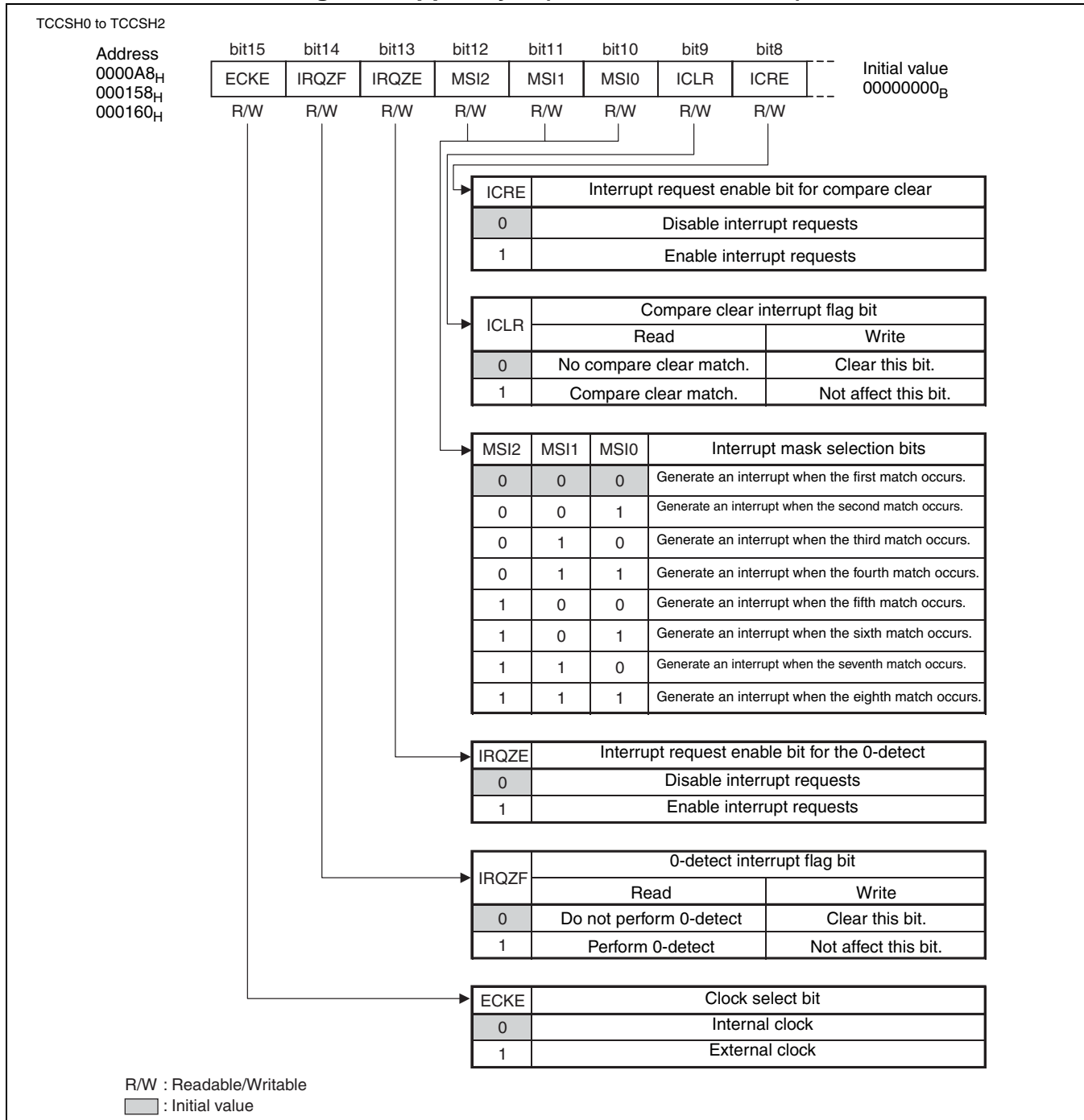
16-bit free-run timer is initialized immediately when the following factors occur.

- Reset
- Clear bit of the timer state control register (TCCSL) (SCLR: bit4 = 1)
- Match of the compare clear register and the timer count value in the up count mode (the lower of the timer state control register (TCCSL), MODE: bit5 = 0)

### 12.4.3 Timer State Control Register (TCCSH0 to TCCSH2, TCCSL0 to TCCSL2)

The timer state control register (TCCSH, TCCSL) is a 16-bit register which is used to control the operation of the 16-bit free-run timer.

#### ■ Timer State Control Register, Upper Byte (TCCSH0 to TCCSH2)





**Table 12.4-1 Timer State Control Register, Upper Byte (TCCSH) (1 / 2)**

Bit name		Function
bit15	ECKE: Clock select bit	<ul style="list-style-type: none"> <li>This bit is used to select the internal clock or the external clock as a count clock of the 16-bit free-run timer.</li> <li>Setting this bit to "0" selects the internal clock. To select a count clock frequency, you also need to select a clock frequency select bits of the TCCSL register (CLK3 to CLK0: bit3 to bit0).</li> <li>Setting this bit to "1" selects the external clock. The external clock is input from CKI pins. Therefore, write "0" to the bit7 of the port direction register (DDR1) to enable the external clock input.</li> </ul> <p>Note: The count clock is changed immediately when this bit is set. Therefore, this bit must be changed when the output compare and the input capture are stopped.</p>
bit14	IRQZF: 0-detect interrupt flag bit	<ul style="list-style-type: none"> <li>When the count value of the 16-bit free-run timer is 0000<sub>H</sub>, this bit is set to "1".</li> <li>When this bit is set to "0": Clears the bit.</li> <li>Setting this bit to "1" has no effect on this bit.</li> <li>When this bit is read to a read modify write instruction, "1" is always read.</li> </ul> <p>Note: This bit is not set by a software clear (writing "1" to bit4 (SCLR) of the lower timer status control register (TCCSL)). In up/down count mode (bit5 (MODE) in the lower timer state control register (TCCSL) = 1), this bit is set to "1" when an interrupt set in the interrupt mask selection bits (bit12 to bit10 (MSI2 to MSI0) in the upper timer state control register (TCCSH) <math>\neq</math> 000<sub>B</sub>) occurs. When no interrupt occurs, this bit is not set to "1". In the up count mode (MODE: bit5 = 0), this bit is set every time the 0-detection occurs regardless of the value of the MSI2 to MSI0: bit12 to bit10.</p>
bit13	IRQZE: Interrupt request enable bit for the 0-detect	When this bit and the interrupt flag bit (IRQZF: bit14) are set to "1", an interrupt request to the CPU can be generated.
bit12 to bit10	MSI2 to MSI0: Interrupt mask selection bits	<ul style="list-style-type: none"> <li>These bits are used to specify the number of times to mask the compare clear interrupt in up-count mode (MODE=0). In up/down count mode (MODE = 1), they are used to set the number of masking of the 0-detection interrupt.</li> <li>Setting this bit to "0" does not mask the interrupt source.</li> </ul> <p>Note: When the interrupt source has been masked twice, and then the third interrupt is processed, these bits must be set to "010<sub>B</sub>". Reading returns the value of the mask counter.</p>

**Table 12.4-1 Timer State Control Register, Upper Byte (TCCSH) (2 / 2)**

Bit name		Function
bit9	ICLR: Compare clear interrupt flag bit	<ul style="list-style-type: none"> <li>• This bit is set to "1" when the value of the compare clear matches the value of the 16-bit free-run timer.</li> <li>• When this bit is set to "0": Clears the bit.</li> <li>• Setting this bit to "1" has no effect on this bit.</li> <li>• When this bit is read to a read modify write instruction, "1" is always read.</li> </ul> <p>Note: In up-count mode (bit5 (MODE) in the lower timer status control register (TCCSL) = 0), this bit is set to "1" when an interrupt set in the interrupt mask selection bits occurs. When no interrupt occurs, this bit is not set to "1". In the up/down count mode (MODE = 1), this bit is set every time a compare clear occurs regardless of the value of the MSI2 to MSI0.</p>
bit8	ICRE: Interrupt request enable bit for compare clear	An interrupt request to the CPU can be generated when this bit and the compare clear interrupt flag bit (ICLR: bit9) is set to "1".

## ■ Timer State Control Register, Lower Byte (TCCSL0 to TCCSL2)

TCCSL0 to TCCSL2

Address  
0000A9<sub>H</sub>  
000159<sub>H</sub>  
000161<sub>H</sub>

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
BFE	STOP	MODE	SCLR	CLK3	CLK2	CLK1	CLK0	01000000 <sub>B</sub>
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

CLK3	CLK2	CLK1	CLK0	Clock frequency select bits					
				Count Clock	$\phi=32\text{MHz}$	$\phi=16\text{MHz}$	$\phi=8\text{MHz}$	$\phi=4\text{MHz}$	$\phi=1\text{MHz}$
0	0	0	0	$\phi$	31.25ns	62.5ns	125ns	0.25 $\mu\text{s}$	1 $\mu\text{s}$
0	0	0	1	$\phi/2$	62.5ns	125ns	0.25ms	0.5 $\mu\text{s}$	2 $\mu\text{s}$
0	0	1	0	$\phi/4$	125ns	0.25 $\mu\text{s}$	0.5 $\mu\text{s}$	1 $\mu\text{s}$	4 $\mu\text{s}$
0	0	1	1	$\phi/8$	0.25 $\mu\text{s}$	0.5 $\mu\text{s}$	1 $\mu\text{s}$	2 $\mu\text{s}$	8 $\mu\text{s}$
0	1	0	0	$\phi/16$	0.5 $\mu\text{s}$	1 $\mu\text{s}$	2 $\mu\text{s}$	4 $\mu\text{s}$	16 $\mu\text{s}$
0	1	0	1	$\phi/32$	1 $\mu\text{s}$	2 $\mu\text{s}$	4 $\mu\text{s}$	8 $\mu\text{s}$	32 $\mu\text{s}$
0	1	1	0	$\phi/64$	2 $\mu\text{s}$	4 $\mu\text{s}$	8 $\mu\text{s}$	16 $\mu\text{s}$	64 $\mu\text{s}$
0	1	1	1	$\phi/128$	4 $\mu\text{s}$	8 $\mu\text{s}$	16 $\mu\text{s}$	32 $\mu\text{s}$	128 $\mu\text{s}$
1	0	0	0	$\phi/256$	8 $\mu\text{s}$	16 $\mu\text{s}$	32 $\mu\text{s}$	64 $\mu\text{s}$	256 $\mu\text{s}$
The other setting disabled				—	—	—	—	—	—

 $\phi$  : Machine cycle

SCLR	Timer clear bit	
	Read	Write
0	Reading always returns "0".	Do not initialize counter
1		Initialize counter to "0000 <sub>H</sub> "

MODE	Timer count mode bit
0	Up-count mode
1	Up/down count mode

STOP	Timer enable bit
0	Enable counting (Start counting).
1	Disable counting (Stop counting).

BFE	Compare clear buffer enable bit
0	Disable compare clear buffer
1	Enable compare clear buffer

R/W : Readable/Writable  
 : Initial value

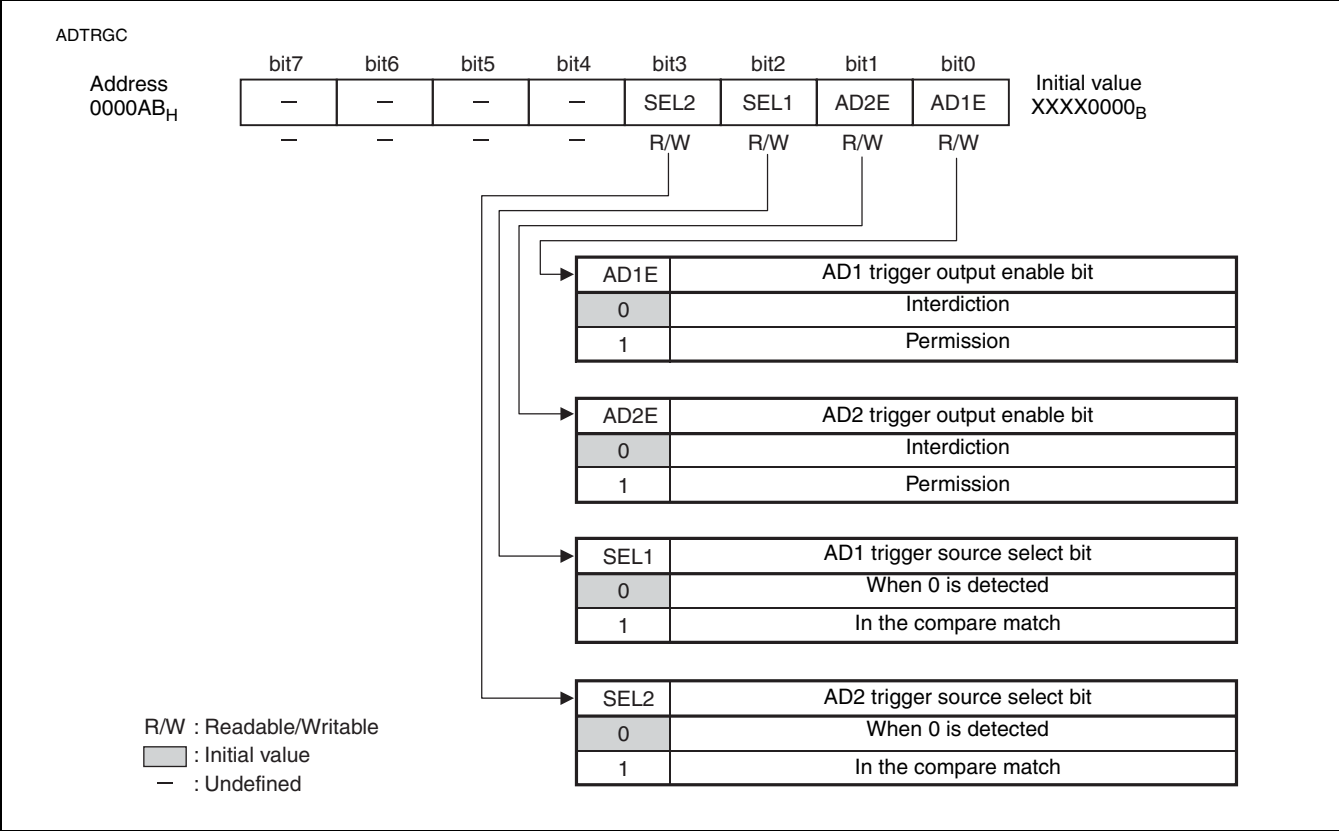
**Table 12.4-2 Timer State Control Register, Lower Byte (TCCSL)**

Bit name		Function
bit7	BFE: Compare clear buffer enable bit	<ul style="list-style-type: none"> <li>This bit is used to enable a compare clear buffer.</li> <li>Setting this bit to "0" disables the compare clear buffer. Accordingly, you can write directly to the compare clear registers (CPCLRH and CPCLRL).</li> <li>Setting this bit to "1" enables the compare clear buffer. The data written to and stored in the compare clear buffer is transferred to the compare clear register when a count value of "0" is detected in the 16-bit free-run timer.</li> </ul>
bit6	STOP: Timer enable bit	<ul style="list-style-type: none"> <li>This bit is used to start/stop the 16-bit free-run timer counting.</li> <li>Setting this bit to "0" starts the 16-bit free-run timer counting.</li> <li>Setting this bit to "1" stops the 16-bit free-run timer counting.</li> </ul> <p>Note: When the 16-bit free-run timer stops, the operation of the output compare also stops.</p>
bit5	MODE: Timer count mode bit	<ul style="list-style-type: none"> <li>This bit is used to select a count mode of the 16-bit free-run timer.</li> <li>Selecting this bit to "0" selects the up-count mode. The timer continues to perform incremental counting until the count value matches a compare clear register and is reset "0000<sub>H</sub>". Then, the timer restarts to perform incremental counting.</li> <li>Setting this bit to "1" selects the up/down count mode. The timer continues to perform incremental counting until the count value matches a compare clear register. Then, the mode changes to the down count. After that, the mode changes to the up count again when the count value reaches to "0000<sub>H</sub>".</li> <li>This bit can be written even if the timer is operating or stopped. When the timer is running, the value written to this bit is stored in a buffer and the count mode changes based on the buffer value the next time the timer value goes to "0000<sub>H</sub>".</li> </ul>
bit4	SCLR: Timer clear bit	<ul style="list-style-type: none"> <li>This bit is used to initialize the 16-bit free-run timer to "0000<sub>H</sub>".</li> <li>When this bit is set to "1": The 16-bit free-run timer is initialized to "0000<sub>H</sub>" at the next count clock.</li> <li>The read value is always "0".</li> </ul> <p>Note: Writing "1" to this bit does not generate a 0-detect interrupt. No timer clear is performed if you set "1" and then write "0" before the next count clock.</p>
bit3 to bit0	CLK3 to CLK0: Clock frequency select bits	<ul style="list-style-type: none"> <li>These bits are used to select a count clock frequency of the 16-bit free-run timer.</li> <li>The count clock is changed immediately when these bits are set. Therefore, these bits must be changed when the output compare and the input capture are stopped.</li> </ul>

### 12.4.4 A/D Trigger Control Register (ADTRGC)

Controls A/D trigger signal output when a free-run timer compare match or 0-detect occurs.

■ A/D Trigger Control Register (ADTRGC)



**Table 12.4-3 A/D Trigger Control Register (ADTRGC)**

Bit name		Function
bit7 to bit4	Undefined bits	<ul style="list-style-type: none"> <li>The read value is indeterminate.</li> <li>Writing to these bits has no effect on operation.</li> </ul>
bit3	SEL2: AD2 trigger source select bit	This bit is the select bit if AD2 trigger is output when a 0-detection of the free-run timer or a compare match occurs.
bit2	SEL1: AD1 trigger source select bit	This bit is the select bit if AD1 trigger is output when a 0-detection of the free-run timer or a compare match occurs.
bit1	AD2E: AD2 trigger enable bit	<ul style="list-style-type: none"> <li>When this bit is set to "0", the AD2 trigger signal is not output.</li> <li>When this bit is set to "1", the AD2 trigger signal can be output.</li> </ul>
bit0	AD1E: AD1 trigger enable bit	<ul style="list-style-type: none"> <li>When this bit is set to "0", the AD1 trigger signal is not output.</li> <li>When this bit is set to "1", the AD1 trigger signal can be output.</li> </ul>

## 12.4.5 Output Compare Buffer Register (OCCPBH0 to OCCPBH5, OCCPBL0 to OCCPBL5) /Output Compare Register (OCCPH0 to OCCPH5, OCCPL0 to OCCPL5)

The output compare buffer register (OCCPBH, OCCPBL) is a 16-bit buffer register for the output compare register (OCCPH, OCCPL).

Both the register (OCCPBH, OCCPBL) and the register (OCCPH, OCCPL) exist in the same address.

### ■ Output Compare Buffer Register (OCCPBH0 to OCCPBH5, OCCPBL0 to OCCPBL5)

Output compare buffer register (Upper)

OCCPBH0 to OCCPBH5

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
000090 <sub>H</sub>	OP15	OP14	OP13	OP12	OP11	OP10	OP09	OP08	00000000 <sub>B</sub>
000092 <sub>H</sub>	W	W	W	W	W	W	W	W	
000094 <sub>H</sub>									
000096 <sub>H</sub>									
000098 <sub>H</sub>									
00009A <sub>H</sub>									

Output compare buffer register (Lower)

OCCPBL0 to OCCPBL5

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
	OP07	OP06	OP05	OP04	OP03	OP02	OP01	OP00	00000000 <sub>B</sub>
	W	W	W	W	W	W	W	W	

W: Write only

The output compare buffer register is a buffer register for the output compare register (OCCPH, OCCPL). When the buffer function is disabled (the lower of the compare control register (OCSL0, OCSL2, OCSL4), BUF1, BUF0: bit3, bit2 = 11<sub>B</sub>), or the free-run timer is stopped, the value of the output compare buffer register is transferred to the output compare register immediately. When the buffer function is enabled (the lower of the compare control register (OCSL0, OCSL2, OCSL4), BUF1, BUF0: bit3, bit2=00<sub>B</sub>), the value is transferred when the compare clear match or 0-detect occurs according to the transfer selection bits (BTS1, BTS0: bit14, bit13) in the upper of the compare control register (OCSH1, OCSH3, OCSH5).

To access this register, use a half-word or word access instruction.

The free-run timer in the above explanation refers to the operation of the free-run timer selected for output compare.

## ■ Output Compare Register (OCCPH0 to OCCPH5, OCCPL0 to OCCPL5)

### Output compare register (Upper)

#### OCCPH0 to OCCPH5

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
000090 <sub>H</sub>	OP15	OP14	OP13	OP12	OP11	OP10	OP09	OP08	00000000 <sub>B</sub>
000092 <sub>H</sub>	R	R	R	R	R	R	R	R	
000094 <sub>H</sub>									
000096 <sub>H</sub>									
000098 <sub>H</sub>									
00009A <sub>H</sub>									

### Output compare register (Lower)

#### OCCPL0 to OCCPL5

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
000091 <sub>H</sub>	OP07	OP06	OP05	OP04	OP03	OP02	OP01	OP00	00000000 <sub>B</sub>
000093 <sub>H</sub>	R	R	R	R	R	R	R	R	
000095 <sub>H</sub>									
000097 <sub>H</sub>									
000099 <sub>H</sub>									
00009B <sub>H</sub>									

R: Read only

The output compare register is a 16-bit register used to compare with the count value of the 16-bit free-run timer. Set the value of the output compare buffer register (OCCPBH, OCCPBL) before the timer operation is enabled.

When the value of the output compare register matches the count value of the 16-bit free-run timer, a compare signal is generated and the output compare interrupt flag bit (the lower of the compare control register (OCSL0, OCSL2, OCSL4), IOP1, IOP0: bit7, bit6) is set. When the output level is set (the upper of the compare control register (OCSH1, OCSH3, OCSH5), ODT1, ODT0: bit9, bit8), an output level waveform generator (RTO0 to RTO5) corresponding to the output compare register (OCCPH0 to OCCPH5, OCCPL0 to OCCPL5) can be reversed.

The compare signal is not generated when the value of this register matches the peak value while the 16-bit free-run timer is in up/down count mode.

### ● Up/down mode

- In CMOD = 0

When this register is set to "FFFF<sub>H</sub>", the RT output goes to "0" regardless of the 16-bit free-run timer value and inversion mode. The output goes to "1" when "0000<sub>H</sub>" is set.

- In CMOD = 1

When this register is set to "FFFF<sub>H</sub>", the RT output goes to "1" regardless of the 16-bit free-run timer value and inversion mode. The output goes to "0" when "0000<sub>H</sub>" is set.

To access this register, use a half-word or word access instruction.

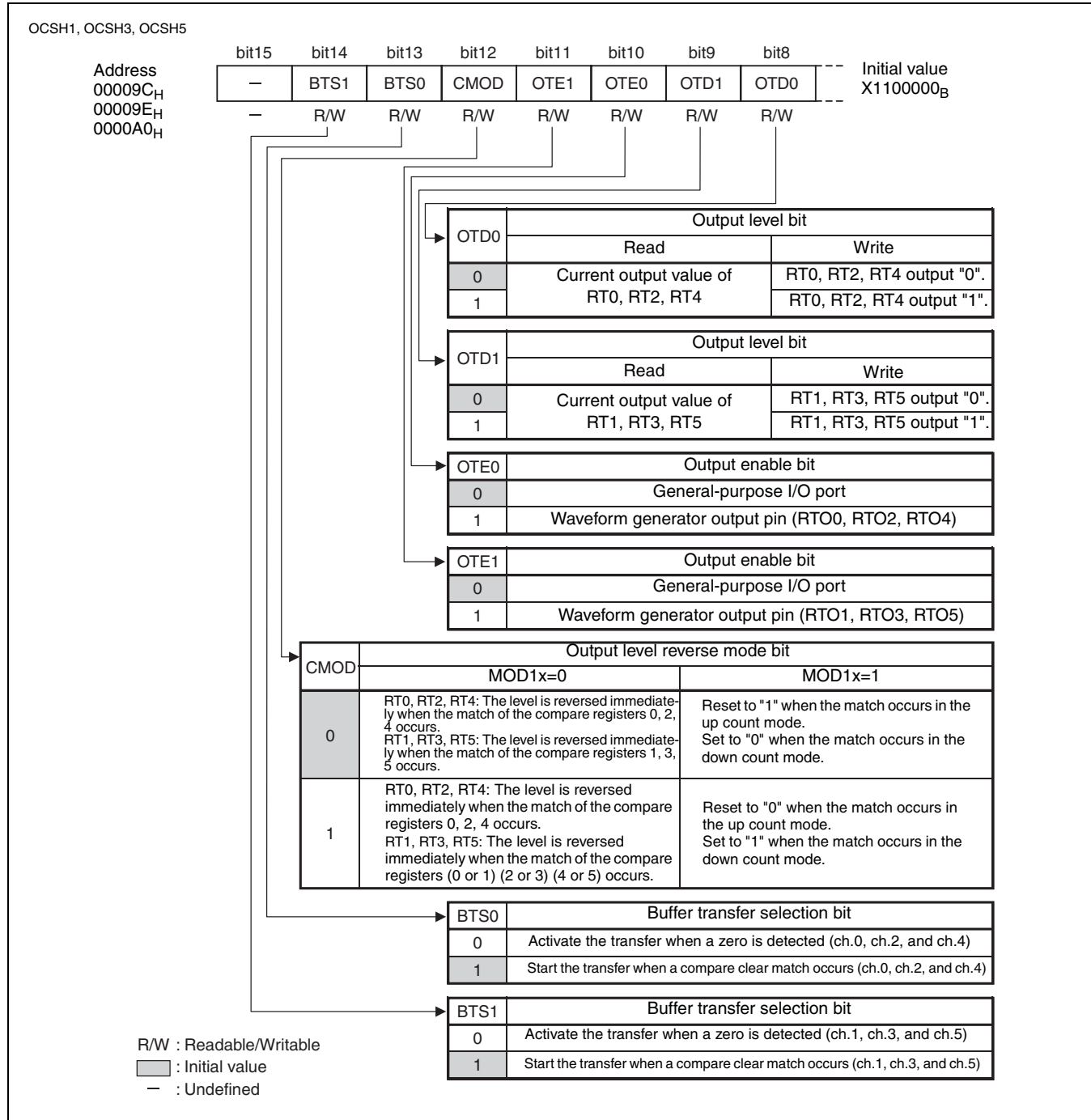
The free-run timer in the above explanation refers to the operation of the free-run timer selected for output compare.



## 12.4.6 Compare Control Register (OCSH0 to OCSH5, OCSL0 to OCSL5)

The compare control register is used to control the output level, output enable, output level reverse mode, compare operation enable, compare match interrupt enable, and compare match interrupt flag of RT0 to RT5.

### ■ Compare Control Register, Upper Byte (OCSH1, OCSH3, OCSH5)



**Table 12.4-4 Compare Control Register, Upper Byte (OCSH1, OCSH3, OCSH5) (1 / 2)**

Bit name		Function
bit15	Undefined bit	<ul style="list-style-type: none"> <li>The read value is indeterminate.</li> <li>Writing to this bit has no effect on operation.</li> </ul>
bit14	BTS1: Buffer transfer selection bit	<ul style="list-style-type: none"> <li>This bit is used to select the time when the data is transferred from the output compare buffer registers (OCCPBH1, OCCPBH3, OCCPBH5, OCCPBL1, OCCPBL3, OCCPBL5) to the output compare registers (OCCPH1, OCCPH3, OCCPH5, OCCPL1, OCCPL3, OCCPL5).</li> <li>Setting this bit to "0" starts the data transfer when the count value "0" of the 16-bit free-run timer is detected.</li> <li>Setting this bit to "1" starts the data transfer when a compare clear match of the 16-bit free-run timer occurs.</li> </ul>
bit13	BTS0: Buffer transfer selection bit	<ul style="list-style-type: none"> <li>This bit is used to select the time when the data is transferred from the output compare buffer registers (OCCPBH0, OCCPBH2, OCCPBH4, OCCPBL0, OCCPBL2, OCCPBL4) to the output compare registers (OCCPH0, OCCPH2, OCCPH4, OCCPL0, OCCPL2, OCCPL4).</li> <li>Setting this bit to "0" starts the data transfer when the count value "0" of the 16-bit free-run timer is detected.</li> <li>Setting this bit to "1" starts the data transfer when a compare clear match of the 16-bit free-run timer occurs.</li> </ul>
bit12	CMOD: Output level reverse mode bit	<ul style="list-style-type: none"> <li>This bit is used to change the pin output level inversion mode immediately after a match occurs when pin output is enabled (OTE1=1 or OTE0=1).</li> <li>When this bit is set to "0": <ul style="list-style-type: none"> <li><b>The compare mode control register (OCMOD): MOD1x = 0</b> <ul style="list-style-type: none"> <li>RT0, RT2, RT4: The level is reversed immediately when the compare registers 0, 2, 4 match the 16-bit free-run timer.</li> <li>RT1, RT3, RT5: The level is reversed immediately when the compare registers 1, 3, 5 match the 16-bit free-run timer.</li> </ul> </li> <li><b>The compare mode control register (OCMOD): MOD1x = 1</b> <ul style="list-style-type: none"> <li>Set to "1" when the match occurs in the up-count mode.</li> <li>Reset to "0" when the match occurs in the down-count mode.</li> </ul> </li> </ul> </li> <li>When this bit is set to "1": <ul style="list-style-type: none"> <li><b>The compare mode control register (OCMOD): MOD1x = 0</b> <ul style="list-style-type: none"> <li>RT0, RT2, RT4: The level is reversed immediately when the compare registers 0, 2, 4 match the 16-bit free-run timer.</li> <li>RT1, RT3, RT5: The level is reversed immediately when the compare registers (0 or 1) (2 or 3) (4 or 5) match the 16-bit free-run timer.</li> <li>When the value of the compare register 0, 2, 4 and 1, 3, 5 is the same, the operation is the same operation as only one compare register is used.</li> </ul> </li> <li><b>The compare mode control register (OCMOD): MOD1x = 1</b> <ul style="list-style-type: none"> <li>Reset to "0" when the match occurs in the up-count mode.</li> <li>Set to "1" when the match occurs in the down-count mode.</li> </ul> </li> </ul> </li> </ul>

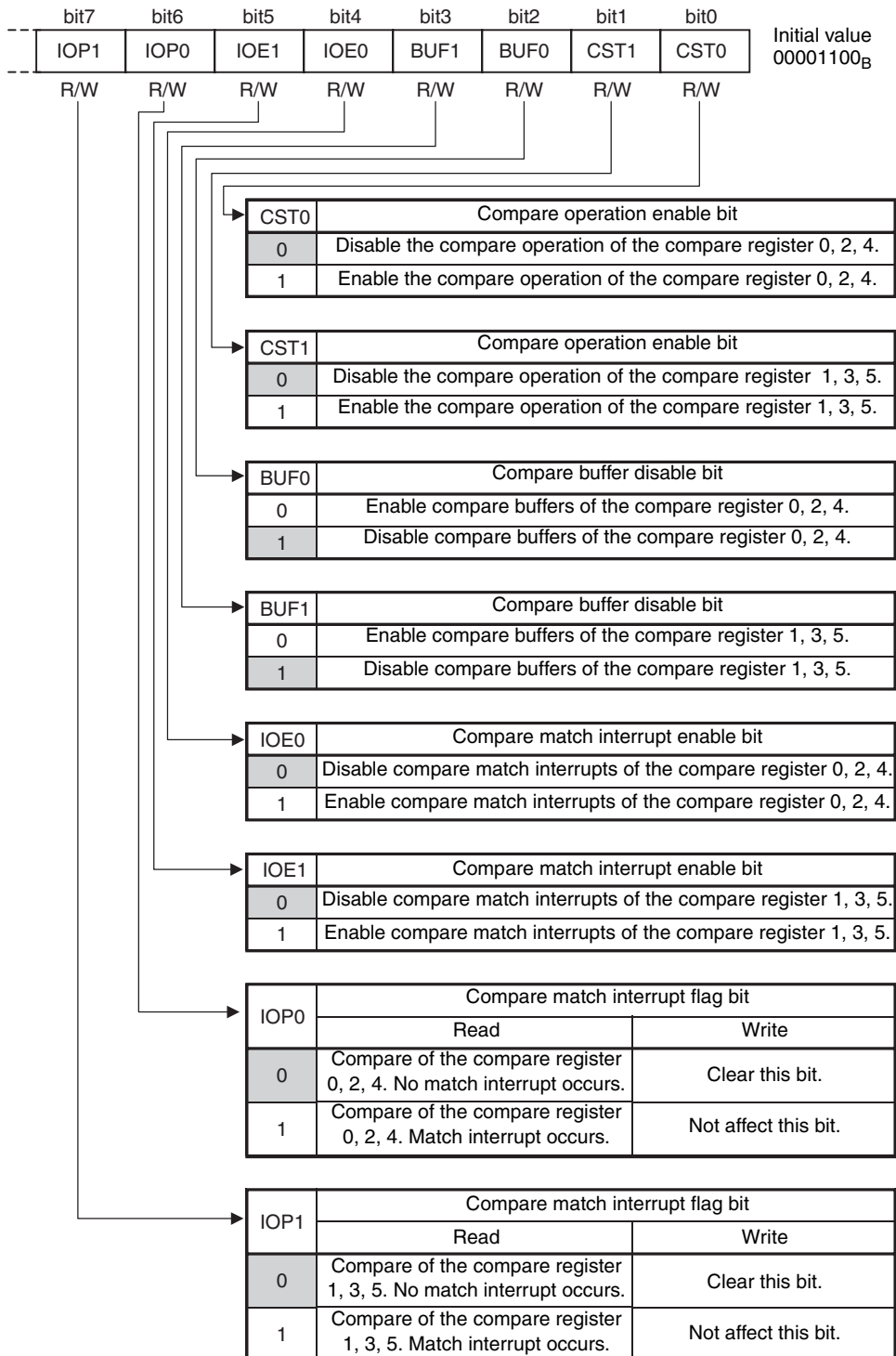
**Table 12.4-4 Compare Control Register, Upper Byte (OCSH1, OCSH3, OCSH5) (2 / 2)**

Bit name		Function
bit11	OTE1: Output enable bit	<ul style="list-style-type: none"> <li>This bit is used to enable the waveform generator output (RTO1, RTO3, RTO5) to ports.</li> <li>The initial value of this bit is "0".</li> </ul> <p>Note: When the waveform generator is disabled (the lower of the 16-bit dead timer control register (DTCR0, DTCR1, DTCR2), TMD2 to TMD0, TMD5 to TMD3, TMD8 to TMD6: bit2 to bit0 = 000<sub>B</sub>), RTO1, RTO3, RTO5 output the same value as the output compare.</p>
bit10	OTE0: Output enable bit	<ul style="list-style-type: none"> <li>This bit is used to enable the waveform generator output (RTO0, RTO2, RTO4) to ports.</li> <li>The initial value of this bit is "0".</li> </ul> <p>Note: When the waveform generator is disabled (the lower of the 16-bit dead timer control register (DTCR0, DTCR1, DTCR2), TMD2 to TMD0, TMD5 to TMD3, TMD8 to TMD6: bit2 to bit0 = 000<sub>B</sub>), RTO0, RTO2, RTO4 output the same value as the output compare.</p>
bit9	OTD1: Output level bit	<ul style="list-style-type: none"> <li>This bit is used to change a pin output level of the output compare 1, 3, 5 (RT1, RT3, RT5).</li> <li>The initial value of the compare pin output is "0".</li> <li>Always halt compare operation before writing to the value of this bit. The read value of this bit indicates the output compare value of RT1, RT3, RT5.</li> </ul>
bit8	OTD0: Output level bit	<ul style="list-style-type: none"> <li>This bit is used to change a pin output level of the output compare 0, 2, 4 (RT0, RT2, RT4).</li> <li>The initial value of the compare pin output is "0".</li> <li>Always halt compare operation before writing to the value of this bit. The read value of this bit indicates the output compare value of RT0, RT2, RT4.</li> </ul>

## ■ Compare Control Register, Lower Byte (OCSL0, OCSL2, OCSL4)

OCSL0, OCSL2, OCSL4

Address  
00009D<sub>H</sub>  
00009F<sub>H</sub>  
0000A1<sub>H</sub>



R/W : Readable/Writable

 : Initial value

**Table 12.4-5 Compare Control Register, Lower Byte (OCSL0, OCSL2, OCSL4) (1 / 2)**

Bit name		Function
bit7	IOP1: Compare match interrupt flag bit	<ul style="list-style-type: none"> <li>• This bit is an interrupt flag which indicates that the compare register 1, 3, 5 matches the value of the 16-bit free-run timer.</li> <li>• This bit is set to "1" when the value of the compare register matches the value of the 16-bit free-run timer.</li> <li>• When this bit is set while the compare match interrupt enable bit (IOE1: bit5) is enabled, the output compare interrupt occurs.</li> <li>• When this bit is set to "0": Clears the bit.</li> <li>• Setting this bit to "1" has no effect on this bit.</li> <li>• When this bit is read to a read modify write instruction, "1" is always read.</li> </ul>
bit6	IOP0: Compare match interrupt flag bit	<ul style="list-style-type: none"> <li>• This bit is an interrupt flag which indicates that the compare register 0, 2, 4 matches the value of the 16-bit free-run timer.</li> <li>• This bit is set to "1" when the value of the compare register matches the value of the 16-bit free-run timer.</li> <li>• When this bit is set while the compare match interrupt enable bit (IOE0: bit4) is enabled, the output compare interrupt occurs.</li> <li>• When this bit is set to "0": Clears the bit.</li> <li>• Setting this bit to "1" has no effect on this bit.</li> <li>• When this bit is read to a read modify write instruction, "1" is always read.</li> </ul>
bit5	IOE1: Compare match interrupt enable bit	<ul style="list-style-type: none"> <li>• This bit is used to enable the output compare interrupt of the compare register 1, 3, 5.</li> <li>• Setting this bit to "0" disables the compare match interrupt.</li> <li>• An output compare interrupt is generated if this bit is "1" when the compare match interrupt flag bit (IOP1:bit7) is set.</li> </ul>
bit4	IOE0: Compare match interrupt enable bit	<ul style="list-style-type: none"> <li>• This bit is used to enable the output compare interrupt of the compare register 0, 2, 4.</li> <li>• Setting this bit to "0" disables the compare match interrupt.</li> <li>• An output compare interrupt is generated if this bit is "1" when the compare match interrupt flag bit (IOP0:bit6) is set.</li> </ul>
bit3	BUF1: Compare buffer disable bit	<ul style="list-style-type: none"> <li>• This bit is used to disable the buffer function of the output compare register 1, 3, 5.</li> <li>• Setting this bit to "0" enables the buffer function.</li> <li>• Setting this bit to "1" disables the buffer function.</li> </ul>
bit2	BUF0: Compare buffer disable bit	<ul style="list-style-type: none"> <li>• This bit is used to disable the buffer function of the output compare register 0, 2, 4.</li> <li>• Setting this bit to "0" enables the buffer function.</li> <li>• Setting this bit to "1" disables the buffer function.</li> </ul>

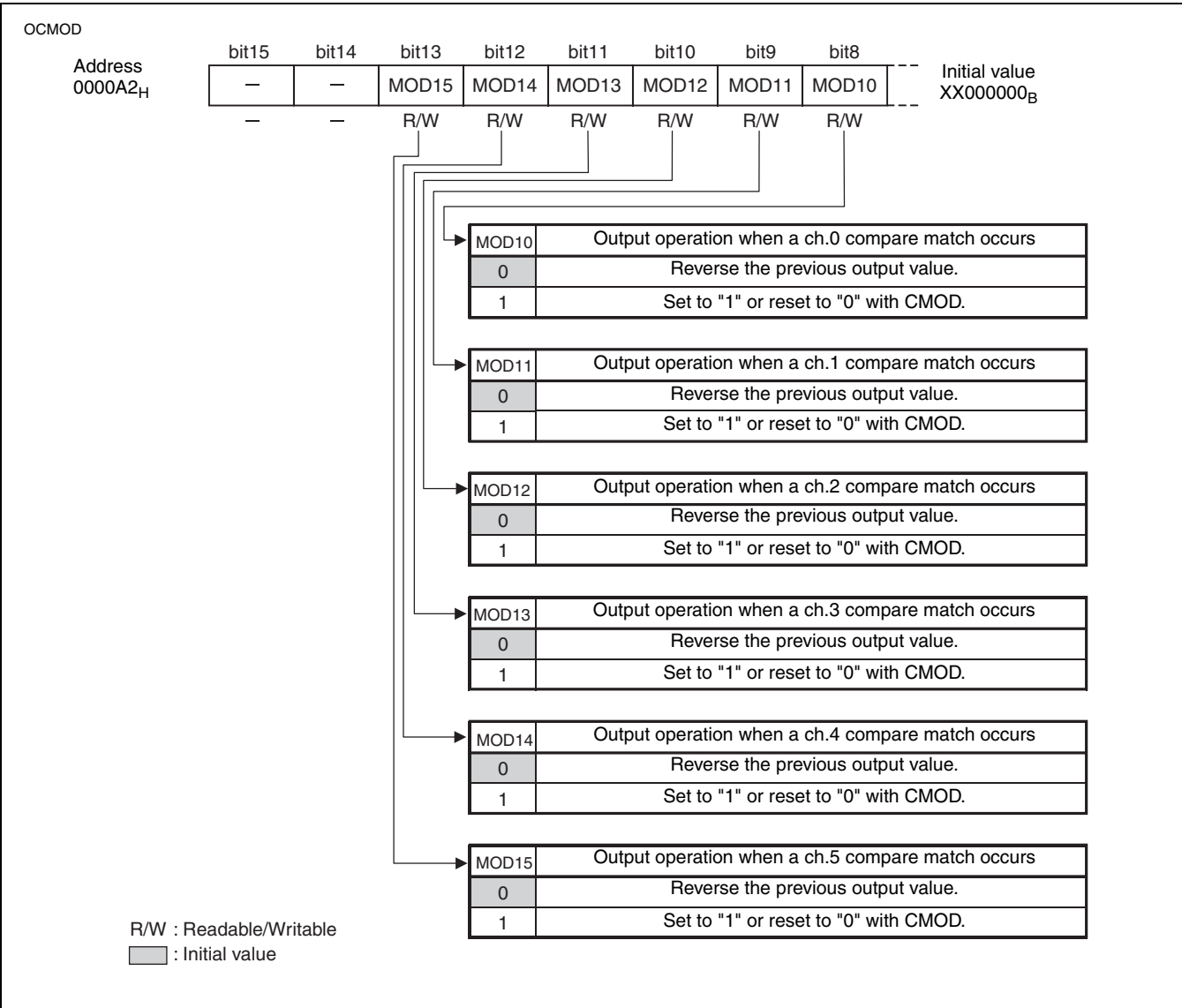
**Table 12.4-5 Compare Control Register, Lower Byte (OCSL0, OCSL2, OCSL4) (2 / 2)**

Bit name		Function
bit1	CST1: Compare operation enable bit	<ul style="list-style-type: none"> <li>• This bit is used to enable the compare operation between the 16-bit free-run timer and compare register 1, 3, 5.</li> <li>• Before enabling compare operation, always write values to compare registers 1, 3, and 5, and to the timer data registers (TCDTH and TCDTL).</li> <li>• Setting this bit to "0" disables the compare operation of the compare register.</li> <li>• Setting this bit to "1" enables the compare operation of the compare register.</li> </ul> <p>Note: The output compare is synchronized with the 16-bit free-run timer clock. Therefore, when the 16-bit free-run timer is stopped, the compare operation is also stopped.</p>
bit0	CST0: Compare operation enable bit	<ul style="list-style-type: none"> <li>• This bit is used to enable the compare operation between the 16-bit free-run timer and compare register 0, 2, 4.</li> <li>• Before enabling compare operation, always write values to compare registers 0, 2, and 4, and to the timer data registers (TCDTH and TCDTL).</li> <li>• Setting this bit to "0" disables the compare operation of the compare register.</li> <li>• Setting this bit to "1" enables the compare operation of the compare register.</li> </ul> <p>Note: The output compare is synchronized with the 16-bit free-run timer clock. Therefore, when the 16-bit free-run timer is stopped, the 0-detection and the compare operation are also stopped.</p>

# 12.4.7 Compare Mode Control Register (OCMOD)

The compare mode control register controls the mode for inverting the output level when a compare match occurs and whether to set or reset.

## ■ Compare Mode Control Register (OCMOD)



**Table 12.4-6 Compare Mode Control Register (OCMOD)**

Bit name		Function
bit15, bit14	Undefined bits	<ul style="list-style-type: none"> <li>The read value is indeterminate.</li> <li>Writing to these bits has no effect on the operation.</li> </ul>
bit13	MOD15: ch.5 compare match mode setting bit	<ul style="list-style-type: none"> <li>These bits specify the operation when the compare match of the output compare output occurs.</li> <li>The initial value is "0".</li> <li>When the bits are set to "0", reverse the output value temporarily when the match occurs.</li> <li>When the bits are set to "1", set the output value to "1" or reset the output value to "0" when the match occurs. CMOD bit of the compare control register (OCSH) sets the set/reset switch.</li> <li>Before data is written, be sure to stop the compare operation.</li> <li>CMOD is set for ch.0 and ch.1, ch.2 and ch.3, ch.4 and ch.5. <ul style="list-style-type: none"> <li>Reset/set is not available to ch.0 and ch.1 separately.</li> <li>Reset/set is not available to ch.2 and ch.3 separately.</li> <li>Reset/set is not available to ch.4 and ch.5 separately.</li> </ul> </li> </ul>
bit12	MOD14: ch.4 compare match mode setting bit	
bit11	MOD13: ch.3 compare match mode setting bit	
bit10	MOD12: ch.2 compare match mode setting bit	
bit9	MOD11: ch.1 compare match mode setting bit	
bit8	MOD10: ch.0 compare match mode setting bit	



### 12.4.8 Input Capture Data Register (IPCPH0 to IPCPH3, IPCPL0 to IPCPL3)

The input capture data register is used to store the count value of the free-run timer on detection of a valid edge of the input waveform.

#### ■ Input Capture Data Register (IPCPH0 to IPCPH3, IPCPL0 to IPCPL3)

Input capture data register (Upper)									
IPCPH0 to IPCPH3									
Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
0000AC <sub>H</sub>	CP15	CP14	CP13	CP12	CP11	CP10	CP09	CP08	XXXXXXXX <sub>B</sub>
0000AE <sub>H</sub>	R	R	R	R	R	R	R	R	
0000B0 <sub>H</sub>									
0000B2 <sub>H</sub>									
Input capture data register (Lower)									
IPCPL0 to IPCPL3									
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0000AD <sub>H</sub>	CP07	CP06	CP05	CP04	CP03	CP02	CP01	CP00	XXXXXXXX <sub>B</sub>
0000AF <sub>H</sub>	R	R	R	R	R	R	R	R	
0000B1 <sub>H</sub>									
0000B3 <sub>H</sub>									
R:	Read only								
X:	Undefined								

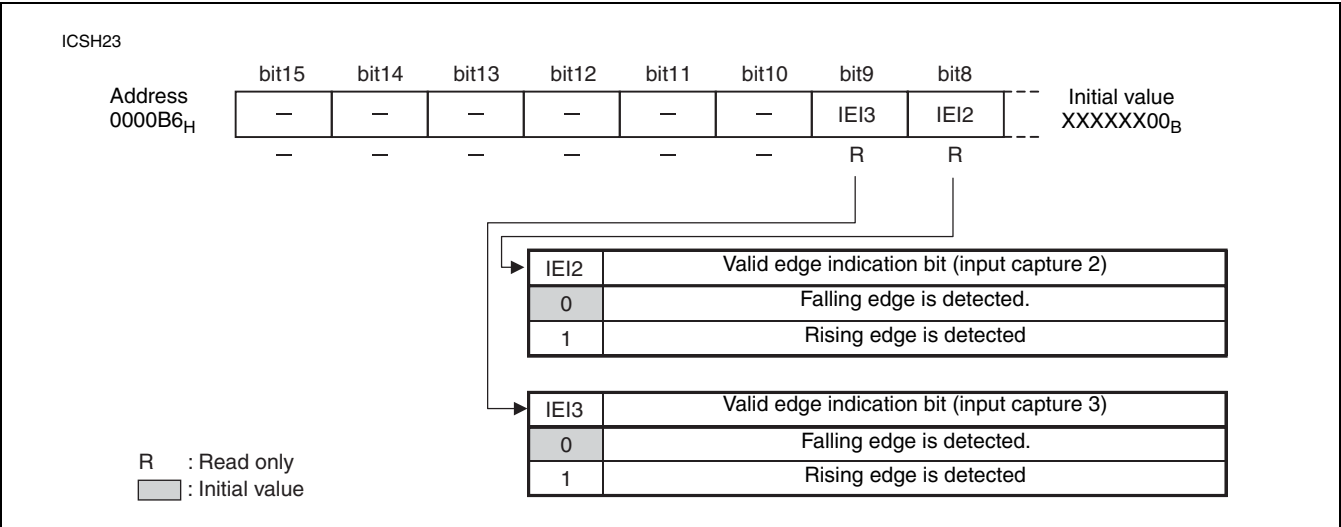
This register is used to store the value of the free-run timer each time the specified edge is detected on the waveform input to the corresponding external pin. (Always use half-word or word access instructions to access this register. Writing data to this register is not permitted.)

The free-run timer in the above explanation refers to the operation of the free-run timer selected for input capture.

### 12.4.9 Input Capture State Control/PPG Output Control Register (ICSH23, ICSL23, PICSH01, PICSL01)

The input capture state control/PPG output control register (ICSH23, ICSL23, PICSH01, PICSL01) is used to control the edge selection, the interrupt request enable, the interrupt request flag, and the PPG output. This register is also used to indicate a valid edge which was detected on the input capture 2 and 3.

#### ■ Input Capture State Control Register (ch.2, ch.3), Upper Byte (ICSH23)



**Table 12.4-7 Input Capture State Control Register (ch.2, ch.3), Upper Byte (ICSH23)**

Bit name		Function
bit15 to bit10	Undefined bits	<ul style="list-style-type: none"> <li>The read value is indeterminate.</li> <li>Writing to these bits has no effect on operation.</li> </ul>
bit9	IEI3: Valid edge indication bit (Input capture 3)	<ul style="list-style-type: none"> <li>This bit specifies the valid edge indication bit for capture register 3 and indicates whether a rising edge or falling edge was detected.</li> <li>"0" is written to this bit when a falling edge is detected.</li> <li>"1" is written to this bit when a rising edge is detected.</li> <li>This bit is a read-only bit.</li> </ul> <p>Note: When the lower of the input capture state control register (ICSL23), EG31, EG30: bit3, bit2 = 00<sub>B</sub>, the read value has no meaning.</p>
bit8	IEI2: Valid edge indication bit (Input capture2)	<ul style="list-style-type: none"> <li>This bit specifies the valid edge indication bit for capture register 2 and indicates whether a rising edge or falling edge was detected.</li> <li>"0" is written to this bit when a falling edge is detected.</li> <li>"1" is written to this bit when a rising edge is detected.</li> <li>This bit is a read-only bit.</li> </ul> <p>Note: When the lower of the input capture state control register (ICSL23), EG21, EG20: bit1, bit0 = 00<sub>B</sub>, the read value has no meaning.</p>

## Input Capture State Control Register (ch.2, ch.3), Lower Byte (ICSL23)

ICSL23

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
ICP3	IP2	ICE3	ICE2	EG31	EG30	EG21	EG20	00000000 <sub>B</sub>
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Address  
0000B7<sub>H</sub>

EG21	EG20	Edge selection bits (input capture 2)
0	0	No edge is detected (stop).
0	1	Rising edge is detected.
1	0	Falling edge is detected.
1	1	Both edges are detected.

EG31	EG30	Edge selection bits (input capture 3)
0	0	No edge is detected (stop).
0	1	Rising edge is detected.
1	0	Falling edge is detected.
1	1	Both edges are detected.

ICE2	Interrupt request enable bit (input capture 2)
0	Disable interrupt requests.
1	Enable interrupt requests.

ICE3	Interrupt request enable bit (input capture 3)
0	Disable interrupt requests.
1	Enable interrupt requests.

ICP2	Interrupt request flag bit (input capture 2)	
	Read	Write
0	No valid edge is detected.	This bit is cleared.
1	Valid edge is detected.	Not affect this bit.

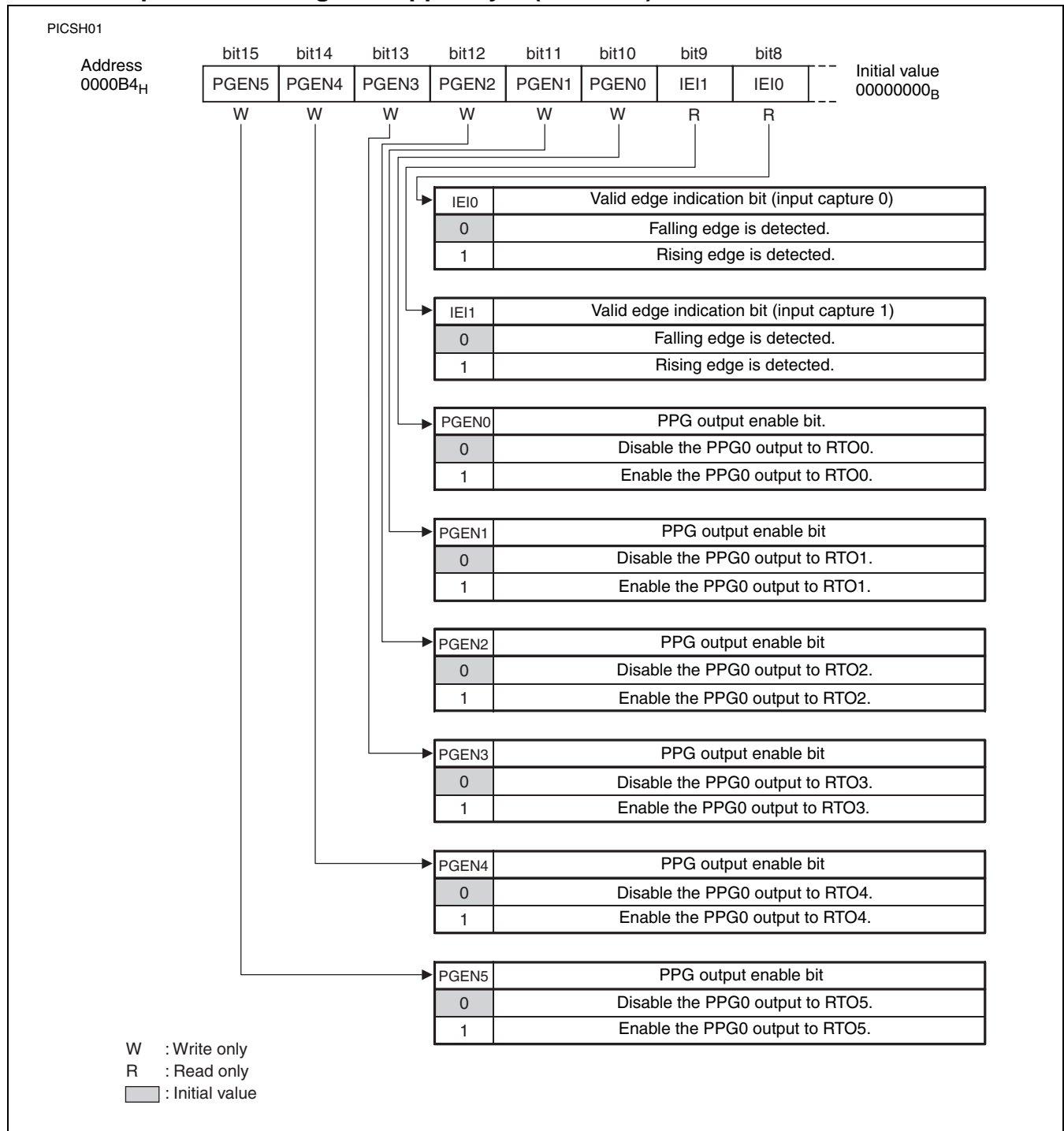
ICP3	Interrupt request flag bit (input capture 3)	
	Read	Write
0	No valid edge is detected.	This bit is cleared.
1	Valid edge is detected.	Not affect this bit.

R/W : Readable/Writable  
 : Initial value

**Table 12.4-8 Input Capture State Control Register (ch.2, ch.3), Lower Byte (ICSL23)**

Bit name		Function
bit7	ICP3: Interrupt request flag bit (Input capture 3)	<ul style="list-style-type: none"> <li>• This bit is used as an interrupt request flag of the input capture 3.</li> <li>• This bit is set to "1" immediately when a valid edge of an external input pin is detected.</li> <li>• When a valid edge is detected while the interrupt request enable bit (ICE3: bit5) is set, the interrupt can be generated immediately.</li> <li>• When this bit is set to "0": Clears the bit.</li> <li>• Setting this bit to "1" has no effect on this bit.</li> <li>• When this bit is read to a read modify write instruction, "1" is always read.</li> </ul>
bit6	ICP2: Interrupt request flag bit (Input capture 2)	<ul style="list-style-type: none"> <li>• This bit is used as an interrupt request flag of the input capture 2.</li> <li>• This bit is set to "1" immediately when a valid edge of an external input pin is detected.</li> <li>• When a valid edge is detected while the interrupt request enable bit (ICE2: bit4) is set, the interrupt can be generated immediately.</li> <li>• When this bit is set to "0": Clears the bit.</li> <li>• Setting this bit to "1" has no effect on this bit.</li> <li>• When this bit is read to a read modify write instruction, "1" is always read.</li> </ul>
bit5	ICE3: Interrupt request enable bit (Input capture 3)	<ul style="list-style-type: none"> <li>• This bit is used to enable an input capture interrupt request of the input capture 3.</li> <li>• Setting this bit to "0" disables interrupt request.</li> <li>• When the interrupt request flag bit (ICP3: bit7) is set while this bit is set to "1", the input capture 3 interrupt is generated.</li> </ul>
bit4	ICE2: Interrupt request enable bit (Input capture 2)	<ul style="list-style-type: none"> <li>• This bit is used to enable an input capture interrupt request of the input capture 2.</li> <li>• Setting this bit to "0" disables interrupt request.</li> <li>• When the interrupt request flag bit (ICP2: bit6) is set while this bit is set to "1", the input capture 2 interrupt is generated.</li> </ul>
bit3, bit2	EG31, EG30: Edge selection bits (Input capture 3)	<ul style="list-style-type: none"> <li>• These bits are used to specify the active edge polarity for the external input to input capture 3.</li> <li>• These bits are used also to enable an operation of the input capture 3.</li> </ul>
bit1, bit0	EG21, EG20: Edge selection bits (Input capture 2)	<ul style="list-style-type: none"> <li>• These bits are used to specify the active edge polarity for the external input to input capture 2.</li> <li>• These bits are used also to enable an operation of the input capture 2.</li> </ul>

## ■ PPG Output Control Register Upper Byte (PICSH01)



**Table 12.4-9 PPG Output Control Register, Upper Byte (PICSH01)**

Bit name		Function
bit15 to bit10	PGEN5 to PGEN0: PPG output enable bits	<ul style="list-style-type: none"> <li>These bits are used to select the PPG0 output to RTO0, RTO1, RTO2, RTO3, RTO4, RTO5.</li> <li>These bits are write-only.</li> </ul>
bit9	IEI1: Valid edge indication bit (Input capture 1)	<ul style="list-style-type: none"> <li>This bit specifies the valid edge indication bit for capture register 1 and indicates whether a rising edge or falling edge was detected.</li> <li>"0" is written to this bit when a falling edge is detected.</li> <li>"1" is written to this bit when a rising edge is detected.</li> </ul> <p>This bit is a read-only bit.</p> <p>Note: When the lower of the input capture state control register (PICSL01), EG11, EG10: bit3, bit2 = 00<sub>B</sub>, the read value has no meaning.</p>
bit8	IEI0: Valid edge indication bit (Input capture 0)	<ul style="list-style-type: none"> <li>This bit specifies the valid edge indication bit for capture register 0 and indicates whether a rising edge or falling edge was detected.</li> <li>"0" is written to this bit when a falling edge is detected.</li> <li>"1" is written to this bit when a rising edge is detected.</li> <li>This bit is a read-only bit.</li> </ul> <p>Note: When the lower of the input capture state control register (PICSL01), EG01, EG00: bit1, bit0 = 00<sub>B</sub>, the read value has no meaning.</p>

## Input Capture State Control Register (ch.0, ch.1), Lower Byte (PICSL01)

PICSL01

Address  
0000B5<sub>H</sub>

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
ICP1	ICP0	ICE1	ICE0	EG11	EG10	EG01	EG00	00000000 <sub>B</sub>
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

EG01	EG00	Edge selection bits (input capture 0)
0	0	No edge is detected (stop).
0	1	Rising edge is detected.
1	0	Falling edge is detected.
1	1	Both edges are detected.

EG11	EG10	Edge selection bits (input capture 1)
0	0	No edge is detected (stop).
0	1	Rising edge is detected.
1	0	Falling edge is detected.
1	1	Both edges are detected.

ICE0	Interrupt request enable bit (input capture 0)
0	Disable interrupt requests.
1	Enable interrupt requests.

ICE1	Interrupt request enable bit (input capture 1)
0	Disable interrupt requests.
1	Enable interrupt requests.

ICP0	Interrupt request flag bit (input capture 0)	
	Read	Write
0	No valid edge is detected.	This bit is cleared.
1	Valid edge is detected.	Not affect this bit.

ICP1	Interrupt request flag bit (input capture 1)	
	Read	Write
0	No valid edge is detected.	This bit is cleared
1	Valid edge is detected.	Not affect this bit.

R/W : Readable/Writable

 : Initial value



**Table 12.4-10 Input Capture State Control Register (ch.0, ch.1), Lower Byte (PICSL01)**

Bit name		Function
bit7	ICP1: Interrupt request flag bit (Input capture 1)	<ul style="list-style-type: none"> <li>• This bit is used as an interrupt request flag of the input capture 1.</li> <li>• This bit is set to "1" immediately when a valid edge of an external input pin is detected.</li> <li>• When a valid edge is detected while the interrupt request enable bit (ICE1: bit5) is set, the interrupt is generated immediately.</li> <li>• When this bit is set to "0": Clears the bit.</li> <li>• Setting this bit to "1" has no effect on this bit.</li> <li>• When this bit is read to a read modify write instruction, "1" is always read.</li> </ul>
bit6	ICP0: Interrupt request flag bit (Input capture 0)	<ul style="list-style-type: none"> <li>• This bit is used as an interrupt request flag of the input capture 0.</li> <li>• This bit is set to "1" immediately when a valid edge of an external input pin is detected.</li> <li>• When a valid edge is detected while the interrupt request enable bit (ICE0: bit4) is set, the interrupt is generated immediately.</li> <li>• When this bit is set to "0": Clears the bit.</li> <li>• Setting this bit to "1" has no effect on this bit.</li> <li>• When this bit is read to a read modify write instruction, "1" is always read.</li> </ul>
bit5	ICE1: Interrupt request enable bit (Input capture 1)	<ul style="list-style-type: none"> <li>• This bit is used to enable an input capture interrupt request of the input capture 1.</li> <li>• Setting this bit to "0" disables interrupt request.</li> <li>• When the interrupt request flag bit (ICP1: bit7) is set while this bit is set to "1", the input capture 1 interrupt is generated.</li> </ul>
bit4	ICE0: Interrupt request enable bit (Input capture 0)	<ul style="list-style-type: none"> <li>• This bit is used to enable an input capture interrupt request of the input capture 0.</li> <li>• Setting this bit to "0" disables interrupt request.</li> <li>• When the interrupt request flag bit (ICP0: bit6) is set while this bit is set to "1", the input capture 0 interrupt is generated.</li> </ul>
bit3, bit2	EG11, EG10: Edge selection bits (Input capture 1)	<ul style="list-style-type: none"> <li>• These bits are used to specify the active edge polarity for the external input to input capture 1.</li> <li>• These bits are used also to enable an operation of the input capture 1.</li> </ul>
bit1, bit0	EG01, EG00: Edge selection bits (Input capture 0)	<ul style="list-style-type: none"> <li>• These bits are used to specify the active edge polarity for the external input to input capture 0.</li> <li>• These bits are used also to enable an operation of the input capture 0.</li> </ul>

## 12.4.10 16-bit Dead Timer Register (TMRRH0 to TMRRH2, TMRRL0 to TMRRL2)

The 16-bit dead timer register stores the compare value of the 16-bit dead timer.

### ■ 16-bit Dead Timer Register (TMRRH0 to TMRRH2, TMRRL0 to TMRRL2)

16-bit dead timer register (Upper)

TMRRH0 to TMRRH2

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
0000BC <sub>H</sub>	TR15	TR14	TR13	TR12	TR11	TR10	TR09	TR08	XXXXXXXX <sub>B</sub>
0000BE <sub>H</sub>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0000C0 <sub>H</sub>									

16-bit dead timer register (Lower)

TMRRL0 to TMRRL2

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0000BD <sub>H</sub>	TR07	TR06	TR05	TR04	TR03	TR02	TR01	TR00	XXXXXXXX <sub>B</sub>
0000BF <sub>H</sub>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
0000C1 <sub>H</sub>									

R/W: Readable/Writable

X: Undefined

These registers are used to store the compare value for the 16-bit dead timer.

These register values are reloaded when the 16-bit dead timer starts the operation.

If the values are rewritten to these registers during the timer operation, these new values are enabled in the next timer start/operation.

To access these registers, use a half-word or word access instruction.

In the dead time timer mode, these registers are used to set the non-overlap time.

$$\text{Non-overlap time} = (\text{Setting value}) \times \text{Selected clock}$$

Note:

"0000<sub>H</sub>" cannot be set.

In the timer mode, these registers are used to set the GATE time of the PPG0 timer operation.

$$\text{GATE time} = (\text{Setting value}) \times \text{Selected clock}$$

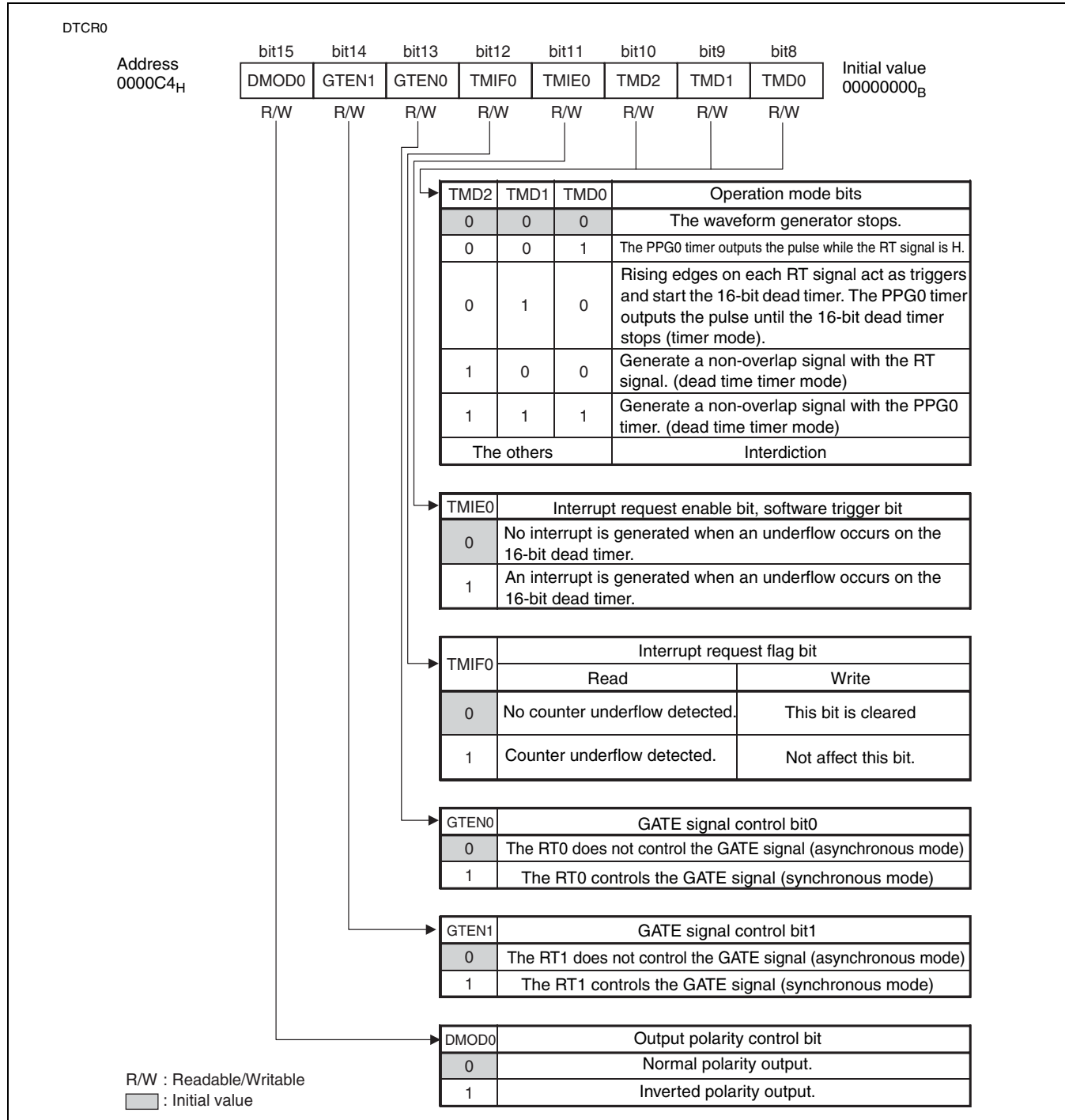
Note:

"0000<sub>H</sub>" cannot be set.

## 12.4.11 16-bit Dead Timer Control Register (DTCR0 to DTCR2)

The 16-bit dead timer control register (DTCR0, DTCR1, DTCR2) is used to control the operation mode of the waveform generator, the interrupt request enable, the interrupt request flag, the GATE signal enable, and the output level polarity.

### ■ 16-bit Dead Timer Control Register, Upper Byte (DTCR0)



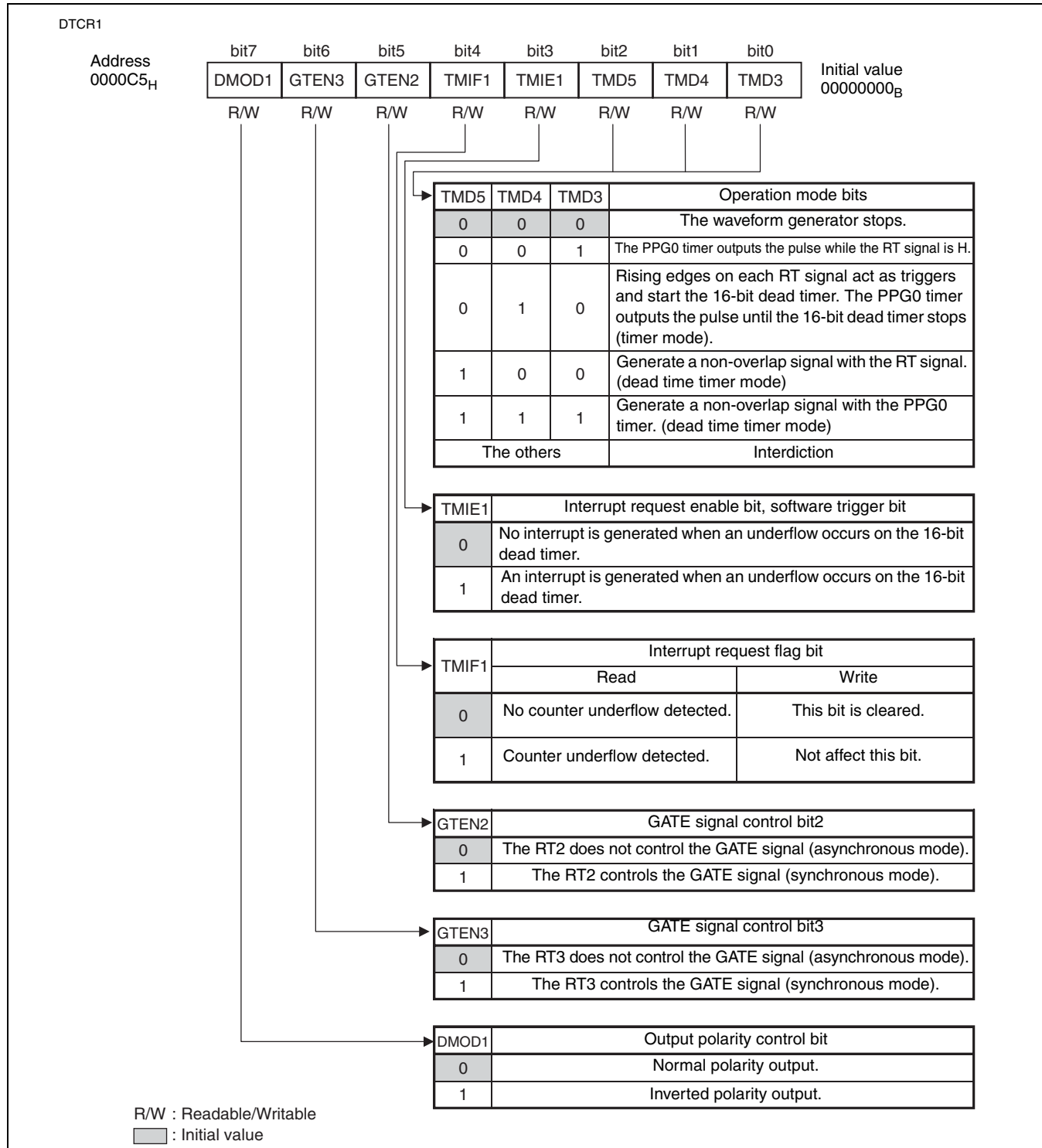
**Table 12.4-11 16-bit Dead Timer Control Register, Upper Byte (DTCR0) (1 / 2)**

Bit name		Function
bit15	DMOD0: Output polarity control bit	<ul style="list-style-type: none"> <li>This bit is used to set the U/V/W output in the dead time timer mode.</li> <li>Setting this bit reverses the U/V/W output polarity.</li> </ul> <p>Note: When the dead time timer mode is not selected, (TMD2: bit10 = 0), this bit has no meaning.</p>
bit14	GTEN1: GATE signal control bit1	This bit is used to control the GATE signal output of the PPG0 timer with RT1.
bit13	GTEN0: GATE signal control bit0	This bit is used to control the GATE signal output of the PPG0 timer with RT0.
bit12	TMIF0: Interrupt request flag bit	<ul style="list-style-type: none"> <li>This bit is used as an interrupt request flag of the 16-bit dead timer.</li> <li>This bit is set to "1" when an underflow occurs on the 16-bit dead timer.</li> <li>Writing "0" to this bit clears the bit. Writing "1" to this bit has no effect on this bit.</li> <li>When this bit is read to a read modify write instruction, "1" is always read.</li> </ul> <p>Note: This bit works only when the register values (TMD2 to TMD0: bit10 to bit8) are 000<sub>B</sub> or 001<sub>B</sub>, and always becomes "0" when they are other values. If a software clear (writing "0") and hardware set (underflow on 16-bit dead timer 0) occur simultaneously, the software operation has precedence and the bit is cleared.</p>
bit11	TMIE0: Interrupt request enable bit, software trigger bit	<ul style="list-style-type: none"> <li>This bit is used as a software trigger bit and an interrupt enable bit of the 16-bit dead timer.</li> <li>TMD2 to TMD0: bit10 to bit8 = 000<sub>B</sub> or 001<sub>B</sub>: This bit is used as a software trigger of the 16-bit dead timer. When this bit changes from "0" to "1", it becomes a trigger of the 16-bit dead timer, reloads the value, and starts the down count.</li> <li>When this bit is "1" and the interrupt request flag bit (TMIF0: bit12) is "1", an interrupt request is sent to the CPU.</li> </ul> <p>Note: To trigger the 16-bit dead timer again, you must write "0" to this bit before writing "1".</p>

**Table 12.4-11 16-bit Dead Timer Control Register, Upper Byte (DTCR0) (2 / 2)**

Bit name		Function
bit10 to bit8	TMD2 to TMD0: Operation mode bits	<ul style="list-style-type: none"> <li>These bits are used to select the operation mode of the waveform generator.</li> <li>TMD2 to TMD0: bit10 to bit8 = 000<sub>B</sub>: The RT0 and RT1 signals of the output compare are respectively output from the RTO0 and RTO1. The 16-bit dead timer is also used as a reload timer.</li> <li>TMD2 to TMD0: bit10 to bit8 = 001<sub>B</sub>: The RT0 and RT1 signals of the output compare are respectively output from RTO0 and RTO1 when the PPG0 output is disabled (the upper of the PPG output control/input capture state control register (PICSH01), PGEN0: bit10 = 0, PGEN1: bit11 = 0). The 16-bit dead timer is also used as a reload timer.</li> </ul> <p>Note: Always select two-channel mode for RT1 (set bit12 (CMOD) = 1 in the upper compare control register (OCSH1)) to use the waveform generator in dead time timer mode. TMD2 to TMD0: bit10 to bit8 = 111<sub>B</sub>: The RTO0 and RTO1 outputs are independent of the register settings (the upper of the PPG output control/input capture state control register (PICSH01), PGEN0: bit10 = 0, PGEN1: bit11 = 0).</p>

## ■ 16-bit Dead Timer Control Register, Lower Byte (DTCR1)



**Table 12.4-12 16-bit Dead Timer Control Register, Lower Byte (DTCR1) (1 / 2)**

Bit name		Function
bit7	DMOD1: Output polarity control bit	<ul style="list-style-type: none"> <li>This bit is used to set the U/V/W output in the dead time timer mode.</li> <li>Setting this bit reverses the U/V/W output polarity.</li> </ul> <p>Note: When the dead time timer mode is not selected, (TMD5: bit2 = 0), this bit has no meaning.</p>
bit6	GTEN3: GATE signal control bit3	This bit is used to control the GATE signal output of the PPG0 timer with RT3.
bit5	GTEN2: GATE signal control bit2	This bit is used to control the GATE signal output of the PPG0 timer with RT2.
bit4	TMIF1: Interrupt request flag bit	<ul style="list-style-type: none"> <li>This bit is used as an interrupt request flag of the 16-bit dead timer.</li> <li>This bit is set to "1" when an underflow occurs on the 16-bit dead timer.</li> <li>Writing "0" to this bit clears the bit. Writing "1" to this bit has no effect on this bit.</li> <li>When this bit is read to a read modify write instruction, "1" is always read.</li> </ul> <p>Note: This bit works only when the register values (TMD5 to TMD3: bit2 to bit0) are "000<sub>B</sub>" or "001<sub>B</sub>", and always becomes 0 when they are other values. If a software clear (writing "0") and hardware set (underflow on 16-bit dead timer 1) occur simultaneously, the software operation has precedence and the bit is cleared.</p>
bit3	TMIE1: Interrupt request enable bit, software trigger bit	<ul style="list-style-type: none"> <li>This bit is used as a software trigger bit and an interrupt enable bit of the 16-bit dead timer.</li> <li>TMD5 to TMD3: bit2 to bit0 = 000<sub>B</sub> or 001<sub>B</sub>: This bit is used as a software trigger of the 16-bit dead timer. When this bit changes from "0" to "1", it becomes a trigger of the 16-bit dead timer, reloads the value, and starts the down count.</li> <li>When this bit is "1" and the interrupt request flag bit (TMIF1: bit4) is "1", an interrupt request is sent to the CPU.</li> </ul> <p>Note: To trigger the 16-bit dead timer again, you must write "0" to this bit before writing "1".</p>

**Table 12.4-12 16-bit Dead Timer Control Register, Lower Byte (DTCR1) (2 / 2)**

Bit name		Function
bit2 to bit0	TMD5 to TMD3: Operation mode bits	<ul style="list-style-type: none"> <li>These bits are used to select the operation mode of the waveform generator.</li> <li>TMD5 to TMD3: bit2 to bit0 = 000<sub>B</sub>: The RT2 and RT3 signals of the output compare are respectively output from the RTO2 and RTO3. The 16-bit dead timer is also used as a reload timer.</li> <li>TMD5 to TMD3: bit2 to bit0 = 001<sub>B</sub>: The RT2 and RT3 signals of the output compare are respectively output from the RTO2 and RTO3 when the PPG0 output is disabled (the upper of the PPG output control/input capture state control register (PICSH01), PGEN2: bit12=0, PGEN3: bit13=0). The 16-bit dead timer is also used as a reload timer.</li> </ul> <p>Note: Always select two-channel mode for RT3 (set bit12 (CMOD) = 1 in the upper compare control register (OCSH3)) to use the waveform generator in dead time timer mode. TMD5 to TMD3: bit2 to bit0 = 111<sub>B</sub>: The RTO2 and RTO3 outputs are independent of the register settings (the upper of the PPG output control/input capture state control register (PICSH01), PGEN2: bit12 = 0, PGEN3: bit13 = 0).</p>



## ■ 16-bit Dead Timer Control Register, Upper Byte (DTCR2)

DTCR2

Address  
0000C6<sub>H</sub>

bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8
DMOD2	GTEN5	GTEN4	TMIF2	TMIE2	TMD8	TMD7	TMD6
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Initial value  
00000000<sub>B</sub>

TMD8	TMD7	TMD6	Operation mode bits
0	0	0	The waveform generator stops.
0	0	1	The PPG0 timer outputs the pulse while the RT signal is "H".
0	1	0	Rising edges on each RT signal act as triggers and start the 16-bit dead timer. The PPG0 timer outputs the pulse until the 16-bit dead timer stops (timer mode).
1	0	0	Generate a non-overlap signal with the RT signal. (dead time timer mode)
1	1	1	Generate a non-overlap signal with the PPG0 timer. (dead time timer mode)
The others			Interdiction

TMIE2	Interrupt request enable bit, software trigger bit
0	No interrupt is generated when an underflow occurs on the 16-bit dead timer.
1	An interrupt is generated when an underflow occurs on the 16-bit dead timer.

TMIF2	Interrupt request flag bit	
	Read	Write
0	No counter underflow detected.	This bit is cleared.
1	Counter underflow detected.	Not affect this bit.

GTEN4	GATE signal control bit4
0	The RT4 does not control the GATE signal (asynchronous mode)
1	The RT4 controls the GATE signal (synchronous mode)

GTEN5	GATE signal control bit5
0	The RT5 does not control the GATE signal (asynchronous mode)
1	The RT5 controls the GATE signal (synchronous mode)

DMOD2	Output polarity control bit
0	Normal polarity output.
1	Inverted polarity output.

R/W : Readable/Writable

 : Initial value

**Table 12.4-13 16-bit Dead Timer Control Register, Upper Byte (DTCR2) (1 / 2)**

Bit name		Function
bit15	DMOD2: Output polarity control bit	<ul style="list-style-type: none"> <li>This bit is used to set the U/V/W output in the dead time timer mode.</li> <li>Setting this bit reverses the U/V/W output polarity.</li> </ul> <p>Note: When the dead time timer mode is not selected, (TMD8: bit10 = 0), this bit has no meaning.</p>
bit14	GTEN5: GATE signal control bit5	This bit is used to control the GATE signal output of the PPG0 timer with RT5.
bit13	GTEN4: GATE signal control bit4	This bit is used to control the GATE signal output of the PPG0 timer with RT4.
bit12	TMIF2: Interrupt request flag bit	<ul style="list-style-type: none"> <li>This bit is used as an interrupt request flag of the 16-bit dead timer.</li> <li>This bit is set to "1" when an underflow occurs on the 16-bit dead timer.</li> <li>Writing "0" to this bit clears the bit. Writing "1" to this bit has no effect on this bit.</li> <li>When this bit is read to a read modify write instruction, "1" is always read.</li> </ul> <p>Note: This bit works only when the register values (TMD8 to TMD6: bit10 to bit8) are "000<sub>B</sub>" or "001<sub>B</sub>", and always becomes "0" when they are other values. If a software clear (writing "0") and hardware set (underflow on 16-bit dead timer 2) occur simultaneously, the software operation has precedence and the bit is cleared.</p>
bit11	TMIE2: Interrupt request enable bit, software trigger bit	<ul style="list-style-type: none"> <li>This bit is used as a software trigger bit and an interrupt enable bit of the 16-bit dead timer.</li> <li>TMD8 to TMD6: bit10 to bit8 = 000<sub>B</sub> or 001<sub>B</sub>: This bit is used as a software trigger of the 16-bit dead timer. When this bit changes from "0" to "1", it becomes a trigger of the 16-bit dead timer, reloads the value, and starts the down count.</li> <li>When this bit is "1" and the interrupt request flag bit (TMIF2: bit12) is "1", an interrupt request is sent to the CPU.</li> </ul> <p>Note: To trigger the 16-bit dead timer again, you must write "0" to this bit before writing "1".</p>

**Table 12.4-13 16-bit Dead Timer Control Register, Upper Byte (DTCR2) (2 / 2)**

Bit name		Function
bit10 to bit8	TMD8 to TMD6: Operation mode bits	<ul style="list-style-type: none"> <li>These bits are used to select the operation mode of the waveform generator.</li> <li>TMD8 to TMD6: bit10 to bit8 = 000<sub>B</sub>: The RT4 and RT5 signals of the output compare are respectively output from the RTO4 and RTO5. The 16-bit dead timer is also used as a reload timer.</li> <li>TMD8 to TMD6: bit10 to bit8 = 001<sub>B</sub>: The RT4 and RT5 signals of the output compare are respectively output from the RTO4 and RTO5 when the PPG0 output is disabled (the upper of the PPG output control/input capture state control register (PICSH01), PGEN4: bit14 = 0, PGEN5: bit15 = 0). The 16-bit dead timer is also used as a reload timer.</li> </ul> <p>Note: Always select two-channel mode for RT5 (set bit12 (CMOD) = 1 in the upper compare control register (OCSH5)) to use the waveform generator in dead time timer mode. TMD8 to TMD6: bit10 to bit8 = 111<sub>B</sub>: The RTO4 and RTO5 outputs are independent of the register settings (the upper of the PPG output control/input capture state control register (PICSH01), PGEN4: bit14 = 0, PGEN5: bit15 = 0).</p>

## 12.4.12 Waveform Control Register (SIGCR1/SIGCR2)

The waveform control register is used to control the operating clock frequency, enable the noise cancellation feature, enable DTTI pin input, and control DTTI pin interrupts.

### ■ Waveform Control Register 1 (SIGCR1)

SIGCR1

Address  
0000C9<sub>H</sub>

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
DTIE	DTIF	NRSL	DCK2	DCK1	DCK0	NWS1	NWS0	00000000 <sub>B</sub>
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

NWS1	NWS0	DTTI pin input noise width selection bits
0	0	Cancel 4-cycle noise.
0	1	Cancel 8-cycle noise.
1	0	Cancel 16-cycle noise.
1	1	Cancel 32-cycle noise.

DCK2	DCK1	DCK0	Operation clock selection bits
0	0	0	$\phi$ (62.5ns, $\phi$ =16MHz)
0	0	1	$\phi/2$ (125ns, $\phi$ =16MHz)
0	1	0	$\phi/4$ (250ns, $\phi$ =16MHz)
0	1	1	$\phi/8$ (500ns, $\phi$ =16MHz)
1	0	0	$\phi/16$ (1 $\mu$ s, $\phi$ =16MHz)
1	0	1	$\phi/32$ (2 $\mu$ s, $\phi$ =16MHz)
1	1	0	$\phi/64$ (4 $\mu$ s, $\phi$ =16MHz)
1	1	1	Interdiction

 $\phi$ : Machine cycle

NRSL	Noise cancellation feature enabled bit
0	The DTTI pin input noise cancellation circuit is disabled.
1	The DTTI pin input noise cancellation circuit is enabled.

DTIF	DTTI pin interrupt flag bit	
	Read	Write
0	Not interrupt request	This bit is cleared.
1	With interrupt request	Not affect this bit.

DTIE	DTTI pin input enabled bit
0	DTTI pin input is disabled.
1	DTTI pin input is enabled.

R/W : Readable/Writable  
 : Initial value

**Table 12.4-14 Waveform Control Register1 (SIGCR1)**

Bit name		Function
bit7	DTIE: DTTI pin input enabled bit	This bit is used to enable the output level control DTTI pin input signal of RTO0 to RTO5 pins.
bit6	DTIF: DTTI pin interrupt flag bit	<ul style="list-style-type: none"> <li>This bit is the DTTI pin interrupt flag.</li> <li>When DTTI pin input is enabled (DTIE: bit7 = 1) and DTTI pin "L" level is detected, this bit is set, and an interrupt request is sent to the CPU.</li> <li>When this bit is set to "0": Clears the bit.</li> <li>Setting this bit to "1" has no effect on this bit.</li> <li>"1" is always read during read-modify-write.</li> </ul> <p>Note: If noise cancellation is enabled (NRSL: bit 5 = 1), this bit is set to "1" by an "L" level input to the DTTI pin if the pulse width is longer than the specified noise pulse width. If a software clear (write 0) and hardware set (DTTI pin "L" level detected) occur simultaneously, the software clear is given precedence over the hardware set, and this bit is cleared.</p>
bit5	NRSL: Noise cancellation feature enabled bit	<ul style="list-style-type: none"> <li>This bit is used to enable the noise cancellation feature.</li> <li>Setting this bit to "0" disables noise cancellation feature.</li> <li>Setting this bit to "1" enables noise cancellation feature.</li> <li>The noise canceling circuit transmit DTTI pin input signal to DTTI control circuit if it remains at the "L" level until an overflow occurs on the counter. The counter is an n-bit counter operated by Low level input of DTTI pin input. n is set by NWS1 and NWS0 bit: 1, 0 based on the settings of bit1 and bit0, the value of n is 2, 3, 4, or 5.</li> </ul> <p>Note: Approximately <math>2^n</math> machine cycles are required to disable a noise cancellation feature. When a noise cancellation feature is enable, DTTI pin input is disabled when in a mode that stops the internal clock (e.g. stop mode).</p>
bit4 to bit2	DCK2 to DCK0: Operation clock selection bits	These bits are used to select the operating clock for the 16-bit dead timer.
bit1, bit0	NWS1, NWS0: DTTI pin noise width selection bits	These bits are used to select the width of noise pulses to reject on the DTTI pin.

■ Waveform Control Register 2 (SIGCR2)

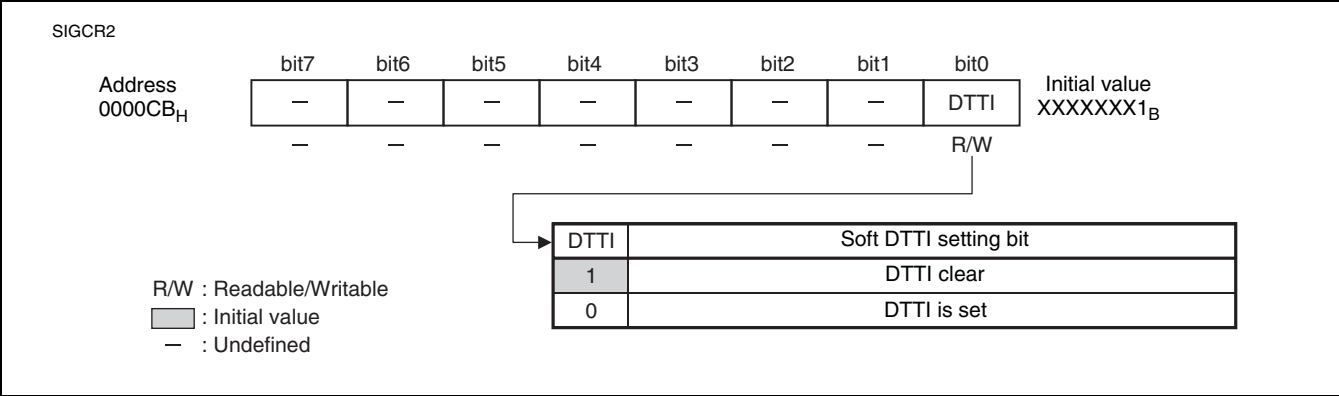


Table 12.4-15 Waveform Control Register2 (SIGCR2)

Bit name		Function
bit7 to bit1	Undefined bits	<ul style="list-style-type: none"><li>The read value is indeterminate.</li><li>Writing to these bits has no effect on operation.</li></ul>
bit0	DTTI: Soft DTTI setting bit	<ul style="list-style-type: none"><li>Write "0" to set DTTI.</li><li>Write "1" to this bit to clear it.</li></ul> <p>Note: Writing "1" to this bit does not affect the DTTI pin interrupt or change the DTTI pin interrupt flag bit (DTIF).</p>

### 12.4.13 A/D activation compare register (ADCOMP1, ADCOMP2, ADCOMPC1, ADCOMPC2)

Compare registers 1 and 2 activate A/D converters 1 and 2 when their values match that of the free-run timer 0. The compare register is used to write compare values.

#### ■ Compare Register 1, 2 (ADCOMP1, ADCOMP2)

Compare register 1, 2 (Upper)

ADCOMP1, ADCOMP2

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
ch.1: 0000CE <sub>H</sub>	CMP15	CMP14	CMP13	CMP12	CMP11	CMP10	CMP09	CMP08	00000000 <sub>B</sub>
ch.2: 0000D0 <sub>H</sub>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Compare register 1, 2 (Lower)

ADCOMP1, ADCOMP2

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
ch.1: 0000CF <sub>H</sub>	CMP07	CMP06	CMP05	CMP04	CMP03	CMP02	CMP01	CMP00	00000000 <sub>B</sub>
ch.2: 0000D1 <sub>H</sub>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable

The compare register is used to write data for comparison with the 16-bit free-run timer 0 count value. It is possible to activate A/D when the free-run timer 0 and compare values match.

The value written to the compare register is used for a comparison immediately.

Always use word or half-word access to write to the compare register.

## ■ Control Register 1, 2 (ADCOMPC1, ADCOMPC2)

### Control Register 1

#### ADCOMPC1

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0000D3 <sub>H</sub>	–	–	–	–	–	CE2	CE1	–	XXXXX00X <sub>B</sub>
	–	–	–	–	–	R/W	R/W	–	

### Control Register 2

#### ADCOMPC2

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
0000D2 <sub>H</sub>	–	–	SEL21	SEL20	SEL11	SEL10	–	–	XX0000XX <sub>B</sub>
	–	–	R/W	R/W	R/W	R/W	–	–	

R/W: Readable/Writable

X: Undefined value

–: Undefined

[bit2, bit1, bit0] CE2,CE1: A/D compare activation enable bits

CEn	A/D compare activation enable bits
0	Disables compare start on A/D unit n [Initial value]
1	Enables compare start on A/D unit n

(n = 1, 2 : A/D unit number)

[bit13 to bit10] SEL21,SEL20,SEL11, and SEL10: A/D compare activation trigger selection bits

SELn1	SELn0	A/D compare activation trigger selection bits
0	0	Starts when a free-run timer match occurs. [Initial value]
0	1	Only starts when a match occurs on an up-count of the free-run timer.
1	0	Only starts when a match occurs on a down-count of the free-run timer.
1	1	Setting disabled

(n = 1, 2 : A/D unit number)



### 12.4.14 Free-run Timer Selector Register (FSR0 to FSR2)

The free-run timer selector registers specify which of the three free-run timer channels to use for each input capture and output compare.

■ Free-run Timer Selector Register (FSR0 to FSR2)

FSR2									
Address	bit23	bit22	bit21	bit20	bit19	bit18	bit17	bit16	Initial value
000169 <sub>H</sub>	ICU31	ICU30	ICU21	ICU20	ICU11	ICU10	ICU01	ICU00	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
FSR1									
Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
00016A <sub>H</sub>	–	–	–	–	OCU51	OCU50	OCU41	OCU40	XXXX0000 <sub>B</sub>
	–	–	–	–	R/W	R/W	R/W	R/W	
FSR0									
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
00016B <sub>H</sub>	OCU31	OCU30	OCU21	OCU20	OCU11	OCU10	OCU01	OCU00	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
R/W: Readable/Writable									
X: Undefined value									
–: Undefined									

[bit23 to bit16] ICUn1,ICUn0: Free-run timer selection bits for the input capture

ICUn1	ICUn0	Free-run timer selection bits for the input capture
0	0	Selects free-run timer ch.0 [Initial value]
0	1	Selects free-run timer ch.1
1	0	Selects free-run timer ch.2
1	1	Setting disabled

(n = 0, 1, 2, 3: Input capture channel number)

Note: Do not change the setting during input capture operation.

[bit11 to bit0] OCUn1,OCUn0: Free-run timer selection bits for the output compare

OCUn1	OCUn0	Free-run timer selection bits for the output compare
0	0	Selects free-run timer ch.0 [Initial value]
0	1	Selects free-run timer ch.1
1	0	Selects free-run timer ch.2
1	1	Setting disabled

(n=0, 1, 2, 3, 4, 5: Output compare channel number)

Note: Do not change the setting during output compare operation.

## 12.5 Multifunction Timer Interrupt

The multifunction timer can generate 16-bit free-run timer interrupts, 16-bit output compare interrupts, 16-bit input capture interrupts, and waveform generator interrupts.

### ■ 16-bit Free-run Timer Interrupt

See Table 12.5-1 for 16-bit free-run timer interrupt control bits and interrupt causes.

**Table 12.5-1 16-bit Free-run Timer Interrupt Control Bits and Interrupt Causes**

	16-bit free-run timer	
	Compare clear	0-detect
Interrupt request flag bit	Timer status register Upper (TCCSH0, TCCSH1, TCCSH2) ICLR: bit9	Timer status register Upper (TCCSH0, TCCSH1, TCCSH2) IRQZF: bit14
Interrupt request enable bit	Timer status register Upper (TCCSH0, TCCSH1, TCCSH2) ICRE: bit8	Timer status register Upper (TCCSH0, TCCSH1, TCCSH2) IRQZE: bit13
Interrupt cause	The 16-bit free-run timer value and compare-clear register (CPCLRH0/CPCLRL0, CPCLRH1/CPCLRL1, CPCLRH2/CPCLRL2) match.	16-bit free-run timer goes to "0"

When the 16-bit free-run timer value and compare-clear register (CPCLRH0/CPCLRL0, CPCLRH1/CPCLRL1, CPCLRH2/CPCLRL2) match, the timer status register (TCCSH0, TCCSH1, TCCSH2) ICLR: bit9 is set to 1. When interrupt requests are enabled in this state (TCCSH0, TCCSH1, and TCCSH2 registers ICRE:bit8=1), the interrupt request is output to the interrupt controller.

When the timer value is "0000<sub>H</sub>", the timer status control register (TCCSH0, TCCSH1, TCCSH2) IRQZF: bit14 is set to "1". When interrupt requests are enabled in this state (TCCSH0, TCCSH1, and TCCSH2 registers IRQZE:bit13=1), the interrupt request is output to the interrupt controller.

## ■ 16-bit Output Compare Interrupt

See Table 12.5-2 for 16-bit output compare interrupt control bits and interrupt causes.

**Table 12.5-2 16-bit Output Compare 0 to 5 Interrupt Control Bits and Interrupt Causes**

	16-bit output compare 0, 1	16-bit output compare 2, 3	16-bit output compare 4, 5
Interrupt request flag bit	Compare control register Lower (OCSL0) IOP1, IOP0 (bit7, bit6)	Compare control register Lower (OCSL2) IOP1, IOP0 (bit7, bit6)	Compare control register Lower (OCSL4) IOP1, IOP0 (bit7, bit6)
Interrupt request enable bit	Compare control register Lower (OCSL0) IOE1, IOE0 (bit5, bit4)	Compare control register Lower (OCSL2) IOE1, IOE0 (bit5, bit4)	Compare control register Lower (OCSL4) IOE1, IOE0 (bit5, bit4)
Interrupt source	The 16-bit free-run timer value and output compare register (OCCPH0, OCCPH1, OCCPL0, OCCPL1) match.	The 16-bit free-run timer value and output compare register (OCCPH2, OCCPH3, OCCPL2, OCCPL3) match.	The 16-bit free-run timer value and output compare register (OCCPH4, OCCPH5, OCCPL4, OCCPL5) match.

When the 16-bit free-run timer value and output compare register (OCCPH0 to OCCPH5/OCCPL0 to OCCPL5) match, the compare control register low-order (OCSL0, OCSL2, and OCSL4) IOP1 and IOP0: bit7 and bit6 are set to 1. When interrupt requests are enabled in this state (OCSL0, OCSL2, and OCSL4 registers IOE1 and IOE0: bit5/bit4 = 11<sub>B</sub>), the interrupt request is output to the interrupt controller.

## ■ 16-bit Input Capture Interrupt

See Table 12.5-3 for 16-bit input capture interrupt control bits and interrupt causes.

**Table 12.5-3 16-bit Input Capture 0 to 3 Interrupt Control Bits and Interrupt Causes**

	16-bit input capture 0, 1	16-bit input capture 2, 3
Interrupt request flag bit	Input capture status control register Lower (PICSL01) ICP1, ICP0 (bit7, bit6)	Input capture status control register Lower (ICSL23) ICP3, ICP2 (bit7, bit6)
Interrupt request enable bit	Input capture status control register Lower (PICSL01) ICE1, ICE0 (bit5, bit4)	Input capture status control register Lower (ICSL23) ICP3, ICP2 (bit5, bit4)
Interrupt source	Valid edges are detected by IC pins 0 and 1.	Valid edges are detected by IC pins 2 and 3.

With 16-bit input capture, when a valid edge is detected by IC pins 0, 1, 2, and 3, the input capture-status control registers (PICSL01 and ICSL23) ICP3, ICP2, ICP1, and ICP0: bit7 and bit6 are both set to "11<sub>B</sub>". When interrupt requests are enabled in this state (PICSL01 and ICSL23 registers ICE3, ICE2, ICE1, ICE0: bit5 and bit4 are both 11<sub>B</sub>), interrupt requests are output to the interrupt controller.

## ■ Waveform Generator Interrupts

See Table 12.5-4 for waveform generator interrupt control bits and interrupt causes.

**Table 12.5-4 Waveform Generator Interrupt Control Bits and Interrupt Causes**

	Waveform generator	
	16-bit dead timer 0, 1, 2	DTTI pin
Interrupt request flag bit	16-bit dead timer control register Upper, Lower (DTCR0, DTCR1, DTCR2) TMIF0 to TMIF2 (Upper is bit12, Lower is bit4)	Waveform control register1 (SIGCR1) DTIF (bit6)
Interrupt request enable bit	16-bit dead timer control register Upper, Lower (DTCR0, DTCR1, DTCR2) TMIE0 to TMIE2 (Upper is bit11, Lower is bit3)	Waveform control register1 (SIGCR1) DTIF (bit7)
Interrupt source	Underflow in 16-bit dead timer 0, 1, 2	Low level detected in DTTI pin.

The waveform generator sets TMIF0 to TMIF2 (upper bit12 and lower bit4) in the 16-bit dead timer control register (DTCR0, DTCR1, DTCR2) to "1" when an underflow occurs on the 16-bit dead timer and the TMD8 to TMD0 bits in the DTCR0, DTCR1, DTCR2 registers (upper bit10 to bit8, lower bit2 to bit0) are set to "000<sub>B</sub>" or "001<sub>B</sub>". When interrupt requests are enabled in this state (DTCR0, DTCR1, and DTCR2 registers TMIE0 to TMIE2 (upper bit is 11, lower bit is 3) = 1), the interrupt request is output to the interrupt controller.

## 12.6 Operation of the Multifunction Timer

---

The operation of the multifunction timer is described below.

---

### ■ Operation of the Multifunction Timer

- 16-bit free-run timer

When the 16-bit free-run timer enables count operation, the counter begins counting up from the value set in the timer data register (TCDTH/TCDTL). The count value is used as the standard time of the 16-bit output compare and 16-bit input capture.

- 16-bit output compare

16-bit output compare is used to compare the value set in the output compare register with the 16-bit free-run timer value. If a match is detected, the interrupt flag is set, and the output level is reversed.

- 16-bit input capture

16-bit input capture is used to detect specified valid edges.

When a valid edge is detected, the interrupt flag is set, and the value of the 16-bit free-run timer is retrieved, and stored in the input capture data register.

- Waveform generator

The waveform generator generates a variety of waveforms (including dead times) using real-time output (RTO0 - RTO5), the 16-bit PPG timer 0, and 16-bit dead timer.

- A/D activation compare

An A/D trigger is generated when the 16-bit free-run timer 0 reaches the specified value.

## 12.6.1 Operation of 16-bit Free-run Timer

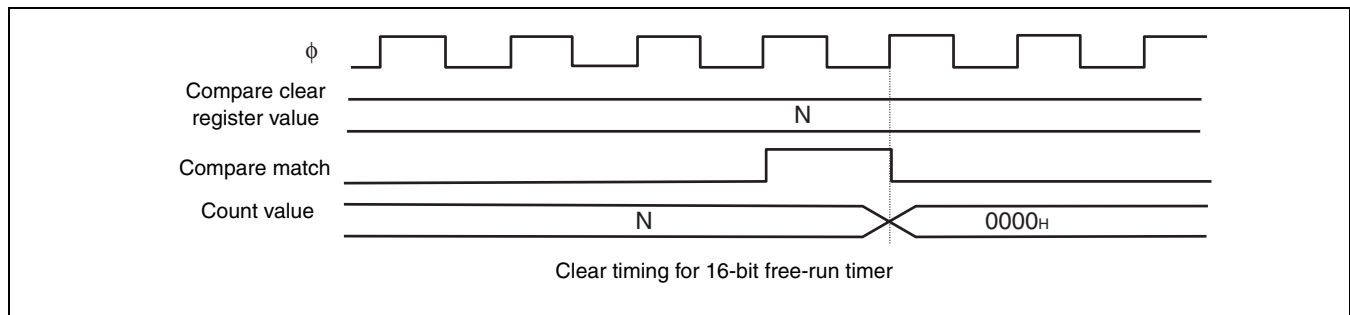
Three 16-bit free-run timer units are provided and these start up-counting from the value set in the timer data register (TCDTH0, TCDTH1, TCDTH2 and TCDTL0, TCDTL1, TCDTL2) after a reset completes. The count value is used as the standard time of the 16-bit output compare and 16-bit input capture.

### ■ Timer Clear

The count value of the 16-bit free-run timer is cleared when one of the following holds:

- When a match with the compare-clear register is detected via up-count mode (TCCSL registers MODE: bit5 = 0)
- When "1" is written to bit4 (SCLR) of the TCCSL register during operation
- When "0000<sub>H</sub>" is written to the TCDTH/TCDTL register when operation is halted
- When a reset occurs

After a reset, the counter is immediately cleared. The counter is cleared, synchronized with the count timing, when cleared by software or when a match with the compare clear register occurs.



### ■ Timer Mode

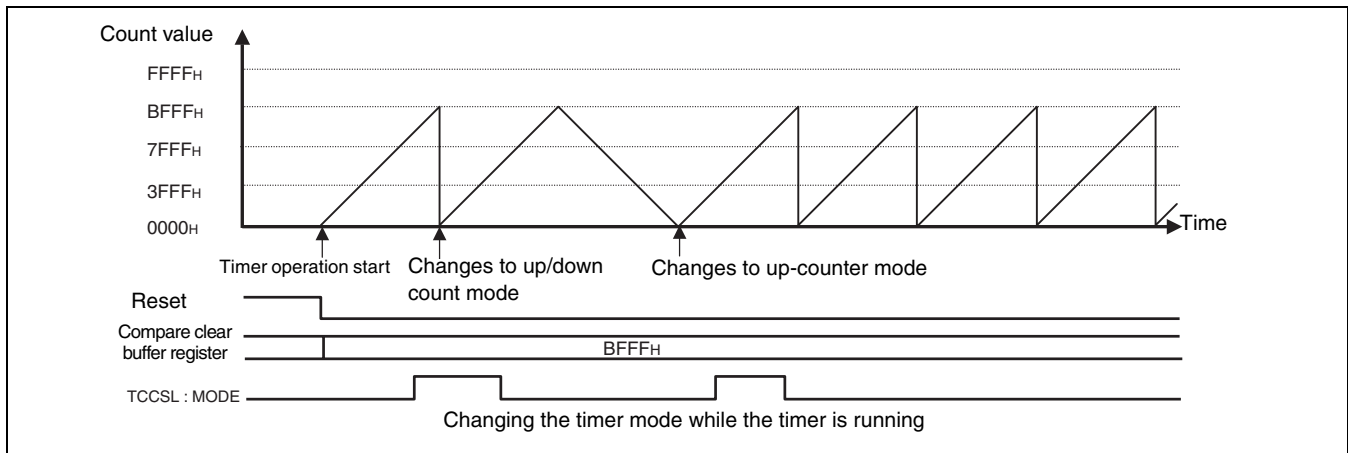
The following modes can be selected for the 16-bit free-run timer.

- Up-count mode (TCCSL registers MODE: bit5 = 0)
- Up/down count mode (TCCSL registers MODE: bit5=1)

In the up-count mode, the counter starts counting from the preset timer data register (TCDTH/TCDTL) and continues counting up until the count value matches the value in the compare clear register (CPCLRH/CPCLRL). Then, the counter is cleared to "0000<sub>H</sub>" and the counter restarts counting up.

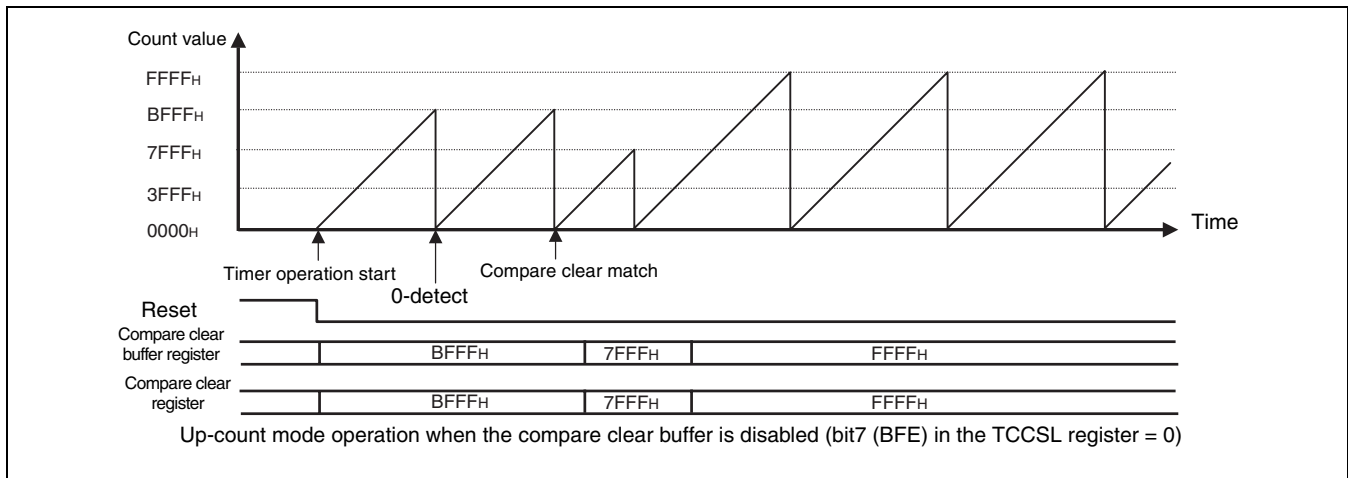
In the up/down count mode, the counter starts counting from the preset timer data register (TCDTH/TCDTL) and continues counting up until the count value matches the value in the compare clear register (CPCLRH/CPCLRL). Then, counting changes from the up-count mode to the down-count mode, the counter value performs counting down until it reaches to "0000<sub>H</sub>", and the counter restarts counting up.

You can write to the mode bit (bit5 (MODE) in the TCCSL register) at any time regardless of whether the timer is running or halted. The value written to the bit when the timer is running is stored in a buffer and the actual count mode does not change until the timer value reaches "0000<sub>H</sub>".

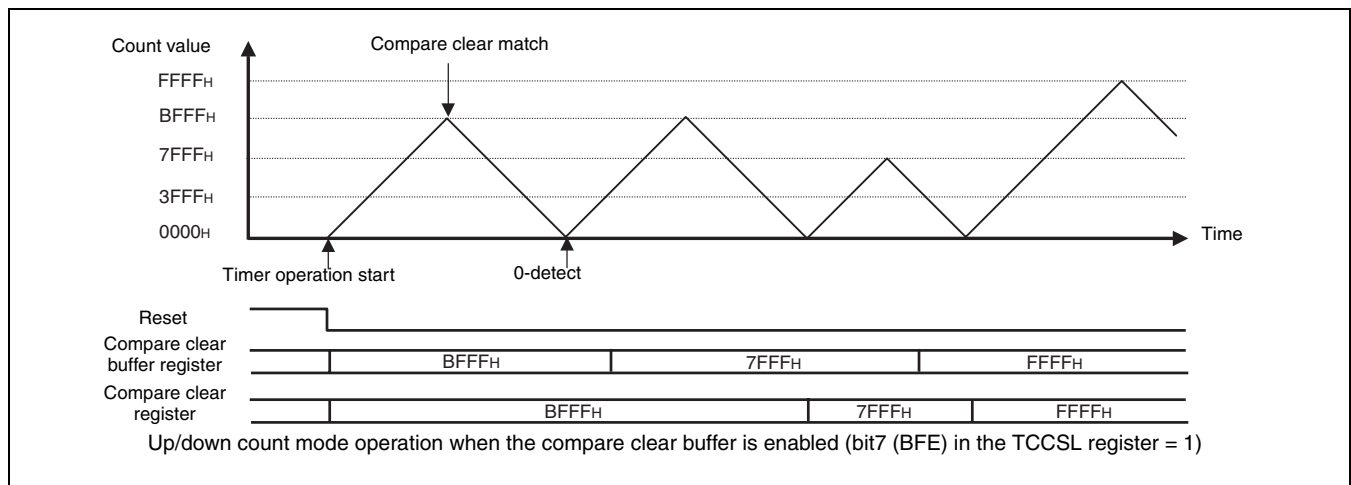
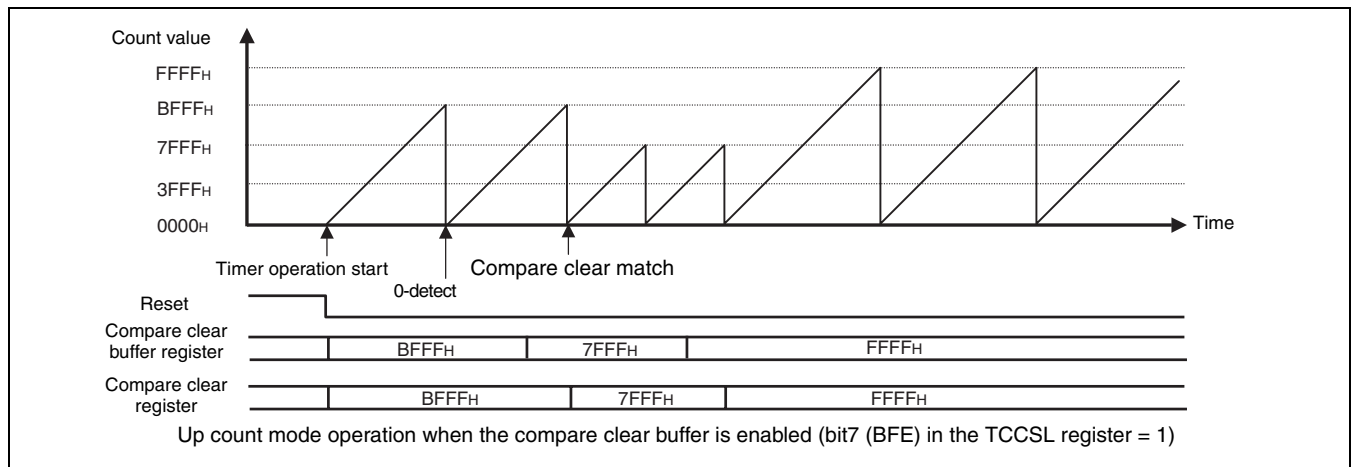


### ■ Compare Clear Buffer

The compare-clear register (CPCLR<sub>H</sub>/CPCLR<sub>L</sub>) has a buffer feature that can be enabled or disabled. When the buffer function is enabled (bit7 (BFE) in the TCCSL register = 1), data written to the compare clear buffer register (CPCLRB<sub>H</sub>/CPCLRB<sub>L</sub>) is transferred to the CPCLR<sub>H</sub>/CPCLR<sub>L</sub> register when zero is detected on the 16-bit free-run timer. The CPCLRB<sub>H</sub>/CPCLRB<sub>L</sub> register becomes invisible when the buffer function is disabled (bit7 (BFE) in the TCCSL register = 0) and data is written directly to the CPCLR<sub>H</sub>/CPCLR<sub>L</sub> register.







## ■ Timer Interrupt

The 16-bit free-run timer can generate the following two interrupts.

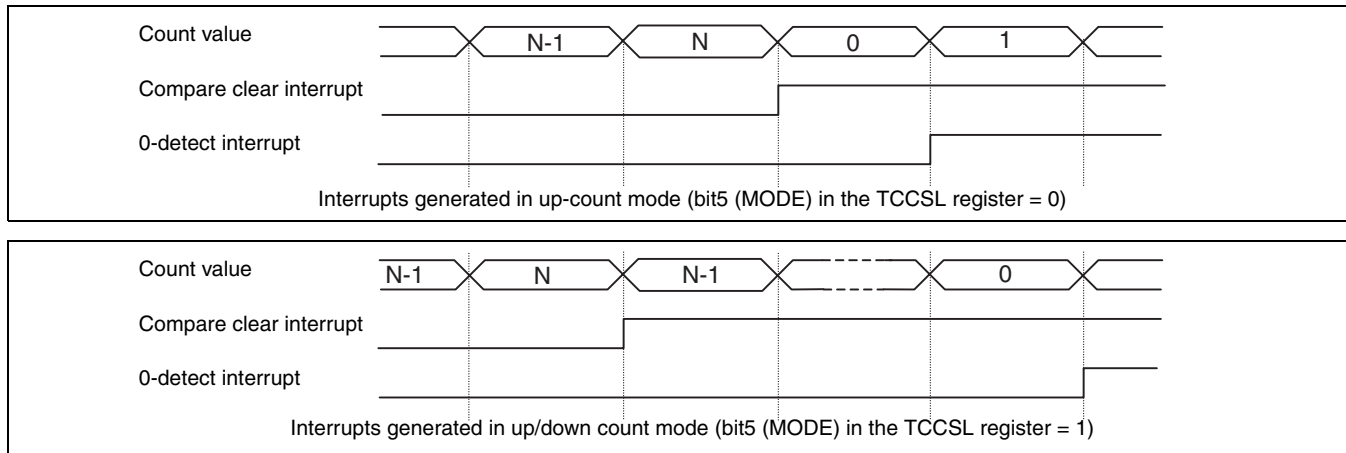
- Compare clear interrupt
- 0-detect interrupt

Compare-clear interrupts are generated when the timer value matches the value of the compare-clear register (COCLR/COCLRL).

0-detect interrupts are generated when the timer value reaches "0000H".

**Note:**

A software clear (setting bit4 (SCLR) in the TCCSL register = 1) does not generate a 0-detect interrupt.



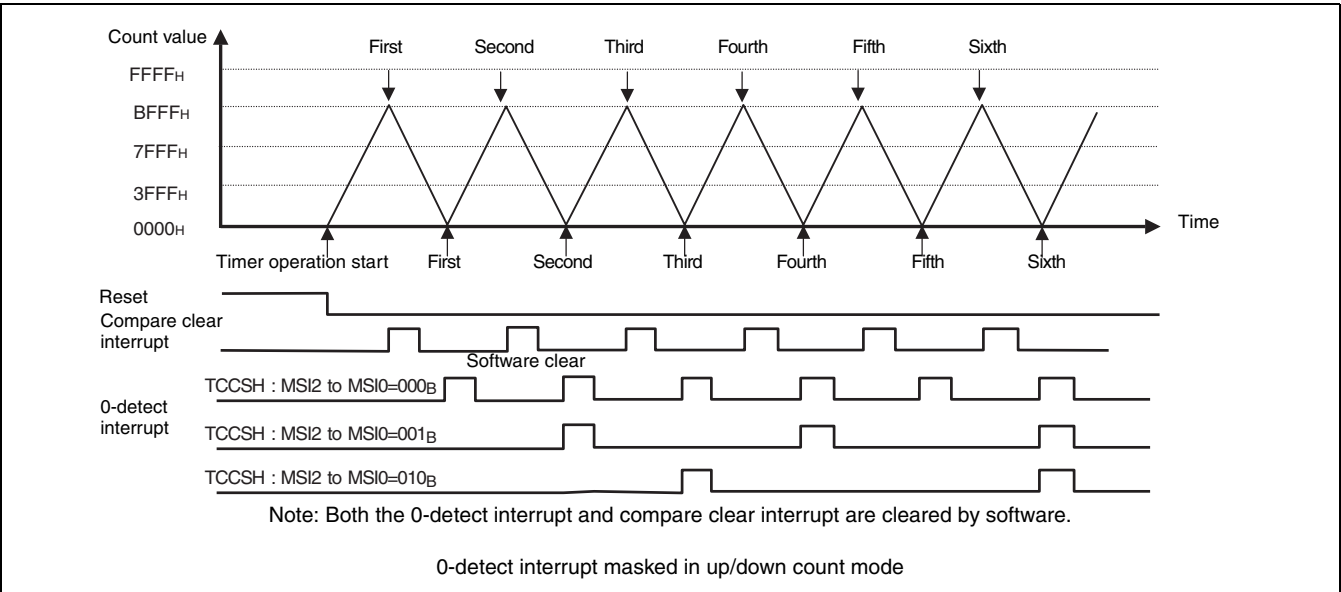
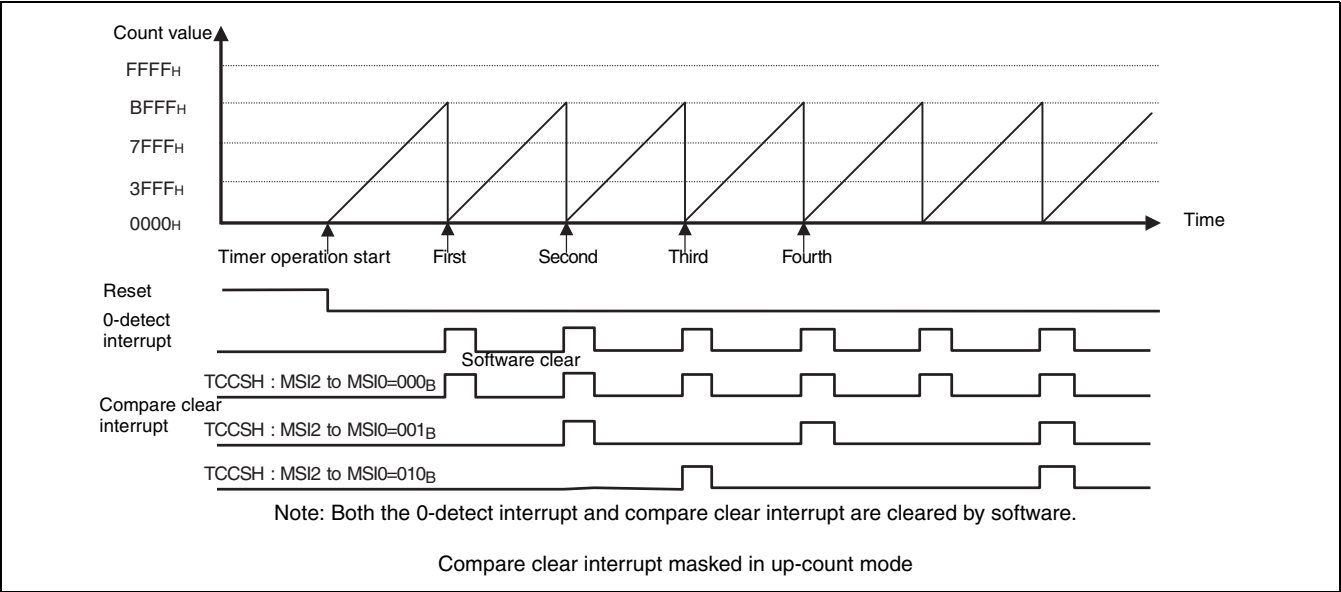
## ■ Interrupt Mask Function

It is possible to mask interrupt requests by setting the TCCSH registers MSI2 to MSI0: bit12-bit10. MSI2 bit to MSI0 bit are a 3-bit reload down counter that reload when the count value reaches "000<sub>B</sub>". The count value can also be loaded by writing directly to the MSI2 to MSI0 bits. The mask count is the value set in MSI2 bit to MSI0 bit. When MSI2 bit to MSI0 bit are "000<sub>B</sub>", interrupt causes are not masked.

The interrupt cause depends on the count mode (TCCSL registers MODE: bit5). In up-count mode, it is only possible to mask compare-clear interrupts, and 0-detect interrupts are generated each time "0" is detected. In up/down count mode, it is only possible to mask 0-detect interrupts, and compare-clear interrupts are generated each time a compare clear is detected.

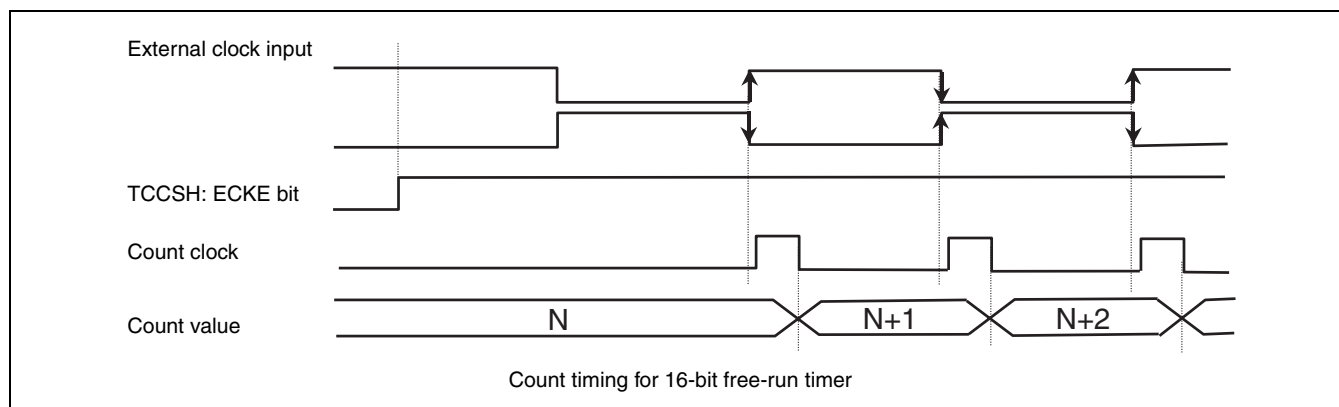
Note:

A software clear (setting bit4 (SCLR) in the TCCSL register = 1) does not generate a 0-detect interrupt.



## ■ Selected External Count Clock

The 16-bit free-run timer is incremented based on the input clock (internal clock or external clock). When the external clock is selected in external clock mode (bit15 (ECKE) in the TCCSH register = 1), the 16-bit free-run timer starts counting up on rising edges if the initial value of the external input is "1". Subsequently, it counts up on both edges. If the initial value of external input is "0", it starts counting up on a falling edge. Subsequently, it counts up on both edges.



Note:

The external clock input is counted on both edges.

## ■ A/D Activation Via Free-run Timer 0

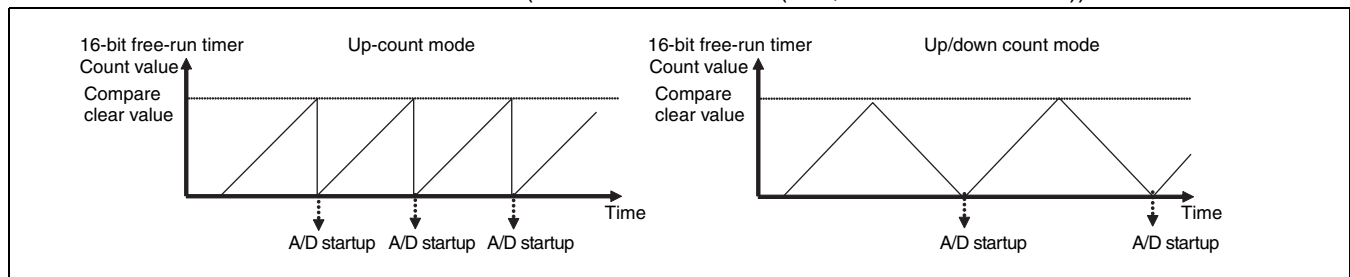
It is possible to activate A/D1 and A/D2 upon a compare match or 0-detection of the 16-bit free-run timer 0. The activation trigger can be selected by means of the A/D trigger cause selection bits (SEL1 and SEL2: bit2 and bit3) of the A/D trigger control register (ADTRGC).

It is possible to halt A/D activation signals, even upon compare match or 0-detection, via the A/D trigger output enable/disable bits (AD1E and AD2E: bit0 and bit1) of the A/D trigger control register (ADTRGC).

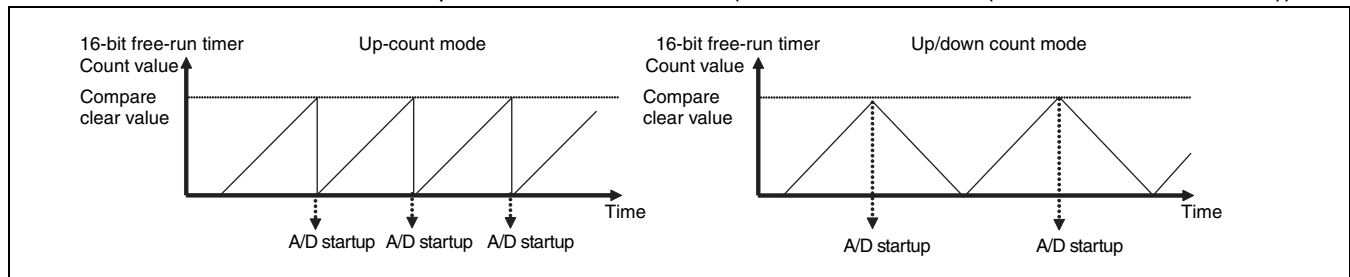
### Note:

When A/D activation signal output is disabled, if A/D activation signal output is enabled, then when an activation trigger compare match or 0-detection is output, the A/D activation signal will be output at the same time that the authorization is made.

#### ● Activate A/D when 0 is detected (ADTRGC: SEL<sub>n</sub> = 0 (n=1, 2: A/D unit number))



#### ● Activate A/D when a compare clear match occurs (ADTRGC: SEL<sub>n</sub> = 1 (n=1, 2: A/D unit number))



## 12.6.2 Operation of 16-bit Output Compare

Output compare is used to compare the value set in the compare clear register with the 16-bit free-run timer value. If a match is detected, the interrupt flag is set, and the output level is reversed.

If the free-run timer is in up/down count mode, match signals are ignored when the count peak and compare register value match.

### ■ Free-run Timer Selection for the 16-bit Output Compare

- One of the three free-run timers can be set for each channel of the 16-bit output compare.
- In the following explanation, the free-run timer refers to the selected free-run timer.
- Do not modify this setting while the output compare is operating.

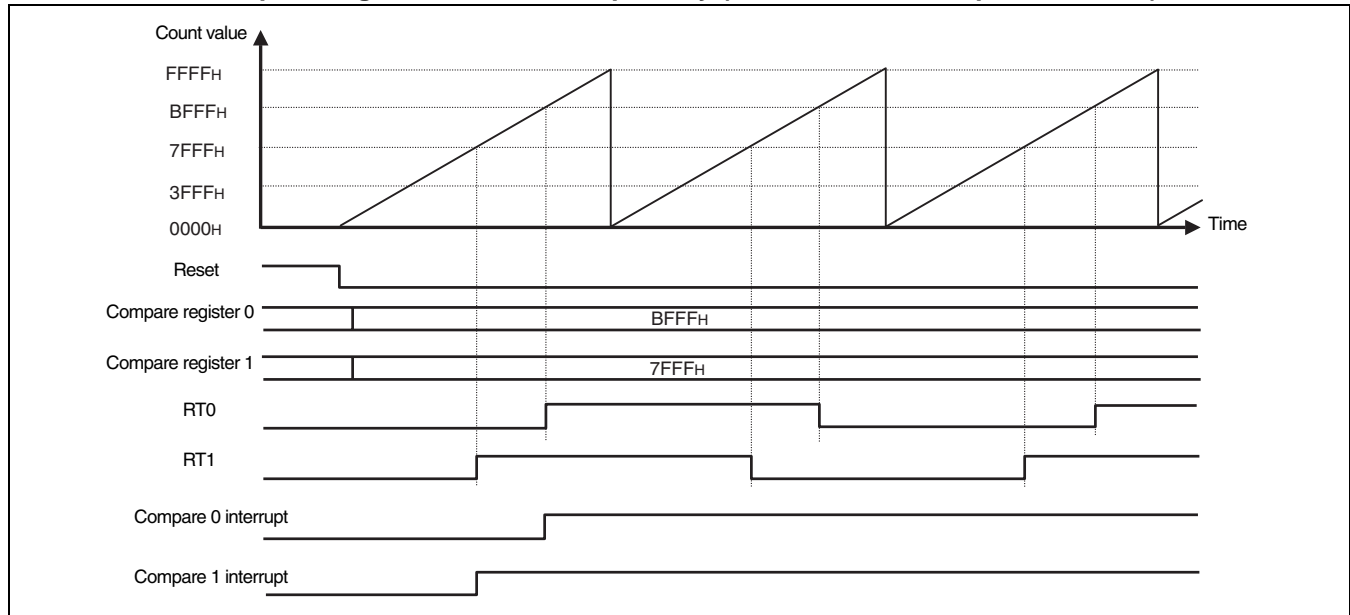
FSR0 to FSR2 register		Selected free-run timer
OCUn1	OCUn0	
0	0	Selects free-run timer ch.0 [Initial value]
0	1	Selects free-run timer ch.1
1	0	Selects free-run timer ch.2
1	1	Setting disabled

(n = 0, 1, 2, 3, 4, 5 : Output compare channel number)

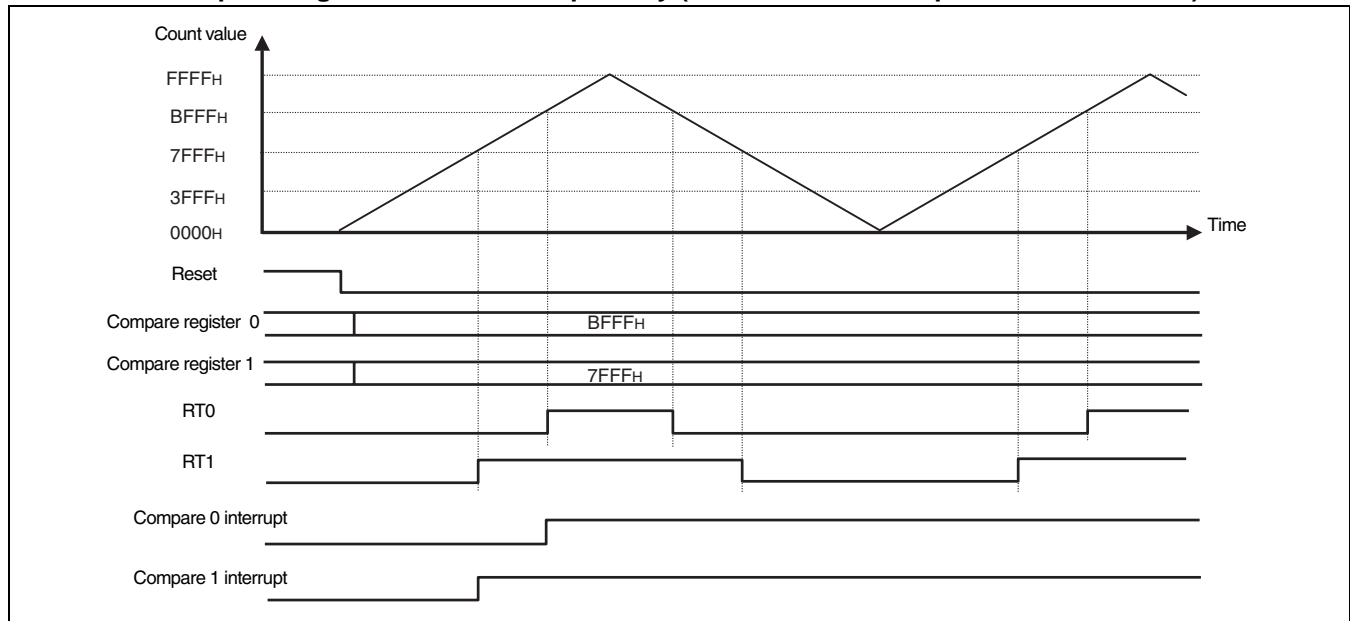
### ■ 16-bit Output Compare Operation (Inversion Mode, MOD1x = 0)

- Compare operation can be performed on each channel (compare control register higher-orders (OCSH1, OCSH3, and OCSH5) CMOD: bit12 = 0)

**Figure 12.6-1 Example of Output Waveform when the Initial Output is "0" and Compare Register 0 and Compare Register 1 are Used Separately (Free-run Timer in Up-count Mode)**

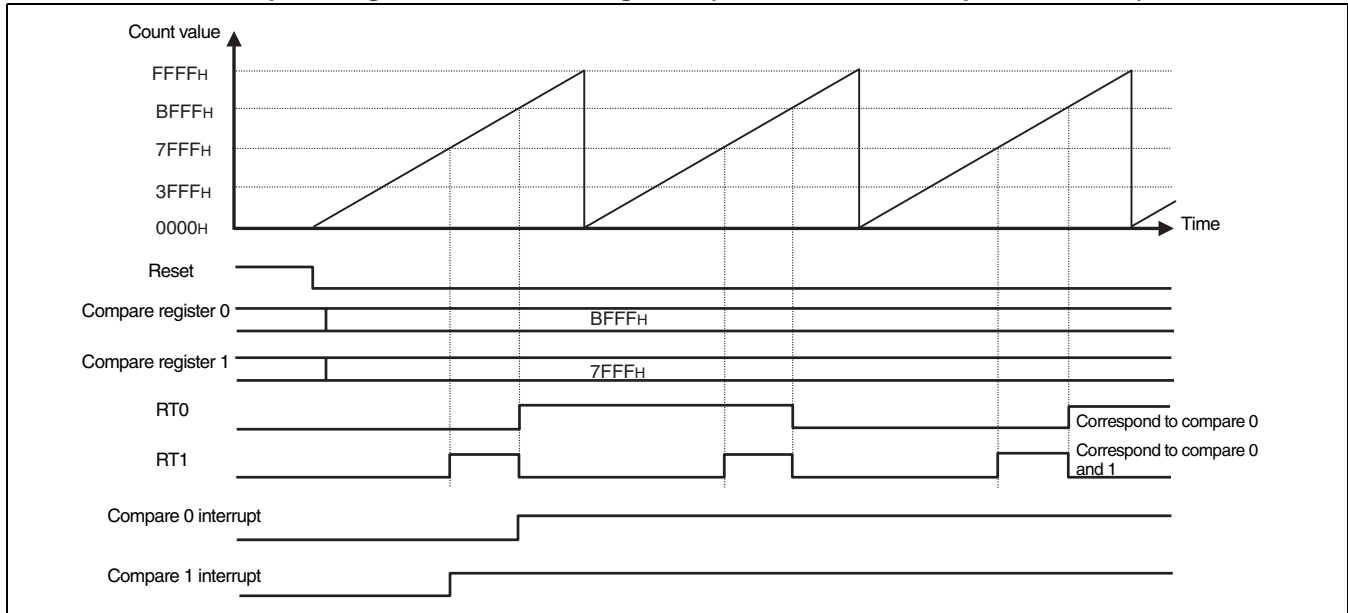


**Figure 12.6-2 Example of Output Waveform when the Initial Output is "0" and Compare Register 0 and Compare Register 1 are Used Separately (Free-run Timer in Up/Down Count Mode)**

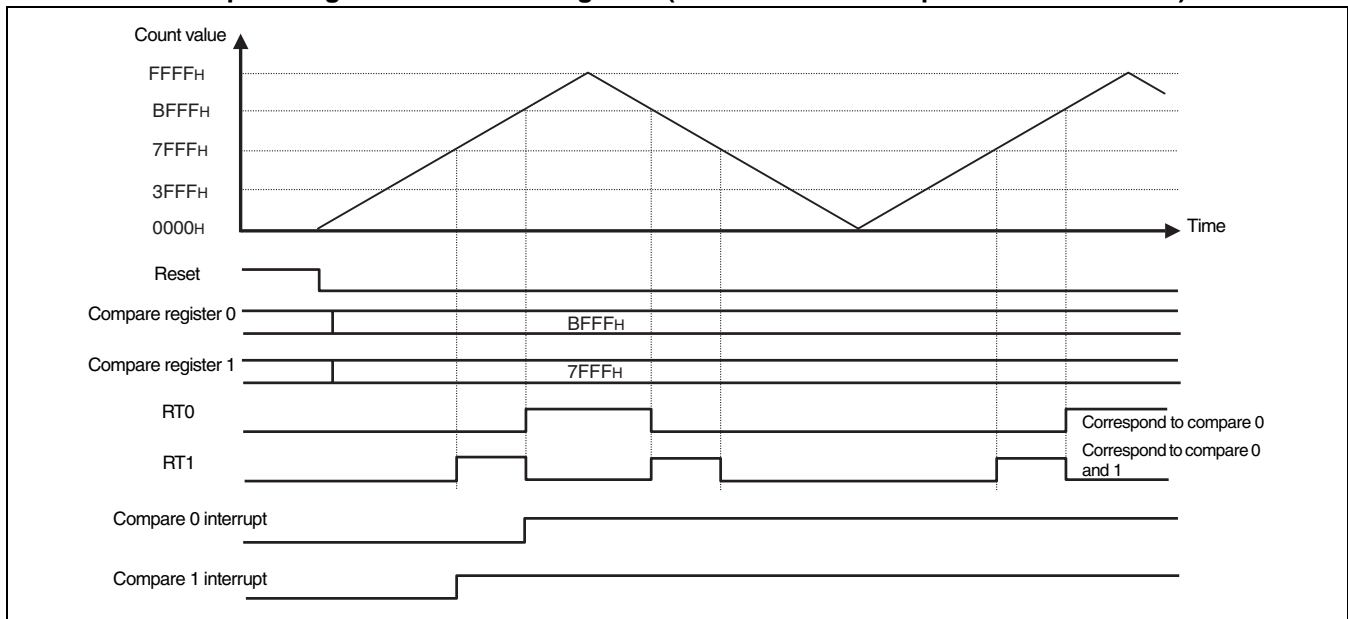


- The output level can use a pair of compare register (compare control register higher-orders (OCSH1, OCSH3, and OCSH5) CMOD: bit12 = 1)

**Figure 12.6-3 Example of Output Waveform when the Initial Output is "0" and Compare Register 0 and Compare Register 1 are Used Together (Free-run Timer in Up-count Mode)**



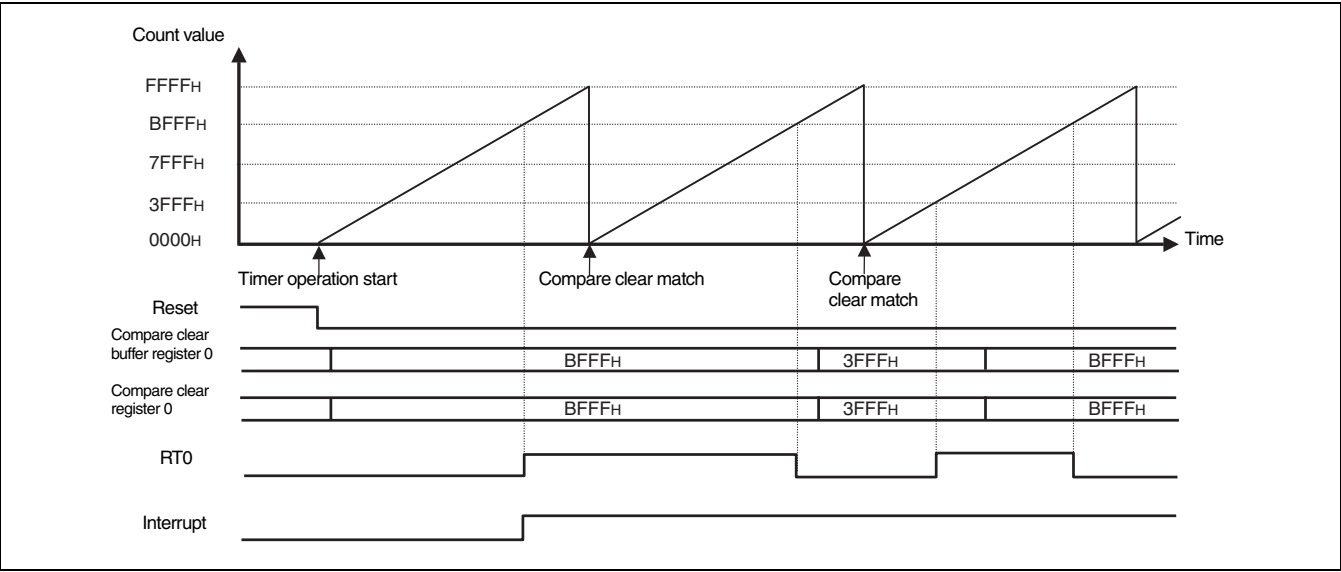
**Figure 12.6-4 Example of Output Waveform when the Initial Output is "0" and Compare Register 0 and Compare Register 1 are Used Together (Free-run Timer in Up/Down Count Mode)**





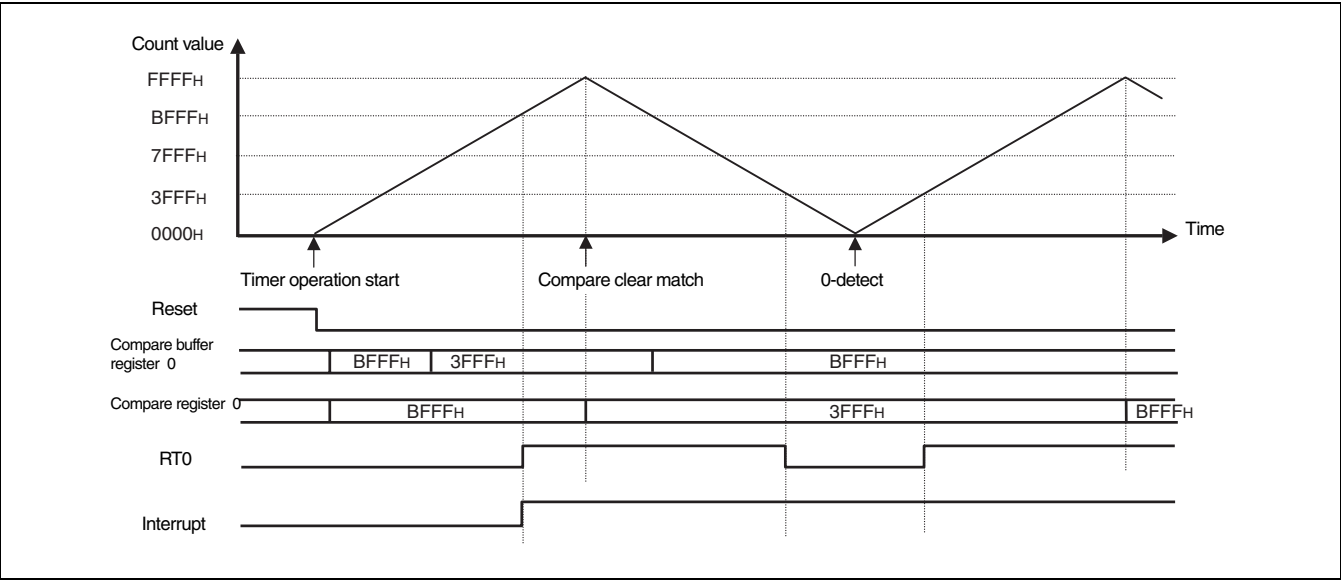
● Output level when compare buffer is disabled

**Figure 12.6-5 Example of Output Waveform when the Compare Buffer is Disabled  
(Free-run Timer in Up-count Mode)**

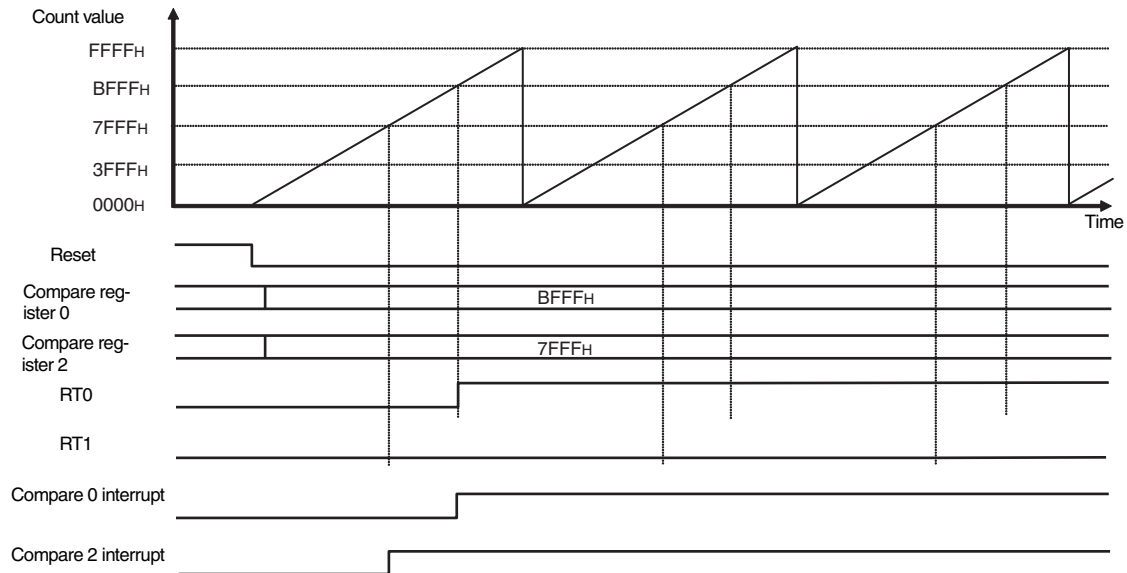


● Output level when compare buffer is selected, and a compare clear match occurs:

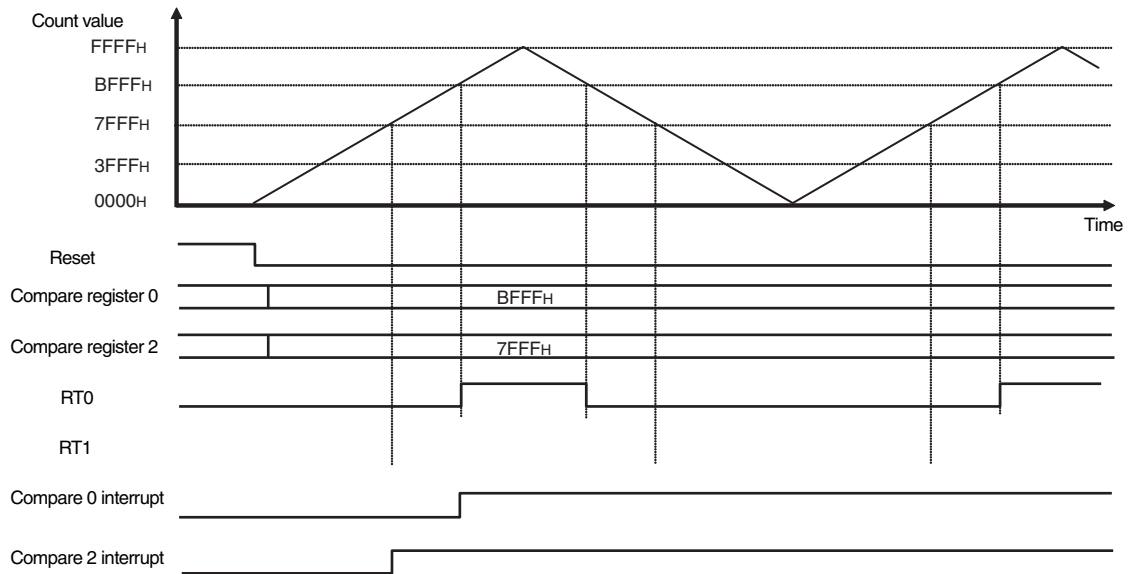
**Figure 12.6-6 Example of Output Waveform when the Compare Buffer is Enabled  
(Free-run Timer in Up/Down Count Mode)**



## ■ 16-bit Output Compare Operation (Set/Reset Mode, MOD1x = 1)



ch.0: Set on up-count, reset on down-count  
 ch.2: Reset on up-count, set on down-count  
 Note: Already "1" when match occurred on ch.0. ch.2 is always "0".



ch.0: Set on up-count, reset on down-count  
 ch.2: Reset on up-count, set on down-count

■ 16-bit Output Compare Timing

When a match occurs between the free-run timer and compare register, the output compare generates the compare match signal to reverse the output and generates an interrupt. When a compare match occurs, output is reversed in synchronization with the count timing of the counter.

Figure 12.6-7 Compare Register Interrupt Timing

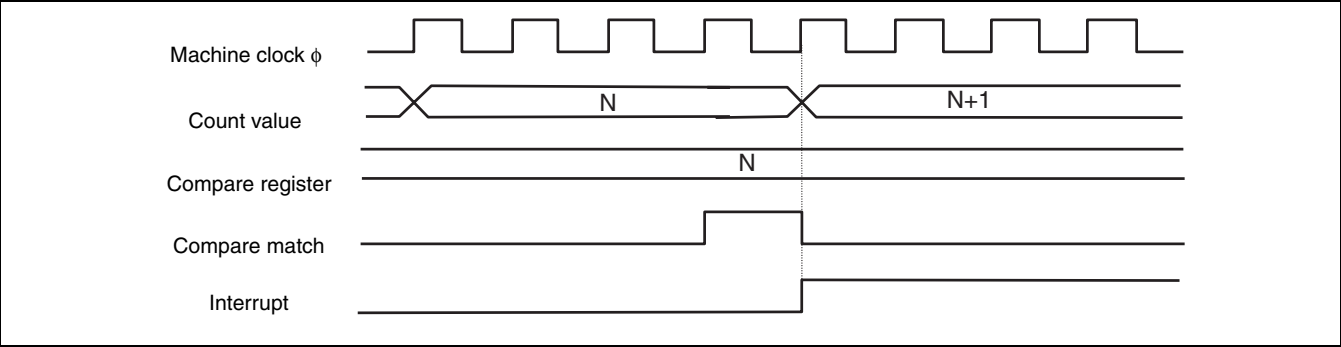
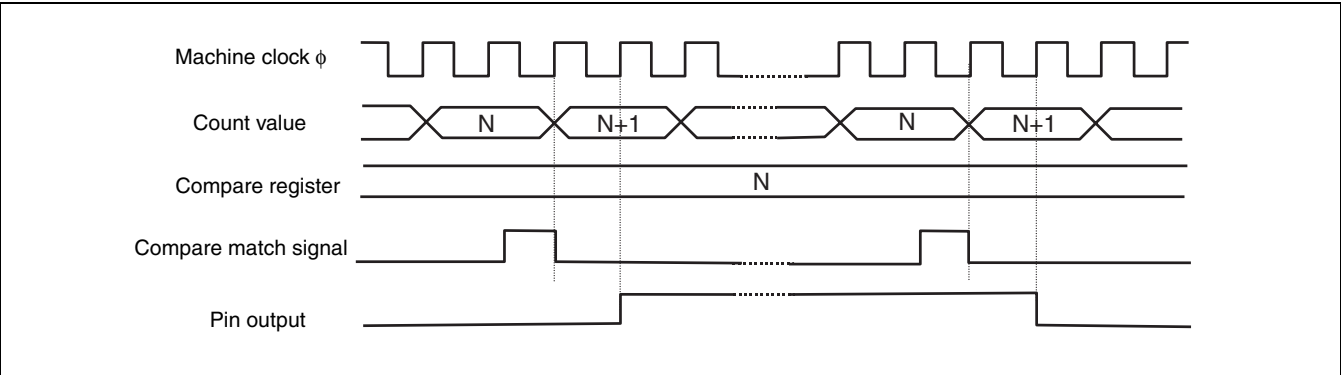
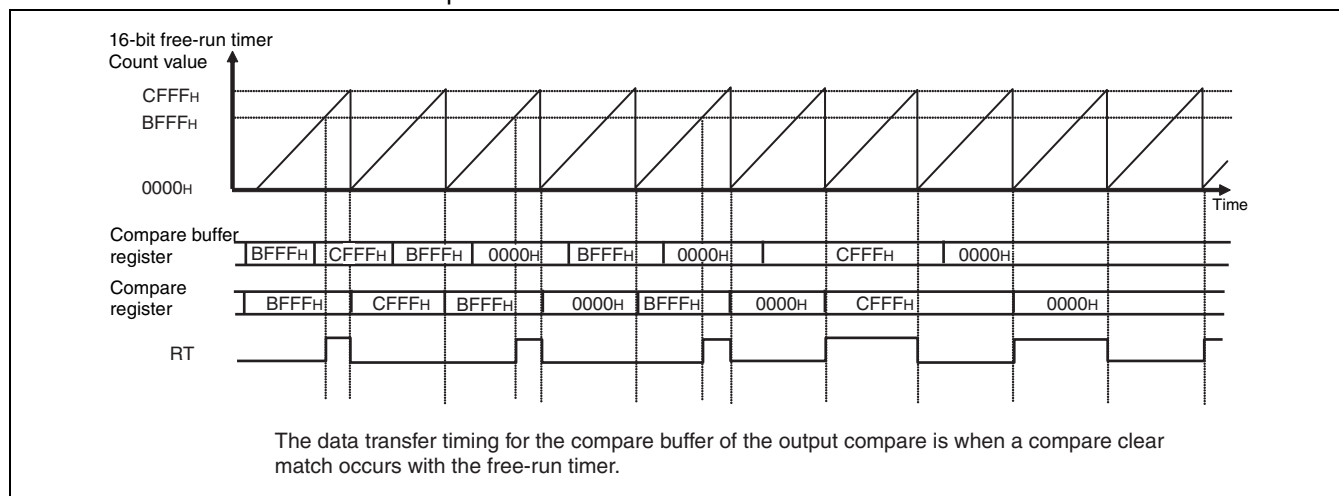


Figure 12.6-8 Pin Output Change Timing

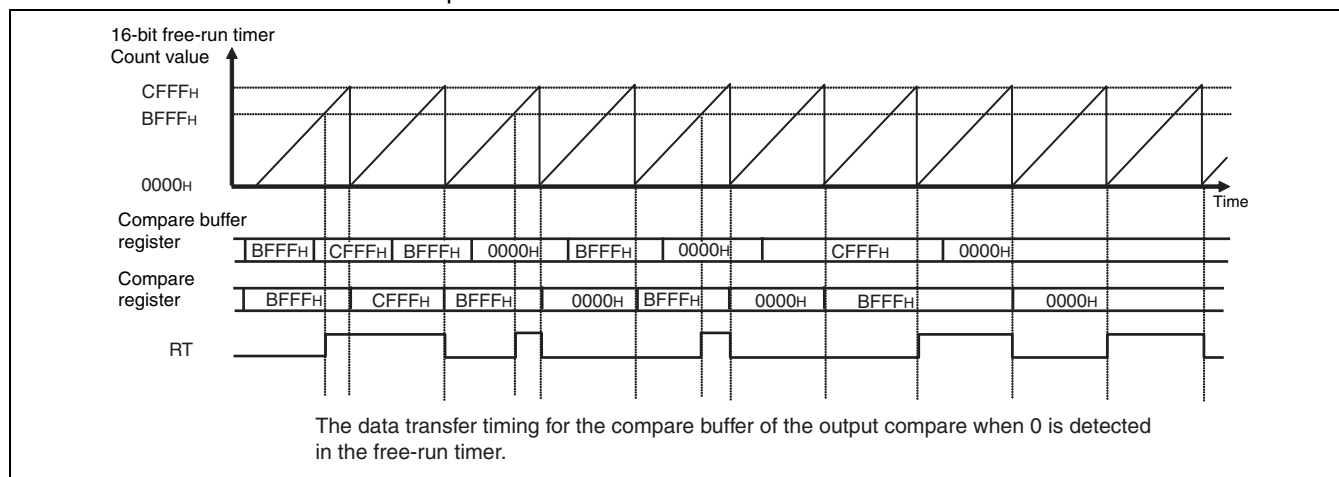


## ■ Operation of 16-bit Output Compare and Free-run Timer

### ● When the free-run timer up-counts



### ● When the free-run timer up-counts

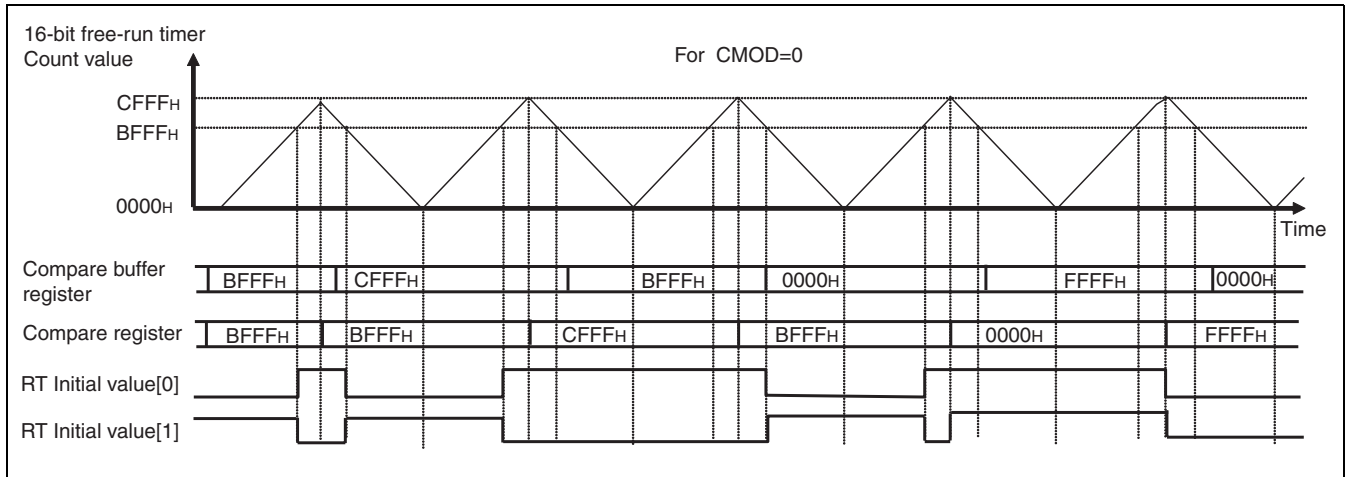


### ● When the free-run timer up/down counts

The data transfer timing for the compare buffer of the output compare is when a compare clear match occurs with the free-run timer.

When the output compare output mode is set to reverse the output when a match occurs

- RT is set to "1" when the compare register is set to "0000<sub>H</sub>" regardless of the free-run timer count value (reset to "0" when CMOD = 1).
- RT is reset to "0" when the compare register is set to "FFFF<sub>H</sub>", regardless of the free-run timer count value (set to "1" when CMOD = 1).
- No comparison is performed when the value of the compare clear register in the free-run timer is the same as the compare register in the output compare. In this case, RT is reset to "0" when both the compare clear register and compare register are set to "FFFF<sub>H</sub>", regardless of the free-run timer count value.

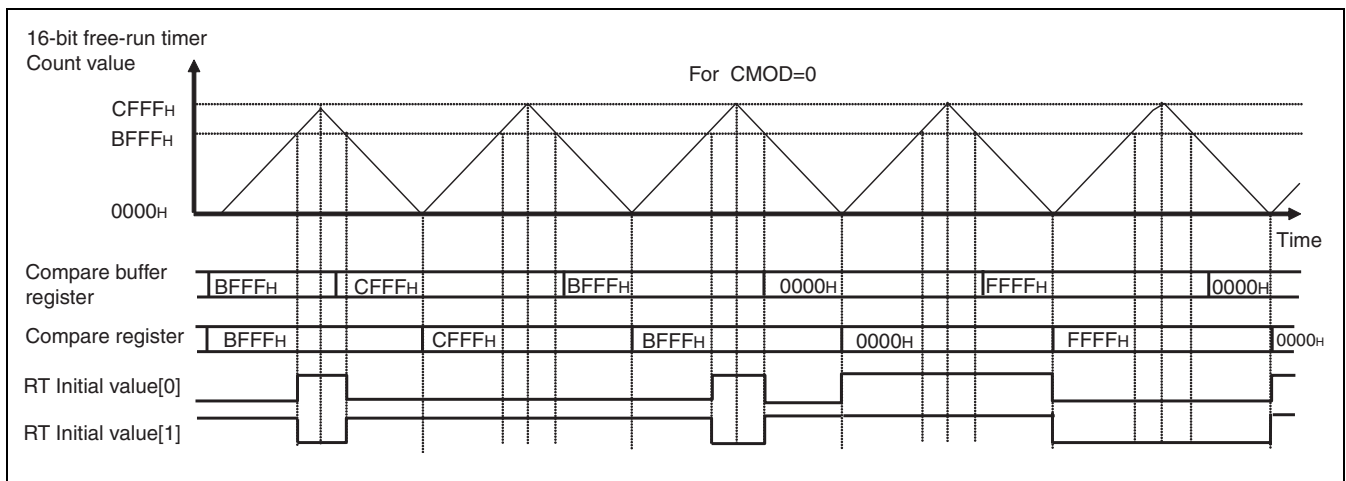


● When the free-run timer up/down counts

The data transfer timing for the compare buffer of the output compare is when 0 is detected in the free-run timer.

When the output compare output mode is set to reverse the output when a match occurs

- RT is set to "1" when the compare register is set to "0000<sub>H</sub>" regardless of the free-run timer count value (reset to "0" when CMOD = 1).
- RT is reset to "0" when the compare register is set to "FFFF<sub>H</sub>", regardless of the free-run timer count value (set to "1" when CMOD = 1).
- No comparison is performed when the value of the compare clear register in the free-run timer is the same as the compare register in the output compare. In this case, RT is reset to "0" when both the compare clear register and compare register are set to "FFFF<sub>H</sub>", regardless of the free-run timer count value.

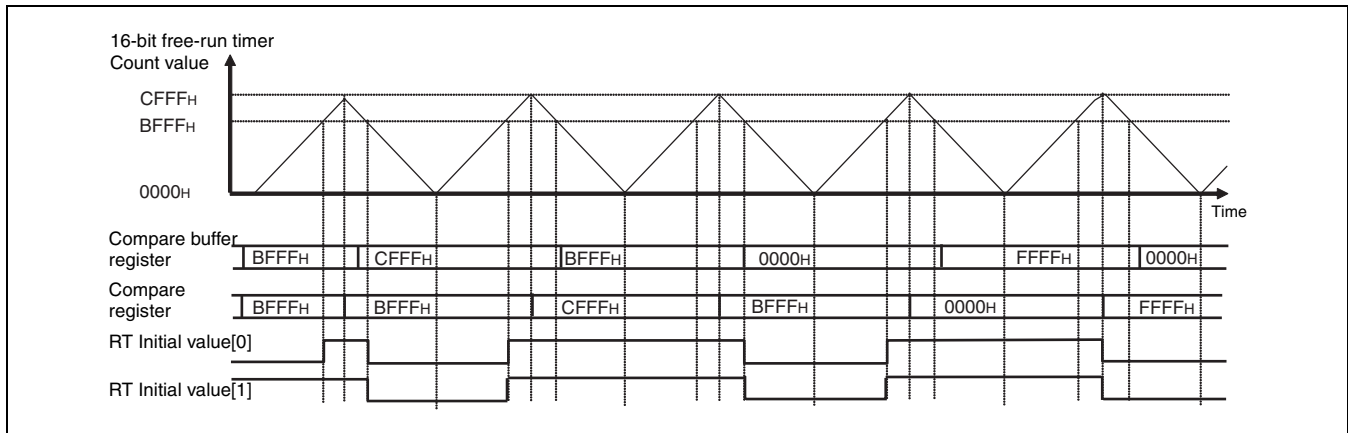


### ● When the free-run timer up/down counts

The data transfer timing for the compare buffer of the output compare is when a compare clear match occurs with the free-run timer.

The output compare output is set to "1" for a match due to an up-count and reset to "0" for a match due to a down-count (CMOD=0).

- RT is set to "1" when the compare register is set to "0000<sub>H</sub>" regardless of the free-run timer count value.
- RT is reset to "0" when the compare register is set to "FFFF<sub>H</sub>", regardless of the free-run timer count value.
- No comparison is performed when the value of the compare clear register in the free-run timer is the same as the compare register in the output compare. In this case, RT is reset to "0" when both the compare clear register and compare register are set to "FFFF<sub>H</sub>", regardless of the free-run timer count value.

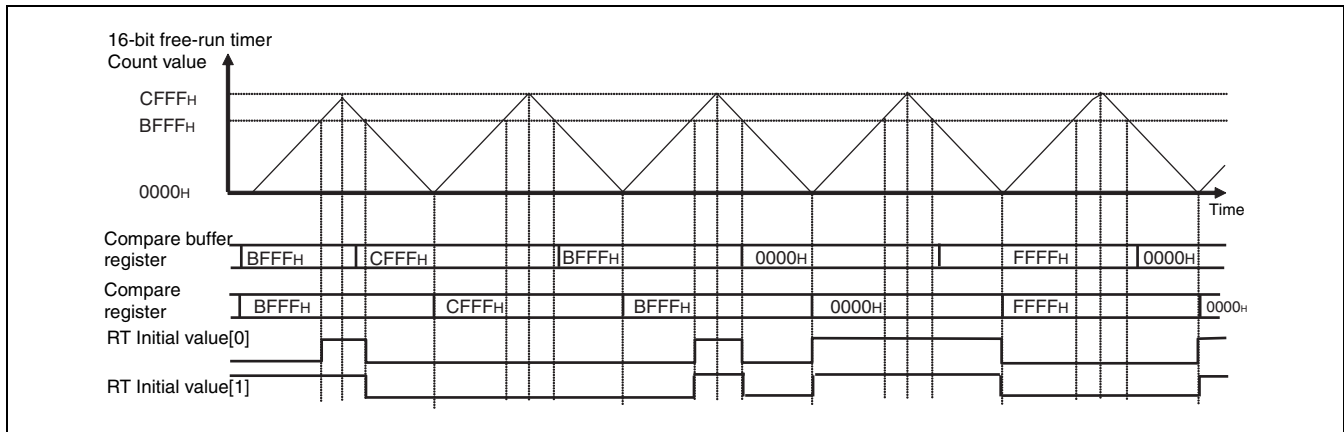


### ● When the free-run timer up/down counts

The data transfer timing for the compare buffer of the output compare is when 0 is detected in the free-run timer.

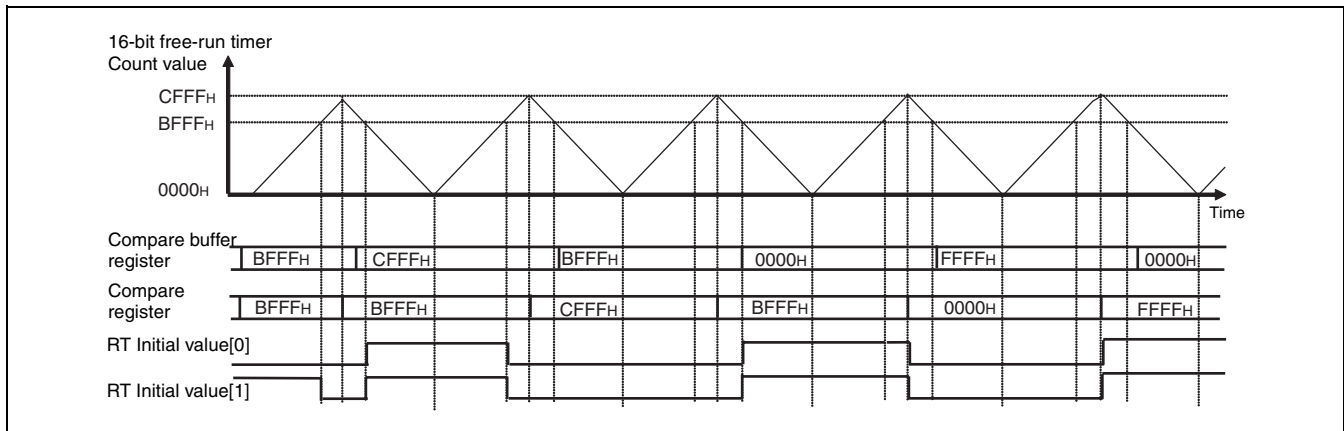
The output compare output is set to "1" for a match due to an up-count and reset to "0" for a match due to a down-count (CMOD=0).

- RT is set to "1" when the compare register is set to "0000<sub>H</sub>" regardless of the free-run timer count value.
- RT is reset to "0" when the compare register is set to "FFFF<sub>H</sub>", regardless of the free-run timer count value.
- No comparison is performed when the value of the compare clear register in the free-run timer is the same as the compare register in the output compare. In this case, RT is reset to "0" when both the compare clear register and compare register are set to "FFFF<sub>H</sub>", regardless of the free-run timer count value.



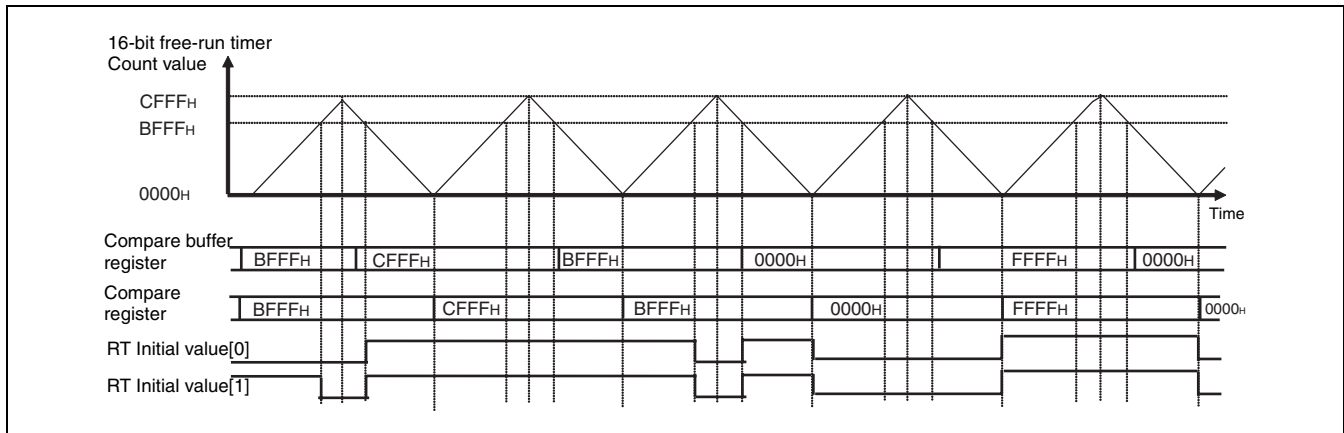
● When the free-run timer up/down counts

- The data transfer timing for the compare buffer of the output compare is when a compare clear match occurs with the free-run timer.
- Output compare output when up-count match is reset to 0, and down-count match is set to "1" (CMOD=1) :
  - RT is reset to "0" when the compare register is set to "0000<sub>H</sub>", regardless of the free-run timer count value.
  - RT is set to "1" when the compare register is set to "FFFF<sub>H</sub>" regardless of the free-run timer count value.
  - No comparison is performed when the value of the compare clear register in the free-run timer is the same as the compare register in the output compare. In this case, RT is reset to "0" when both the compare clear register and compare register are set to "FFFF<sub>H</sub>", regardless of the free-run timer count value.



● When the free-run timer up/down counts

- The data transfer timing for the compare buffer of the output compare is when zero is detected in the free-run timer.
- Output compare output when up-count match is reset to "0", and down-count match is set to "1" (CMOD=1) :
  - RT is reset to "0" when the compare register is set to "0000<sub>H</sub>", regardless of the free-run timer count value.
  - RT is set to "1" when the compare register is set to "FFFF<sub>H</sub>" regardless of the free-run timer count value.
  - No comparison is performed when the value of the compare clear register in the free-run timer is the same as the compare register in the output compare. In this case, RT is reset to "0" when both the compare clear register and compare register are set to "FFFF<sub>H</sub>", regardless of the free-run timer count value.





### 12.6.3 16-bit Input Capture Operation

Input capture is used to detect specified valid edges. When a valid edge is detected, the interrupt flag is set, and the value of the 16-bit free-run timer is loaded into the capture register.

#### ■ Free-run Timer Selection for the 16-bit Input Capture

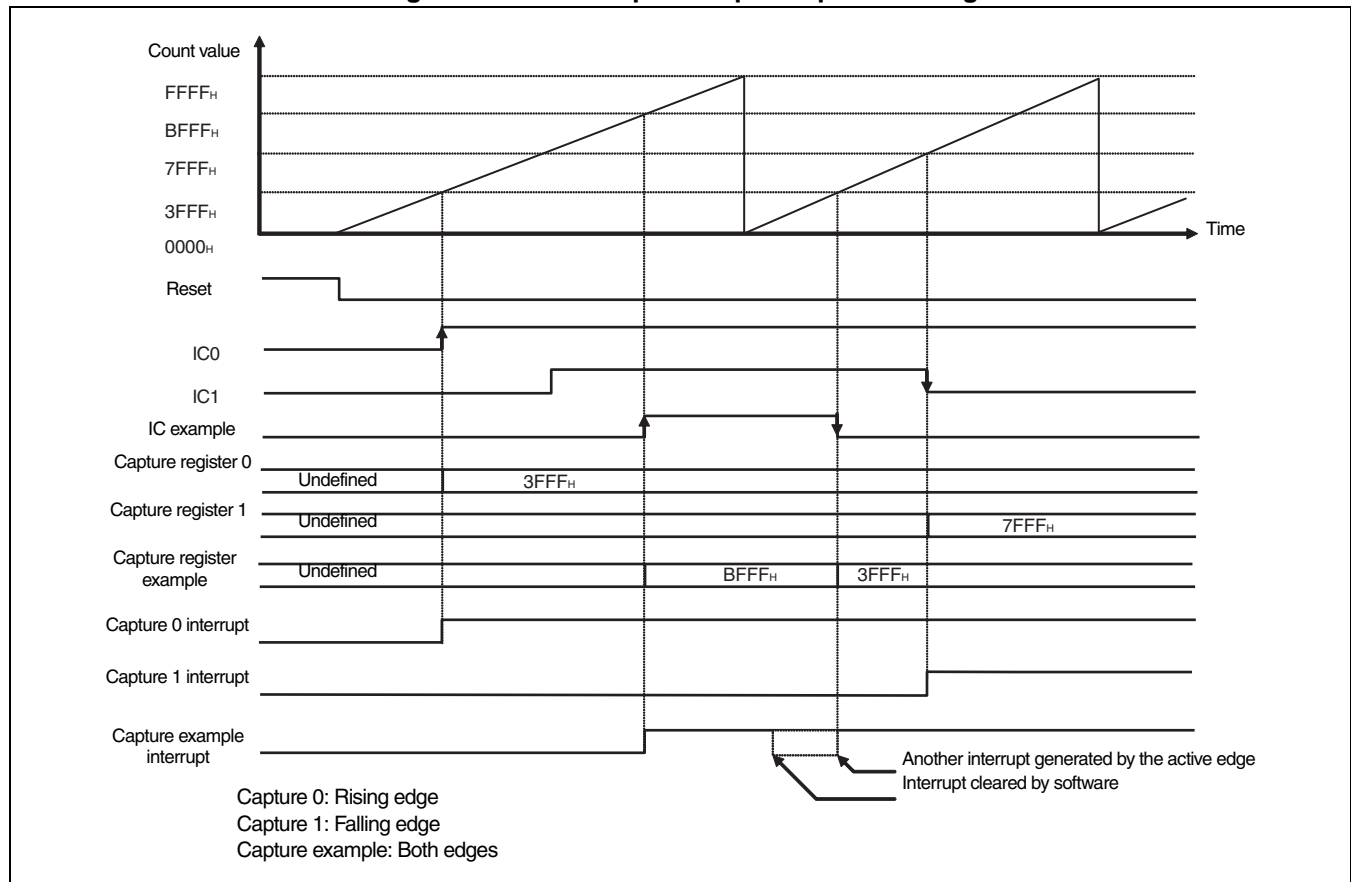
- One of the three free-run timers can be set for each channel of the 16-bit input capture.
- Do not modify this setting while the input capture is operating.

FSR2 register		Selected free-run timer
ICUn1	ICUn0	
0	0	Selects free-run timer ch.0 [Initial value]
0	1	Selects free-run timer ch.1
1	0	Selects free-run timer ch.2
1	1	Setting disabled

(n=0, 1, 2, 3: Input capture channel number)

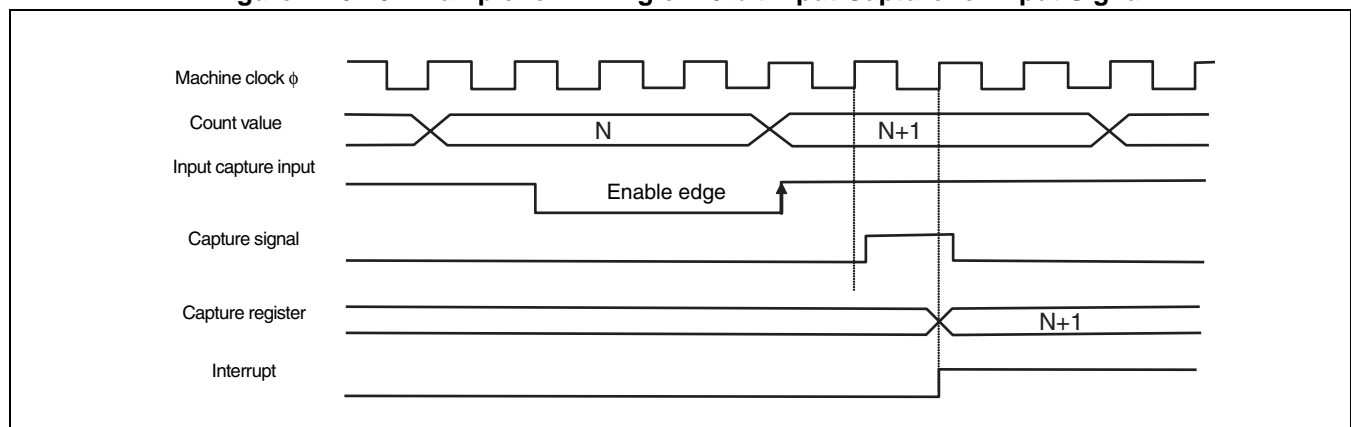
## ■ 16-bit Input Capture Operation

### Figure 12.6-9 Example of Input Capture Timing



## ■ Input Timing for 16-bit Input Capture

**Figure 12.6-10 Example for Timing of 16-bit Input Capture for Input Signal**



## 12.6.4 Waveform Generator Operation

The waveform generator can generate a variety of waveforms (including dead times) using real-time output (RTO0 to RTO5), the 16-bit PPG timer 0, and 16-bit dead timers 0, 1, and 2.

### ■ Output Status of RTO0 to RTO5 and GATE

Table 12.6-1 RTO0 to RTO5/GATE Output Status and Bit Settings (1 / 2)

TMD2	TMD1	TMD0	GTENx	PGENx	RTOx	GATE
0	0	0	X	X	Real-time output RTOx (16-bit output compare output)	Always "0"
0	0	1	X	0	Real-time output RTOx (16-bit output compare output)	(RTx & GTENx) *3
0	0	1	0	1	RTOx outputs a PPG0 pulse for duration H. *1	Always "0"
0	0	1	1	1	RTOx outputs the PPG0 pulse activated by the GATE signal for duration H.	(RT0/RT1/ RT2/RT3/ RT4/RT5)
0	1	0	X	0	16-bit dead timer 0 starts on a rising edge on RT0 or RT1 and "H" is output until the 16-bit dead timer 0 underflows.	"H" is output during timer operation *4
			X		16-bit dead timer 1 starts on a rising edge on RT2 or RT3 and "H" is output until the 16-bit dead timer 1 underflows.	
			X		16-bit dead timer 2 starts on a rising edge on RT4 or RT5 and "H" is output until the 16-bit dead timer 2 underflows.	
0	1	0	0	1	16-bit dead timer 0 starts on a rising edge on RT0 or RT1 and the PPG0 pulse is output until the 16-bit dead timer 0 underflows. *1	Always "0"
			0		16-bit dead timer 1 starts on a rising edge on RT2 or RT3 and the PPG0 pulse is output until the 16-bit dead timer 1 underflows. *1	
			0		16-bit dead timer 2 starts on a rising edge on RT4 or RT5 and the PPG0 pulse is output until the 16-bit dead timer 2 underflows. *1	
0	1	0	1	1	16-bit dead timer 0 starts on a rising edge on RT0 or RT1 and the PPG0 pulse activated by the GATE signal is output until the 16-bit dead timer 0 underflows.	"H" is output during timer operation *4
			1		16-bit dead timer 1 starts on a rising edge on RT2 or RT3 and the PPG0 pulse activated by the GATE signal is output until the 16-bit dead timer 1 underflows.	
			1		16-bit dead timer 2 starts on a rising edge on RT4 or RT5 and the PPG0 pulse activated by the GATE signal is output until the 16-bit dead timer 2 underflows.	

**Table 12.6-1 RTO0 to RTO5/GATE Output Status and Bit Settings (2 / 2)**

TMD2	TMD1	TMD0	GTENx	PGENx	RTOx	GATE
1	0	0	X	X	RT1 generates a non-overlapping signal. *2	Always "0"
			X		RT3 generates a non-overlapping signal. *2	
			X		RT5 generates a non-overlapping signal. *2	
1	1	1	0	X	A non-overlapping signal is generated by means of PPG 0.	Always "0"
1	1	1	1	X	A non-overlapping signal is generated by means of the PPG 0 activated by the GATE signal.	(RT0/RT1/ RT2/RT3/ RT4/RT5)
The others					Always "0"	Always "0"

x = 0 to 5

\*1: PPG0 needs to be started beforehand.

\*2: In order to generate a non-overlapping signal, first select 2-channel mode (compare control registers higher-order (OCSH1, OCSH3, and OCSH5) CMOD: bit12 = 1) for RT1, RT3, and RT5.

\*3: The GATE signal is generated from the RTx whose GTENx bit is set to "1".

\*4: The GATE signal is generated while the timer activated by the RTx whose GTENx bit is set to "1" is operating. If more than one GATEx bit is set to "1", the GATE signal is the OR of the signals of each of the operating timers.

---

**Note:**

RTO0 and RTO1 are controlled by the 16-bit dead timer control register higher-order (DTCR0) TMD2 to TMD0: bit10 to bit8, RTO2 and RTO3 are controlled by the lower-order of the DTCR1 register TMD5 to TMD3: bit2 to bit0, and RTO4 and RT5 are controlled by the higher-order of the DTCR2 register TMD8 to TMD6: bit10 to bit8.

---

## ■ PPG0 Output Control

PPG0 output to RTO0 pin to RTO5 pin can be enabled by means of the PPG output control/input capture-status control registers higher-order (PICSH01) PGEN5 to PGEN0: bit15 to bit10.

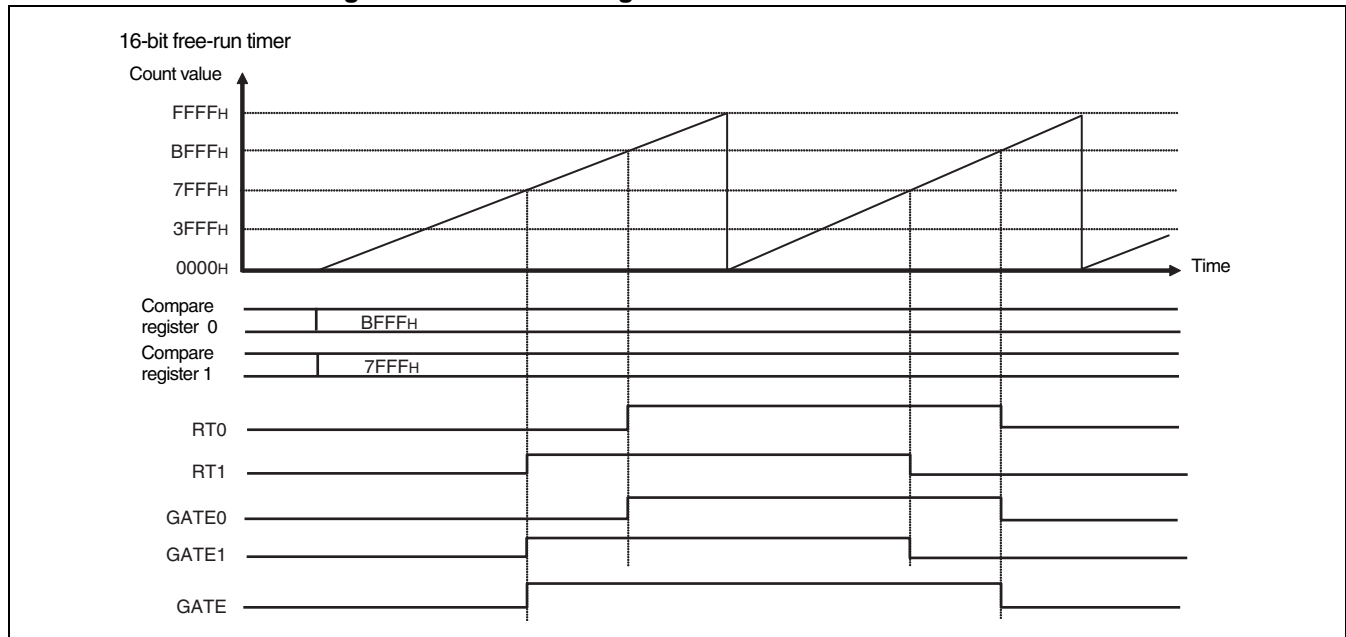
## ■ PPG0 Output Via Gate Trigger

The waveform generator can generate a GATE signal via real-time output RTO0 to RTO5, and the 16-bit dead timer 0, 1, and 2 can operate the PPG0 count as a trigger. Two real-time outputs (RTO0/RTO2/RTO4, RTO1/RTO3/RTO5) are controlled by one 16-bit dead timer 0, 1, and 2, generating six separate gate signals. Six gate signals are used logical sum and the GATE signal is generated, causing trigger of the PPG0 count.

Also, if PGEN0 to PGEN5 signals are used, it is possible to output 6 different waveforms to RTO0 pin to RTO5 pin using PPG0 alone.

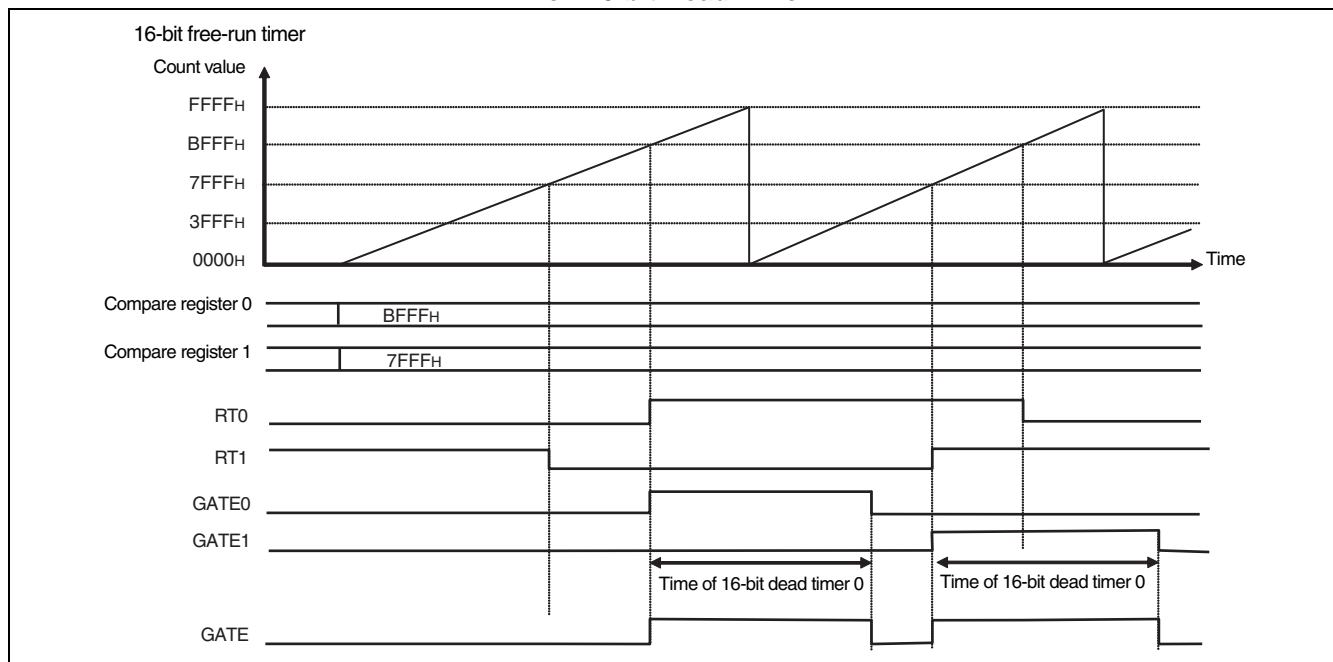
- GATE signal generation when GATENx is active and each RTx is at the "H" level (TMD8 to TMD0 (upper bit10 to bit8, lower bit2 to bit0) in the 16-bit dead timer control registers (DTCR0, DTCR1, DTCR2) are "001<sub>B</sub>" or "111<sub>B</sub>")

**Figure 12.6-11 GATE Signal Generation when RTx is "H"**



- GATE signal generation from rising edge on RTx until underflow on 16-bit dead timer 0, 1, 2 when GTENx is active (TMD8 to TMD0 in the DTCR0, DTCR1, DTCR2 registers = 010<sub>B</sub>)

**Figure 12.6-12 GATE Signal Generation from Rising Edge on RTx Until Underflow Occurs on 16-bit Dead Timer**



**Note:**

Each 16-bit dead timer is used for two RTs. In other words, 16-bit dead timer 0 is used for RT0 and RT1; 16-bit dead timer 1 is used for RT2 and RT3; and 16-bit dead timer 2 is used for RT4 and RT5. Consequently, you must not try to use a RT to activate a timer that is already operating. Attempting to do this will extend the GATE signal and therefore result in misoperation.

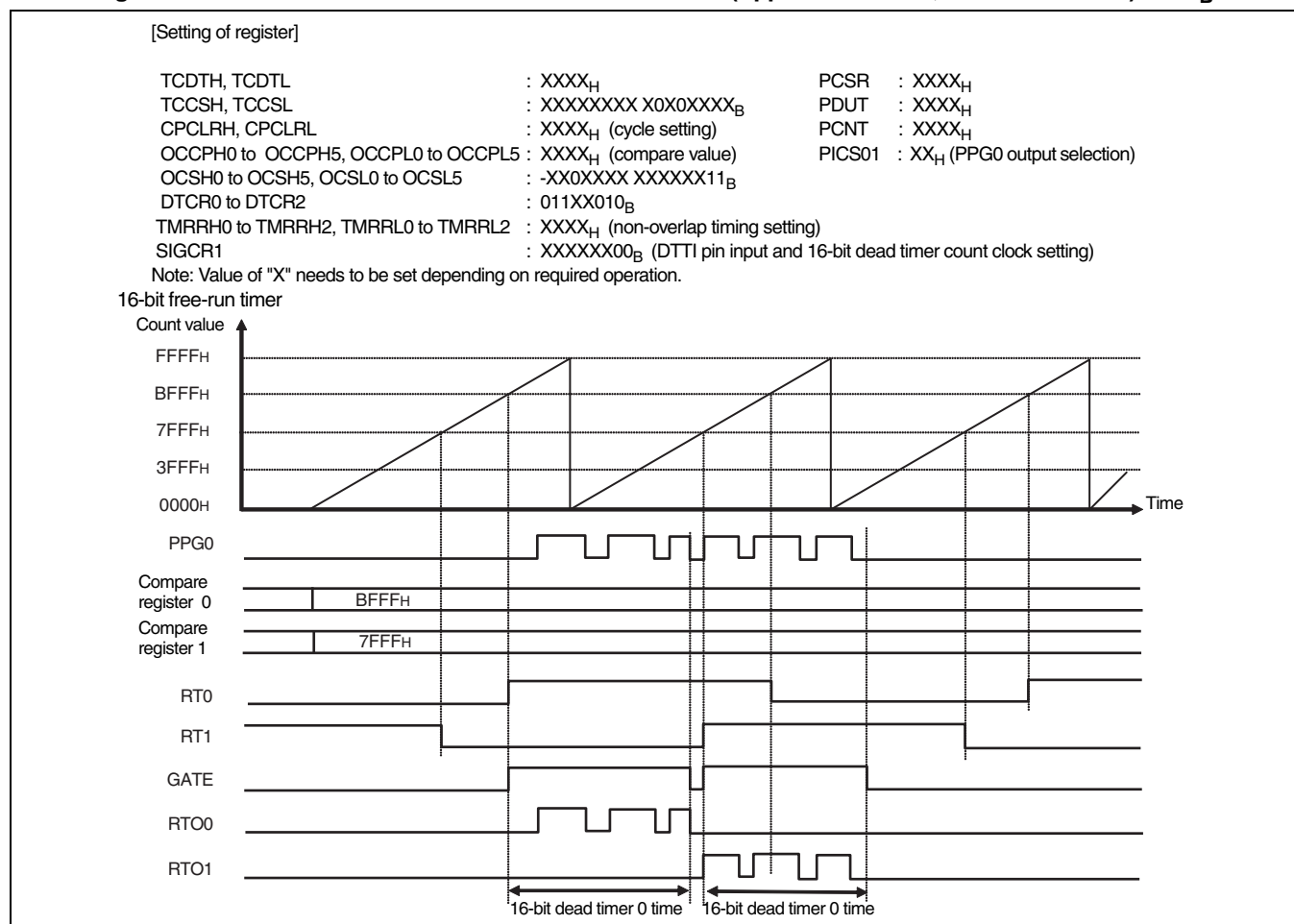
### 12.6.4.1 Operation of Timer Mode

When RTO0 to RTO5 pins rising edge are detected, the value is reloaded into the 16-bit dead timer, and the 16-bit dead timer starts counting down. PPG timer 0 continues to output to the RTO0 to RTO5 pins until an underflow occurs on the 16-bit dead timer.

#### ■ Operation of Timer Mode

Generation of PPG0 output pulse from a rising edge on RT until an underflow on the 16-bit dead timer (TMD8 to TMD0 (upper bit10 to bit8, lower bit2 to bit0) in the DTCR0, DTCR1, DTCR2 registers are "010<sub>B</sub>").

**Figure 12.6-13 Waveform Generated when TMD2 to TMD0 (Upper bit10 to bit8, Lower bit2 to bit0) = 010<sub>B</sub>**



Note:

Each 16-bit dead timer is used for two RTs. In other words, 16-bit dead timer 0 is used for RT0 and RT1; 16-bit dead timer 1 is used for RT2 and RT3; and 16-bit dead timer 2 is used for RT4 and RT5. Consequently, you must not try to use a RT to activate a PPG0 that is already operating. Attempting to do this will extend the GATE signal and therefore result in misoperation.

### 12.6.4.2 Operation during Dead Time Timer Mode

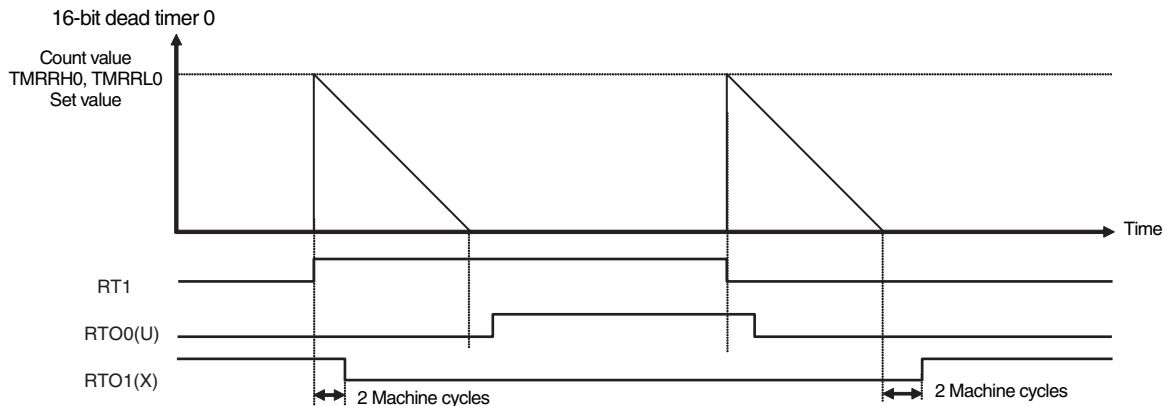
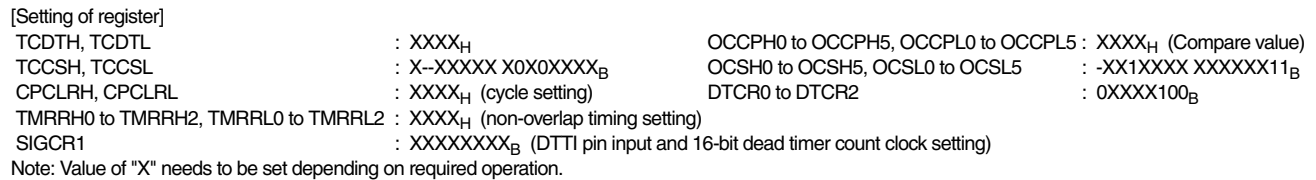
The dead-time generator inputs real-time output (RT1, RT3, and RT5) or PPG0 timer pulse output and outputs a non-overlapping signal (reverse signal) to the output pins (RTO0 to RTO5).

#### ■ Operation during Dead Time Timer Mode

- This non-overlapping signal is generated via normal-polarity RT1, RT3, and RT5 (16-bit dead timer control registers (DTCR0, DTCR1, and DTCR2) TMD8 to TMD0 (higher-order bits are 10 to 8; lower-order bits are 2 to 0) = 100<sub>B</sub>)

When the DTCR0, DTCR1, and DTCR2 registers DMOD2 to DMOD0 select the non-overlapping signal with a value of 0 (normal polarity), a delay equivalent to the non-overlap time set in the 16-bit dead timer registers (TMRRH0 to TMRRH2/TMRRL0 to TMRRL2) is applied. This delay is generated on the rising edge and falling edge of the RT1, RT3, RT5 pins. If the RT1, RT3, and RT5 pulse width is less than the specified non-overlap time, the 16-bit dead timer starts counting down again from TMRRH0 to TMRRH2/TMRRL0 to TMRRL2 register value of the next RT edges.

**Figure 12.6-14 Non-overlapping Signal Generation Using Normal Polarity RT1, RT3, and RT5**



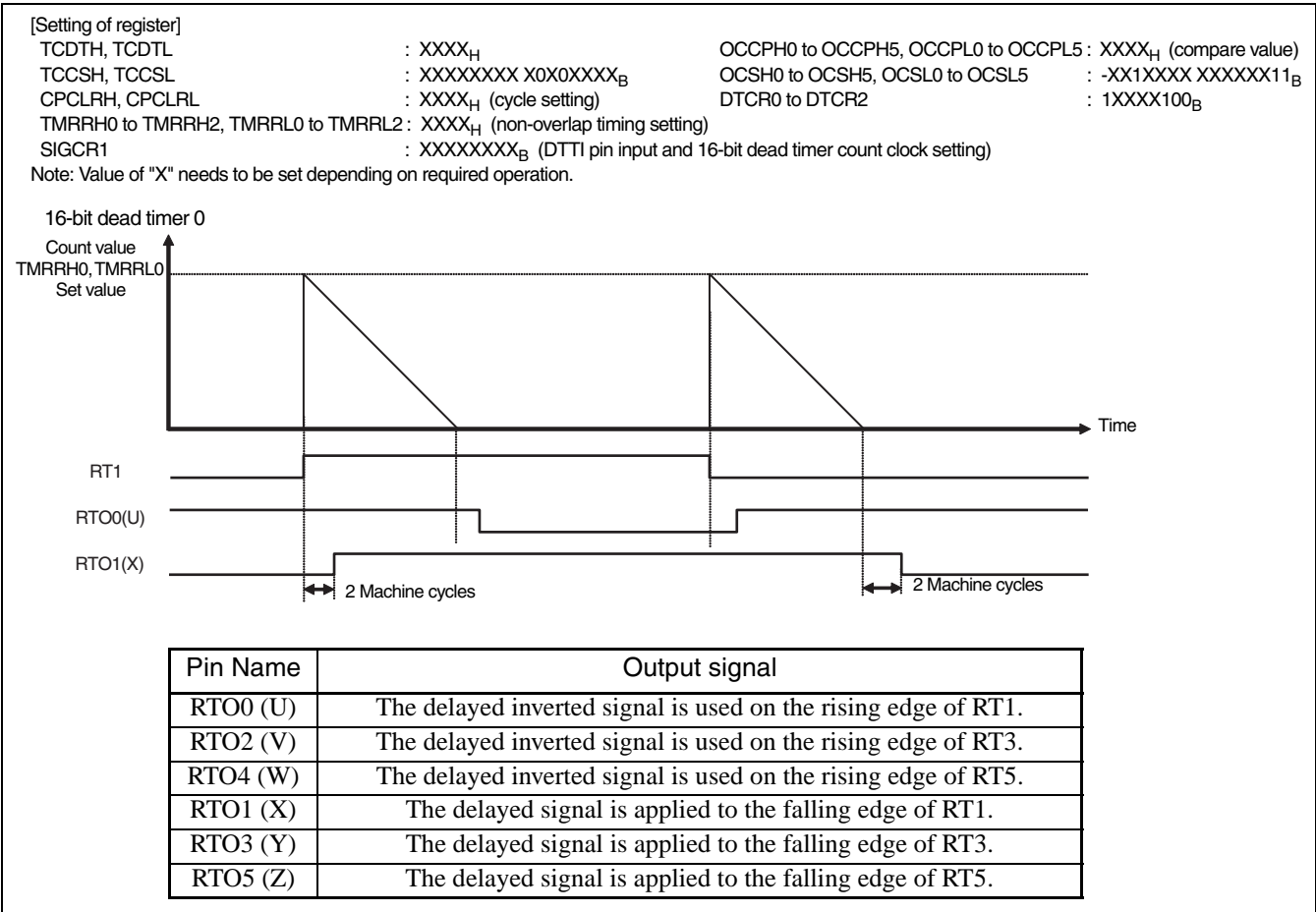
Pin Name	Output signal
RTO0 (U)	The delayed signal is used on the rising edge of RT1.
RTO2 (V)	The delayed signal is used on the rising edge of RT3.
RTO4 (W)	The delayed signal is used on the rising edge of RT5.
RTO1 (X)	The delayed reverse signal is applied to the falling edge of RT1.
RTO3 (Y)	The delayed reverse signal is applied to the falling edge of RT3.
RTO5 (Z)	The delayed reverse signal is applied to the falling edge of RT5.



- This non-overlapping signal is generated via reverse-polarity RT1, RT3, and RT5 (16-bit dead timer control registers (DTCR0, DTCR1, and DTCR2) TMD8 to TMD0 (higher-order bits are 10 to 8; lower-order bits are 2 to 0) = 100<sub>B</sub>)

When the DTCR0, DTCR1, and DTCR2 registers DMOD2 to DMOD0 (higher-order bit is 15, lower-order bit is 7) select the non-overlapping signal with a value of 1 (reverse polarity), a delay equivalent to the non-overlap time set in the 16-bit dead timer registers (TMRRH0 to TMRRH2/TMRRL0 to TMRRL2) is applied. This delay is generated on the rising edge and falling edge of the RT1, RT3, RT5. If the RT1, RT3, and RT5 pulse width is less than the specified non-overlap time, the 16-bit dead timer starts counting down again from TMRRH0 to TMRRH2/TMRRL0 to TMRRL2 value of the next RT edges.

Figure 12.6-15 Non-overlapping Signal Generation Using Normal Polarity RT1, RT3, and RT5



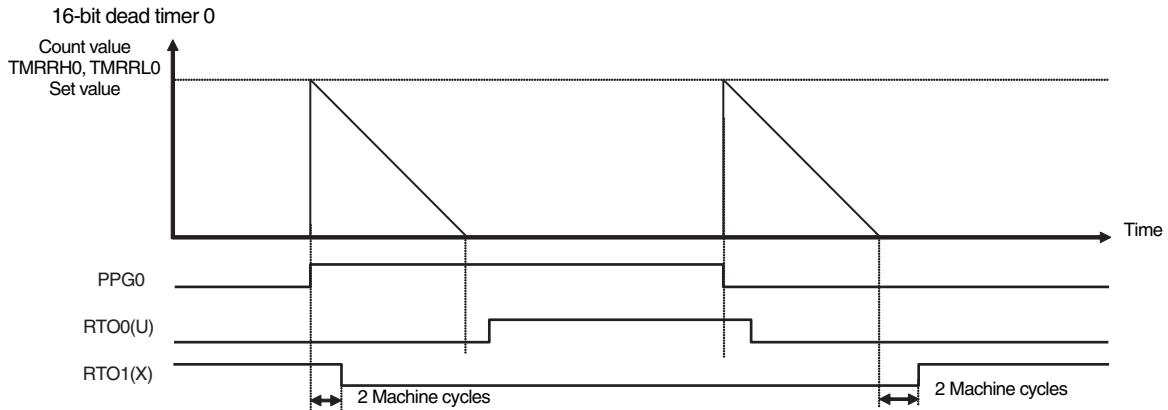
- This non-overlapping signal is generated via normal-polarity PPG (16-bit dead timer control registers (DTCR0, DTCR1, and DTCR2) TMD8 to TMD0 (higher-order bits are 10 to 8; lower-order bits are 2 to 0) = 111<sub>B</sub>)

When the DTCR0, DTCR1, and DTCR2 registers DMOD2 to DMOD0 (higher-order bit is 15; lower-order bit is 7) select the non-overlapping signal with a value of "0" (normal polarity), a delay equivalent to the non-overlap time set in the 16-bit dead timer registers (TMRRH0 to TMRRH2/TMRRL0 to TMRRL2) is applied. This delay is generated on the rising edge of the PPG0 timer pulse signal or inverted signal. If the PPG0 timer pulse width is less than the specified non-overlap time, the 16-bit dead timer starts counting down again from the TMRRH0 to TMRRH2/TMRRL0 to TMRRL2 value of the edge following that PPG0 pulse.

**Figure 12.6-16 Non-overlapping Signal Generation Using Normal Polarity PPG0 Timer**

[Setting of register]  
 TCDTH, TCDTL : XXXX<sub>H</sub> PCSR : XXXX<sub>H</sub>  
 TCCSH, TCCSL : XXXXXXXX X0X0XXXX<sub>B</sub> PDUT : XXXX<sub>H</sub>  
 CPCLRH, CPCLRL : XXXX<sub>H</sub> (cycle setting) PCNT : XXXX<sub>H</sub>  
 OCCPH0 to OCCPH5, OCCPL0 to OCCPL5 : XXXX<sub>H</sub> (compare value)  
 OCSH0 to OCSH5, OCSL0 to OCSL5 : -XX1XXXX XXXXXX11<sub>B</sub>  
 DTCR0 to DTCR2 : 0XXXX111<sub>B</sub>  
 TMRRH0 to TMRRH2, TMRRL0 to TMRRL2 : XXXX<sub>H</sub> (non-overlap timing setting)  
 SIGCR1 : XXXXXXXX<sub>B</sub> (DTT1 pin input and 16-bit dead timer count clock setting)

Note: Value of "X" needs to be set depending on required operation.



Pin Name	Output signal
RTO0 (U)	The delayed signal is used on the rising edge of PPG0.
RTO2 (V)	The delayed signal is used on the rising edge of PPG0.
RTO4 (W)	The delayed signal is used on the rising edge of PPG0.
RTO1 (X)	The delayed reverse signal is applied to the falling edge of PPG0.
RTO3 (Y)	The delayed reverse signal is applied to the falling edge of PPG0.
RTO5 (Z)	The delayed reverse signal is applied to the falling edge of PPG0.

- This non-overlapping signal is generated via reverse-polarity PPG (16-bit dead timer control registers (DTCR0, DTCR1, and DTCR2) TMD8 to TMD0 (higher-order bits are 10 to 8; lower-order bits are 2 to 0) =111<sub>B</sub>)

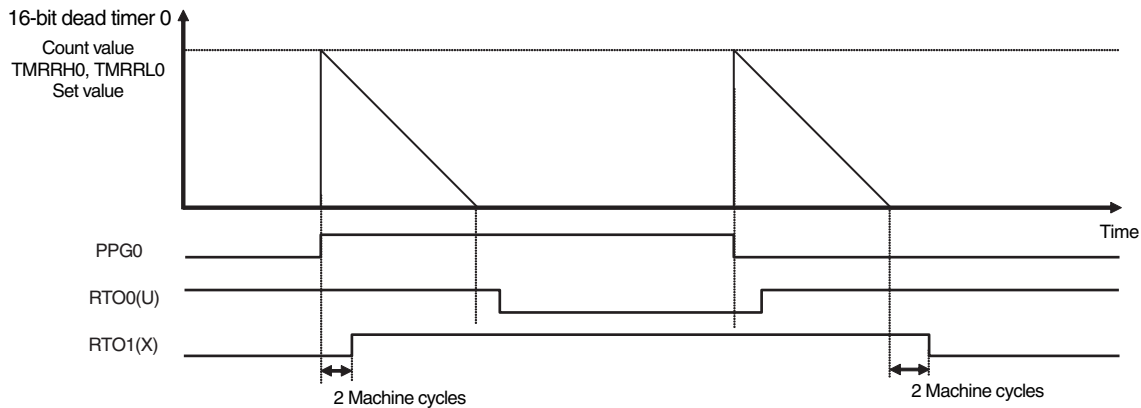
When the DTCR0, DTCR1, and DTCR2 registers DMOD2 to DMOD0 (higher-order bit is 15; lower-order bit is 7) select the non-overlapping signal with a value of "1" (reverse polarity), a delay equivalent to the non-overlap time set in the 16-bit dead timer registers (TMRRH0 to TMRRH2/TMRRL0 to TMRRL2) is applied. This delay is generated on the rising edge of the PPG0 timer pulse signal or inverted signal. If the PPG0 timer pulse width is less than the specified non-overlap time, the 16-bit dead timer starts counting down again from the TMRRH0 to TMRRH2/TMRRL0 to TMRRL2 value of the edge following that PPG0 pulse.

**Figure 12.6-17 Non-overlapping Signal Generation Using Inverted Polarity PPG0 Timer**

[Setting of register]

TCDTH, TCDTL	: XXXX <sub>H</sub>	PCSR	: XXXX <sub>H</sub>
TCCSH, TCCSL	: XXXXXXXX X0X0XXXX <sub>B</sub>	PDUT	: XXXX <sub>H</sub>
CPCLRH, CPCLRL	: XXXX <sub>H</sub> (cycle setting)	PCNT	: XXXX <sub>H</sub>
OCCPH0 to OCCPH5, OCCPL0 to OCCPL5	: XXXX <sub>H</sub> (compare value)		
OCSH0 to OCSH5, OCSL0 to OCSL5	: -XX1XXXX XXXXX11 <sub>B</sub>		
DTCR0 to DTCR2	: 1XXXX111 <sub>B</sub>		
TMRRH0 to TMRRH2, TMRRL0 to TMRRL2	: XXXX <sub>H</sub> (non-overlap timing setting)		
SIGCR1	: XXXXXXXX <sub>B</sub> (DTTI pin input and 16-bit dead timer count clock setting)		

Note: Value of "X" needs to be set depending on required operation.



Pin Name	Output signal
RTO0 (U)	The delayed inverted signal is used on the rising edge of PPG0.
RTO2 (V)	The delayed inverted signal is used on the rising edge of PPG0.
RTO4 (W)	The delayed inverted signal is used on the rising edge of PPG0.
RTO1 (X)	The delayed signal is applied to the falling edge of PPG0.
RTO3 (Y)	The delayed signal is applied to the falling edge of PPG0.
RTO5 (Z)	The delayed signal is applied to the falling edge of PPG0.

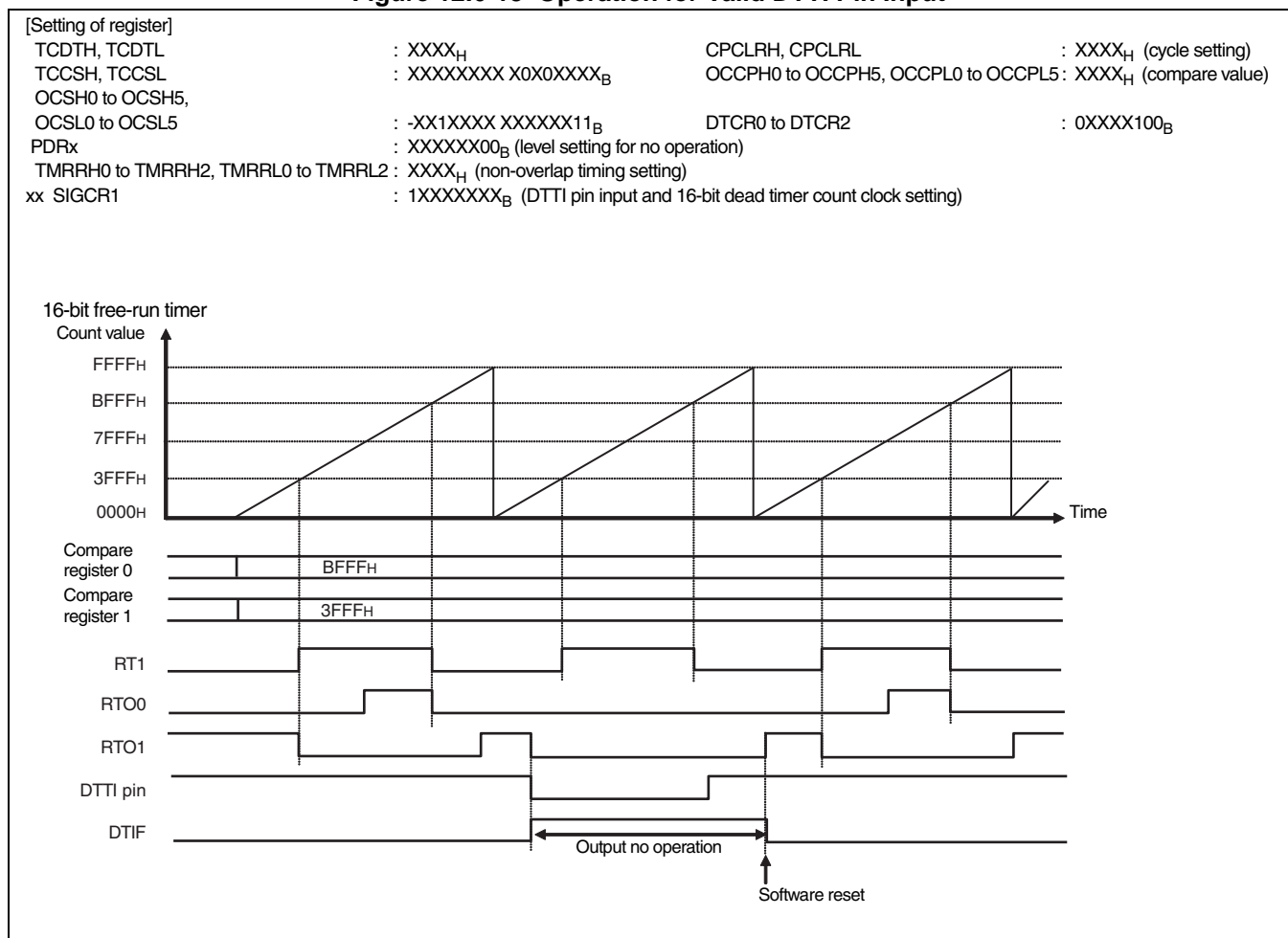
### 12.6.4.3 DTTI pin Control Operation

You can control RTO0 to RTO5 output by means of the DTTI pins by setting "1" in the waveform control register 1 (SIGCR1) DTIE: bit7. When a DTTI pin "L" level is detected, RTO0 to RTO5 output is locked at non-operating level until the interrupt flag (SIGCR1 register DTIF: bit6) is cleared. When RTO0 to RTO5 are at non-operating level, these pins can be set via software using the port data registers (PDR), which share them. Additionally, if they are used as input ports using the data direction register (DDR), Hi-Z is output.

#### ■ DTTI Pin Input Operation

Even when "L" is detected in DTTI pin input, although the timer continues to operate while the waveform generator is operational, waveforms are not output to the external RTO0 to RTO5 pins.

Figure 12.6-18 Operation for Valid DTTI Pin Input



## ■ DTTI Bit Operation of Waveform Control Register 2 (SIGCR2)

- Setting waveform control register 2 DTTI: bit 0 to "0" causes the outputs of RTO0 to RTO5 to go to the non-operating level until "1" is written to this bit.
- RTO0 to RTO5 can only be controlled by setting the DTTI bit to "0", regardless of the state of the DTIE bit.
- RTO0 to RTO5 go to the non-operating level immediately after the DTTI bit is set to "0". This is not affected by the noise cancellation feature.
- When the DTTI bit is set to "0", the DTTI pin interrupt is not generated and the interrupt flag (DTIF) is not set to "1".
- If an "L" level input is detected on the DTTI pin while the DTTI bit is set to "0", the DTTI pin interrupt flag is set and a DTTI pin interrupt request is generated.

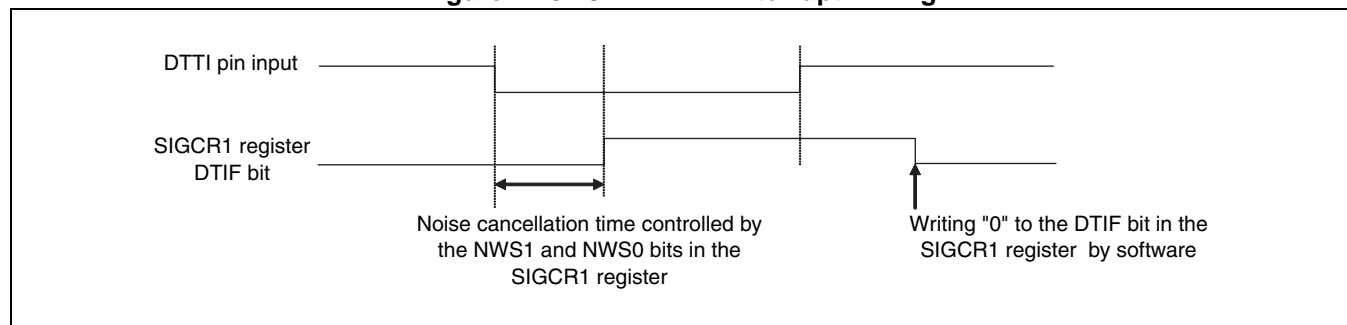
## ■ DTTI Pin Noise Cancellation Feature

The DTTI pin input noise cancellation feature is enabled when "1" is set in the waveform control register 1 (SIGCR1) NRSL: bit5. When the noise cancellation feature is enabled, there is a delay of 4, 8, 16, or 32 machine cycles (selected via the SIGCR1 register NWS1 and NWS0: bit1 and bit0), for an amount of time necessary to lock the output pins (RTO0 to RTO5) to non-operating level. Since the noise cancellation circuit uses resources, in modes where oscillation is stopped (e.g. stopped mode), input is disabled, even when DTTI pin input is enabled.

## ■ DTTI Pin Interrupt

When DTTI pin "L" level is detected, after the noise cancellation time has elapsed, the DTTI pin interrupt flag (SIGCR1 register DTIF: bit6) is set to "1", and an interrupt request is sent to the interrupt controller.

**Figure 12.6-19 DTTI Pin Interrupt Timing**



### Notes:

- If the SIGCR1 register NWS1 and NWS0 bits' values change within the noise cancellation time, a larger (NWS1 and NWS0) noise cycle value is enabled.
- The SIGCR1 register DTIF: bit6 can only be cleared via software.

## 12.6.5 A/D Activation Compare Operation

An A/D activation can be performed when the value of 16-bit free-run timer 0 reaches the specified value.

### ■ A/D Activation

Two A/D converter units can be activated.

- A/D activation compare 1: Start A/D unit 1.
- A/D activation compare 2: Start A/D unit 2.

### ■ A/D Compare Activation Enabled

When the compare register value is set, and "1" is set in the control registers (ADCOMPC1) CE1 and CE2: bit1 and bit2, when the free-run timer 0 and compare register value match, an A/D activation signal is generated for A/D.

When "0" is set in CE1 and CE2, even if the free-run timer 0 and compare register value match, an A/D activation signal is not generated for A/D.

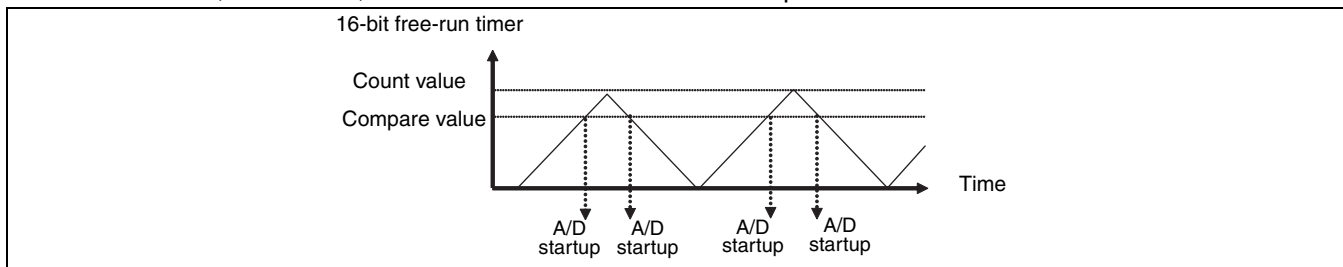
If a value is written to the compare register while activation is enabled, a comparison with the newly written value occurs immediately.

Generating an activation signal when a match occurs with free-run timers 1 and 2 is not available.

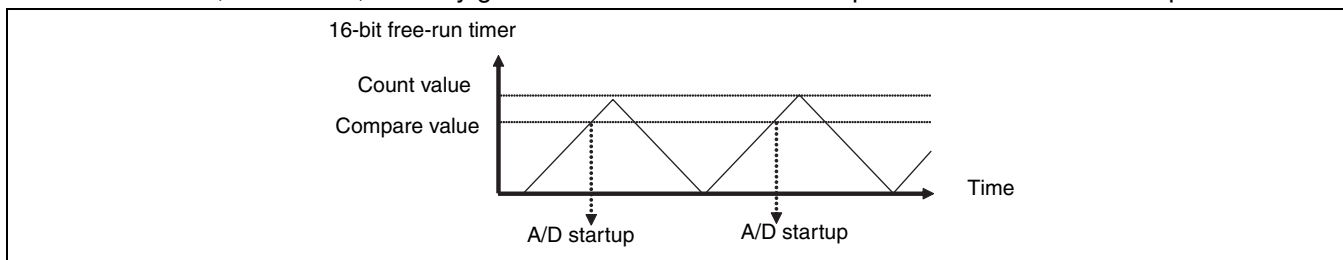
### ■ A/D Compare Activation Mode

The A/D activation mode is set in the SELn1 and SELn0 (n=1, 2) bits of the ADCOMPC2 register.

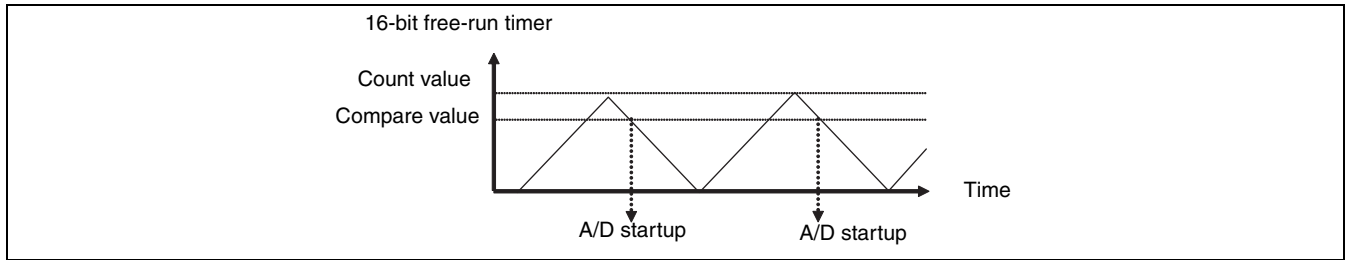
- SELn1, SELn0 = 0, 0: Generate activation when a compare match occurs



- SELn1, SELn0 = 0, 1: Only generate activation when a compare match occurs for an up-count



- SELn1, SELn0 = 1, 0: Only generate activation when a compare match occurs for a down-count



- SELn1,SELn0=1, 1: Setting prohibited

## 12.7 Notes on Using the Multifunction Timers

---

Heed the following cautions when using the multifunction timer.

---

### ■ Notes on Using the 16-bit Free-run Timers

#### ● Cautions for setting via the program

- When a reset is executed, although the timer value becomes "0000<sub>H</sub>", the 0-detect interrupt flag is not set.
- Since the timer-mode bit (TCCSL registers MODE: bit5) has a buffer, and so timer modes changed after 0-detect are enabled.
- A software clear (setting bit4 (SCLR) in the TCCSL register = 1) initializes the timer but does not generate a 0-detect interrupt.
- When the compare value and count value match, if the count starts, the compare-clear flag is not set.

#### ● Cautions for interrupt

- If "1" is set in the timer status control registers higher-order (TCCSH) IRQZF: bit14, then interrupt requests are enabled (TCCSH register's IRQZE: bit13 = 1), control cannot return from the interrupt processing. Always clear the IRQZF: bit14.
- If "1" is set in the timer status control registers higher-order (TCCSH) ICLR: bit9, then interrupt requests are enabled (TCCSH register's ICRE: bit8 = 1), control cannot return from the interrupt processing. Always clear the ICLR: bit9.

### ■ Notes on Using the 16-bit Output Compare

#### ● Cautions for interrupt

If "11<sub>B</sub>" is set in the compare control registers lower-order (OCSL0, OCSL2, and OCSL4) IOP1, IOP0 : bit7 and bit6, then interrupt requests are enabled (OCSL register's IOE1, IOE0 : bit6 and bit5 = 11<sub>B</sub>), control cannot return from the interrupt processing. Always clear the IOP0, IOP1 bits.

### ■ Cautions for Use of 16-bit Input Capture

#### ● Cautions for interrupt

- If "1" is set in the input capture state control registers lower-order (PICSL01 and ICSL23) ICP3, ICP2, ICP1, and ICP0 (both bit7 and bit6), then interrupt requests are enabled (PICSL01 and ICSL23 registers' ICE3, ICE2, ICE1, and ICE0 (both bit5 and bit4) = 11<sub>B</sub>), control cannot return from the interrupt processing. Be sure to clear ICP3, ICP2, ICP1, and ICP0 (both bit7 and bit6).
- When the input capture pin (IC) level changes the time between setting the bit for ICP3, ICP2, ICP1, and ICP0 and processing of the interrupt routine, the valid edge indication bits of the ICP3, ICP2, ICP1, and ICP0 (ICSH23 register's IEI3 and IEI2: bit9 and bit8 and PICSH01 register's IEI1 and IEI0: bit9, bit8) indicate the newly detected edge.



## ■ Notes on Using the Waveform Generator

### ● Cautions for setting via the program

- Confirm that the trigger source and 16-bit dead timer are not counting when the bit values of TMD8, TMD5, TMD2 (higher-order bit is 10; lower-order bit is 2), TMD7, TMD4, TMD1 (higher-order bit is 9; lower-order bit is 1), and TMD6, TMD3, TMD0 (higher-order bit is 8; lower-order bit is 0) in the 16-bit dead timer control registers (DTCR0, DTCR1, and DTCR2) are changed while the waveform generator is operating (DTCR0, DTCR1, and DTCR2 register's TMD2 to TMD0, TMD5 to TMD3, TMD8 to TMD6 are "001<sub>B</sub>", "010<sub>B</sub>", "100<sub>B</sub>" or "111<sub>B</sub>"). If this operation is not performed, an unintended waveform will be output from the RTO pin due to the output scheduled by the previous trigger. However, the RTO output returns to normal operation when a timer underflow occurs or when triggered again by the new trigger source.
- The trigger source is at "H" level for RT when TMD8 to TMD0 (higher-order bits are 10 to 8; lower-order bits are 2 to 0) in the DTCR0, DTCR1, and DTCR2 registers are "001<sub>B</sub>"; it is rising edge of RT when these bits are "010<sub>B</sub>"; it is rising or falling edge of RT when these bits are "100<sub>B</sub>"; it is rising or falling edge of PPG0 when these bits are "111<sub>B</sub>".

For example, if TMD bits 8 to 0 change from "100<sub>B</sub>" to "111<sub>B</sub>", the following steps can be executed.

1. Set the 16-bit dead timer register (TMRRH0 to TMRRH2/TMRRL0 to TMRRL2) to an extremely small value like 0001<sub>H</sub>.
  2. Set the RTO1, RTO3, or RTO5 output to "L" or "H" and wait for an underflow on timer 0, 1, or 2.
  3. Change the mode bits (TMD8 to TMD0) and corresponding settings.
  4. A corrected output waveform appears at the RTO pins after 1 machine cycle.
- If the value of the 16-bit dead timer register (TMRRH0 to TMRRH2/TMRRL0 to TMRRL2) is modified while the timer is counting, the new value is not used until the next timer trigger. When accessing the timer registers, be sure to use half-word or word transfer commands.
  - Only change the waveform control register 1's (SIGCR1) DCK2 to DCK0: bit4 to bit2 when the timers are not counting.
  - Only change the waveform control register 1's (SIGCR1) NWS1 and NWS0: bit1 and bit0 when the noise cancellation feature is disabled.

### ● Cautions for interrupt

- If "1" is set in the 16-bit dead timer control register (DTCR0, DTCR1, and DTCR2) TMIF2 to TMIF0 (higher-order bit is 12; lower-order bit is 4), then interrupt requests are enabled (DTCR0, DTCR1, and DTCR2 register's TMIE2 to TMIE0 (higher-order bit is 11; lower-order bit is 3) = 1), control cannot return from the interrupt processing. Always clear the TMIF bit.
- Control cannot return from interrupt processing after setting 1 in the waveform control register 1 (SIGCR1) DTIF:bit6. Always clear the DTIF bit.

## 12.8 Example Program for Multifunction Timer

The following is a sample multifunction timer program.

### ■ Example Program for the 16-bit Free-run Timer

#### ● Processing

- When the 16-bit free-run timer is 4 ms, generate a compare-clear interrupt.
- This timer is used to re-generate a compare-clear timer during up-count mode.
- 16 MHz is for the machine clock, and 62.5 ns is for the count clock.

#### ● Coding example

```

ICR32    .EQU      000460H    ; Interrupt control register for the 16-bit free-run
                                ; timer compare clear

TCCSH    .EQU      0000A8H    ; Timer control status register
CPCLRBH  .EQU      0000A4H    ; Compare-clear buffer register

; ----- Main Program -----
                ORG          C0000H

START:
;           :                      ; Assumes that the stack pointer (SP) has already been
                                ; initialized.

                ANDCCR      #0EFH    ; Disables the interrupt.
                LDI         #ICR32,r0

LDI         #00H,r1
STB         r1,@r0                ; Interrupt levels 16(strength)
LDI         #CPCLRBH,r0           ; Set value to the compare clear buffer register so that
LDI         #0FA00H,r1            ; compare clear interrupts will be generated at 4ms
                STH         r1,@r0    ; intervals in 16-bit free-run timer up-count mode
                LDI         #TCCSH,r3 ; Set up-count down mode,
                LDI         #0110H,r1 ; set 62.5ns count clock,
                STH         r1,@r3    ; enable compare clear interrupt,
                                ; clear compare clear interrupt flag bit,
                                ; disable interrupt mask,
                                ; clear timer, and enable operation

                STILM       #14H      ; Set the ILM in PS to level 20
                ORCCR       #10H      ; Interruption permission

LOOP        LDI         #00H,r0      ; Infinite loop
                LDI         #01H,r1
                BRA         LOOP      ;

; ----- Interrupt Program -----

```

```

WARI    LDI        #0100H,r1
        ANDH       r1,@r3    ; Clear interrupt request flag.
;
;
;      User processing
;
;      RETI          ; Returns from interrupt.

; ----- Vector Settings -----
VECT    .ORG FFFF8H
        .DATA.W    WARI    ; Set interrupt routine.
        .ORG       FFFF8H
        .DATA.W    0x07000000 ; Set single-chip mode.
        .DATA.W    START   ; Set reset vectors
        .END

```

## ■ Example Program for the 16-bit Output Compare

### ● Processing

- When count value of the 16-bit free-run timers matches the output compare value, an output compare match is generated.
- Use when the 16-bit free-run timer is in up/down count mode.
- 16 MHz is for the machine clock, and 62.5 ns is for the 16-bit free-run timer 0 count clock.

### ● Coding example

```

ICR44   EQU        00046CH    ; Output compare 0/1 interrupt register
TCCSH   EQU        0000A8H    ; Timer control status register
CPCLRBH EQU        0000A4H    ; Compare-clear buffer register
OCCPBH0 EQU        000090H    ; Output compare buffer register 0
OCCPBH1 EQU        000092H    ; Output compare buffer register 1
OCSH1   EQU        00009CH    ; Compare control register

; ----- Main Program -----
START:
;
;      ; Assumes that the stack pointer (SP) has already been
;      ; initialized.
        ANDCCR     #0EFH      ; Disables the interrupt.
        LDI        #ICR44,r0
        LDI        #00H,r1
        STB        r1,@r0     ; Interrupt levels 16 (strength)
        LDI        #CPCLRBH,r0 ; Set compare clear buffer register
        LDI        #0FFFFH,r1 ; for 16-bit free-run timer
        STH        r1,@r0

```

```

LDI      #OCCPBH0,r0 ; Set the output compare register 0.
LDI      #0BFFFH,r1
STH      r1,@r0
LDI      #OCCPBH1,r0 ; Set the output compare register 1.
LDI      #07FFFH,r1
STH      r1,@r0
LDI      #OCSH1,r3    ; Enable output compare output.
LDI      #6C33H,r2    ; Enable compare match interrupts 0/1.
STH      r2,@r3       ; Clear the interrupt flag bit.

LDI      #TCCSH,r0    ; Set up-count down mode,
LDI      #0010H,r1    ; clear timer, and enable operation
STH      r1,@r0
STILM    #14H         ; Set the ILM in PS to level 20
ORCCR    #10H         ; Interruption permission
LOOP     LDI      #00H,r0 ; Infinite loop
LDI      #01H,r1
BRA      LOOP         ;

; ----- Interrupt Program -----
WARI :
        ANDH     r2,@r3    ; Clear interrupt register flag.
        ;
        ;      User processing
        ;
        RETI          : Returns from interrupt.

; ----- Vector Settings -----
VECT     .ORG FFFF8H
        .DATA.W  WARI      ; Set interrupt routine.
        .ORG FFFF8H
        .DATA.W  0x07000000 ; Set single-chip mode.
        .DATA.W  START     ; Set reset vectors.
        .END

```



# **CHAPTER 13**

---

## ***U-TIMER***

### ***(16-BIT TIMER FOR UART BAUD RATE GENERATION)***

This chapter explains the overview of the U-TIMER, the configuration and functions of registers, and U-TIMER operation.

13.1 Overview

13.2 Operation Explanation

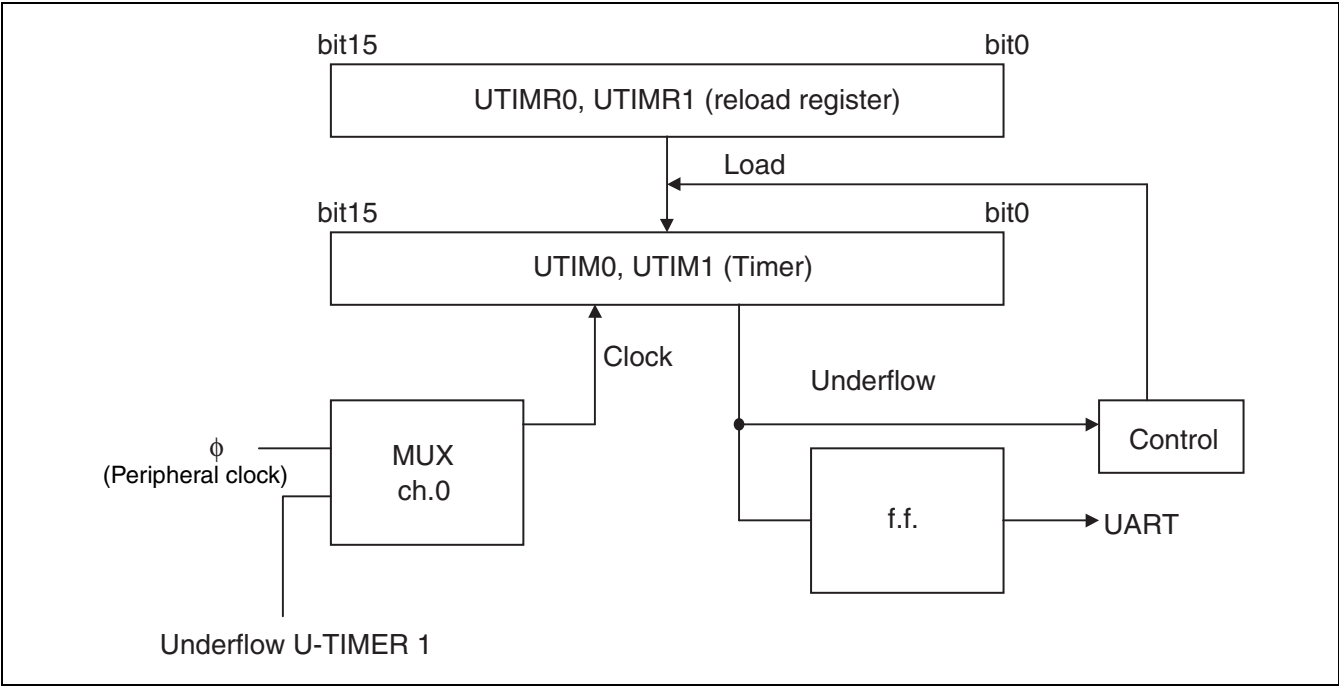
### 13.1 Overview

The U-TIMER is a 16-bit timer for generating baud rates for the UART. An arbitrary baud rate can be set with the combination of the operating frequency of the chip and the reload value of the U-TIMER. The U-TIMER can also be used as an interval timer as it generates an interrupt when count underflow occurs. MB91265A series contains two channels of this timer. When used as interval timers, two pairs of U-TIMERS can be cascaded to count up to  $2^{32} \times \phi$  intervals.

#### ■ Register List

bit15	bit8	bit7	bit0	
UTIM0, UTIM1				(R)
UTIMR0, UTIMR1				(W)
UTIMC0, UTIMC1				(R/W)

#### ■ Block Diagram of U-TIMER



## ■ Register Explanation

### ● U-TIMER (UTIM: UTIM0, UTIM1)

#### UTIM (Upper)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
ch.0: 000064 <sub>H</sub>	b15	b14	b13	b12	b11	b10	b9	b8	00000000 <sub>B</sub>
ch.1: 00006C <sub>H</sub>	R	R	R	R	R	R	R	R	

#### UTIM (Lower)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
ch.0: 000064 <sub>H</sub>	b7	b6	b5	b4	b3	b2	b1	b0	00000000 <sub>B</sub>
ch.1: 00006C <sub>H</sub>	R	R	R	R	R	R	R	R	

R: Read only

The UTIM register indicates a timer value. Use a 16-bit transfer instruction to access this register.

### ● Reload register (UTIMR: UTIMR0, UTIMR1)

#### UTIMR (Upper)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
ch.0: 000064 <sub>H</sub>	b15	b14	b13	b12	b11	b10	b9	b8	00000000 <sub>B</sub>
ch.1: 00006C <sub>H</sub>	W	W	W	W	W	W	W	W	

#### UTIMR (Lower)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
ch.0: 000064 <sub>H</sub>	b7	b6	b5	b4	b3	b2	b1	b0	00000000 <sub>B</sub>
ch.1: 00006C <sub>H</sub>	W	W	W	W	W	W	W	W	

W: Write only

The UTIMR register contains the value to be reloaded by the UTIM register when it causes underflow.

Be sure to use a 16-bit transfer instruction to access this register.

### ● U-TIMER Control register (UTIMC: UTIMC0, UTIMC1)

#### UTIMC

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
ch.0: 000067 <sub>H</sub>	UCC1	–	–	UTIE	UNDR	CLKS	UTST	UTCR	0--00000 <sub>B</sub>
ch.1: 00006F <sub>H</sub>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable

–: Undefined bit

The UTIMC register controls the operation of the U-TIMER.

Be sure to use a byte transfer instruction to access this register.



**[bit7] UCC1 (U-timer Count Control 1)**

The UCC1 bit controls how the U-TIMER counts.

UCC1	Operation
0	Normal operating. $\alpha = 2n+2$ [Initial value]
1	+1 mode $\alpha = 2n+3$

n... Setting value of UTIMR

$\alpha$ : Output clock cycles for UART

The U-TIMER can set either the normal number of clock cycles of  $2(n + 1)$  or the one divided by an odd number for the UART.

Setting the UCC1 bit to 1 generates  $2n + 3$  clock cycles.

Setting examples: 1. UTIMR = 5, UCC1 = 0 : generation cycle =  $2n + 2 = 12$  cycles

2. UTIMR = 25, UCC1 = 1 : generation cycle =  $2n + 3 = 53$  cycles

3. UTIMR = 60, UCC1 = 0 : generation cycle =  $2n + 2 = 122$  cycles

To use the U-TIMER as an interval timer, set the UCC1 bit to "0".

**[bit6, bit5] (Undefined)****[bit4] UTIE (U-Timer Interrupt Enable)**

The UTIE bit enables an interrupt when U-TIMER underflow occurs.

0: Interrupt disabled [Initial value]

1: Interrupt enabled

**[bit3] UNDR (UNDeR flow flag)**

The UNDR bit serves as a flag indicating that underflow has occurred. An underflow interrupt occurs when the UNDR bit is set with the UTIE bit containing "1". The UNDR bit is cleared either at a reset or when "0" is written.

The UNDR bit returns "1" whenever read by a read modify write instruction.

Note that an attempt to write "1" to the UNDR bit is ignored.

**[bit2] CLKS (CLock Select)**

The CLKS bit specifies the cascading of U-TIMER ch.0 and ch.1.

0: Use the peripheral clock ( $\phi$ ) as the clock source. [initial value]

1: Use the underflow signal (f.f in the block diagram) of ch.0 as the U-TIMER source clock timing signal.

The CLKS bit is enabled only for ch.1. For ch.0, leave it always set to "0".

Note:  $\phi$  (peripheral clock = CLKP) varies in cycle depending on the gear setting.

**[bit1] UTST (U-Timer SStart)**

This bit enables the operation of the U-TIMER.

0: Stops

Writing "0" stops the U-TIMER even during operation. [Initial value]

1: Operation

The U-TIMER continues to operate even when "1" is written during operation.

**[bit0] UTCR (U-Timer Clear)**

Writing "0" to the UTCR bit clears the U-TIMER to "0000<sub>H</sub>" (also clears f.f. to "0").

The bit returns "1" whenever read.

---

**Notes:**

- The U-TIMER reloads the initial value automatically when the UTST start bit is asserted (to start the U-TIMER) in the stop state.
  - When the UTCR clear bit and UTST start bit are asserted at the same time in the stop state, the counter is cleared to "0" and then underflow occurs when it is decremented first after being cleared.
  - Asserting the UTCR clear bit during operation clears the counter to "0" as well. The output waveform may therefore include short pulse of a glitch which can cause the UART or the high-order one of the cascaded U-TIMERS to malfunction. When the output clock is being used, do not use the clear bit to clear during operation.
  - Setting the low-order UTIMR (reload register) in cascade mode to "0" or "1" prevents normal counting.
  - If you assert bit1 (U-TIMER start bit, UTST) and bit0 (U-TIMER clear bit, UTCR) of the U-TIMER control register at the same time with the timer stopped, bit3 (underflow flag, UNDR) of the same register is set at the post-clear counter load timing. In addition, the internal baud rate clock signal goes high at the same timing.
  - If the interrupt request set timing and clear timing coincide with each other, the flag set operation overrides the clear operation.
  - To leave ch.0 unused in cascade mode or to use this module as a simple timer, always write "0" to bit2 (reference clock selection bit, CLKS) of the U-TIMER control register. The "CLKS" setting must be changed with the module inactive.
  - If the U-TIMER reload register write timing and reload timing coincide with each other, the counter loads old data. The counter loads new data at the next reload timing.
  - If the timer clear timing and timer count/reload timing coincide with each other, the timer clear operation overrides the other.
  - To use clock synchronous mode, set the UTIMR reload value to "3" or more.
-

## 13.2 Operation Explanation

This section explains the calculation of the baud rate and the cascade mode for the UART.

### ■ Calculation of Baud Rate

Each UART uses the underflow flip-flop (f.f. in the block diagram) of the corresponding U-TIMER (U-TIMER0 for UART0 or U-TIMER1 for UART1) as the baud rate clock source.

#### ● Asynchronous (start-stop synchronization) mode

The UART uses the U-TIMER's output divided by 16.

$$\text{bps} = \frac{\phi}{(2n+2) \times 16} \dots\dots\dots \text{UCC1}=0$$

$$\text{bps} = \frac{\phi}{(2n+3) \times 16} \dots\dots\dots \text{UCC1}=1$$

n: UTIMR (Reload Value)  
ϕ: Peripheral machine clock frequency (depending on the gear setting)

Max bps: 1031250bps at 33MHz clock

#### ● Clock synchronous mode

$$\text{bps} = \frac{\phi}{(2n+2)} \dots\dots\dots \text{UCC1}=0$$

$$\text{bps} = \frac{\phi}{(2n+3)} \dots\dots\dots \text{UCC1}=1$$

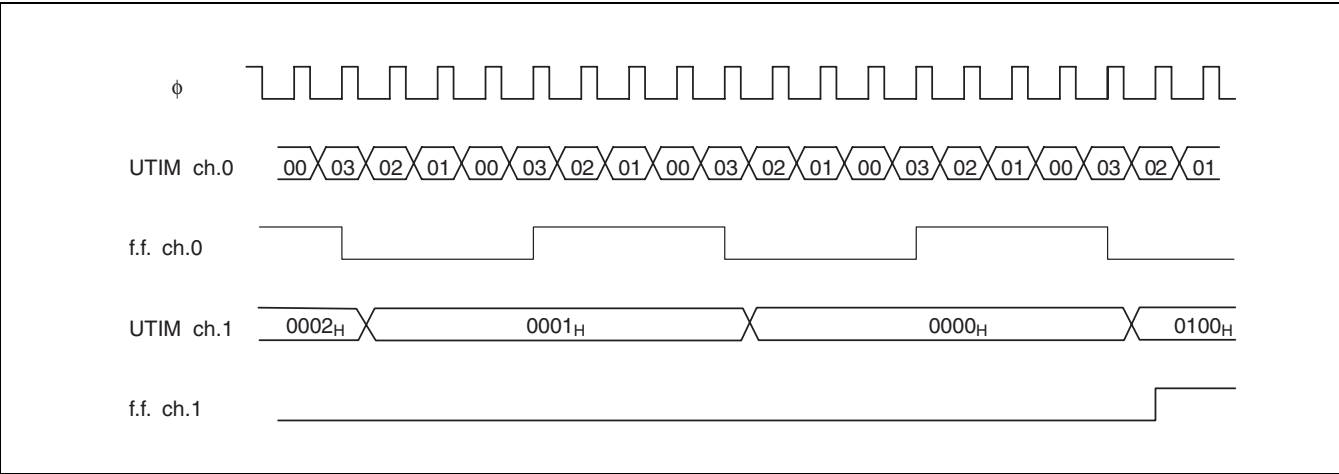
n: UTIMR (reload value)  
Note: n must be "3" or more.  
ϕ: Peripheral machine clock frequency (depending on the gear setting)

Max bps: 4125000bps at 33MHz clock

### ■ Cascade Mode

U-TIMER ch.0 and ch.1 can be used in cascade mode.

Example: Set UTIMR ch.0 to "0003<sub>H</sub>" and UTIMR ch.1 to "0100<sub>H</sub>".



# **CHAPTER 14**

---

## **UART**

**This chapter explains the overview of the UART, the configuration and functions of registers, and UART operation.**

- 14.1 Overview
- 14.2 Register Details Explanation
- 14.3 Operation of UART
- 14.4 Application Example
- 14.5 Examples of Baud Rate and U-TIMER Reload Value Settings

## 14.1 Overview

---

**The UART serves as a serial I/O port for asynchronous (start-stop synchronization) communication or clock synchronous communication. MB91265A series contains two channels of UART.**

---

### ■ Features of UART

- With full-duplex double buffer
- Capable of asynchronous (start-stop synchronization) or clock synchronous communication
- Support for multiprocessor mode
- Completely programmable baud rates. Capable of setting an arbitrary baud rate using an internal timer (See "CHAPTER 13 U-TIMER (16-BIT TIMER FOR UART BAUD RATE GENERATION)")
- Capable of setting a baud rate freely using an external clock
- Error detection function (parity, framing, overrun)
- Transfer signal is NRZ sign.
- Capable of starting DMA transfer using an interrupt

## ■ Register List

bit15	bit8	bit7	bit0
SCR		SMR	
SSR		SIDR(R)/SODR(W)	
8 bits		8 bits	

(R/W, W)

(R/W, R, W)

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
D7	D6	D5	D4	D3	D2	D1	D0

Serial Input Data Register  
Serial Output Data Register  
(SIDR/SODR)

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
PE	ORE	FRE	RDRF	TDRE	BDS	RIE	TIE

Serial Status Register  
(SSR)

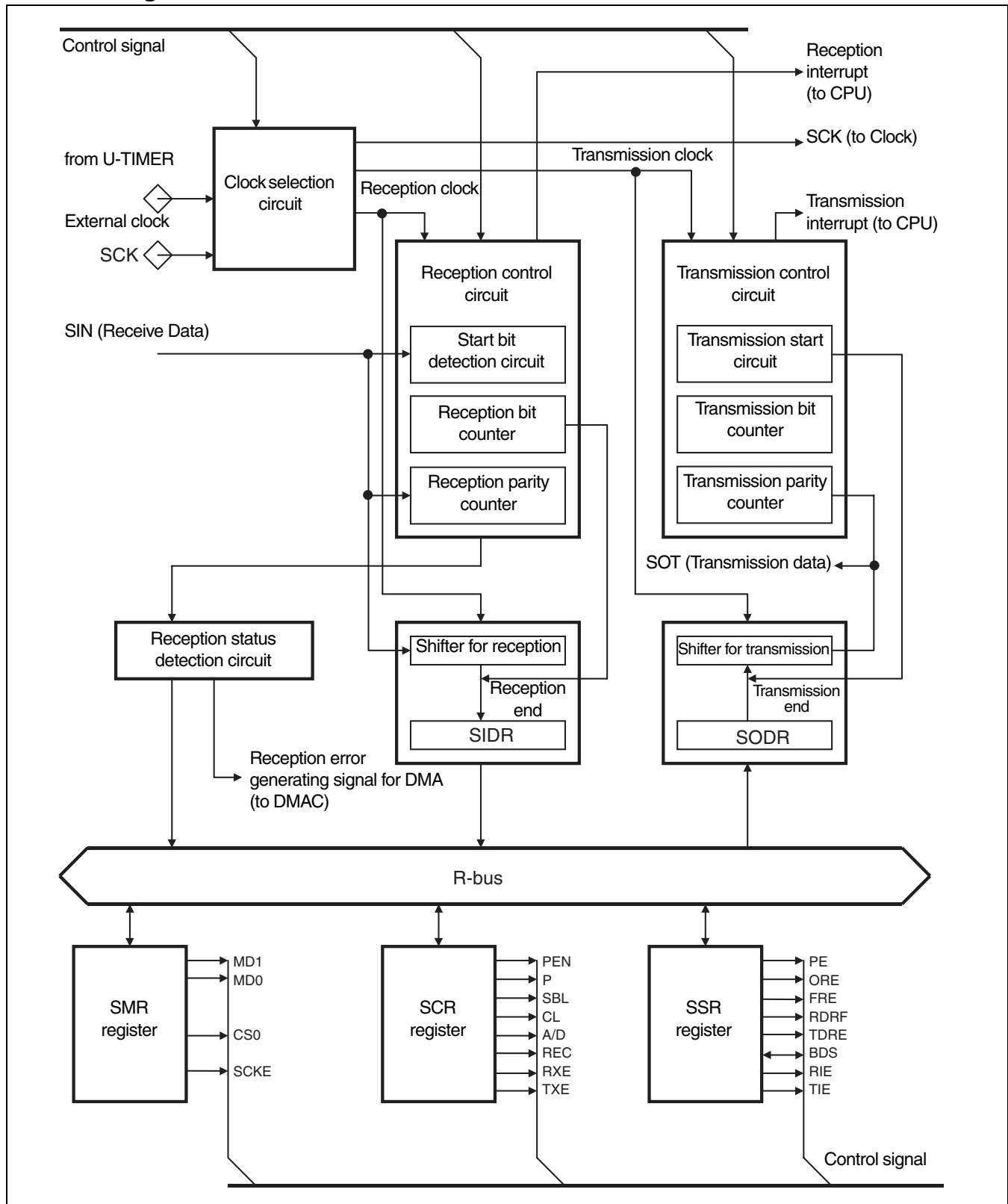
bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
MD1	MD0	-	-	CS0	-	SCKE	-

Serial Mode Register  
(SMR)

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
PEN	P	SBL	CL	A/D	REC	RXE	TXE

Serial Control Register  
(SCR)

## ■ Block Diagram of UART



## 14.2 Register Details Explanation

This section explains the registers used by UART in detail.

### ■ Serial Mode Register (SMR: SMR0, SMR1)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
ch.0:000063 <sub>H</sub>	MD1	MD0	–	–	CS0	–	SCKE	–	00--0-0 <sub>B</sub>
ch.1:00006B <sub>H</sub>	R/W	R/W	R/W	R/W	W		R/W		

R/W: Readable/Writable  
W: Write only

The SMR register specifies the UART operation mode. Set the operation mode while the UART is inactive. Do not write to this register with the UART operating.

[bit7, bit6] MD1,MD0 (MoDe select)

The MD1 and MD0 bits select the UART operation mode.

**Table 14.2-1 Selection of Operation Mode**

Mode	MD1	MD0	Operating mode
0	0	0	Asynchronous (start-stop synchronization): Normal mode [Initial value]
1	0	1	Asynchronous (start-stop synchronization): Multiprocessor mode
2	1	0	Clock synchronous mode
-	1	1	Setting disabled

Note:

Mode 1 is clock asynchronous mode (multiprocessor mode) in which two or more slave CPUs are connected to one host CPU. This resource cannot identify the data format of data received. In multiprocessor mode, therefore, only the master is supported.

Set the PEN bit of the SCR register to "0" because parity checking cannot be used.

[bit5, bit4]

Always write "1".

[bit3] CS0(Clock Select)

The operation clock of UART is selected.

0: Internal timer (U-TIMER) [Initial value]

1: External clock



[bit2]

Always write "0".

[bit1] SCKE (SCLK Enable)

The SCKE bit selects whether the SCK pin is used as the clock input pin or clock output pin for communication in clock synchronous mode (mode 2).

For clock asynchronous mode or external clock mode, set this bit to "0".

0: Functions as clock input pin [Initial value].

1: Functions as clock output pin.

---

Note:

For use as the clock input pin, set the CS0 bit to "1" to select the external clock.

---

[bit0] –: Undefined bit

It is an unused bit.

## ■ Serial Control Register (SCR: SCR0, SCR1)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
ch.0:000062 <sub>H</sub>	PEN	P	SBL	CL	A/D	REC	RXE	TXE	00000100 <sub>B</sub>
ch.1:00006A <sub>H</sub>	R/W	R/W	R/W	R/W	R/W	W	R/W	R/W	
R/W: Readable/Writable									
W: Write only									

The SCR register controls the transfer protocol for serial communication.

### [bit7] PEN (Parity Enable)

The PEN bit selects whether to perform data communication with parity added in serial communication.

0: No parity [Initial value]

1: With parity

### Note:

Parity can be added only in asynchronous (start-stop synchronization), normal communication mode (mode 0). Parity cannot be added in multiprocessor mode (mode 1) or clock synchronous mode (mode 2).

### [bit6] P (Parity)

The P bit selects even or odd parity be added for data communication.

0: Even parity [Initial value]

1: Odd parity

### [bit5] SBL (Stop Bit Length)

The SBL bit specifies the bit length of the stop bit(s) to be used as a frame end mark in asynchronous (start-stop synchronization) communication.

0: 1 stop bit [Initial value]

1: 2 stop bits

### [bit4] CL (Character Length)

The CL bit specifies the data length of each frame to be transmitted/received.

0: 7-bit data [Initial value]

1: 8-bit data

### Note:

7-bit data can be handled only in asynchronous (start-stop synchronization), normal communication mode (mode 0). For multiprocessor mode (mode 1) or clock synchronous mode (mode 2), select the 8-bit data length.

### [bit3] A/D (Address/Data)

Specify the data format of frames to be transmitted/received in asynchronous (start-stop synchronization), multiprocessor communication mode (mode 1).

0: Data frame [Initial value]

1: Address frame

### [bit2] REC (Receiver Error Clear)

Writing "0" to the REC bit clears the error flags (PE, ORE, FRE) in the SSR register.

An attempt to write "1" to this bit is ignored. The bit returns "1" whenever read.

### [bit1] RXE (Receiver Enable)

The RXE bit controls the reception by the UART.

0: Disables reception [Initial value].

1: Enables reception.

---

#### Note:

If the UART is reception-disabled during reception (with the receive shift register containing data), the UART stops reception immediately after storing received data in the receive data buffer SDR register upon completion of reception of the current frame.

---

### [bit0] TXE (Transmitter Enable)

The TXE bit controls the transmission by the UART.

0: Disables transmission [Initial value].

1: Enables transmission.

---

#### Note:

If the UART is transmission-disabled during transmission (with data being output from the transmit register), the UART stops transmission the moment the transmit data buffer SODR register runs out of data.

---

## ■ Serial Input Data Register (SIDR: SIDR0, SIDR1) / Serial Output Data Register (SODR: SODR0, SODR1)

### SIDR

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
ch.0:000061 <sub>H</sub>	D7	D6	D5	D4	D3	D2	D1	D0	XXXXXXXX <sub>B</sub>
ch.1:000069 <sub>H</sub>	R	R	R	R	R	R	R	R	

### SODR

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
ch.0:000061 <sub>H</sub>	D7	D6	D5	D4	D3	D2	D1	D0	XXXXXXXX <sub>B</sub>
ch.1:000069 <sub>H</sub>	W	W	W	W	W	W	W	W	

R: Read only  
W: Write only  
X: Undefined

The SIDR/SODR register is a data buffer registers for reception/transmission.

When the data length is 7 bits, the data in bit7 (D7) of the SIDR/SODR register is invalid. When the SIDR or SODR register is accessed with the BDS bit set to 1, the upper and lower portions of data on the bus are replaced with each other and accordingly bit0 (D0) apparently seems to be ignored.

Write to the SODR register when the TDRE flag in the SSR register contains "1".

### Note:

Writing to this address means writing to the SODR register; reading it means reading the SIDR register.

## ■ Serial Status Register (SSR: SSR0, SSR1)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
ch.0:000060 <sub>H</sub>	PE	ORE	FRE	RDRF	TDRE	BDS	R1E	T1E	00001000 <sub>B</sub>
ch.1:000068 <sub>H</sub>	R	R	R	R	R	R/W	R/W	R/W	

R/W: Readable/Writable  
R: Read only

The SSR register comprises flags which indicate the operation states of the UART.

### [bit7] PE (Parity Error)

This bit serves as the interrupt request flag that is set when a parity error occurs during reception.

To clear the flag once set, write "0" to the REC bit (bit10) of the SCR register.

When this bit is set, the data in the SIDR register becomes invalid.

0: No parity error [Initial value]

1: Parity error occurring

### [bit6] ORE (Over Run Error)

This bit serves as the interrupt request flag that is set when an overrun error occurs during reception.

To clear the flag once set, write "0" to the REC bit (bit10) of the SCR register.

When this bit is set, the data in the SIDR register becomes invalid.

0: No overrun error [Initial value]

1: Overrun error occurring

### [bit5] FRE (FRaming Error)

This bit serves as the interrupt request flag that is set when a framing error occurs during reception.

To clear the flag once set, write "0" to the REC bit (bit10) of the SCR register.

When this bit is set, the data in the SIDR register becomes invalid.

0: No framing error [Initial value]

1: Framing error occurring

---

### Notes:

- Switching between the internal and external baud rate clocks using bit3 of the serial mode register takes effect immediately after the bit is written to. Therefore, write to the bit with the UART inactive.
  - Bit3 of the serial mode register is write-only.
- 

### [bit4] RDRF (Receiver Data Register Full)

This bit serves as the interrupt request flag indicating that receive data exists in the SIDR register.

The flag is set when receive data is loaded into the SIDR register; it is cleared automatically when the data is read from the SIDR register.

0: No data received [Initial value]

1: Data received

**[bit3] TDRE (Transmitter Data Register Empty)**

This bit serves as the interrupt request flag indicating that transmit data can be written to the SODR register.

The flag is cleared when transmit data is written to the SODR register. The flag is set again when the transfer of written data loaded to the transmission shifter starts, indicating that the next piece of transmit data can be written.

0: Disable writing transmit data.

1: Enable writing transmit data. [Initial value]

**[bit2] BDS (Bit Direction Select)**

This bit is a transfer direction selection bit.

0: Transfer is carried out from the lowest bit (LSB) side [Initial value].

1: Transfer is carried out from the highest bit (MSB) side.

**Note:**

When the serial data register is read/write-accessed, the upper and lower portions of data are replaced with each other. If this bit is updated after data is written to the SODR register, therefore, the data becomes invalid.

When the SODR register and BDS bit are updated at the same time in half-words (16 bits), the SODR register is written to according to the pre-update BDS value.

**[bit1] RIE (Receiver Interrupt Enable)**

This bit controls reception interrupts.

0: Disable interrupts [Initial value].

1: Enable interrupts.

**Note:**

Reception interrupt trigger events are the occurrence of an error detected with the PE/ORE/FRE flag set and normal reception with the RDRF flag set as well.

**[bit0] TIE (Transmitter Interrupt Enable)**

This bit controls transmission interrupts.

0: Disable interrupts [Initial value].

1: Enable interrupts.

**Note:**

The transmission interrupt trigger event is a transmit request issued with the TDRE flag set.

## 14.3 Operation of UART

This section describes the operation of each operation mode.

The UART has two operating modes: asynchronous (start-stop synchronization) mode and clock synchronous mode. Asynchronous mode consists of normal mode and multiprocessor mode.

### ■ Operating Mode

The UART has the operation modes listed in Table 14.3-1, from which one can be selected by setting the values in the SMR and SCR registers.

**Table 14.3-1 Operation Modes**

Mode	Parity	Data length	Operating mode	Length of stop bit
0	Provided/Not provided	7 bits	Asynchronous (start-stop synchronization): Normal mode	1 bit or 2 bits
	Provided/Not provided	8 bits		
1	Not provided	8 bits + 1 bit	Asynchronous (start-stop synchronization): Multiprocessor mode	
2	Not provided	8 bits	Clock synchronous mode	Not provided

Note, however, that the stop bit length in asynchronous (start-stop synchronization) mode can be specified only for transmission. For reception, the stop bit length is always 1 bit. Do not set any mode other than those listed above as the UART does not work.

### ■ Clock Selection of UART

#### ● Internal timer

When the U-TIMER is selected with the CS0 bit set to "0", the baud rate is determined by the reload value set for the U-TIMER. The baud rate is calculated as follows:

Asynchronous (start-stop synchronization)  $\phi / (16 \times \beta)$

Clock synchronous  $\phi / \beta$

$\phi$ : Peripheral machine clock frequency

$\beta$ : Number of clock cycles set with U-TIMER ( $2n + 2$  or  $2n + 3$ , where  $n$  is reload value ( $n \geq 3$ ))

The baud rate in asynchronous (start-stop synchronization) mode allows transfer within the range from - 1% to + 1% of the set baud rate.

#### ● External clock

When the external clock is selected with the CS0 bit set to "1", the baud rate is determined as follows, assuming  $f$  as the external clock frequency:

Asynchronous (start-stop synchronization)  $f / 16$

Clock synchronous  $f$

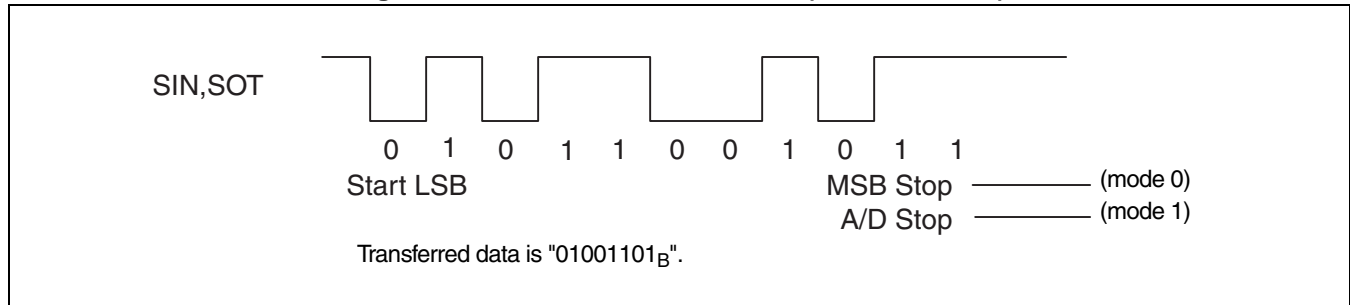
Note that  $f$  can be up to 3.125 MHz.

## ■ Asynchronous (start-stop Synchronization) Mode

### ● Transfer data format

The UART handles only data in the NRZ (Non Return to Zero) format. Figure 14.3-1 shows the data format.

**Figure 14.3-1 Transfer Data Format (Mode0, Mode1)**



As shown in Figure 14.3-1, data transfer always begins with a start bit ("L" level data), followed by the specified bit length of data in the LSB-first mode, and ends with a stop bit ("H" level data). When the external clock has been selected, keep the clock signal always input.

While the data length can be set to 7 or 8 bits in normal mode (mode 0), it must be 8 bits in multiprocessor mode (mode 1). In multiprocessor mode, in addition, no parity bit can be added. The A/D bit is however always added instead.

### ● Reception operation

The UART is receiving data whenever the RXE bit (bit1) of the SCR register is "1".

When a start bit appears on the receiving line, one frame of data is received in the data format specified in the SCR register. If an error occurs after the one-frame reception, the relevant error flag is set and the RDRF flag (bit4 of the SSR register) is set. If the RIE bit (bit1) of the same SSR register contains "1" at this time, a receive interrupt occurs to the CPU. Check the individual flags in the SSR register and either read the SISR register when reception is normal or perform necessary processing if an error has occurred.

The RDRF flag is cleared when the SISR register is read.

### ● Transmission operation

When the TDRE flag (bit3) of the SSR register contains "1", transmit data is written to the SODR register. The data is transmitted if the TXE bit (bit0) of the SCR register is "1" at this time.

When the data set in the SODR register is loaded into the transmission shift register and its transmission is started, the TDRE flag is set again, so that the next piece of transmit data can be set. If the TIE bit (bit0) of the same SSR register contains "1" at this time, transmit interrupt occurs to the CPU, requesting it to set transmit data in the SODR register.

The TDRE flag is cleared as soon as data is set in the SODR register.

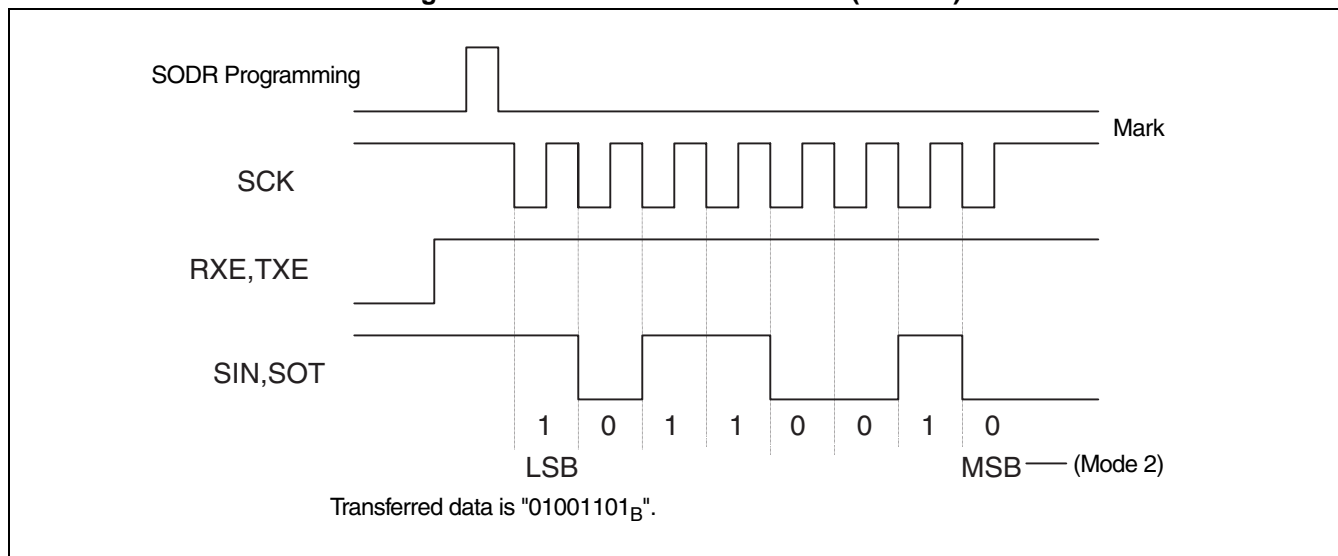


## ■ Clock Synchronous Mode

### ● Transfer data format

The UART handles only data in the NRZ (Non Return to Zero) format. Figure 14.3-2 shows the relationship between transmission/reception clock signal and data.

**Figure 14.3-2 Transfer Data Format (mode 2)**



When data is transmitted with the internal clock (U-TIMER) selected, a synchronous clock signal for data reception is generated automatically.

When the external clock has been selected, supply the clock signal for an exact one byte after making sure that the SODR register as a UART transmission data buffer of transmission side contains data (with the TDRE flag set to "0"). Be sure to set the mark level before and after transmission.

The data length must be 8 bits only, where no parity bit can be added. Note also that any error other than overrun errors is not detected because the neither the start nor stop bit is included.

### ● Initialization

Describes the setting value of each control register when using the clock synchronous mode.

#### (1) SMR register

MD1,MD0 : 10<sub>B</sub>  
 CS0 : Specify the clock input.  
 SCKE : "1" for internal timer; "0" for external clock

#### (2) SCR register

PEN : "0"  
 P, SBL, and A/D : These bits do not have the meaning.  
 CL : "1"  
 REC : "0" (for initialization)  
 RXE and TXE : At least, it is "1" as for either

#### (3) SSR register

RIE : "1" to use interrupts, "0" not to use interrupts.  
 TIE : "0"

- Starting communications

Writing to the SODR register starts communication. Even for only reception, be sure to write temporary transmit data to the SODR register.

- Terminating communications

This terminating can be confirmed by RDRF flag of SSR register having changed into "1". Check whether communication has been performed normally, with the ORE bit of the SSR register.

## ■ Interrupt Generation and Flag Setting Timings

The UART has five flags and two interrupt sources.

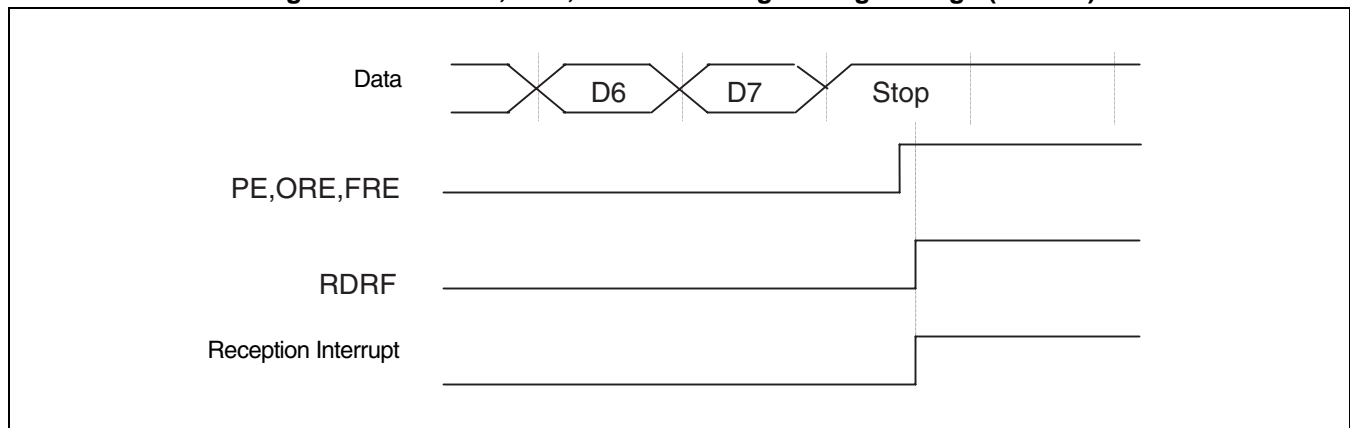
Five flags are PE/ORE/FRE/RDRF/TDRE. PE, ORE, and FRE stand for parity error, overrun error, and framing error, respectively. Each of these flags is set when the corresponding error occurs during reception. It is cleared by writing "0" to the REC bit of the SCR register. The RDRF flag is set when receive data is loaded into the S IDR register; it is cleared when the loaded data is read from the S IDR register. Note, however, that mode 1 does not support parity detection and mode 2 supports neither parity detection nor framing error detection. The TDRE flag is set when the SODR register becomes empty, or write-enabled; it is cleared when data is written to the SODR register.

One of the two interrupt sources is for reception and the other is for transmission. During reception, an interrupt request is generated in response to the PE/ORE/FRE/RDRF flag. During transmission, an interrupt request is generated in response to the TDRE flag. The remainder of this section shows the timings at which the interrupt flags are set in individual operation modes.

- During reception in mode 0

The PE, ORE, FRE, and RDRF flags are set upon detection of the last stop bit at the end of reception transfer, generating an interrupt request to the CPU. Data in the S IDR register is invalid when the PE, ORE, or FRE flag is active.

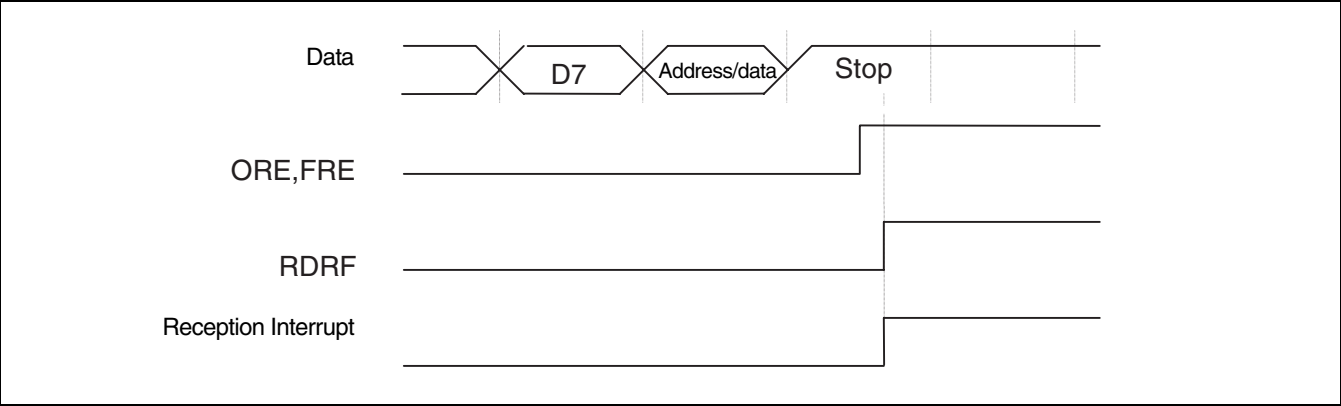
**Figure 14.3-3 ORE, FRE, and RDRF Flag Setting Timings (Mode 0)**



● During reception in mode 1

The ORE, FRE, and RDRF flags are set upon detection of the last stop bit at the end of reception transfer, generating an interrupt request to the CPU. As the acceptable data length is 8 bits, the address/data indicator in the last, ninth bit is treated as invalid data. Data in the SISR register is invalid when the ORE or FRE flag is active.

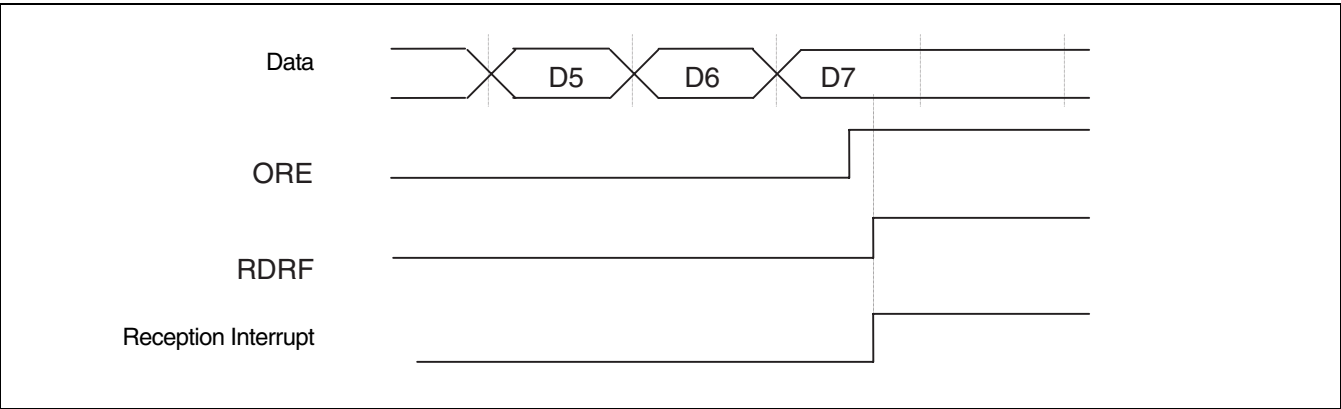
Figure 14.3-4 ORE, FRE, and RDRF Flag Setting Timings (Mode 1)



● During reception in mode 2

The ORE and RDRF flags are set upon detection of the last data (D7) at the end of reception transfer, generating an interrupt request to the CPU. Data in the SISR register is invalid when the ORE flag is active.

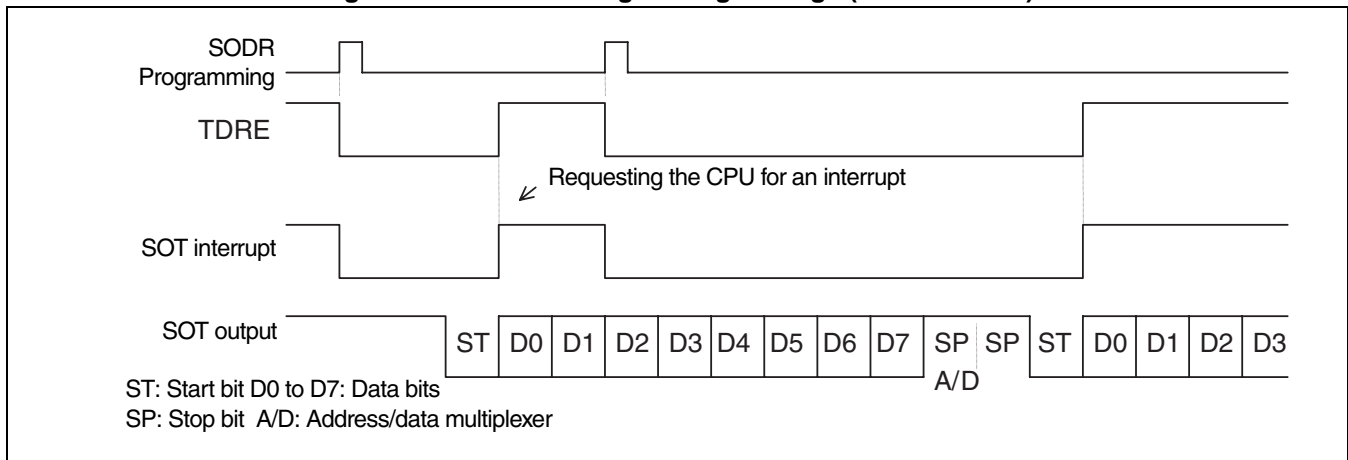
Figure 14.3-5 ORE and RDRF Flag Setting Timings (Mode 2)



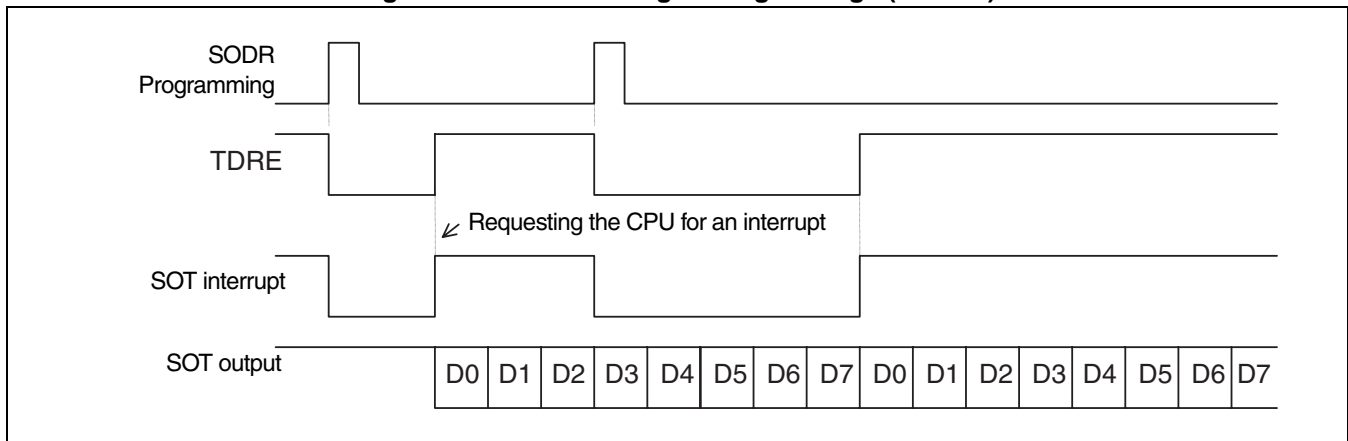
● During transmission in modes 0, 1, and 2

The TDRE flag is cleared when data is written to the SODR register. When the data is transferred to the internal shift register so that the next data can be written, an interrupt request to the CPU is generated. When "0" is written to the TXE bit (and the RXE bit in mode 2) of the SCR register during transmission, the TDRE bit of the SSR register is set to "1", disabling the UART for transmission after the transmission shifter stops. After "0" is written to the TXE bit (and the RXE bit in mode 2) of the SCR register during transmission, the data written to the SODR register is transmitted before the transmission stops.

**Figure 14.3-6 TDRE Flag Setting Timings (Mode 0 and 1)**



**Figure 14.3-7 TDRE Flag Setting Timings (Mode 2)**



■ **Precautions when Using**

Writing to the SODR register starts communication. Even for only reception, be sure to write temporary transmit data to the SODR register.

Set the communication mode while the UART is inactive. Any data sent or received while setting modes is not guaranteed.

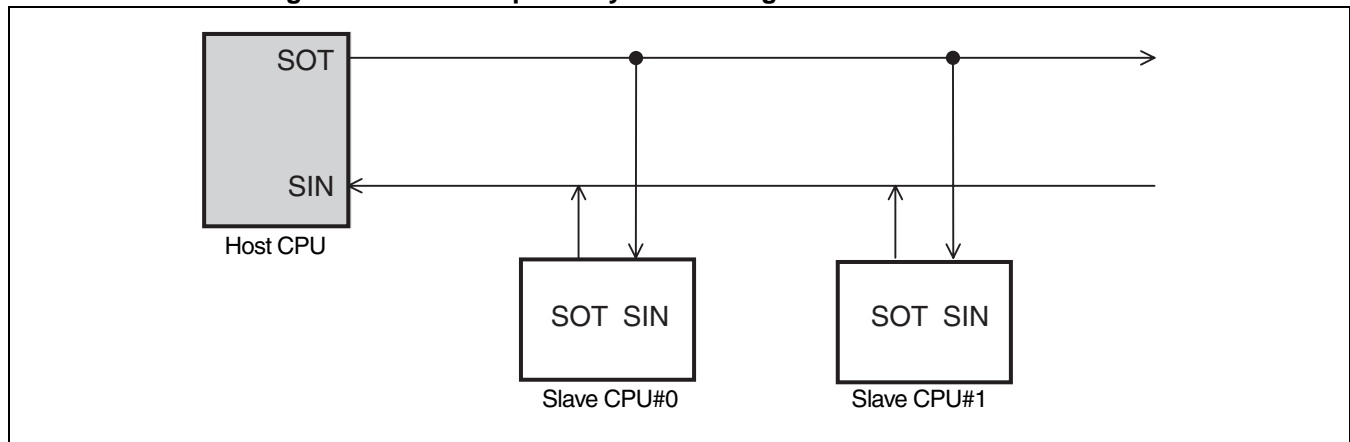
## 14.4 Application Example

This section describes application example of the UART and mode 1 of UART.

### ■ Application Example

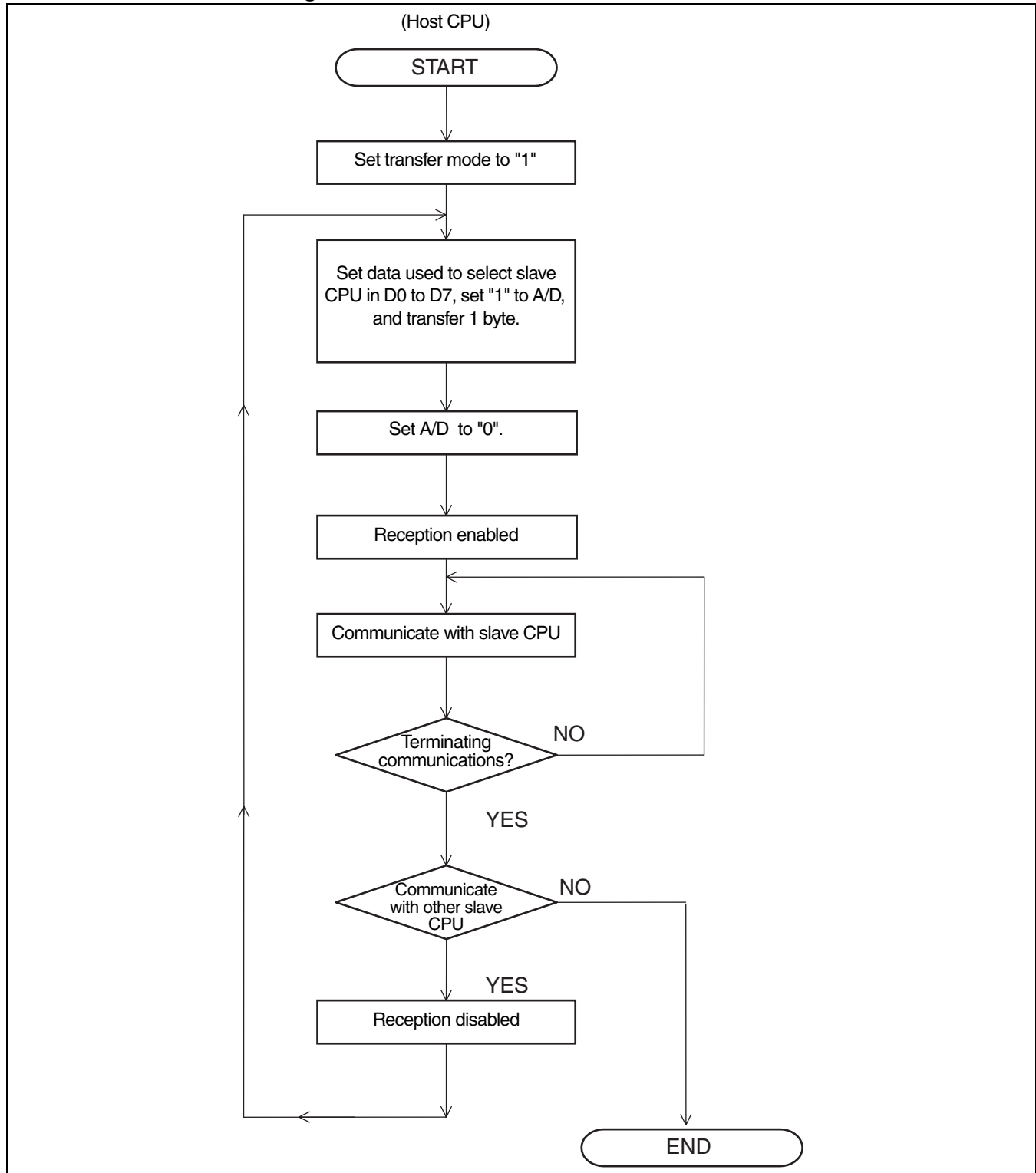
Mode 1 is used, for example, when two or more slave CPUs are connected to a single host CPU (See Figure 14.4-1). This resource supports only the communication interface on the host side.

**Figure 14.4-1 Example of System Configuration for Use in Mode 1**



Communication starts when the host CPU transfers address data. The address data is obtained when the A/D bit of the SCR register is "1", with which the destination slave CPU is selected to communicate with the host CPU. Normal data is obtained when the A/D bit of the SCR register is "0". Figure 14.4-2 is a communication flowchart.

For mode 1, set the PEN bit of the SCR register to "0" because parity checking is not available.

**Figure 14.4-2 Mode-1 Communication Flowchart**

## 14.5 Examples of Baud Rate and U-TIMER Reload Value Settings

This section shows the examples of baud rate and U-TIMER reload value settings.

### ● Examples of baud rate and U-TIMER reload value settings

The frequencies in the table are peripheral machine clock frequencies. UCC1 represents the value set in the UCC1 bit of the UTIMC register of the U-TIMER.

"-" in the table indicates that the corresponding setting cannot be used as an error exceeds  $\pm 1\%$ .

**Table 14.5-1 Asynchronous (start-stop Synchronization) Mode**

Baud rate	$\mu\text{s}$	33MHz	20MHz	16.5MHz	10MHz
1200	833.33	858(UCC1=0)	520(UCC1=0)	428(UCC1=1)	259(UCC1=1)
2400	416.67	428(UCC1=1)	259(UCC1=1)	214(UCC1=0)	129(UCC1=0)
4800	208.33	214(UCC1=0)	129(UCC1=0)	106(UCC1=0)	64(UCC1=0)
9600	104.17	106(UCC1=1)	64(UCC1=0)	52(UCC1=1)	31(UCC1=1)
19200	52.08	52(UCC1=1)	31(UCC1=1)	26(UCC1=0)	-
38400	26.04	26(UCC1=0)	-	12(UCC1=1)	-
57600	17.36	17(UCC1=0)	-	8(UCC1=0)	-
10400	96.15	98(UCC1=0)	59(UCC1=0)	48(UCC1=1)	29(UCC1=0)
31250	32.00	32(UCC1=0)	19(UCC1=0)	15(UCC1=1)	9(UCC1=0)
62500	16.00	15(UCC1=1)	9(UCC1=0)	-	4(UCC1=0)

**Table 14.5-2 CLK Synchronous Mode**

Baud rate	$\mu\text{s}$	33MHz	20MHz	16.5MHz	10MHz
250K	4.00	65(UCC1=0)	39(UCC1=0)	32(UCC1=0)	19(UCC1=0)
500K	2.00	32(UCC1=0)	19(UCC1=0)	15(UCC1=1)	9(UCC1=0)
1M	1.00	15(UCC1=1)	9(UCC1=0)	* 7(UCC1=0)	4(UCC1=0)

\*: There is an error margin of  $\pm 1\%$  or more.

# **CHAPTER 15**

---

## **C-CAN**

**This chapter explains the functions and the operation of C-CAN.**

- 15.1 Features of C-CAN
- 15.2 Block Diagram of C-CAN
- 15.3 Registers of C-CAN
- 15.4 Function of C-CAN Register
- 15.5 C-CAN Function



## 15.1 Features of C-CAN

---

**The C-CAN conforms to the CAN protocol version 2.0 part A and B that is the standard protocol for serial communication. It is widely used in industrial applications such as automobile and FA.**

---

### ■ Features of C-CAN

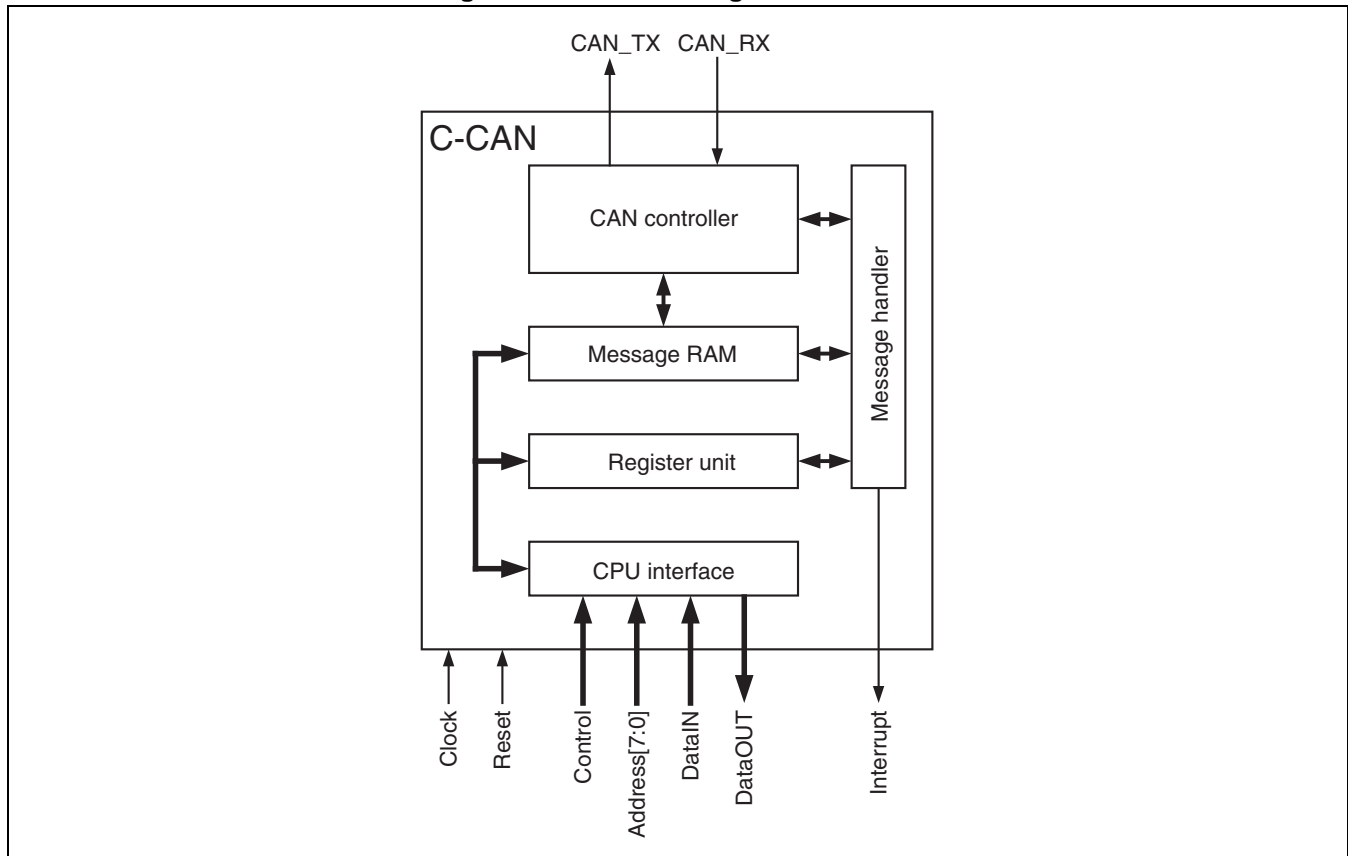
The C-CAN implements the following features:

- Supports CAN protocol version 2.0 part A and B
- Bit rates up to 1 Mbps
- Each message object has its own identifier mask
- Supports programmable FIFO mode
- Maskable interrupt
- Supports programmable loop back mode for self-test operation
- Read/write to message buffer by using interface register

## 15.2 Block Diagram of C-CAN

Figure15.2-1 shows the block diagram of C-CAN.

Figure15.2-1 Block Diagram of C-CAN



### ■ CAN Controller

It controls serial registers for serial/parallel conversion used for transfer of CAN protocol and transmission/reception message.

### ■ Message RAM

It stores the message object.

### ■ Register Group

It is all registers used in C-CAN

### ■ Message Handler

It controls message RAM and CAN controller.

### ■ CPU Interface

It controls the interface in FR internal bus.

## 15.3 Registers of C-CAN

---

The C-CAN implements the following registers:

- CAN control register (CTRLR)
  - CAN status register (STATR)
  - CAN error counter (ERRCNT)
  - CAN bit timing register (BTR)
  - CAN interrupt register (INTR)
  - CAN test register (TESTR)
  - CAN prescaler extended register (BRPER)
  - IFx command request register (IFxCREQ)
  - IFx command mask register (IFxCMSK)
  - IFx mask registers 1 and 2 (IFxMSK1, IFxMSK2)
  - IFx arbitration 1 and 2 (IFxARB1, IFxARB2)
  - IFx message control register (IFxMCTR)
  - IFx data registers A1, A2, B1 and B2 (IFxDTA1, IFxDTA2, IFxDTB1, IFxDTB2)
  - CAN transmission request registers 1 and 2 (TREQR1, TREQR2)
  - CAN New Data registers 1 and 2 (NEWDT1, NEWDT2)
  - CAN interrupt pending registers 1 and 2 (INTPND1, INTPND2)
  - CAN message enable registers 1 and 2 (MSGVAL1, MSGVAL2)
  - CAN clock prescaler register (CANPRE)
-

## ■ List of Overall Control Register

**Table 15.3-1 List of Overall Control Register**

Address	Register				Remarks
	+0	+1	+2	+3	
Base-addr + 00 <sub>H</sub>	CAN control register (CTRLR)		CAN status register (STATR)		STAR: BOff, EWarn, EPass = Read Only RxOk, TxOk, LEC = Read/Write
	bit[15:8]	bit[7:0]	bit[15:8]	bit[7:0]	
	Reserved bit	See Section "15.4.1.1 CAN Control Register (CTRLR)".	Reserved bit	See Section "15.4.1.2 CAN Status Register (STATR)".	
	Reset: 00 <sub>H</sub>	Reset: 01 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	
Base-addr + 04 <sub>H</sub>	CAN error counter (ERRCNT)		CAN bit timing register (BTR)		ERRCNT: Read Only  BTR: Writable at Init (CTRLR) = CCE (CTRLR) = 1.
	bit[15:8]	bit[7:0]	bit[15:8]	bit[7:0]	
	RP, REC[6:0]	TEC[7:0]	TSeg2[2:0], TSeg1[3:0]	SJW[1:0], BRP[5:0]	
	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 23 <sub>H</sub>	Reset: 01 <sub>H</sub>	
Base-addr + 08 <sub>H</sub>	CAN interrupt register (INTR)		CAN test register (TESTR)		INTR: Read Only  TESTR: Writable at Test (CTRLR) = 1. "Rx" specifies the level value of CAN_RX pin.
	bit[15:8]	bit[7:0]	bit[15:8]	bit[7:0]	
	IntId[15:8]	IntId[7:0]	Reserved bit	See Section "15.4.1.6 CAN Test Register (TESTR)".	
	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub> & 0br0000000	
Base-addr + 0C <sub>H</sub>	CAN prescaler extended register (BRPER)		Reserved bit		BRP: Writable at CCE (CTRLR) = 1.
	bit[15:8]	bit[7:0]	bit[15:8]	bit[7:0]	
	Reserved bit	BRP[3:0]	-	-	
	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	

## ■ List of Message Interface Register

**Table 15.3-2 List of Message Interface Register (1 / 3)**

Address	Register				Remarks
	+0	+1	+2	+3	
Base-addr + 10 <sub>H</sub>	IF1 command request register (IF1CREQ)		IF1 command mask register (IF1CMSK)		
	bit[15:8]	bit[7:0]	bit[15:8]	bit[7:0]	
	Busy	Mess. No. [5:0]	Reserved bit	See Section "15.4.2.2 IFx Command Mask Register (IFxCMSK)".	
	Reset: 00 <sub>H</sub>	Reset: 01 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	
Base-addr + 14 <sub>H</sub>	IF1 mask register 2 (IF1CMSK2)		IF1 mask register 1 (IF1CMSK1)		
	bit[15:8]	bit[7:0]	bit[15:8]	bit[7:0]	
	MXtd, MDir, Msk[28:24]	Msk[23:16]	Msk[15:8]	Msk[7:0]	
	Reset: FF <sub>H</sub>	Reset: FF <sub>H</sub>	Reset: FF <sub>H</sub>	Reset: FF <sub>H</sub>	
Base-addr + 18 <sub>H</sub>	IF1 arbitration register 2 (IF1ARB2)		IF1 arbitration register 1 (IF1ARB1)		
	bit[15:8]	bit[7:0]	bit[15:8]	bit[7:0]	
	MsgVal, Xtd, Dir, ID[28:24]	ID[23:16]	ID[15:8]	ID[7:0]	
	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	
Base-addr + 1C <sub>H</sub>	IF1 message control register (IF1MCTR)		Reserved bit		
	bit[15:8]	bit[7:0]	bit[15:8]	bit[7:0]	
	See Section "15.4.2.5 IFx Message Control Register (IFxMCTR)".	See Section "15.4.2.5 IFx Message Control Register (IFxMCTR)".	-	-	
	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	
Base-addr + 20 <sub>H</sub>	IF1 data A register 1 (IF1DTA1)		IF1 data A register 2 (IF1DTA2)		Big Endian byte ordering
	bit[7:0]	bit[15:8]	bit[7:0]	bit[15:8]	
	Data[0]	Data[1]	Data[2]	Data[3]	
	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	
Base-addr + 24 <sub>H</sub>	IF1 data B register 1 (IF1DTB1)		IF1 data B register 2 (IF1DTB2)		Big Endian byte ordering
	bit[7:0]	bit[15:8]	bit[7:0]	bit[15:8]	
	Data[4]	Data[5]	Data[6]	Data[7]	
	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	
Base-addr + 30 <sub>H</sub>	IF1 data A register 2 (IF1DTA2)		IF1 data A register 1 (IF1DTA1)		Little Endian byte ordering
	bit[15:8]	bit[7:0]	bit[15:8]	bit[7:0]	
	Data[3]	Data[2]	Data[1]	Data[0]	
	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	

**Table 15.3-2 List of Message Interface Register (2 / 3)**

Address	Register				Remarks
	+0	+1	+2	+3	
Base-addr + 34 <sub>H</sub>	IF1 data B register 2 (IF1DTB2)		IF1 data B register 1 (IF1DTB1)		Little Endian byte ordering
	bit[15:8]	bit[7:0]	bit[15:8]	bit[7:0]	
	Data[7]	Data[6]	Data[5]	Data[4]	
	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	
Base-addr + 40 <sub>H</sub>	IF2 command request register (IF2CREQ)		IF2 command mask register (IF2CMSK)		
	bit[15:8]	bit[7:0]	bit[15:8]	bit[7:0]	
	Busy	Mess. No. [5:0]	Reserved bit	See Section "15.4.2.2 IFx Command Mask Register (IFxCMSK)".	
	Reset: 00 <sub>H</sub>	Reset: 01 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	
Base-addr + 44 <sub>H</sub>	IF2 mask register 2 (IF2CMSK2)		IF2 mask register 1 (IF2CMSK1)		
	bit[15:8]	bit[7:0]	bit[15:8]	bit[7:0]	
	MXtd. MDir, Msk[28:24]	Msk[23:16]	Msk[15:8]	Msk[7:0]	
	Reset: FF <sub>H</sub>	Reset: FF <sub>H</sub>	Reset: FF <sub>H</sub>	Reset: FF <sub>H</sub>	
Base-addr + 48 <sub>H</sub>	IF2 arbitration register 2 (IF2ARB2)		IF2 arbitration register 1 (IF2ARB1)		
	bit[15:8]	bit[7:0]	bit[15:8]	bit[7:0]	
	MsgVal, Xtd, Dir,ID[28:24]	ID[23:16]	ID[15:8]	ID[7:0]	
	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	
Base-addr + 4C <sub>H</sub>	IF2 message control register (IF2MCTR)		Reserved bit		
	bit[15:8]	bit[7:0]	bit[7:0]	bit[15:8]	
	See Section "15.4.2.5 IFx Message Control Register (IFxMCTR)".	See Section "15.4.2.5 IFx Message Control Register (IFxMCTR)".	-	-	
	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	
Base-addr + 50 <sub>H</sub>	IF2 data A register 1 (IF2DTA1)		IF2 data A register 2 (IF2DTA2)		Big Endian byte ordering
	bit[7:0]	bit[15:8]	bit[7:0]	bit[15:8]	
	Data[0]	Data[1]	Data[2]	Data[3]	
	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	
Base-addr + 54 <sub>H</sub>	IF2 data B register 1 (IF2DTB1)		IF2 data B register 2 (IF2DTB2)		Big Endian byte ordering
	bit[7:0]	bit[15:8]	bit[7:0]	bit[15:8]	
	Data[4]	Data[5]	Data[6]	Data[7]	
	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	

**Table 15.3-2 List of Message Interface Register (3 / 3)**

Address	Register				Remarks
	+0	+1	+2	+3	
Base-addr + 60 <sub>H</sub>	IF2 data A register 2 (IF2DTA2)		IF2 data A register 1 (IF2DTA1)		Little Endian byte ordering
	bit[15:8]	bit[7:0]	bit[15:8]	bit[7:0]	
	Data[3]	Data[2]	Data[1]	Data[0]	
	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	
Base-addr + 64 <sub>H</sub>	IF2 data B register 2 (IF2DTB2)		IF2 data B register 1 (IF2DTB1)		Little Endian byte ordering
	bit[15:8]	bit[7:0]	bit[15:8]	bit[7:0]	
	Data[7]	Data[6]	Data[5]	Data[4]	
	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	

### ■ List of Message Handler Register

**Table 15.3-3 List of Message Handler Register**

Address	Register				Remarks
	+0	+1	+2	+3	
Base-addr + 80 <sub>H</sub>	CAN transmission request register 2 (TREQR2)		CAN transmission request register 1 (TREQR1)		INTR1, INTR2: Read Only
	bit[15:8]	bit[7:0]	bit[15:8]	bit[7:0]	
	TxRqst[32-25]	TxRqst[24-17]	TxRqst[16-9]	TxRqst[8-1]	
	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	
Base-addr + 90 <sub>H</sub>	CAN New Data register 2 (NEWDT2)		CAN New Data register 1 (NEWDT1)		NEWDT1, NEWDT2: Read Only
	bit[15:8]	bit[7:0]	bit[15:8]	bit[7:0]	
	NewDat[32-25]	NewDat[24-17]	NewData[16-9]	NewData[8-1]	
	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	
Base-addr + A0 <sub>H</sub>	CAN interrupt pending register 2 (INTPND2)		CAN interrupt pending register 1 (INTPND1)		INTPND1, INTPND2: Read Only
	bit[15:8]	bit[7:0]	bit[15:8]	bit[7:0]	
	IntPnd[32-25]	IntPnd[24-17]	IntPnd[16-9]	IntPnd[8-1]	
	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	
Base-addr + B0 <sub>H</sub>	CAN message enable register 2 (MSGVAL2)		CAN message enable register 1 (MSGVAL1)		MSGVAL1, MSGVAL2: Read Only
	bit[15:8]	bit[7:0]	bit[15:8]	bit[7:0]	
	MsgVal[32-25]	MsgVal[24-17]	MsgVal[16-9]	MsgVal[8-1]	
	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	Reset: 00 <sub>H</sub>	

## ■ Clock Prescaler Register

**Table 15.3-4 Clock Prescaler Register**

Address	Register				Remarks
	+0	+1	+2	+3	
0001A8 <sub>H</sub>	CANPRE	-	-	-	CAN Prescaler
	bit[3:0]	-	-	-	
	CANPRE[3:0]	-	-	-	
	Reset: 00 <sub>H</sub>	-	-	-	



## 15.4 Function of C-CAN Register

---

The CAN register can be allocated the address space in 256 bytes (64 words) and access the byte or the word. CPU is accessed to message RAM through the message interface register.

This section publishes the CAN register and describes a detailed function of each register.

---

### ■ Overall Control Register

- CAN control register (CTRLR)
- CAN status register (STATR)
- CAN error counter (ERRCNT)
- CAN bit timing register (BTR)
- CAN interrupt register (INTR)
- CAN test register (TESTR)
- CAN prescaler extended register (BRPER)

### ■ Message Interface Register

- IFx command request register (IFxCREQ)
- IFx command mask register (IFxCMSK)
- IFx mask registers 1 and 2 (IFxMSK1, IFxMSK2)
- IFx arbitration registers 1 and 2 (IFxARB1, IFxARB2)
- IFx message control register (IFxMCTR)
- IFx data registers A1, A2, B1 and B2 (IFxDTA1, IFxDTA2, IFxDTB1, IFxDTB2)

### ■ Message Handler Register

- CAN transmission request registers 1 and 2 (TREQR1, TREQR2)
- CAN data renewal registers 1 and 2 (NEWDT1, NEWDT2)
- CAN interrupt pending registers 1 and 2 (INTPND1, INTPND2)
- CAN message enable registers 1 and 2 (MSGVAL1, MSGVAL2)

### ■ Prescaler Register

- CAN clock prescaler register (CANPRE)

## 15.4.1 Overall Control Register

---

The overall control register controls the CAN protocol and the operation mode and offers status information.

---

### ■ Overall Control Register

- CAN control register (CTRLR)
- CAN status register (STATR)
- CAN error counter (ERRCNT)
- CAN bit timing register (BTR)
- CAN interrupt register (INTR)
- CAN test register (TESTR)
- CAN prescaler extended register (BRPER)

### 15.4.1.1 CAN Control Register (CTRLR)

This register controls the operation mode of the CAN controller.

#### ■ CAN Control Register (CTRLR)

CAN control register (Upper)									
Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
Base+00 <sub>H</sub>	-	-	-	-	-	-	-	-	00000000 <sub>B</sub>
	R	R	R	R	R	R	R	R	

CAN control register (Lower)									
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
Base+01 <sub>H</sub>	Test	CCE	DAR	-	EIE	SIE	IE	Init	00000000 <sub>B</sub>
	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	

R/W:	Readable/Writable
R:	Read only

#### ■ Function of Registers

[bit15 to bit8] - : Reserved bits

These bits read "0" and write "0" at writing.

[bit7] Test: Test mode enable bit

Test	Function
0	Normal operation [Initial value]
1	Test mode

[bit6] CCE: Bit timing register write enable bit

CCE	Function
0	Writing to CAN bit timing register and CAN prescaler extended register are disabled. [Initial value]
1	Writing to CAN bit timing register and CAN prescaler extended register are enabled. When Init bit is "1", this bit is enabled.

[bit5] DAR: Automatic retransmission disable bit

DAR	Function
0	Automatic retransmission of messages is enabled at arbitration lost or error detection. [Initial value]
1	Automatic retransmission is disabled.

CAN controller performs automatic retransmission of the frame by the arbitration lost or error detection in transferring shown in CAN specification (Refer to ISO11898 and "6.3.3"). If automatic retransmission is performed, reset DAR bit to "0". It is necessary to set DAR bit to "1" if CAN is operated in the state of Time Triggered CAN (Refer to TTCAN and ISO11898-1).

---

Note :

In the mode which sets DAR bit to "1", operation of TxRqst bit in message object (see Section "15.4.3 Message Object" for the message object.) differs from that of NewDat bit in message object.

- When frame transmission is started, TxRqst of message object is reset to "0", but NewDat bit is remained to set.
- When frame transmission is completed normally, NewDat is reset to "0".

If the transmission operates arbitration lost or error detection, NewDat is remained to set.

To restart the transmission, it is necessary to set TxRqst to "1" by CPU.

---

[bit4] - : Reserved bit

This bit reads "0" and writes "0" at writing.

[bit3] EIE: Error interrupt code enable bit

EIE	Function
0	A change in the bit BOff or EWarn in CAN status register disables setting the interrupt code to CAN interrupt register. [Initial value]
1	A change in the bit BOff or EWarn in CAN status register enables to set the interrupt code to CAN interrupt register.

[bit2] SIE: Status interrupt code enable bit

SIE	Function
0	A change in the bits TxOk, RxOk or LEC in CAN status register disables setting the interrupt code to CAN interrupt register. [Initial value]
1	A change in the bits TxOk, RxOk or LEC in CAN status register enables to set the interrupt code to CAN interrupt register. The change of the bits TxOk, RxOk and LEC generated by writing from CPU is not set in CAN interrupt register.

[bit1] IE: Interrupt enable bit

IE	Function
0	Interrupt generation is disabled. [Initial value]
1	Interrupt generation is enabled.

[bit0] Init: Initialization bit

Init	Function
0	CAN controller operation is enabled.
1	Initialization [Initial value]

---

Notes:

- The bus off recovery sequence (see CAN Specification Rev. 2.0) cannot be shortened by setting or resetting Init bit. If the device goes bus off, the CAN controller itself sets Init bit to "1", stopping all bus activities. Once Init bit has been cleared to "0" from the bus off state, the bus operation is stopped until the bus idle consecutively occurs 129 times (11 recessive bits generate once). At the end of the bus off recovery sequence, the Error Management Counters will be reset.
  - Concerning write to CAN bit timing register, set "1" to Init and CCE bits.
  - Write "1" to INIT bit and initialize the CAN controller before transmitting to low power consumption mode (stop mode and clock mode).
  - When changing the division ratio of clock supplied to CAN interface by CAN prescaler register, change CAN prescaler register after setting INIT bit to "1".
-

## 15.4.1.2 CAN Status Register (STATR)

This register specifies CAN status and the state of CAN bus.

### ■ CAN Status Register (STATR)

CAN status register (Upper)									
Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
Base+02 <sub>H</sub>	-	-	-	-	-	-	-	-	00000000 <sub>B</sub>
	R	R	R	R	R	R	R	R	
CAN status register (Lower)									
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
Base+03 <sub>H</sub>	BOff	EWarn	EPass	RxOk	TxOk	LEC			00000000 <sub>B</sub>
	R	R	R	R/W	R/W	R/W	R/W	R/W	
R/W: Readable/Writable									
R: Read only									

### ■ Function of Registers

[bit15 to bit8] - : Reserved bits

These bits read "0" and write "0" at writing.

[bit7] BOff: Bus Off bit

BOff	Function
0	CAN controller is not bus off state. [Initial value]
1	CAN controller is bus off state.

[bit6] EWarn: Warning bit

EWarn	Function
0	Both transmission and reception counters are below the error warning limit of 96. [Initial value]
1	At least one of the transmission and reception counters has reached the error warning limit of 96.

[bit5] EPass: Error passive bit

EPass	Function
0	Both counters of transmission and reception are below the error passive limit of 128 (error active state) . [Initial value]
1	Reception counter sets RP bit to 1, and transmission counter has reached the error passive limit of 128 (error passive state).

[bit4] RxOk: Normal message reception bit

RxOk	Function
0	The message is not communicated successfully on CAN bus, or it is in the bus idle state. [Initial value]
1	The message is communicated successfully on CAN bus.

[bit3] TxOk: Normal message transmission bit

TxOk	Function
0	It is in the bus idle state, or the message is not transmitted successfully. [Initial value]
1	The message is transmitted successfully.

---

Note:

RxOk and TxOk bits are reset only by CPU.

---

[bit2 to bit0] LEC: Last error code bits

LEC	State	Function
0	Normal	Transmission or reception is operated successfully. [Initial value]
1	Stuff error	Consecutive 6-bit or more dominant or recessive is detected in the message.
2	Form error	A fixed format part of a received frame has the wrong format.
3	Ack error	Acknowledge from another node is not performed for the transmission message.
4	Bit1 error	For the transmission data of message other than arbitration field, dominant is detected regardless of transmitting recessive.
5	Bit0 error	For the transmission data of message, recessive is detected regardless of transmitting dominant. This bit is set each time 11 recessive bits are detected during bus recovery. Reading this bit can monitor the bus recovery sequence.
6	CRC error	CRC data of the received message and result of calculated CRC do not match.
7	Not detected	When "7" is read for the LEC value after writing "7" to the LEC bit by CPU, this period does not transmit / receive. (bus idle state)

The LEC bit holds a code which indicates the type of the last error to occur on the CAN bus. This bit will be cleared to "0" when a message has been transferred (reception or transmission) without error.

The undetected code "7" may be written by the CPU to check for updates of code.

---

#### Notes:

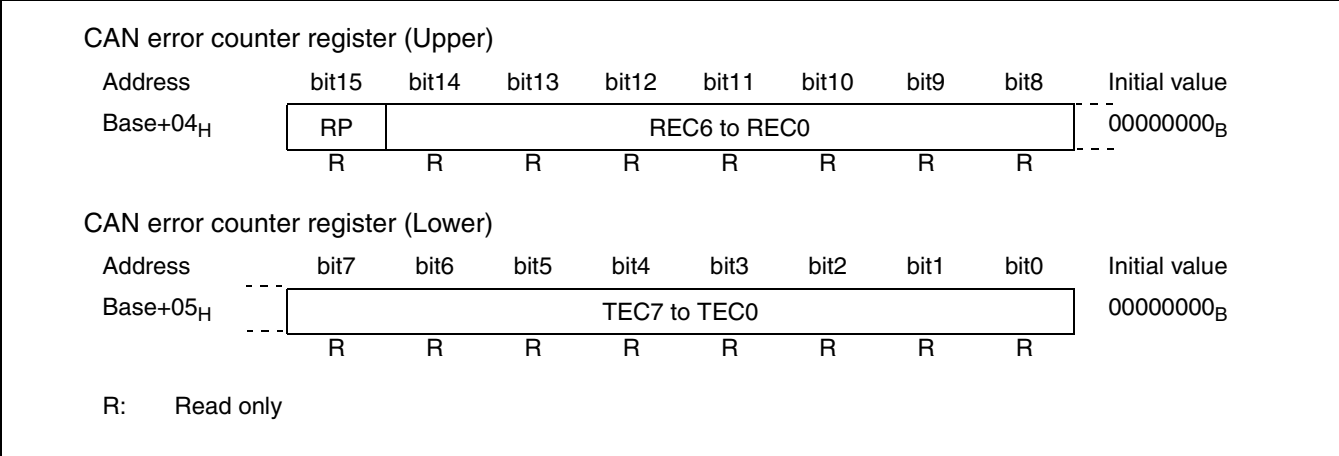
- A status interrupt code (8000<sub>H</sub>) is set to the CAN interrupt register when EIE bit is "1" if BOff or EWarn bit is changed or when SIE bit is "1" if RxOk, TxOk, or LEC bit is changed.
  - Because RxOk and TxOk bits are updated by writing to CPU, RxOk and TxOk bits set by the CAN controller are lost. If RxOk and TxOk bits are used, clear the bits within time of (45×BT) after setting RxOk or TxOk bit to "1". BT indicates 1 bit time.
  - Do not write to the CAN status register if the interrupt occurs due to change of LEC bit when SIE bit is "1".
  - A change of EPass bit or a write to RxOk, TxOk, or LEC bit by CPU will never generate a status interrupt.
  - EWarn bit is remained to set "1" even if BOff bit or EPass bit becomes "1".
  - Reading this register will clear the status interrupt (8000<sub>H</sub>) in the CAN interrupt register.
-



### 15.4.1.3 CAN Error Counter (ERRCNT)

This register shows the reception error passive display, the reception error counter and transmission error counter.

#### ■ CAN Error Counter (ERRCNT)



#### ■ Function of Registers

[bit15] RP: Reception error passive display

RP	Function
0	The reception error counter is not the error passive state. [Initial value]
1	The reception error counter is reached the error passive state defined by CAN specification.

[bit14 to bit8] REC6 to REC0: Reception error counters

Reception error counter value. This range is 0 to 127.

[bit7 to bit0] TEC7 to TEC0: Transmission error counters

Transmission error counter value. This range is 0 to 255.

### 15.4.1.4 CAN Bit Timing Register (BTR)

This register sets prescaler and bit timing.

#### ■ CAN Bit Timing Register (BTR)

CAN bit timing register (Upper)									
Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
Base+06 <sub>H</sub>	-	TSeg2			TSeg1				00100011 <sub>B</sub>
	R	R/W	R/W	R/W	R	R	R	R	
CAN bit timing register (Lower)									
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
Base+07 <sub>H</sub>	SJW		BRP						00000001 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
R/W: Readable/Writable									
R: Read only									

#### ■ Function of Registers

[bit15] - : Reserved bit

This bit reads "0" and writes "0" at writing.

[bit14 to bit12] TSeg2: Time segment 2 setting bits

Enable setting value is 0 to 7. The value of TSeg2+1 becomes time segment 2.

Time segment 2 is equivalent for phase buffer segment (PHASE\_SEG2) of CAN specification.

[bit11 to bit8] TSeg1: Time segment 1 setting bits

Enable setting value is 1 to 15. Setting "0" is prohibited. The value of TSeg1+1 becomes time segment 1.

Time segment 1 is equivalent for propagation segment (PROP\_SEG) + phase buffer segment 1 (PHASE\_SEG1) of CAN specification.

[bit7, bit6] SJW: Resynchronization jump width setting bits

Enable setting value is 0 to 3. The value of SJW+1 becomes resynchronous jump width.

[bit5 to bit0] BRP: Baud rate prescaler setting bits

Enable setting value is 0 to 63. The value of BRP+1 becomes baud rate prescaler.

The basic unit time (t<sub>q</sub>) of CAN controller is determined by dividing the system clock (f<sub>sys</sub>).

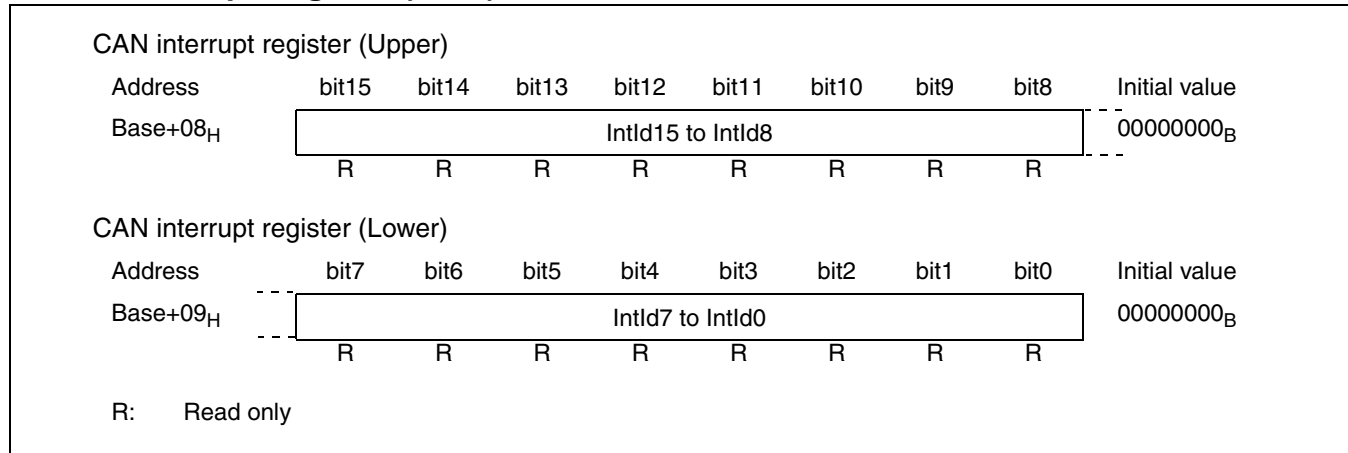
Note:

Set the CAN bit timing register and CAN prescaler extended register while CCE bit and Init bit of CAN control register are set to "1".

### 15.4.1.5 CAN Interrupt Register (INTR)

This register shows the message interrupt code and the status interrupt code.

#### ■ CAN Interrupt Register (INTR)



#### ■ Function of Registers

IntId	Function
0000 <sub>H</sub>	No interrupt
0001 <sub>H</sub> to 0020 <sub>H</sub>	The source of the interrupt indicates the number of message object. (message interrupt code)
0021 <sub>H</sub> to 7FFF <sub>H</sub>	Unused
8000 <sub>H</sub>	Interrupt by the change of CAN status register is indicated. (status interrupt code)
8001 <sub>H</sub> to FFFF <sub>H</sub>	Unused

If several interrupt codes are pending, the CAN interrupt register will point to the pending interrupt code with the highest priority. When the interrupt code with the highest priority is generated even if the interrupt code is set in CAN interrupt register, CAN interrupt register is renewed to the interrupt code with highest priority.

The interrupt code with highest priority is in the order of the status interrupt code (8000<sub>H</sub>), the message interrupt (0001<sub>H</sub>, 0002<sub>H</sub>, 0003<sub>H</sub>, ....., 0020<sub>H</sub>).

When IntId bit is a value other than 0000<sub>H</sub> and IE bit of CAN control register is set to "1", the interrupt signal to CPU becomes active. When the value of IntId becomes 0000<sub>H</sub> (the source of interrupt is reset) or IE bit of CAN control register is reset to "0", the interrupt signal is inactive.

Clearing IntPnd bit of corresponding message object (refer to Section "15.4.3 Message Object" for the message object.) to "0" clears the message interrupt code.

Reading the CAN status register clears the status interrupt code.

### 15.4.1.6 CAN Test Register (TESTR)

This register sets the test mode and monitors RXO pin. Refer to Section "15.5.7 Test mode" for operation.

#### ■ CAN Test Register (TESTR)

CAN test register (Upper)									
Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
Base+0A <sub>H</sub>	-	-	-	-	-	-	-	-	00000000 <sub>B</sub>
	R	R	R	R	R	R	R	R	

CAN test register (Lower)									
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
Base+0B <sub>H</sub>	Rx	Tx1	Tx0	LBack	Silent	Basic	-	-	00000000 <sub>B</sub>
	R	R/W	R/W	R/W	R/W	R/W	R	R	

R/W:	Readable/Writable
R:	Read only

Initial value R of Rx in bit7 displays the level on CAN bus.

#### ■ Function of Registers

[bit15 to bit8] - : Reserved bits

These bits read "0" and write "0" at writing.

[bit7] Rx: RXO pin monitoring bit

Rx	Function
0	CAN bus is dominant.
1	CAN bus is recessive.

[bit6, bit5] Tx1, Tx0: TXO pin control bits

Tx1, Tx0	Function
00	Normal operation [Initial value]
01	Sampling point is outputted to Tx pin.
10	Dominant is outputted to Tx pin.
11	Recessive is outputted to Tx pin.

[bit4] LBack: Loop back mode

LBack	Function
0	Loop back mode is disabled. [Initial value]
1	Loop back mode is enabled.

[bit3] Silent: Silent mode

Silent	Function
0	Silent mode is disabled. [Initial value]
1	Silent mode is enabled.

[bit2] Basic: Basic mode

Basic	Function
0	Basic mode is disabled. [Initial value]
1	Basic mode is enabled. IF1 register and IF2 register are used as transmission message and reception message respectively.

[bit1, bit0] - : Reserved bits

These bits read "0" and write "0" at writing.

---

Notes:

- Write to this register after setting the Test bit of CAN control register to "1". Test mode is enabled when Test bit of CAN control register is "1". The mode is changed from test mode to normal mode when Test bit of CAN control register is set to "0" in the middle of operation.
  - When Tx bit is set to a value other than "00", the message cannot be transmitted.
-

### 15.4.1.7 CAN Prescaler Extended Register (BRPER)

By combining with the prescaler set in CAN bit timing, this register extends the prescaler used in CAN controller.

#### ■ CAN Prescaler Extended Register (BRPER)

CAN prescaler extended register (Upper)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
Base+0C <sub>H</sub>	-	-	-	-	-	-	-	-	00000000 <sub>B</sub>
	R	R	R	R	R	R	R	R	

CAN prescaler extended register (Lower)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
Base+0D <sub>H</sub>	-	-	-	-	BRPE				00000000 <sub>B</sub>
	R	R	R	R	R/W	R/W	R/W	R/W	

R/W: Readable/Writable

R: Read only

#### ■ Function of Registers

[bit15 to bit4] - : Reserved bits

These bits read "0" and write "0" at writing.

[bit3 to bit0] BRPE: Baud rate prescaler extended bits

By combining BRP of CAN bit timing register with BRPE, the baud rate prescaler can be extended to values up to 1023.

The value of {BRPE(MSB:4 bits), BRP(LSB:6 bits)} + 1 becomes the prescaler of CAN controller.

## 15.4.2 Message Interface Register

---

**Two sets of message interface registers which are used to control the CPU access to the message RAM are offered.**

---

There are two sets of message interface registers which are used to control the CPU access to the message RAM. The message interface registers avoid conflict between CPU access to the message RAM and access from the CAN controller by buffering the data to be transferred (message object). A message object (refer to Section "15.4.3 Message Object" for message object) may be transferred between the message interface register and the message RAM in one single transfer.

The function for two sets of message interface registers is identical and can operate independently (except for test basic mode ). For example, they can be used the way that the message interface register of IF2 is used to read from message RAM while the message interface register of IF1 is used to write to message RAM. Table 15.4-1 gives an overview for two sets of message interface registers.

Each set of message interface registers consists of command register (command request and command mask register) and message buffer registers controlled by their own command registers (mask, arbitration, message control and data register). The command mask register specifies the direction of the data transfer and which parts of a message object will be transferred. The command request register is used to select a message number and perform the operation set in the command mask register.

**Table 15.4-1 IF1 and IF2 Message Interface Registers**

Address	IF1 register set	Address	IF2 register set
Base + 10 <sub>H</sub>	IF1 command request	Base + 40 <sub>H</sub>	IF2 command request
Base + 12 <sub>H</sub>	IF1 command mask	Base + 42 <sub>H</sub>	IF2 command mask
Base + 14 <sub>H</sub>	IF1 mask2	Base + 44 <sub>H</sub>	IF2 mask 2
Base + 16 <sub>H</sub>	IF1 mask 1	Base + 46 <sub>H</sub>	IF2 mask 1
Base + 18 <sub>H</sub>	IF1 arbitration 2	Base + 48 <sub>H</sub>	IF2 arbitration 2
Base + 1A <sub>H</sub>	IF1 arbitration 1	Base + 4A <sub>H</sub>	IF2 arbitration 1
Base + 1C <sub>H</sub>	IF1 message control	Base + 4C <sub>H</sub>	IF2 message control
Base + 20 <sub>H</sub>	IF1 data A1	Base + 50 <sub>H</sub>	IF2 data A1
Base + 22 <sub>H</sub>	IF1 data A2	Base + 52 <sub>H</sub>	IF2 data A2
Base + 24 <sub>H</sub>	IF1 data B1	Base + 54 <sub>H</sub>	IF2 data B1
Base + 26 <sub>H</sub>	IF1 data B2	Base + 56 <sub>H</sub>	IF2 data B2

### 15.4.2.1 IFx Command Request Register (IFxCREQ)

This register selects the message number of message RAM and transfers between the message RAM and the message buffer registers. In the test basic mode, IF1 and IF2 are used for transmission control and reception control respectively.

#### ■ IFx Command Request Register (IFxCREQ)

IFx command request register (Upper)									
Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
Base+10 <sub>H</sub>	BUSY	-	-	-	-	-	-	-	00000000 <sub>B</sub>
Base+40 <sub>H</sub>	R/W	R	R	R	R	R	R	R	
IFx command request register (Lower)									
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
Base+11 <sub>H</sub>	-	-	Message Number						00000000 <sub>B</sub>
Base+41 <sub>H</sub>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
R/W: Readable/Writable									
R: Read only									

#### ■ Function of Registers

A message transfer with message RAM and message buffer register (mask, arbitration, message control and data register) is started as soon as the CPU has written the message number to the IFx command request register. With this write operation, BUSY bit is set to "1" and the CPU is notified that a transfer is in progress. BUSY bit is reset to "0" when the transfer has finished.

When BUSY bit is "1" if access from CPU to the message interface register occurs, CPU is waited until BUSY bit becomes "0" (period of 3 to 6 clock cycles after writing to command request register).

In the test basic mode, method of using BUSY bit is different from that of the mode other than this mode. IF1 command request register is used as the message transmission and indicates that the message transmission is started after setting BUSY bit to "1". When the transmission is completed normally, the bit is reset to "0". Also, resetting the bit to "0" enables to suspend the message transfer at anytime.

IF2 command request register is used as the message reception and stores the received message to IF2 message interface register after setting BUSY bit to "1".



[bit15] BUSY: BUSY flag bit

- Mode other than test basic mode

BUSY	Function
0	The data transfer is not in progress between message interface register and message RAM. [Initial value]
1	The data transfer is in progress between message interface register and message RAM.

- Test basic mode

IF1 command request register

BUSY	Function
0	Message transmission is disabled.
1	Message transmission is enabled.

IF2 command request register

BUSY	Function
0	Message reception is disabled.
1	Message reception is enabled.

[bit14 to bit6] - : Reserved bits

These bits read "0" and write "0" at writing.

[bit5 to bit0] Message Number: Message number (for 32-message buffer CAN)

Message Number	Function
00 <sub>H</sub>	Setting is disabled. If setting, it is regarded as 20 <sub>H</sub> and 20 <sub>H</sub> is read.
01 <sub>H</sub> to 20 <sub>H</sub>	Message number for processing is set.
21 <sub>H</sub> to 3F <sub>H</sub>	Setting is disabled. If setting, it is regarded as 01 <sub>H</sub> to 1F <sub>H</sub> and the value of them is read.

---

Note:

BUSY bit is readable and writable. Writing any value to this bit has no effect on operation in the mode other than the test basic mode (see Section "15.5.7 Test mode" for basic mode).

---

## 15.4.2.2 IFx Command Mask Register (IFxCMSK)

This register controls the transfer direction between message interface register and message RAM and selects which of the data is renewed. Also, this register is invalid in test basic mode.

### ■ IFx Command Mask Register (IFxCMSK)

IFx command mask register (Upper)									
Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
Base+12 <sub>H</sub> & Base+42 <sub>H</sub>	-	-	-	-	-	-	-	-	00000000 <sub>B</sub>
	R	R	R	R	R	R	R	R	

IFx command mask register (Lower)									
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
Base+13 <sub>H</sub> & Base+43 <sub>H</sub>	WR/RD	Mask	Arb	Control	CIP	TxRqst/ NewDat	Data A	Data B	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable

R: Read only

### ■ Function of Registers

[bit15 to bit8] - : Reserved bits

These bits read "0" and write "0" at writing.

[bit7] WR/RD: Write/Read control bit

WR/RD	Function
0	Data is read from message RAM. Reading from message RAM is executed by writing to IFx command request register. Data read from message RAM depends on the setting of Mask, Arb, Control, CIP, TxRqst/NewDat, Data A and Data B bits. [Initial value]
1	Data is written to message RAM. Writing to message RAM is executed by writing to IFx command request register. Data written to message RAM depends on the setting of Mask, Arb, Control, CIP, TxRqst/NewDat, Data A and Data B bits.

Note:

The data of message RAM is undefined after reset. Reading the data of message RAM is disabled under the condition that the data of message RAM is undefined.

Bit6 to bit0 in IFx command mask register has the different meaning depending on the setting of transfer direction (WR/RD bit).

● When transfer direction is write (WR/RD=1)

[bit6] Mask: Mask data renewal bit

Mask	Function
0	The mask data of message object* (ID mask + MDir + MXtd) is not renewed. [Initial value]
1	The mask data of message object* (ID mask + MDir + MXtd) is renewed.

[bit5] Arb: Arbitration data renewal bit

Arb	Function
0	The arbitration data of message object* (ID + Dir + Xtd + MsgVal) is not renewed. [Initial value]
1	The arbitration data of message object* (ID + Dir + Xtd + MsgVal) is renewed.

[bit4] Control: Control data renewal bit

Control	Function
0	The control data of message object* (IFx message control register) is not renewed. [Initial value]
1	The control data of message object* (IFx message control register) is renewed.

[bit3] CIP: Interrupt clear bit

The operation to CAN controller is not affected even if setting "0" or "1" to this bit.

[bit2] TxRqst/NewDat: Message transfer request bit

TxRqst/NewDat	Function
0	"0" is set to the message object* and TxRqst bit of CAN transmission request register. [Initial value]
1	"1" is set to the message object* and TxRqst bit of CAN transmission request register (transmission request).

\*: Refer to Section "15.4.3 Message Object".

[bit1] Data A: Data0 to Data3 renewal bit

Data A	Function
0	The data 0 to 3 of message object* is not renewed. [Initial value]
1	The data 0 to 3 of message object* is renewed.

[bit0] Data B: Data4 to Data7 renewal bit

Data B	Function
0	The data 4 to 7 of message object* is not renewed. [Initial value]
1	The data 4 to 7 of message object* is renewed.

\*: Refer to Section "15.4.3 Message Object".

---

Notes:

- If TxRqst/NewDat bit in the IFx command mask register is set to "1", the setting of TxRqst bit in the IFx message control register will be ignored.
  - This register is invalid in the test basic mode.
-

● When the transfer direction is read (WR/RD=0)

[bit6] Mask: Mask data renewal bit

Mask	Function
0	The data (ID mask + MDir + MXtd) is not transferred from message object* to IFx mask registers 1 and 2. [Initial value]
1	The data (ID mask + MDir + MXtd) is transferred from message object* to IFx mask registers 1 and 2.

[bit5] Arb: Arbitration data renewal bit

Arb	Function
0	The data (ID + Dir + Xtd + MsgVal) is not transferred from message object* to IFx arbitration 1 and 2. [Initial value]
1	The data (ID + Dir + Xtd + MsgVal) is transferred from message object* to IFx arbitration 1 and 2.

[bit4] Control: Control data renewal bit

Control	Function
0	The data is not transferred from message object* to IFx message control register. [Initial value]
1	The data is transferred from message object* to IFx message control register.

[bit3] CIP: Interrupt clear bit

CIP	Function
0	Message object* and IntPnd bit of CAN interrupt pending register are retained. [Initial value]
1	Message object* and IntPnd bit of CAN interrupt pending register are cleared to "0".

[bit2] TxRqst/NewDat: Data renewal bit

TxRqst/NewDat	Function
0	Message object* and NewDat bit of CAN data renewal register are retained. [Initial value]
1	Message object* and NewDat bit of CAN data renewal register are cleared to "0".

\*: Refer to Section "15.4.3 Message Object".

[bit1] Data A: Data0 to Data3 renewal bit

Data A	Function
0	Message object* and the data of CAN data registers A1 and A2 are retained. [Initial value]
1	Message object* and the data of CAN data registers A1 and A2 are renewed.

[bit0] Data B: Data4 to Data7 renewal bit

Data B	Function
0	Message object* and the data of CAN data registers B1 and B2 are retained. [Initial value]
1	Message object* and the data of CAN data registers B1 and B2 are renewed.

\*: Refer to Section "15.4.3 Message Object".

---

Notes:

- A read access to a message object can be reset to "0" of IntPnd and NewDat bits. However, IntPnd and NewDat bits which have not been reset by read access yet are stored in IntPnd and NewDat bits of the IFx message control register.
  - This register is invalid in the test basic mode.
-

### 15.4.2.3 IFx Mask Registers 1 and 2 (IFxMSK1, IFxMSK2)

These registers are used for writing /reading the message object mask data of message RAM. Also, the set mask data is invalid in test basic mode.

The function of each bit is described in Section "15.4.3 Message Object".

#### ■ IFx Mask Registers 1 and 2 (IFxMSK1, IFxMSK2)

##### IFx mask register 2 (Upper)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
Base+14 <sub>H</sub> & Base+44 <sub>H</sub>	MXtd	MDir	-	Msk28 to Msk24					11111111 <sub>B</sub>
	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	

##### IFx mask register 2 (Lower)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
Base+15 <sub>H</sub> & Base+45 <sub>H</sub>	Msk23 to Msk16								11111111 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

##### IFx mask register 1 (Upper)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
Base+16 <sub>H</sub> & Base+46 <sub>H</sub>	Msk15 to Msk8								11111111 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

##### IFx mask register 1 (Lower)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
Base+17 <sub>H</sub> & Base+47 <sub>H</sub>	Msk7 to Msk0								11111111 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable

R: Read only

Refer to Section "15.4.3 Message Object" for bit explanation of these registers.

The reserved bit of these registers (bit13 in IFx mask register 2) is read "1" and written "1" at write.

## 15.4.2.4 IFx Arbitration Registers 1 and 2 (IFxARB1, IFxARB2)

These registers are used for writing/reading the message object arbitration data of message RAM. Also, these are invalid in test basic mode.

The function of each bit is described in Section "15.4.3 Message Object".

### ■ IFx Arbitration Registers 1 and 2 (IFxARB1, IFxARB2)

#### IFx arbitration register 2 (Upper)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
Base+18 <sub>H</sub> & Base+48 <sub>H</sub>	MsgVal	Xtd	Dir	ID28 to ID24					00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

#### IFx arbitration register 2 (Lower)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
Base+19 <sub>H</sub> & Base+49 <sub>H</sub>	ID23 to ID16								00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

#### IFx arbitration register 1 (Upper)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
Base+1A <sub>H</sub> & Base+4A <sub>H</sub>	ID15 to ID8								00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

#### IFx arbitration register 1 (Lower)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
Base+1B <sub>H</sub> & Base+4B <sub>H</sub>	ID7 to ID0								00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable

R: Read only

Refer to Section "15.4.3 Message Object" for bit explanation of these registers.

#### Note:

When MsgVal bit of the message object is cleared to "0" during the transmission, TxOk bit of the CAN status register is set to "1" after the transmission is completed. However, because the message object and TxRqst bit of the CAN transmission request register are not cleared to "0", clear TxRqst bit to "0" using the message interface register.



### 15.4.2.5 IFx Message Control Register (IFxMCTR)

This register is used for writing/reading the message object control data of message RAM. Also, IF1 message control register is invalid in test basic mode. NewDat and MsgLst of IF2 message control register operates normally, and DLC bit displays the received DLC message. The other control bit operates as invalid ("0"). The function of each bit is described in Section "15.4.3 Message Object".

#### ■ IFx Message Control Register (IFxMCTR)

IFx message control register (Upper)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
Base+1C <sub>H</sub> & Base+4C <sub>H</sub>	NewDat	MsgLst	IntPnd	UMask	TxE	RxE	RmtEn	TxRqst	00000000 <sub>B</sub>
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

IFx message control register (Lower)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
Base+1D <sub>H</sub> & Base+4D <sub>H</sub>	EoB	-	-	-	DLC3 to DLC0				00000000 <sub>B</sub>
	R/W	R	R	R	R/W	R/W	R/W	R/W	

R/W: Readable/Writable

R: Read only

Refer to Section "15.4.3 Message Object" for the bit explanation of this register.

#### Note:

By setting of WR/RD bit in IFx command mask register, TxRqst, NewDat and IntPnd bits operate as follows:

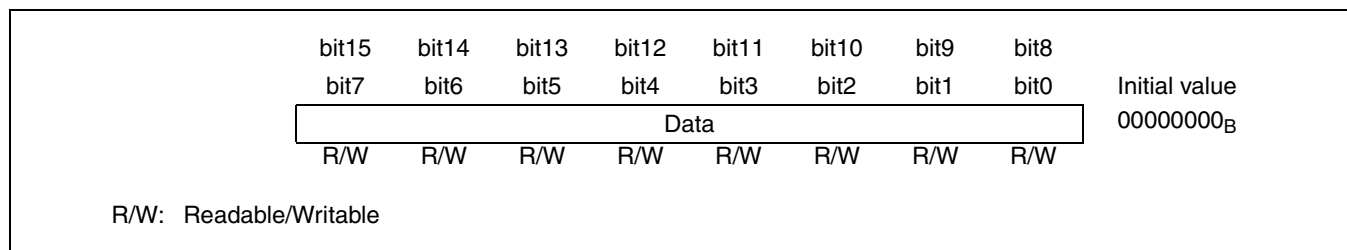
- When transfer direction is write (IFx command mask register: WR/RD=1)
  - TxRqst bit of this register is enabled only when TxRqst/NewDat of IFx command mask register is set to "0".
- When transfer direction is read (IFx command mask register: WR/RD=0)
  - CIP bit of IFx command mask register is set to "1" and the message object and IntPnd bit of CAN interrupt pending register are reset by writing to IFx command request register, IntPnd bit before resetting is stored in this register.
  - TxRqst/NewDat bit of IFx command mask register is set to "1" and the message object and NewDat bit of CAN data renewal register are reset by writing to IFx command request register, NewDat bit before resetting is stored in this register.

### 15.4.2.6 IFx Data Registers A1, A2, B1 and B2 (IFxDTA1, IFxDTA2, IFxDTB1, IFxDTB2)

These registers are used for writing/reading the message object transmission/reception data of message RAM. These are used for only transmission/reception of data frame, not for transmission/reception of remote frame.

#### ■ IFx Data Registers A1, A2, B1 and B2 (IFxDTA1, IFxDTA2, IFxDTB1, IFxDTB2)

	addr+0	addr+1	addr+2	addr+3
IFx Message Data A1 (addresses 20 <sub>H</sub> & 50 <sub>H</sub> )	Data(0)	Data(1)	–	–
IFx Message Data A2 (addresses 22 <sub>H</sub> & 52 <sub>H</sub> )	–	–	Data(2)	Data(3)
IFx Message Data B1 (addresses 24 <sub>H</sub> & 54 <sub>H</sub> )	Data(4)	Data(5)	–	–
IFx Message Data B2 (addresses 26 <sub>H</sub> & 56 <sub>H</sub> )	–	–	Data(6)	Data(7)
IFx Message Data A2 (addresses 30 <sub>H</sub> & 60 <sub>H</sub> )	Data(3)	Data(2)	–	–
IFx Message Data A1 (addresses 32 <sub>H</sub> & 62 <sub>H</sub> )	–	–	Data(1)	Data(0)
IFx Message Data B2 (addresses 34 <sub>H</sub> & 64 <sub>H</sub> )	Data(7)	Data(6)	–	–
IFx Message Data B1 (addresses 36 <sub>H</sub> & 66 <sub>H</sub> )	–	–	Data(5)	Data(4)



## ■ Function of Registers

- Setting of transmission message data

The setting data is transmitted in order of MSB (bit7 and bit15), Data(0), Data(1), ..., Data(7).

- Reception message data

The reception message data is stored in order of MSB (bit7 and bit15), Data(0), Data(1), ..., Data(7).

---

Notes:

- When reception message data is less than 8 bytes, undefined data is written to the remaining byte of data register.
  - Since transfer to the message object is performed using 4-byte unit of Data A or Data B, a part of 4-byte data cannot be updated.
-

## 15.4.3 Message Object

There are 32 message objects in the message RAM. To avoid conflicts between CPU access to the message RAM and access from CAN controller, the CPU cannot directly access the message objects. These accesses are handled via the IFx message interface registers.

This section explains the configuration and function of message object.

### ■ Configuration of Message Object

Message object

UMask	Msk28 to Msk0	MXtd	MDir	EoB	New Dat		MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
MsgVal	ID28 to ID0	Xtd	Dir	DLC3 to DLC0	Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7

Note:

Message objects are not initialized by Init bit of CAN control register and hardware reset. For the hardware reset, the CPU must initialize the message RAM or set the MsgVal bit of message RAM to "0" after release of hardware reset.

### ■ Function of Message Object

ID28 to ID0, Xtd and Dir bits are used for ID and the kind of message when message is transmitted. They are used in the reception filter with Msk28 to Msk0, MXtd and MDir bits when message is reception.

Data or remote frame which passed the reception filter is stored in message object. Xtd indicated extended or standard frame. When Xtd is "1", 29-bit ID (extended frame) is received, and when it is "0", 11-bit ID (standard frame) is received. When the received data frame or the remote frame matches more than one message object, the frame is stored in the smallest message number of the message object. For detail, refer to the reception filter of received message in Section "15.5.3 Message Reception Operation".

MsgVal: Enable message bit

MsgVal	Function
0	Message object is invalid. Message is not transmitted and received.
1	Message object is enabled. Message can be transmitted and received.

## Notes:

- Be sure to initialize MsgVal bit of the message object before resetting Init bit of the CAN control register (CTRLR) to "0" and before changing ID28 to ID0, Xtd, Dir, DLC3 to DLC0.
- When MsgVal bit is set to "0" during the transmission, TxOk bit of CAN status register (STATR) is set to "1" after the transmission is completed. However, since the message object and TxRqst bit of CAN transmission request register (TREQR) are not cleared to "0", clear TxRqst bit to "0" using the message interface register.

UMask: Reception mask enable bit

UMask	Function
0	Msk28 to Msk0, MXtd and MDir are not used.
1	Msk28 to Msk0, MXtd and MDir are used.

## Notes:

- Change the UMask bit when Init bit of CAN control register is "1" or MsgVal bit is "0".
- When Dir bit is "1" and RmtEn bit is "0", the operation is different by setting of UMask.
  - If UMask is "1" when the acceptance filter passed and the remote frame is received, TxRqst bit is reset to "0". At that time, the received ID, IDE, RTR, and DLC store in the message object, NewDat bit is set to "1", and data is not changed (handled the same as data frame).
  - If UMask is "0", a value of TxRqst bit is kept as it is and the remote frame is ignored due to reception of the remote frame.

ID28 to ID0: Message ID

ID	Function
ID28 to ID0	29-bit ID (extended frame) is indicated.
ID28 to ID18	11-bit ID (standard frame) is indicated.

Msk28 to Msk0: ID mask

Msk	Function
0	The bit corresponding to ID of message object is masked.
1	The bit corresponding to ID of message object is not masked.

Xtd: Extended ID enable bit

Xtd	Function
0	11-bit ID (standard frame) is used for message object.
1	29-bit ID (extended frame) is used for message object.

MXtd: Extended ID mask bit

MXtd	Function
0	Extended ID bit (IDE) is masked in reception filter.
1	Extended ID bit (IDE) is not masked in reception filter.

---

Note:

If 11-bit ID (standard frame) is set in message object, ID of received data frame is written to ID28 to ID18. Msk28 to Msk18 are used for ID mask.

---

Dir: Message direction bit

Dir	Function
0	Reception direction is indicated. When TxRqst is set to "1", remote frame is transmitted. When TxRqst is set to "0", data frame which has passed the reception filter is received.
1	Transmission direction is indicated. When TxRqst is set to "1", data frame is transmitted. When TxRqst is set to "0" and RmtEn is set to "1", CAN controller sets TxRqst to "1" itself by reception of remote frame which has passed the reception filter.

MDir: Message direction mask bit

MDir	Function
0	Message direction bit (Dir) is masked by reception filter.
1	Message direction bit (Dir) is not masked by reception filter.

---

Note:

MDir bit is always set to "1".

---

EoB: End of buffer bit (For detail, refer to Section "15.5.4 Function of FIFO Buffer".)

EoB	Function
0	Message object is used as FIFO buffer and is not the last message.
1	Single message object or last message object of FIFO buffer is indicated.

---

Notes:

- EoB bit is used to configure the FIFO buffer of 2 to 32 messages.
  - Single message object (when not using FIFO) should be set "1" to EoB bit.
- 

NewDat: Data renewal bit

NewDat	Function
0	Enable data is not existed.
1	Enable data is existed.

MsgLst: Message lost

MsgLst	Function
0	Message lost is not generated.
1	Message lost is generated.

---

Note:

MsgLst bit is enabled only when Dir bit is "0" (reception direction).

---

RxIE: Reception interrupt flag enable bit

RxIE	Function
0	After the frame is received successfully, IntPnd is not changed.
1	After the frame is received successfully, IntPnd is set to "1".

TxE: Transmission interrupt flag enable bit

TxE	Function
0	After the frame is transmitted successfully, IntPnd is not changed.
1	After the frame is transmitted successfully, IntPnd is set to "1".

IntPnd: Interrupt pending bit

IntPnd	Function
0	There is no interrupt source.
1	There is interrupt source. If there is no other interrupt with high priority, IntId bit of CAN interrupt register indicates the message object.

RmtEn: Remote enable

RmtEn	Function
0	TxRqst is not changed by reception of remote frame.
1	When Dir bit is "1" and remote frame is received, TxRqst is set to "1".

---

Note:

When Dir bit is "1" and RmtEn bit is "0", the operation is different by the setting of UMask.

- If UMask is "1" when the acceptance filter passed and the remote frame is received, TxRqst bit is reset to "0". At that time, the received ID, IDE, RTR, and DLC store in the message object, NewDat bit is set to "1", and data is not changed (handled the same as data frame).
  - If Umask is "0", a value of TxRqst bit is kept as it is and the remote frame is ignored due to reception of the remote frame.
- 

TxRqst: Transmission request bit

TxRqst	Function
0	The transmission idle state (neither transmitting nor transmission wait state)
1	Transmitting or transmission wait state



## DLC3 to DLC0: Data length code

DLC	Function
0 to 8	Data frame length is 0 to 8 bytes.
9 to 15	Setting is prohibited. If setting, it becomes 8-byte length.

## Note:

When the data frame is received, received DLC is stored in DLC bit.

## Data0 to Data7: Data0 to Data7

Data	Function
Data0	The first data byte of CAN data frame
Data1	The second data byte of CAN data frame
Data2	The third data byte of CAN data frame
Data3	The fourth data byte of CAN data frame
Data4	The fifth data byte of CAN data frame
Data5	The sixth data byte of CAN data frame
Data6	The seventh data byte of CAN data frame
Data7	The eighth data byte of CAN data frame

## Notes:

- Serial output to CAN bus is outputted from MSB (bit7 or bit15).
- When the reception message data is less than 8 bytes, undefined data is written to the remaining byte of data register.
- Since transfer to the message object is performed using 4-byte unit of Data A or Data B, a part of 4-byte data cannot be updated.

## 15.4.4 Message Handler Register

---

**All message handler registers are read-only. TxRqst, NewDat, IntPnd, and MsgVal bits of each message object and the Intld bit are status information.**

---

### ■ Message Handler Register

- CAN transmission request registers 1 and 2 (TREQR1, TREQR2)
- CAN data renewal registers 1 and 2 (NEWDT1, NEWDT2)
- CAN interrupt pending registers 1 and 2 (INTPND1, INTPND2)
- CAN message enable registers 1 and 2 (MSGVAL1, MSGVAL2)

### 15.4.4.1 CAN Transmission Request Registers (TREQR1, TREQR2)

These registers show TxRqst bit of all the message object. Reading TxRqst bit can check which transmission request of message object is pending.

#### ■ CAN Transmission Request Registers (TREQR1, TREQR2)

CAN transmission request register 2 (Upper)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
Base+80 <sub>H</sub>	TxRqst32 to TxRqst25								00000000 <sub>B</sub>
	R	R	R	R	R	R	R	R	

CAN transmission request register 2 (Lower)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
Base+81 <sub>H</sub>	TxRqst 24 to TxRqst17								00000000 <sub>B</sub>
	R	R	R	R	R	R	R	R	

CAN transmission request register 1 (Upper)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
Base+82 <sub>H</sub>	TxRqst 16 to TxRqst9								00000000 <sub>B</sub>
	R	R	R	R	R	R	R	R	

CAN transmission request register 1 (Lower)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
Base+83 <sub>H</sub>	TxRqst 8 to TxRqst1								00000000 <sub>B</sub>
	R	R	R	R	R	R	R	R	

R: Read only

## ■ Function of Registers

TxRqst32 to TxRqst1: Transmission request bits

TxRqst	Function
0	The transmission idle state (neither transmitting nor transmission wait state)
1	Transmitting or transmission wait state

Set/reset condition of TxRqst bit is shown in the following.

- Set condition
  - When WR/RD and TxRqst of IFx command mask register are set to "1", TxRqst of specific object can be set by writing to IFx command request register.
  - When WR/RD and TxRqst of IFx command mask register are set to "1" and "0" respectively, and TxRqst of IFx message control register is set to "1", TxRqst of specific object can be set by writing to IFx command request register.
  - When Dir and RmtEn bits are set to "1", this bit is set by reception of the remote frame which has passed the reception filter.
- Reset condition
  - When WR/RD and TxRqst of IFx command mask register are set to "1" and "0" respectively, and TxRqst of IFx message control register is set to "0", TxRqst of specific object can be reset by writing to IFx command request register.
  - When frame transmission is completed normally, this bit is reset.
  - When Dir is "1", RmtEN is "0", and Umask is "1", this bit is reset by reception of the remote frame which has passed the reception filter.

### 15.4.4.2 CAN Data Renewal Registers (NEWDT1, NEWDT2)

These registers show NewDat bit of all the message object. Reading NewDat bit can check which data of message object is renewed.

■ CAN Data Renewal Registers (NEWDT1, NEWDT2)

CAN data renewal register 2 (Upper)									
Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
Base+90 <sub>H</sub>	NewDat32 to NewDat25								00000000 <sub>B</sub>
	R	R	R	R	R	R	R	R	
CAN data renewal register 2 (Lower)									
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
Base+91 <sub>H</sub>	NewDat24 to NewDat17								00000000 <sub>B</sub>
	R	R	R	R	R	R	R	R	
CAN data renewal register 1 (Upper)									
Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
Base+92 <sub>H</sub>	NewDat16 to NewDat9								00000000 <sub>B</sub>
	R	R	R	R	R	R	R	R	
CAN data renewal register 1 (Lower)									
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
Base+93 <sub>H</sub>	NewDat8 to NewDat1								00000000 <sub>B</sub>
	R	R	R	R	R	R	R	R	
R:	Read only								

## ■ Function of Registers

NewDat32 to NewDat1: Data renewal bits

NewDat32 to NewDat1	Function
0	Enable data is not existed.
1	Enable data is existed.

Set/reset condition of NewDat bit is shown below.

- Set condition
  - When WR/RD of IFx command mask register and NewDat of IFx message control register are set to "1", specific object of this bit can be set by writing to IFx command request register.
  - This bit is set by reception of the data frame which has passed the reception filter.
  - When Dir is "1", RmtEN is "0", and UMask is "1", this bit is set by reception of the remote frame which has passed the reception filter.
- Reset condition
  - When WR/RD and NewDat of IFx command mask register are set to "0" and "1" respectively, NewDat of specific object can be reset by writing to IFx command request register.
  - When WR/RD of IFx command mask register is set to "1", and NewDat of IFx message control register is set to "0", NewDat of specific object can be reset by writing to IFx command request register.
  - After data is transferred to transmission shift register (internal register), this bit is reset.

### 15.4.4.3 CAN Interrupt Pending Registers (INTPND1, INTPND2)

These registers show IntPnd bit of all the message object. Reading IntPnd bit can check which data of message object is interrupt pending.

#### ■ CAN Interrupt Pending Registers (INTPND1, INTPND2)

CAN interrupt pending register 2 (Upper)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
Base+A0 <sub>H</sub>	IntPnd32 to IntPnd25								00000000 <sub>B</sub>
	R	R	R	R	R	R	R	R	

CAN interrupt pending register 2 (Lower)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
Base+A1 <sub>H</sub>	IntPnd24 to IntPnd17								00000000 <sub>B</sub>
	R	R	R	R	R	R	R	R	

CAN interrupt pending register 1 (Upper)

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
Base+A2 <sub>H</sub>	IntPnd16 to IntPnd9								00000000 <sub>B</sub>
	R	R	R	R	R	R	R	R	

CAN interrupt pending register 1 (Lower)

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
Base+A3 <sub>H</sub>	IntPnd8 to IntPnd1								00000000 <sub>B</sub>
	R	R	R	R	R	R	R	R	

R: Read only

## ■ Function of Registers

IntPnd32 to IntPnd 1: Interrupt pending bits

IntPnd	Function
0	Interrupt sources are not existed.
1	Interrupt sources are existed.

The set/reset condition of IntPnd bit is shown below.

- Set condition
  - When TxIE is set to "1", this bit is set after transmission of the frame is completed normally.
  - When RxIE is set to "1", this bit is set after reception of the frame which has passed the reception filter is completed normally.
- Reset condition
  - When IFx command mask register sets WR/RD and ClrIntPnd to "1", IntPnd of specific object can be reset by writing to IFx command request register.



### 15.4.4.4 CAN Message Enable Registers (MSGVAL1, MSGVAL2)

These registers show MsgVal bit of all the message object. Reading MsgVal bit can check which data of message object is enabled.

■ CAN Message Enable Registers (MSGVAL1, MSGVAL2)

CAN message enable register 2 (Upper)									
Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
Base+B0 <sub>H</sub>	MsgVal32 to MsgVal25								00000000 <sub>B</sub>
	R	R	R	R	R	R	R	R	
CAN message enable register 2 (Lower)									
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
Base+B1 <sub>H</sub>	MsgVal24 to MsgVal17								00000000 <sub>B</sub>
	R	R	R	R	R	R	R	R	
CAN message enable register 1 (Upper)									
Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
Base+B2 <sub>H</sub>	MsgVal16 to MsgVal9								00000000 <sub>B</sub>
	R	R	R	R	R	R	R	R	
CAN message enable register 1 (Lower)									
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
Base+B3 <sub>H</sub>	MsgVal8 to MsgVal1								00000000 <sub>B</sub>
	R	R	R	R	R	R	R	R	
R:	Read only								

## ■ Function of Registers

MsgVal32 to MsgVal1: Message enable bits

MsgVal	Function
0	Message object is invalid. Transmission/reception of message is not performed.
1	Message object is enabled. Transmission/reception of message is enabled.

Set/reset condition of MsgVal bit is shown below.

- Set condition
  - When MsgVal of IFx arbitration register 2 is set to "1", MsgVal of specific object can be set by writing to IFx command request register.
- Reset condition
  - When MsgVal of IFx arbitration register 2 is set to "0", MsgVal of specific object can be reset by writing to IFx command request register.

## 15.4.5 CAN Clock Prescaler Register

This register defines the division ratio of the clock supplied to CAN interface. When changing the value of this register, set the initialization bit (Init) of CAN control register (CTRLR) to "1" and stop all the bus operation.

### ■ CAN Clock Prescaler Register

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
01A8 <sub>H</sub>	-	-	-	-	CANPRE3	CANPRE2	CANPRE1	CANPRE0	00000000 <sub>B</sub>
	R	R	R	R	R/W	R/W	R/W	R/W	

R/W: Readable/Writable  
R: Read only

### ■ Function of Registers

[bit15 to bit12] - : Reserved bits

These bits read "0" and write is not affected on registers.

[bit11 to bit8] CAN clock prescaler setting bits

CANPRE[3:0]	Function
0000	1/2 cycle of system clock is selected as CAN clock. (Initial value: CANPRE[3:0]=0000)
0001	
001x	1/4 cycle of system clock is selected as CAN clock.
01xx	1/8 cycle of system clock is selected as CAN clock.
1000	2/3 cycle of system clock is selected as CAN clock. Duty of clock is 67% [Setting disabled]
1001	1/3 cycle of system clock is selected as CAN clock.
101x	1/6 cycle of system clock is selected as CAN clock.
11xx	1/12 cycle of system clock is selected as CAN clock.

Notes:

- Change the CAN clock prescaler setting bits after the initialization bit of CAN control register is set to "1" and all the bus operation is stopped.
- This register sets the clock supplied to CAN interface to 16 MHz or less.
- CANPRE[3:0]=1000 must not be set.

## 15.5 C-CAN Function

---

**This section explains the operation and function of CAN controller.**

---

The following functions are explained.

- Message Object
- Message Transmission Operation
- Message Reception Operation
- Function of FIFO Buffer
- Function of Interrupt
- Bit Timing
- Test mode
- Software Initialization
- CAN Clock Prescaler

## 15.5.1 Message Object

---

**Message object and interface of message RAM are explained.**

---

### ■ Message Object

Message object setting of message RAM (except for MsgVal, NewDat, IntPnd and TxRqst bits) is not initialized by hardware reset. So, initialize the message object by CPU or set the MsgVal bit invalid (MsgVal=0). Set the CAN bit timing register also when Init bit of CAN control register is "0".

Message object setting is specified in the message interface registers (IFx mask register, IFx arbitration register, IFx message control register, and IFx data register). When a message number is written into the IFx command request register, the data of the corresponding interface register is transferred to the specified message object.

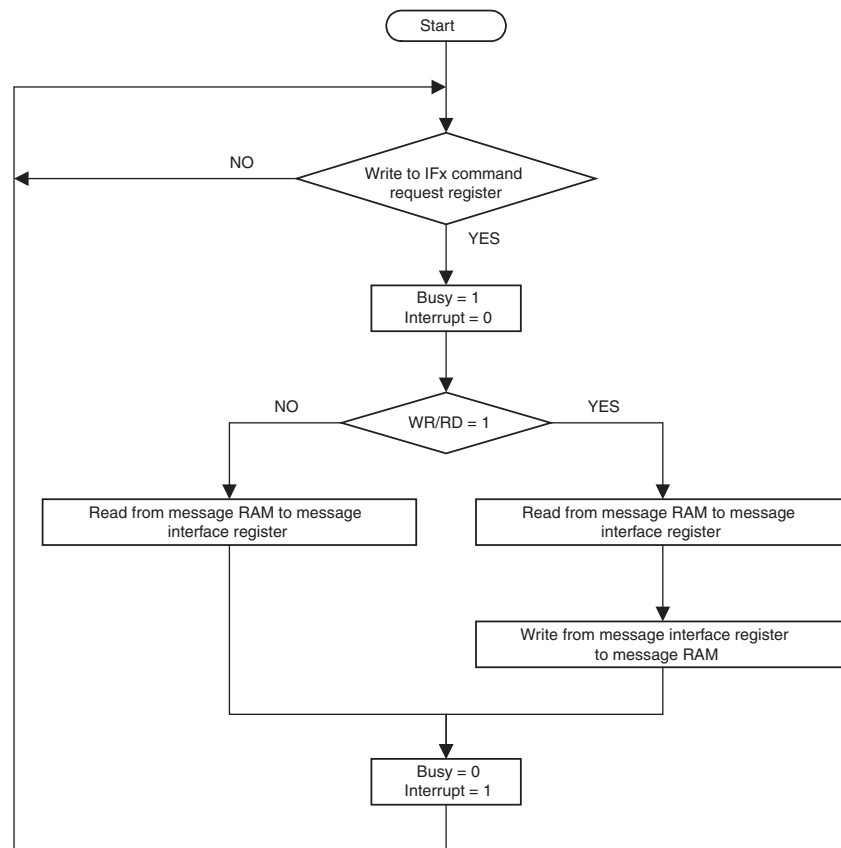
When the Init bit of the CAN control register is cleared to "0", the CAN controller starts operation. The reception messages which passed through the reception filter are stored in the message RAM. The message for which transmission request is pending are transferred from the message RAM to the shift register of the CAN controller, and then sent to the CAN bus.

The CPU reads the reception messages via the message interface register, and updates the transmission messages. An interrupt to the CPU occurs according to the settings of the CAN control register and IFx message control register (message object).

### ■ Message RAM and Data Transmission/Reception

When the data transfer between the message interface register and the message RAM starts, the Busy bit of the IFx command request register is set to "1". When the data transfer is complete, the BUSY bit is cleared to "0". (Refer to Figure15.5-1.)

The IFx command mask register specifies whether to transfer all data of one message object or to transfer partial data. Due to the structure of the message RAM, it is impossible to write a single bit/byte of a message object. All data of a message object is always written to the message RAM. Consequently, the data transfer from the message interface register to the message RAM executes the read-modify-write cycle.

**Figure15.5-1 Data Transfer of Message Interface Register and Message RAM**

## 15.5.2 Message Transmission Operation

Setting procedure of transmission message object and transmission operation are explained.

### ■ Message Transmission

If there is no data transfer between the message interface register and message RAM, MsgVal bit of CAN message enable register and TxRqst bit of CAN transmission request register are evaluated. While the transmission request is pending, the valid message object with the highest priority is transferred to the transmission shift register. At that time, NewDat bit of the message object is reset to "0".

After the transmission is completed normally if no new data was written to the message object (NewDat="0"), TxRqst bit will be reset to "0". If TxIE is set to "1", IntPnd bit is set to "1" after the transmission is performed successfully. CAN controller has lost the arbitration on the CAN bus, or if an error occurred during the transmission, the message will be retransmitted as soon as the CAN bus becomes idle.

### ■ Transmission Priority Level

The transmission priority for the message objects is determined by the message number. Message object 1 has the highest priority while message object 32 (the highest implemented message object number) has the lowest priority. If more than one transmission request is pending, the transfer is performed in the order of smallest number for the corresponding message object.

### ■ Setting of Transmission Message Object

Table 15.5-1 shows the initialization procedure of transmission object.

**Table 15.5-1 Initialization of Transmission Message Object**

MsgVal	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
1	appl.	appl.	appl.	1	1	0	0	0	appl.	0	appl.	0

IFx arbitration registers (ID28 to ID0 and Xtd bits) are given by the application. They define ID of the transmission message and type of message.

When the standard frame (11-bit ID) is specified, ID28 to ID18 are used, but ID17 to ID0 are invalid. When the extended frame (29-bit ID) is specified, ID28 to ID0 are used.

When setting TxIE bit to "1", set IntPnd bit to "1" after successful transmission of the message object. When setting RmtEn bit to "1", set TxRqst bit to "1" after reception of matched remote frame and transmit the data frame automatically.

Data registers (DLC3 to DLC0 and Data0 to Data7) are set by the application.

When Umask=1, IFx mask register (Msk28 to Msk0, Umask, MXtd, and Mdir bits) receives the remote frame that has grouped ID due to mask setting, and then it is used to enable the transmission (set "1" to TxRqst bit).

For details, refer to the remote frame in Section "15.5.3 Message Reception Operation".

---

**Note:**

Setting the Dir bit of IFx mask register to mask enable is disabled.

---

**■ Updating a Transmission Message Object**

The CPU can update the data of the transmission message object via the message interface register.

The data of the transmission message object is written in the unit of 4 bytes of the corresponding IFx data register (IFx data register A or IFx data register B). Therefore, you cannot change only one byte of the transmission message object.

To update 8-byte data only, first write 0087<sub>H</sub> into the IFx command mask register. Then, write a message number into the IFx command request register. The data of the transmission message object is updated (8-byte data) and "1" is written to the TxRqst bit at the same time.

To send the data immediately following the message number which is being sent, set the TxRqst and NewDat bits to "1". By this, the TxRqst bit is not reset to "0" and continuous transmission is enabled.

When NewDat bit is set to "1" together with TxRqst bit, NewDat bit will be reset to "0" as soon as the new transmission has started.

---

**Notes:**

- To update data, write the data in the unit of 4 bytes of IFx data register A or IFx data register B.
  - To update data only, set the NewDat and TxRqst bits to "1".
-



## 15.5.3 Message Reception Operation

---

**Setting procedure of reception message object and reception operation are explained.**

---

### ■ Reception Filter of Reception Message

When the arbitration/control field of the message (ID + IDE + RTR + DLC) is completely shifted to the shift register for the CAN controller reception, the scan of the message RAM starts to compare it with valid message objects to find a match.

During the scan, the arbitration field and mask data (including MsgVal, UMask, NewDat, and EoB) are loaded from the message object of the message RAM, and the arbitration fields of the message object and the shift register are compared including the mask data.

This operation is repeated until "the match of the arbitration fields of the message object and the shift register is found" or "the operation reaches the last word of the message RAM". When a match is found, the scan of the message RAM stops, and the CAN controller starts processing according to the type of the reception frame (data frame or remote frame).

### ■ Reception Priority Level

The reception priority level of the message objects is determined from the message numbers. Message object 1 has the highest priority, and message object 32 (the largest message object number being implemented) has the lowest priority. Consequently, when two or more messages match at the reception filter, the message object with a smaller message number is received.

### ■ Data Frame Reception

The CAN controller transfers the reception message from the shift register to the message RAM of the message object which is found as a match at the reception filter, and stores the message in it. The data to be stored is not only data bytes, but also all the arbitration fields and data length codes. The same is true even when the IFx mask register is set for masking. (The data is stored to retain the ID and data bytes.)

When new data is received, the NewDat bit is set to "1". When the CPU reads a message object, the NewDat bit must be reset to "0". If the NewDat bit is already set to "1" when a message is received, the preceding data is assumed to be lost, and the MsgLst bit is set to "1".

When the RxIE bit is set to "1" and a message buffer is received, the IntPnd bit of the CAN interrupt pending register is set to "1". When this happens, the TxRqst bit of the corresponding message object is reset to "0". The purpose of this is to prohibit transmission processing when a request data frame is received during the transmission processing of a remote frame.

## ■ Remote Frame

The following three types of processing are available when a remote frame is received. The processing is selected according to the setting of the message object which was found to be a match.

- 1) Dir=1 (Transmission direction), RmtEn=1, UMask=1 or 0

The remote frame which was found to be a match is received. Only the TxRqst bit of this message object is set to "1". The automatic reply (transmission) of the data frame is performed for the remote frame. (The message objects other than the TxRqst bit are not changed.)

- 2) Dir=1 (Transmission direction), RmtEn=0, UMask=0

The remote frame is not received even when it matches with the message object. The remote frame is considered to be invalid. (The TxRqst bit of this message object is not changed.)

- 3) Dir=1 (Transmission direction), RmtEn=0, UMask=1

If the received remote frame matches with the message object, the TxRqst bit of the message object is reset to "0", and the remote frame is processed as if it is a reception data frame. The received arbitration field and control field (ID + IDE + RTR + DLC) are stored in the message object of the message RAM, and the NewDat bit of this message object is set to "1". The data field of the message object is not changed.

## ■ Setting of Reception Message Object

Table 15.5-2 shows the initialization procedure of the reception message object.

**Table 15.5-2 Initialization of Reception Message Object**

MsgVal	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
1	appl.	appl.	appl.	1	0	0	0	appl.	0	0	0	0

The IFx arbitration register (ID28 to ID0, Xtd bit) is provided by the application. It defines the reception message ID and message type used for the reception filter.

When a standard frame (11-bit ID) is set, ID28 through ID18 are used; ID17 through ID0 are invalid. When a standard frame is received, ID17 through ID0 are reset to "0". When an expansion frame (29-bit ID) is set, ID28 through ID0 are used.

When the RxIE bit is set to "1" and a reception data frame is stored in the message object, the IntPnd bit is set to "1".

The data length code (DLC3 to DLC0) is provided by the application. When the CAN controller stores a reception data frame into the message object, it stores the reception data length code and the 8-byte data. If the data length code is less than 8, indeterminate data is written as the rest of the message object data.

When Umask=1, the IFx mask register (Msk28 to Msk0, UMask, MXtd, and MDir bits) is used to enable the received data frame having the ID grouped by the mask setting. For details, refer to "Data Frame Reception" in Section "15.5.3 Message Reception Operation".

---

Note:

Mask setting of Dir bit in IFx mask register is disabled.

---

## ■ Process of Reception Message

The CPU can read reception messages any time via the message interface register. In normal times, it writes "007F<sub>H</sub>" into the IFx command mask register. Then, it writes the message number of the message object into the IFx command request register. By this, the reception message of the specified message number is transferred from the message RAM to the message interface register. During this process, the NewDat and IntPnd bits of the message object can be cleared to "0" by setting the IFx command mask register. As for the process of the reception message, the message is received when it is found to be a match by the reception filter. If the message object uses masking by the reception filter, the data set to be masked is removed from the reception filter, and whether to receive the message or not is determined.

The NewDat bit indicates whether a new message is received after the last message object was read. The MsgLst bit indicates that the preceding data is lost because the next reception data is received before the received data was read from the message object. The MsgLst bit is not reset automatically.

When a matching data frame is received by the reception filter during the transmission processing of a remote frame, the TxRqst bit is automatically reset to "0".

## 15.5.4 Function of FIFO Buffer

---

**Configuration and operation of FIFO buffer for message object in the reception message process are explained.**

---

### ■ Configuration of FIFO Buffer

With the exception of the EoB bit, the configuration of receive message objects which belong to a FIFO buffer is the same as the configuration of a receive message object (Refer to the setting of reception message object in Section "15.5.3 Message Reception Operation").

Use FIFO buffer to concatenate two or more reception message objects. To store the reception message to this FIFO buffer, setting of ID and mask for the reception message object must be matched if they are used. First reception message object of FIFO buffer is equal to the lower message number with high priority. Last reception message object of FIFO buffer needs to indicate end of FIFO buffer block after "1" is set to EoB bit ("0" is set to the bit except last message object of message object which uses the configuration of FIFO buffer.

---

Notes:

- Setting of ID and mask for message object used for FIFO buffer must be the same setting.
  - Be sure to set "1" to EoB bit when FIFO buffer is not used.
- 

### ■ Message Reception by FIFO Buffer

If the reception message matches with ID of FIFO buffer, the message is stored to the reception message object for FIFO buffer of the smallest message number.

When the message is stored to the reception message object for FIFO buffer, NewDat bit of this reception message object is set to "1". When EoB bit sets NewDat bit to the reception message object of "0", the reception message object is protected until it reaches to last reception message object (EoB bit = 1).

The CAN controller does not write to FIFO buffer at this time.

Unless "0" is written to NewDat bit of the reception message object with the state which valid data is stored up to last FIFO buffer (release of write protection), next message to be received is written to last message object and the message is overwritten.

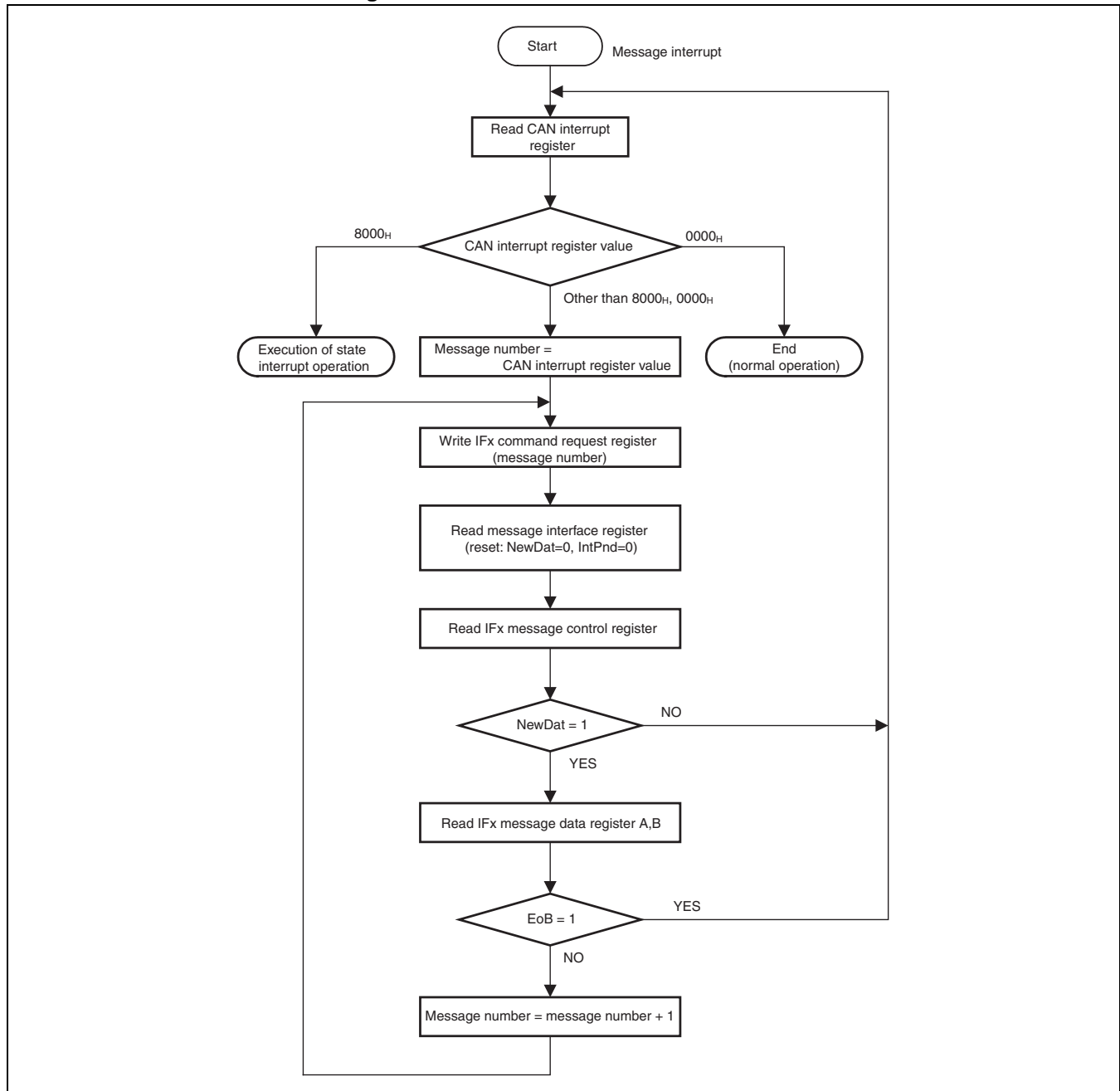
## ■ Read from FIFO Buffer

By writing the reception message number to IFx command request register, the reception message number is transferred to message interface register and the content of the reception message object can be read by CPU. At this time, when setting WR/RD of IFx command mask register to "0" (read) and setting TxRqst/NewDat and IntPnd to 1, reset NewDat and IntPnd bits to "0".

To guarantee the function of FIFO buffer, be sure to read the reception message object of FIFO buffer from the smallest message number.

Figure15.5-2 shows procedure for CPU processing of the message object concatenated by FIFO buffer.

**Figure15.5-2 CPU Process of FIFO Buffer**



## 15.5.5 Function of Interrupt

---

**The interrupt process by status interrupt (IntId=8000<sub>H</sub>) and message interrupt (IntId=the number of message) is explained.**

---

When two or more interrupts are pending, the CAN interrupt register indicates the pending interrupt code of the highest priority. The time order specified to the interrupt codes is ignored. The interrupt code of the highest priority is always indicated. The interrupt code is retained until it is cleared by the CPU. The status interrupt (IntId bit = 8000) has the highest priority.

The priority of the message interrupt is set higher for the message with a smaller message number, and is set lower for the message with a larger message number. The message interrupt is cleared by clearing the IntPnd bit of the message object. The status interrupt is cleared by reading the CAN status register.

The IntPnd bit of the CAN interrupt pending register indicates whether an interrupt is present or not. If there is no pending interrupt, the IntPnd bit indicates "0". When the IntPnd bit is set to "1" while the IE bit of the CAN control register and the TxIE and RxIE bits of the IFx message control register are "1", an interrupt signal to the CPU becomes active. The interrupt signal is retained to be active until the CAN interrupt pending register is cleared to "0" (interrupt factor reset), or until the IE bit of the CAN control register is reset to "0".

The CAN interrupt register set to 8000<sub>H</sub> indicates that the CAN status register was updated by the CAN controller. This interrupt has the highest priority. The interrupt by updating the CAN status register can be used for the control of enabling or disabling the setting of the CAN interrupt register by using the EIE and SIE bits of the CAN control register. The interrupt signal to the CPU can be controlled with the IE bit of the CAN control register.

The RxOk, TxOk, and LEC bits of the CAN status register can be updated (reset) by writing values from the CPU. The writing, however, cannot set or reset an interrupt.

The CAN interrupt register set to the values other than 8000<sub>H</sub> or 0000<sub>H</sub> indicates that the message interrupt is pending, and shows the pending message interrupt which has a high priority.

The CAN interrupt register is updated even when the IE bit is reset.

The cause of the message interrupt to the CPU can be identified by checking the CAN interrupt register or the CAN interrupt pending register. (Refer to Section "15.4.4 Message Handler Register".) When clearing the message interrupt, you can read the message data simultaneously. If you clear the message interrupt indicated by the CAN interrupt register, the interrupt with the second highest priority is set in the CAN interrupt register and waits for the next interrupt processing. If there is no interrupt, the CAN interrupt register indicates 0000<sub>H</sub>.

---

**Notes:**

- Status interrupt (IntId=8000<sub>H</sub>) is cleared by read access of CAN status register.
  - Status interrupt (IntId=8000<sub>H</sub>) by write access of CAN status register does not occur.
-

## 15.5.6 Bit Timing

The overview for the bit timing and the bit timing in CAN controller are explained.

Each CAN node of the CAN network has its own clock oscillator, usually a quartz crystal oscillator. The time parameter of the bit time can be configured individually for each CAN node, creating a common bit rate even though the CAN nodes' oscillator periods ( $f_{OSC}$ ) may be different.

The frequencies of these oscillators are different from normal value due to change in temperature or voltage and deterioration of components. As long as the variations remain inside a specific oscillator tolerance range ( $df$ ), the CAN nodes are able to compensate for the different bit rates by resynchronizing to the bit stream.

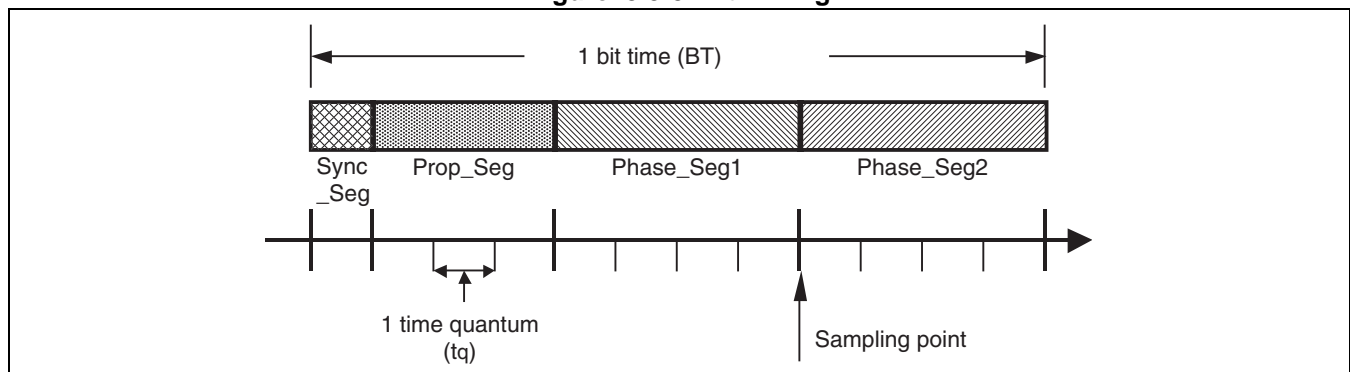
According to the CAN specification, the bit time is divided into four segments (see Figure15.5-3). The Synchronization Segment (Sync-Seg), the Propagation Time Segment (Prop-Seg), the Phase Buffer Segment 1 (Phase-Seg1), and the Phase Buffer Segment 2 (Phase-Seg2). Each segment consists of a specific, programmable number of time quanta (see Table 15.5-3). The length of the time quantum ( $t_q$ ), which is the basic time unit of the bit time, is defined by the CAN controller's system clock  $f_{sys}$  and the Baud Rate Prescaler (BRP) :  $t_q = BRP / f_{sys}$ . The CAN's system clock  $f_{sys}$  is the frequency of Clock input (see Figure15.2-1).

The Synchronization Segment Sync\_Seg is timing of the bit time where edges of the CAN bus level are expected to occur. The Propagation Time Segment Prop\_Seg is intended

to compensate for the physical delay times within the CAN network. The Phase Buffer Segments

Phase\_Seg1 and Phase\_Seg2 specify the Sampling Point. Resynchronization Jump Width (SJW) defines move width of sampling point at the time of resynchronization in order to compensate for edge phase errors.

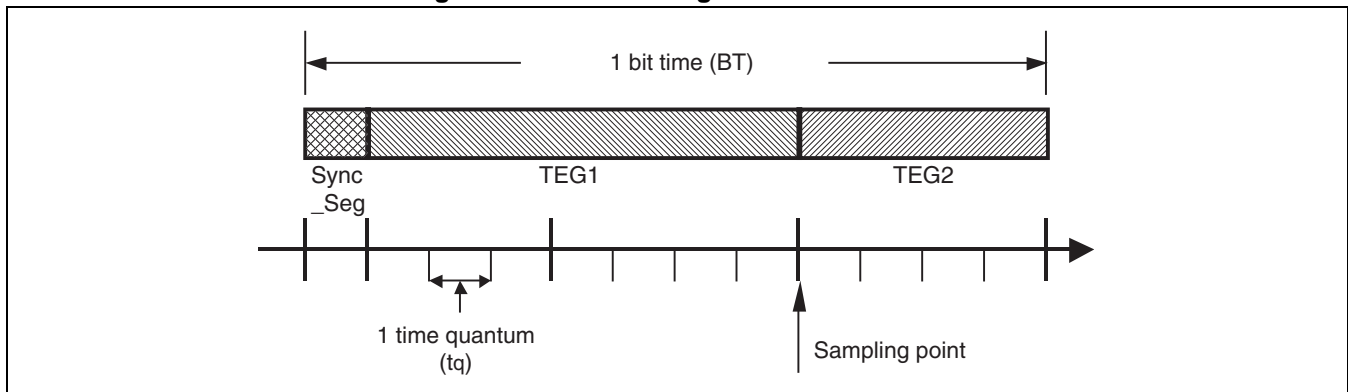
**Figure15.5-3 Bit Timing**



**Table 15.5-3 Parameter of CAN Bit Time**

Parameter	Range	Function
BRP	[1 to 32]	Defines the length of the time quantum $t_q$
Sync_Seg	1 $t_q$	Fixed length, synchronization to system clock
Prop_Seg	[1 to 8] $t_q$	Compensates for the physical delay times
Phase_Seg1	[1 to 8] $t_q$	Guarantee of edge phase error before sampling point May be lengthened temporarily by synchronization
Phase_Seg2	[1 to 8] $t_q$	Guarantee of edge phase error after sampling point May be shortened temporarily by synchronization
SJW	[1 to 4] $t_q$	Resynchronization jump width May not be longer than either phase buffer segment

Bit timing of CAN controller is shown in the following.

**Figure15.5-4 Bit Timing of CAN Controller**



**Table 15.5-4 Parameter of CAN Controller**

Parameter	Range	Function
BRPE,BRP	[0 to 1023]	Defines the length of the time quantum tq Prescaler can be extended up to 1024 by the bit timing register and the prescaler extended register.
Sync_Seg	1 tq	Synchronization to system clock Fixed length
TSEG1	[1 to 15] tq	Segment time before sampling point Equal to Prop_Seg and Phase_Seg1 Can be controlled by bit timing register
TSEG2	[0 to 7] tq	Segment time after sampling point Equal to Phase_Seg2 Can be controlled by bit timing register
SJW	[0 to 3] tq	Resynchronization jump width Can be controlled by bit timing register

Relationship of each parameter is shown below.

$$tq = ([BRPE, BRP] + 1) / f_{sys}$$

$$BT = SYNC\_SEG + TEG1 + TEG2$$

$$= (1 + (TSEG1 + 1) + (TSEG2 + 1)) \times tq$$

$$= (3 + TSEG1 + TSEG2) \times tq$$

## 15.5.7 Test mode

Setting procedure and operation of test mode are explained.

### ■ Test Mode Setting

The test mode is entered by setting Test bit in the CAN control register to "1". In test mode, the Tx1, Tx0, LBack, Silent and Basic bits in the CAN test register are enabled.

All test register functions are disabled when Test bit in the CAN control register is reset to "0".

### ■ Silent Mode

The CAN controller can be set in silent mode by setting Silent bit of CAN test register to "1".

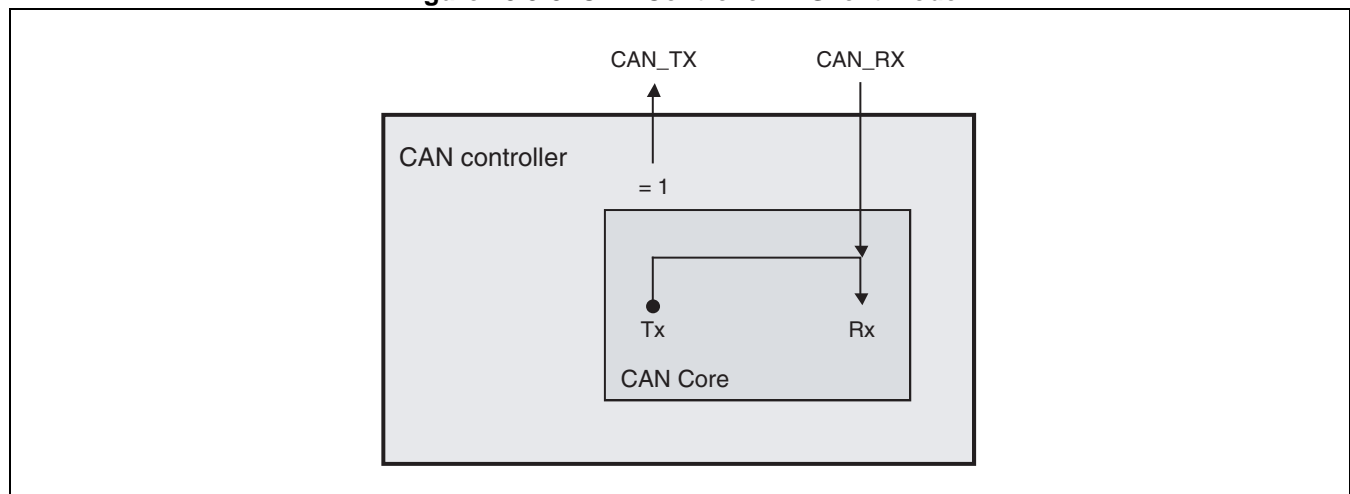
In silent mode, the CAN is able to receive valid data frames and valid remote frames, but it sends only recessive on the CAN bus and it cannot start a transmission of message and ACK.

If the CAN controller is required to send a dominant bit (ACK bit, overload flag, active error flag), it is transmitted at RX side with loop circuit in the CAN controller. This operation causes the loop dominant bit to be transmitted in the CAN controller to receive at the receive side even in recessive state on the CAN bus.

The silent mode can be used to analyze the traffic on a CAN bus without affecting it by the transmission of dominant bits (ACK bits, error flags).

Figure 15.5-5 shows the connection of signals CAN\_TX and CAN\_RX to the CAN controller in silent mode.

**Figure 15.5-5 CAN Controller in Silent Mode**



## ■ Loop Back Mode

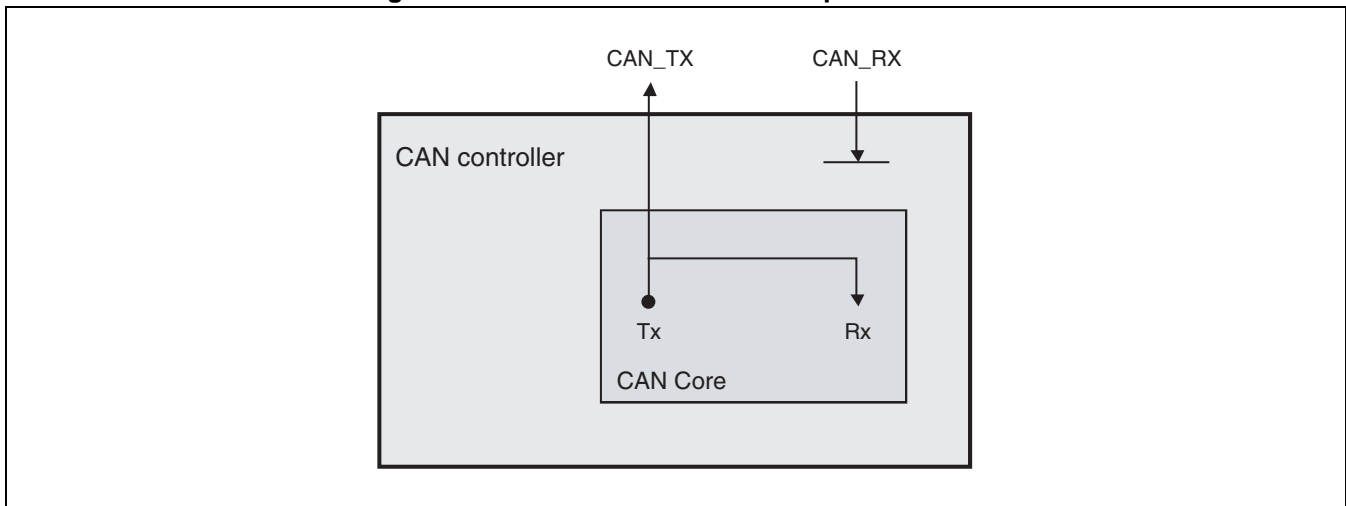
The CAN controller can be set in loop back mode by setting LBack bit of CAN test register to "1".

Loop back mode can be used for self-test function.

In loop back mode, TX side and RX side are connected in the internal CAN controller, the CAN controller treats its own transmitted messages as received messages in RX side and stores them (if they pass acceptance filter) into a receive buffer.

Figure 15.5-6 shows the connection of signals CAN\_TX and CAN\_RX to the CAN controller in loop back mode.

**Figure 15.5-6 CAN Controller of Loop Back Mode**



---

**Note:**

To be independent from external signal, the dominant bit in the acknowledge slot of a data/remote frame is not sampled. CAN controller usually generates acknowledge error, but the CAN controller ignores acknowledge errors in this test mode.

---

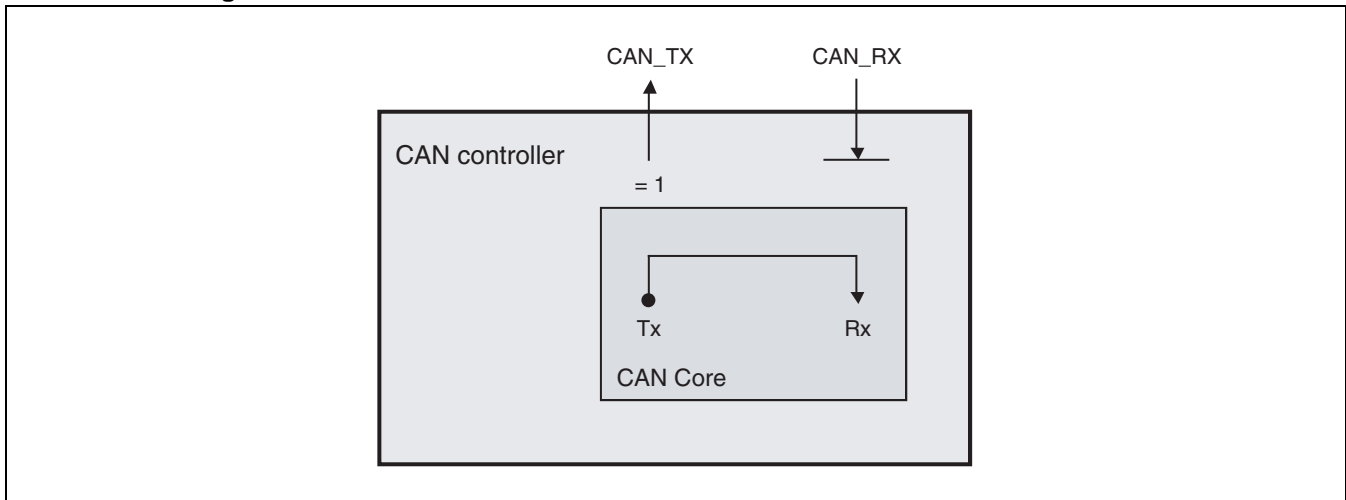
## ■ Combination of Silent Mode and Loop Back Mode

It is also possible to combine loop back mode and silent mode and to operate by setting LBack and Silent bits on the CAN test register to "1" at the same time.

This mode can be used for a "Hot Self test" when the CAN controller tests in loop back mode, this test has no effect on operation of the CAN system because fixed output of recessive to CAN\_TX pin and input from CAN\_RX pin are invalid.

Figure 15.5-7 shows the connection of signals CAN\_TX and CAN\_RX to the CAN controller in case of the combination of loop back mode with silent mode.

**Figure 15.5-7 CAN Controller Combined LOOP Back Mode with Silent Mode**



## ■ Basic Mode

The CAN controller can be set in basic mode by setting basic bit of CAN test register to "1".

In this mode, the CAN controller runs without the message RAM.

The IF1 message interface registers are used for transmit control.

When the message is transmitted, set the content sent to IF1 message interface register. Then, the transmission is request by setting Busy bit of IF1 command request register to "1". When Busy bit is set to "1", IF1 message interface register is locked, or the transmission is pending.

When "1" is set to Busy bit, CAN controller operates as follows.

As soon the CAN bus is bus idle, the content of the IF1 message interface registers are loaded into the transmission shift register and the transmission is started. When the transmission has completed normally, the Busy bit is reset to "0" and the locked IF1 message interface registers are released.

A pending transmission can be aborted at any time by resetting "0" to the Busy bit in the IF1 command request register. If the CPU has reset the Busy bit to "0" during transmission, a possible retransmission in case of arbitration lost or in case of an error is disabled.

The IF2 message interface registers are used for receive control.

All messages are received without using the acceptance filter. The contents of the received message can be read by setting Busy bit of IF2 command request register to "1".

When "1" is set to Busy bit, CAN controller operates as follows.

The received message (content of reception shift register) is stored to IF2 message interface register

without acceptance filter.

When new message is stored in IF2 message interface register, the CAN controller sets NewDat bit to "1". Also, when the bit is "1", the CAN controller sets MsgLst to "1" if another new message is received.

---

### Notes:

- In basic mode, the setting of all the message object related to control and status bits and of the control mode of the IFx command mask registers is disabled.
- The message number of the command request registers is invalid.
- The NewDat and MsgLst bits of the IF2 message control register operates the same as normal operation, DLC3-to-DLC0 will show the received DLC, the other control bits will be read as "0".

---

### ■ Software Control of Pin CAN\_TX

Four output functions are available for the CAN transmit pin CAN\_TX.

- Serial data output (normal output)
- CAN sampling point signal output to monitor the bit timing of CAN controller
- Dominant fixed output
- Recessive fixed output

Fixed output of dominant and recessive has CAN\_RX monitoring function of CAN receive pin and can be used to check the CAN bus's physical layer.

The output mode of CAN\_TX pin can be controlled by setting Tx1 and Tx0 bits of the CAN test register.

---

### Note:

CAN\_TX must be set to serial data output when CAN message transfer or any of the loop back mode, silent mode, or basic mode are selected.

---

## 15.5.8 Software Initialization

---

### Initialization by software is described.

---

The initialization factors in software is shown below.

- Hardware reset
- Setting of Init bit in CAN control register
- Transmission to bus off state

Hardware reset initializes all the message objects other than message RAM (except for MsgVal, NewDat, IntPnd and TxRqst bits). After hardware reset, message RAM should be initialized by the CPU or be set the MsgVal of message RAM to "0". Also, when setting the bit timing register, set the register before clearing Init bit of CAN control register to "0".

Init bit of CAN control register is set to "1" as the following conditions.

- Write "1" from CPU
- Hardware reset
- Bus off

While Init bit is set to "1", all the message transmission/reception from and to the CAN bus is stopped, the CAN\_TX pin of CAN bus output is recessive output (except for CAN\_TX test mode).

When Init bit is set to "1", error counter and register are not changed.

Setting to the bit timing register and to the prescaler extension register for the baud rate control are enabled when both Init and CCE bits in the CAN control register are set to "1".

The software initialization is ended by resetting the Init bit to "0". Only the access from CPU can set Init bit to "0".

The message is transferred after synchronizing with data transfer on the CAN bus when Init bit is reset to "0" until occurrence of consecutive 11-bit recessive (=bus idle) is waited.

Change the mask of a message object, ID, xTd, EoB, RmtEm during normal operation after disabling MsgVal.

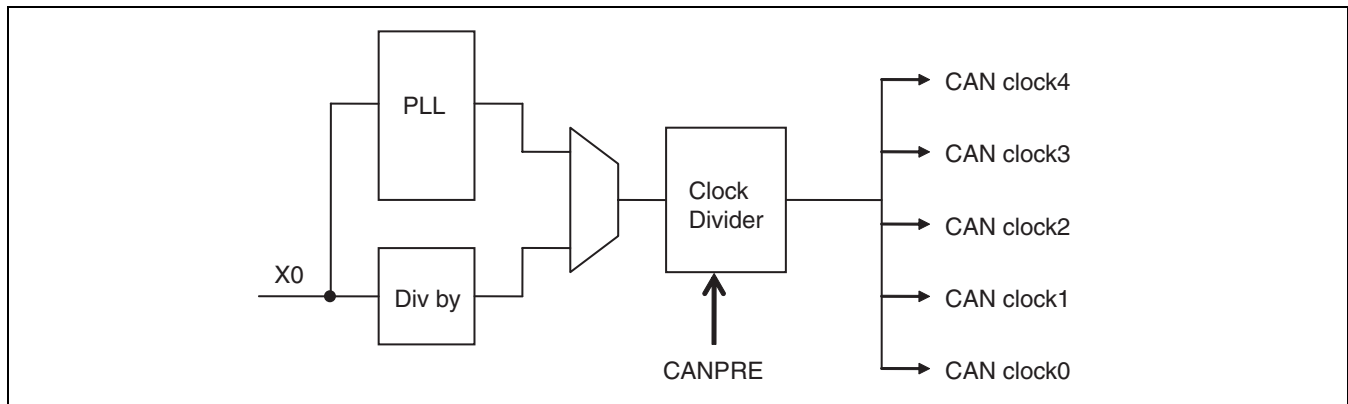
## 15.5.9 CAN Clock Prescaler

Changing of CAN clock in PLL operating is explained.

### ■ Block Diagram

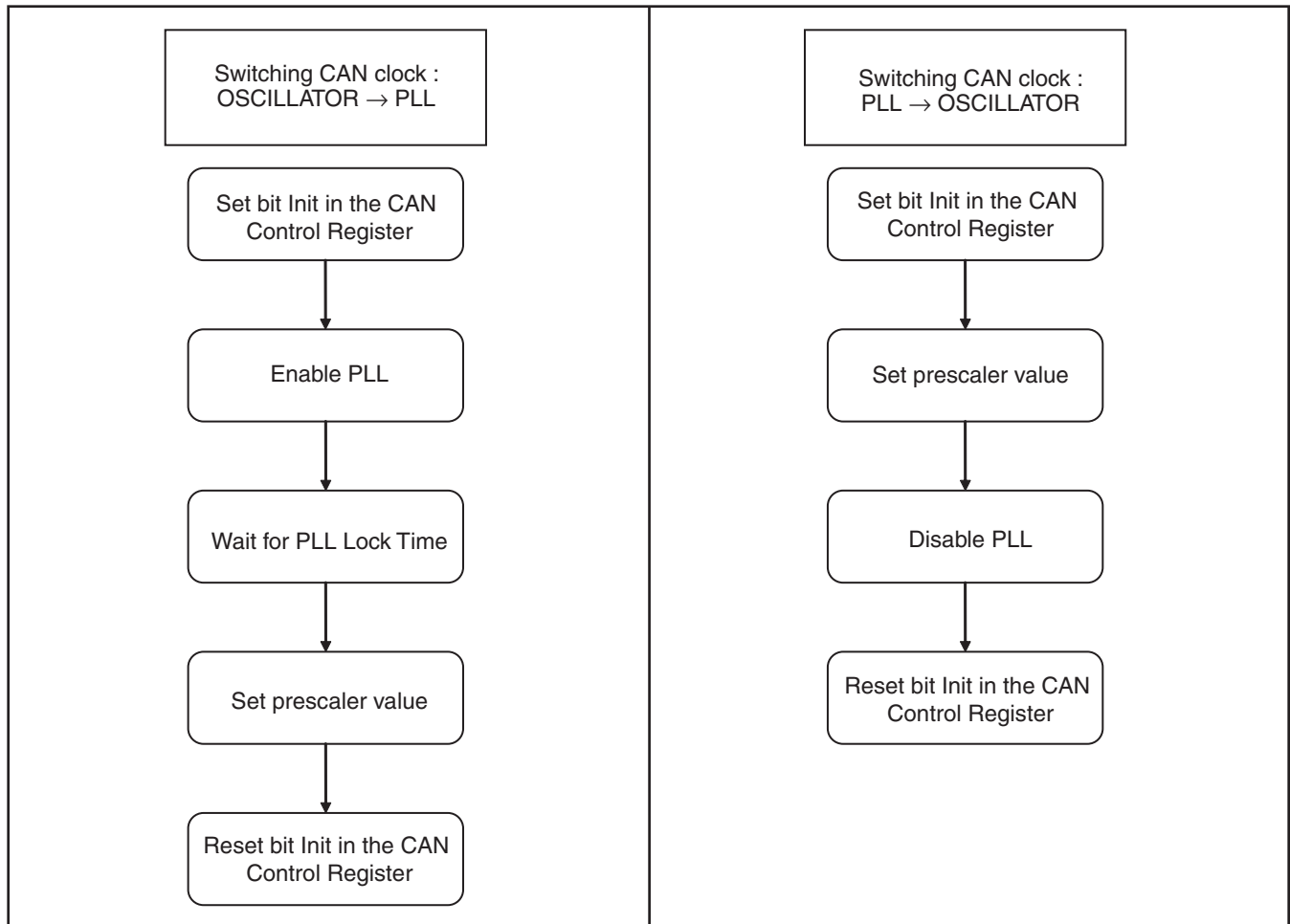
The overview of CAN clock prescaler is shown in the following block diagram.

The division ratio of clock supplied to CAN interface is determined according to setting of CANPRE bit of CAN clock prescaler register.



## ■ Procedure of Changing Clock

For the procedure of changing clock using CAN clock prescaler, the following is recommended.





## ■ CAN Clock Prescaler Setting

The settable value is indicated in CAN clock prescaler.

Clock supplied to CAN interface is the same as the clock which divides the system clock according to the setting value of CAN clock prescaler.

CANPRE[3:0]	Function	System clock : at 32MHz
0000	1/2 cycle of system clock is selected as CAN clock. (Initial value: CANPRE[3:0]=0000)	16MHz
0001		
001x	1/4 cycle of system clock is selected as CAN clock.	8MHz
01xx	1/8 cycle of system clock is selected as CAN clock.	4MHz
1000	2/3 cycle of system clock is selected as CAN clock. Duty of clock is 67%	21.33MHz (setting disabled)
1001	1/3 cycle of system clock is selected as CAN clock.	10.67MHz
101x	1/6 cycle of system clock is selected as CAN clock.	5.33MHz
11xx	1/12 cycle of system clock is selected as CAN clock.	2.67MHz

### Notes:

- Change the CAN clock prescaler setting bits after the initialization bit of CAN control register is set to "1" and all the bus operation is stopped.
- The clock supplied to CAN interface is set to 16 MHz or less by setting of this register.
- CANPRE[3:0]=1000 is disabled to set.

# **CHAPTER 16**

---

## ***8/10-BIT A/D CONVERTER***

**This chapter explains the overview of the 8/10-bit A/D converter, the configuration and functions of registers, and 8/10-bit A/D converter operation.**

- 16.1 Overview
- 16.2 Configuration
- 16.3 Pin
- 16.4 Register
- 16.5 Interrupt
- 16.6 Operation Explanation
- 16.7 A/D Conversion Data Protection Function
- 16.8 Notes on Using 8/10-bit A/D Converter

## 16.1 Overview

---

**The 8/10-bit A/D converter has a feature to convert analog input voltage to a 10-bit or 8-bit digital value, using the RC successive comparison/conversion method. The input signal is selected from individual analog input pins. The conversion trigger can be selected from among three options: software, internal clock, and external pin triggers.**

---

### ■ Functions of the 8/10-bit A/D Converter

There is an A/D conversion feature to convert analog voltage (input voltage) input to the analog input pins into digital values. This feature has the following benefits.

- The conversion time is a minimum of 1.2 $\mu$ s (including the sampling time using the 33MHz machine clock).
- The conversion method used is the RC sequential compare conversion method with sample hold circuit.
- A resolution of 10 or 8 bits can be selected.
- The analog input pin is program-selectable.
- After A/D conversion is completed, an interrupt request is generated.
- No data are lost in the interrupt enable status since the conversion data protection function works even in the continuous conversion.
- One of the following conversion activation causes can be selected: software, 16-bit reload timer 1 or multifunctional timer (rising edge), or external pin triggers (falling edge).

There are three conversion modes.

**Table 16.1-1 Conversion Modes of 8/10-bit A/D Converter**

Conversion mode	Single conversion operation	Scanning conversion operation
Single-shot conversion mode	The specified channel (1 channel only) is converted for one time and terminated.	Continuous plural channels (maximum 7 channels can be specified) are converted for one time and terminated.
Continuous conversion mode	Repeatedly convert specified channel (1 channel only).	Repeatedly convert multiple channels (up to 7 channels can be specified) in succession.
Pause-conversion mode	The specified channel (1 channel only) is converted for one time and suspended until the next start.	Continuous plural channels (maximum 7 channels can be specified) are converted. However, one channel is converted and suspended until the next start.

Notes:

- This model has two units: Unit 1, which has 4 analog-input channels and unit 2, which has 7 analog-input channels.
  - If unit 2 is enabled to start with the multifunction timer, it cannot be started with reload timer 1.
-

## 16.2 Configuration

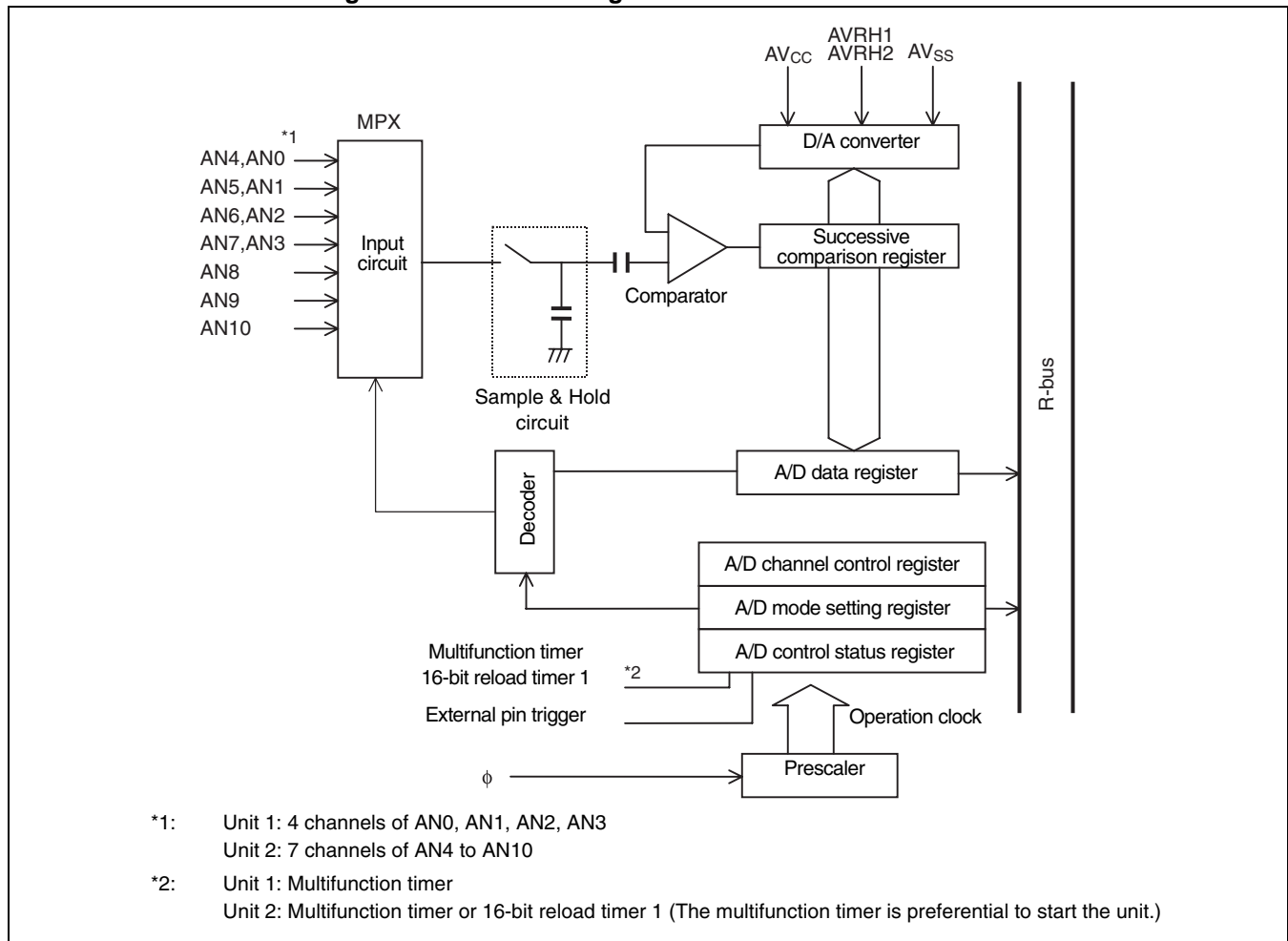
---

The 8/10-bit A/D converter is made up of the following 11 blocks.

- A/D control status registers (ADCS)
  - A/D channel control register (ADCH)
  - A/D mode setting register (ADMD)
  - A/D data register (ADCD)
  - Clock selector (input clock selector for activation of A/D conversion)
  - Decoder
  - Analog channel selector
  - Sample hold circuit
  - D/A converter
  - Comparator
  - Control circuit
-

### ■ Block Diagram of 8/10-bit A/D Converter

**Figure 16.2-1 Block Diagram of 8/10-bit A/D Converter**



- A/D control status registers (ADCS)

Features are available to suspend and confirm conversion, enable/disable interrupt requests, confirm the status of interrupt requests, and select the A/D conversion resolution.

- A/D channel control register (ADCH)

There is a feature to select the A/D channel.

- A/D mode setting register (ADMD)

There is a feature to select a conversion mode and to set the A/D conversion compare time and sampling time.

- A/D data register (ADCD)

This register stores A/D conversion results.

- Clock selector

This is an A/D conversion activation clock selector. 16-bit reload timer ch.1 output, multifunctional timer, and external pin trigger can be selected as the activation clock.

- Unit 2 can be started with 16-bit reload timer ch.1 output or with the multifunction timer.
- Unit 1 can be activated via multifunctional timer.

**● Decoder**

The A/D channel control registers (ADCH) ANE0 to ANE2 and ANS0 to ANS2 bit settings are a circuit to select the analog input pin to use.

**● Analog channel selector**

This block is the circuit to select the pin to be used from among analog input pins.

- Unit 1 has 4 analog inputs.
- Unit 2 has 7 analog inputs.

**● Sample hold circuit**

This circuit holds the input voltage selected by the analog channel selector. The input voltage can be converted without affected by the input voltage fluctuation in the A/D conversion (in the comparison) by holding the sample of input voltage immediately after starting the A/D conversion.

**● D/A converter**

The reference voltage is generated to compare the held sample of input voltage.

**● Comparator**

This compares the input voltage for which sample hold is performed, with the output voltage of the D/A converter to determine which is the greater of the two.

**● Control circuit**

The signal from the comparator (higher or lower) determines the A/D conversion value. When the A/D conversion is terminated, the conversion result is stored in the A/D data register (ADCR) and the interrupt request is generated.

## 16.3 Pin

This section lists the pins of the 8/10-bit A/D converter and provides their block diagram.

### ■ Pins of 8/10-bit A/D Converter

The A/D converter pin serves dual use as a generic port. Table 16.3-1 shows pin functions, I/O formats, and settings when using the 8/10-bit A/D converter.

**Table 16.3-1 Pins of 8 /10-bit A/D Converter**

Functions	Pin Name	Pin Function	I/O Type	Pull-up setting	Standby control	I/O port required for pin setting
ch.0	P50/AN0	Port 5 I/O / analog input	CMOS output/ CMOS hysteresis input or analog input	None	Yes	Input setting of port 5 (DDR5:bit0 to bit7=0) Set to analog input (AICR1: bit0 to bit3=1)
ch.1	P51/AN1					
ch.2	P52/AN2					
ch.3	P53/AN3					
ch.4	P54/AN4					(AICR2: bit0 to bit3=1)
ch.5	P55/AN5					
ch.6	P56/AN6					
ch.7	P57/AN7					
ch.8	P44/AN8	Port 4 I/O / analog input				Input setting of port 4 (DDR4:bit4 to bit6=0) Set to analog input (AICR2:bit4 to bit6=1)
ch.9	P45/AN9					
ch.10	P46/AN10					





# 16.4 Register

This section lists the registers of the 8/10-bit A/D converter.

■ Registers of 8/10-bit A/D Converter

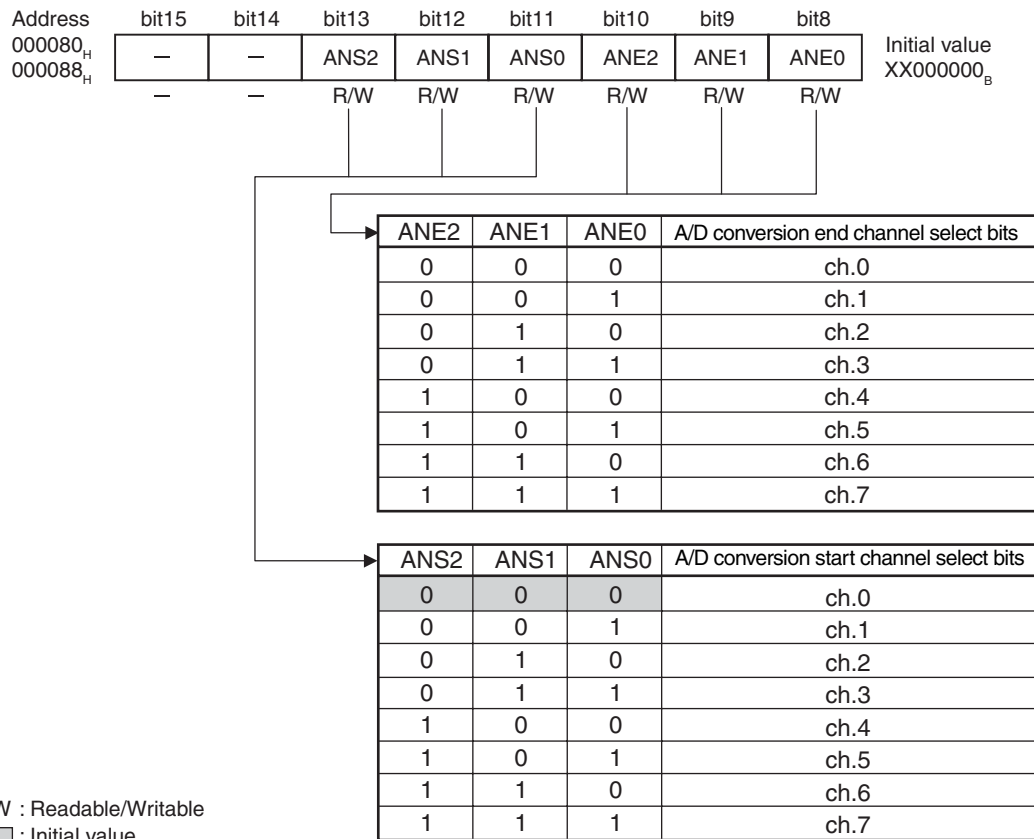
Figure 16.4-1 Registers of 8/10-bit A/D Converter

	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
000086 <sub>H</sub>	AICR1															
00008E <sub>H</sub>	AICR2															
000080 <sub>H</sub>	ADCH1								ADMD1							
000084 <sub>H</sub>	ADCS1															
000082 <sub>H</sub>	ADCD11								ADCD10							
000088 <sub>H</sub>	ADCH2								ADMD2							
00008C <sub>H</sub>	ADCS2															
00008A <sub>H</sub>	ADCD21								ADCD20							

## 16.4.1 A/D Channel Control Register (ADCH)

The A/D channel control register has a feature to select the A/D conversion channel.

### ■ A/D Channel Control Register (ADCH: ADCH1, ADCH2)



Notes:

- A/D unit 1 is ch.0 to ch.3 = AN0 to AN3 (ch.4 to ch.7 are empty)
- A/D unit 2 is ch.0 to ch.6 = AN4 to AN10 (ch.7 is empty)
- Always write "0" to ANE2 and ANS2 of A/D unit 1.
- For A/D unit 2, neither ANE0 - ANE2=1 nor ANS0 - ANS2=1 can be set.
- Be sure to set A/D units 1 and 2 so that ANS0 is less than or equal to ANE0.

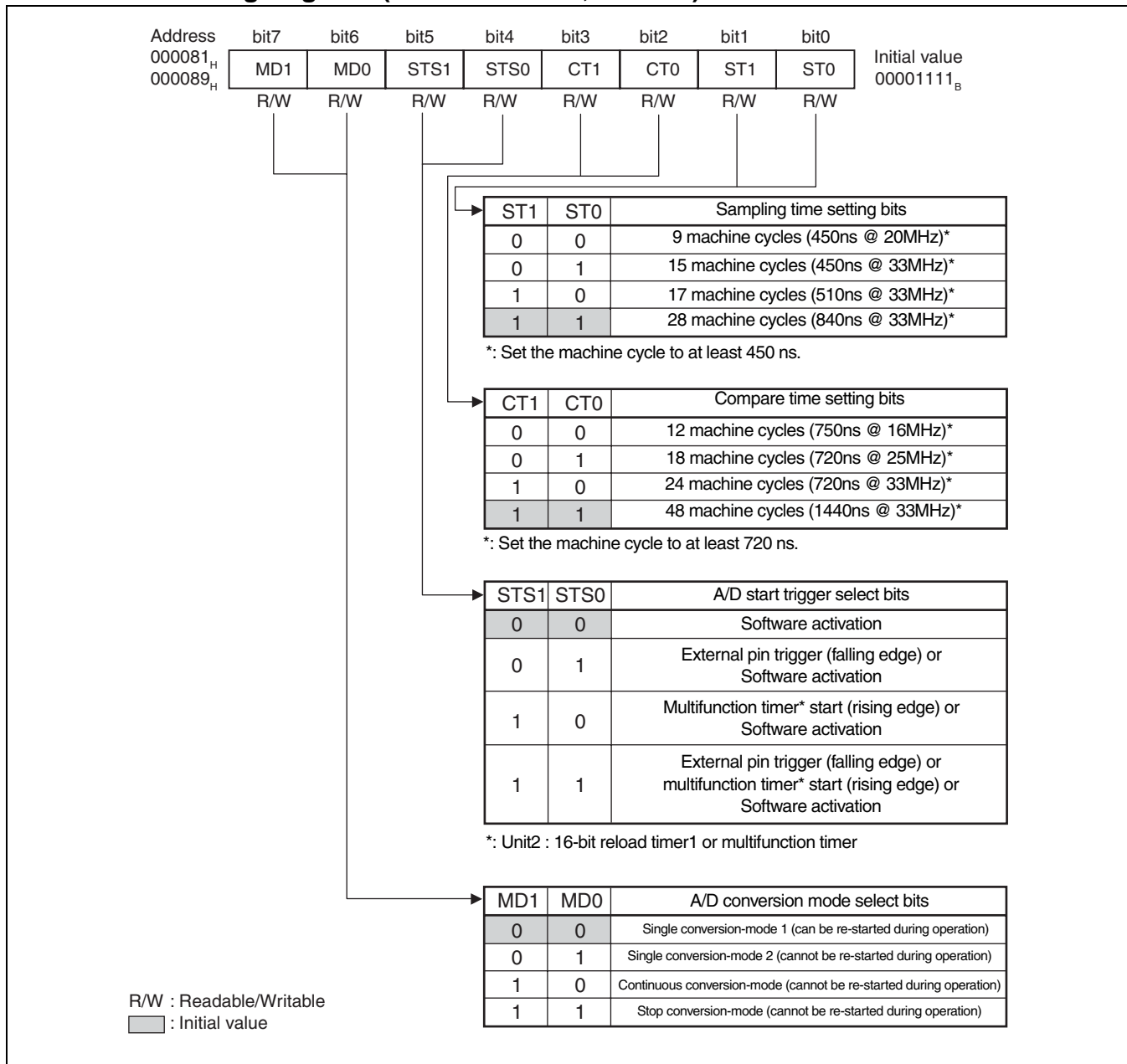
**Table 16.4-1 The Functions of Each of the Bits of the A/D Channel Control Register (ADCH)**

Bit name		Functions
bit15, bit14	Undefined bits	<ul style="list-style-type: none"> <li>The read value is indeterminate.</li> <li>Writing to these bits has no effect on the operation.</li> </ul>
bit13 to bit11	ANS2,ANS1, ANS0: A/D conversion start channel selection bits	<ul style="list-style-type: none"> <li>These bits set the start channel of the A/D conversion and indicate the channel numbers under A/D conversion during conversion operation.</li> <li>When A/D conversion is activated, A/D conversion starts from the channels written to these bits.</li> <li>Channel numbers under the conversion can be read during the A/D conversion. The channel number converted immediately before can be read during the suspension in the stop conversion mode.</li> </ul>
bit10 to bit8	ANE2,ANE1, ANE0: A/D conversion end channel selection bits	<ul style="list-style-type: none"> <li>These bits set the end channel of the A/D conversion.</li> <li>When started, A/D conversion is performed up to the channel selected with these bits.</li> <li>When the same channels with ANS2 to ANS0 are set, only those channels are converted. If continuous conversion mode or stop conversion mode is set when the conversion up to the channels specified in these bits is completed, conversion returns to the start channel set in ANS2 to ANS0. The conversion end channel set here must always be equal to or greater than the conversion start channel.</li> <li>After setting the start channel to the A/D conversion start channel selection bits (ANS2, ANS1, and ANS0), do not set neither the A/D conversion mode selection bits (MD1 and MD0) nor the A/D conversion end channel selection bits (ANE2, ANE1, and ANE0) by the read-modify-write type instruction. The last conversion channel is read from the ANS2, ANS1, and ANS0 bits until the A/D conversion operating starts. Therefore, when the MD1, MD0, ANE2, ANE1, and ANE0 bits are set by the read-modify-write type instruction after setting the start channel to the ANS2, ANS1, and ANS0 bits, the value of ANE2, ANE1, and ANE0 bits may be rewritten.</li> </ul>

## 16.4.2 A/D Mode Setting Register (ADMD)

The A/D mode setting register has a feature to select a conversion mode and to set the A/D conversion compare time and sampling time.

### ■ A/D Mode Setting Register (ADMD: ADMD1, ADMD2)



**Table 16.4-2 Description of the Functions of the A/D Mode Setting Register (ADMD) Bits (1 / 2)**

Bit name	Functions
bit7, bit6	<p data-bbox="521 306 1182 333">• These bits are used for selecting the A/D conversion mode.</p> <p data-bbox="521 352 1468 451">• Single-conversion mode 1, single-conversion mode 2, continuous-conversion mode, or stop-conversion mode is selected in accordance with the values in the MD1 and MD0 bits.</p> <p data-bbox="521 470 997 497">• The operation in each mode is as follows:</p> <p data-bbox="558 516 846 543">Single-conversion mode 1:</p> <p data-bbox="634 554 1468 653">Consecutively performs A/D conversion between the channel specified by ANS2 to ANS0 and the channel specified by ANE2 to ANE0 only once. The converter can be restarted during operation.</p> <p data-bbox="558 672 846 699">Single-conversion mode 2:</p> <p data-bbox="634 709 1468 808">Consecutively performs A/D conversion between the channel specified by ANS2 to ANS0 and the channel specified by ANE2 to ANE0 only once. The converter cannot be restarted during operation.</p> <p data-bbox="558 827 878 854">Continuous-conversion mode:</p> <p data-bbox="634 865 1468 963">Repeatedly performs A/D conversion between the channel specified by ANS2 to ANS0 and the channel specified by ANE2 to ANE0 until operation is forcibly stopped with the BUSY bit. The converter cannot be restarted during operation.</p> <p data-bbox="558 982 805 1010">Stop-conversion mode:</p> <p data-bbox="634 1020 1468 1232">Repeatedly performs A/D conversion between the channel specified by ANS2 to ANS0 and the channel specified by ANE2 to ANE0, temporarily stopping A/D conversion on a per-channel basis, until it is forcibly stopped by the BUSY bit. The converter cannot be restarted during operation. A temporarily stopped A/D conversion is restarted in accordance with the start source selected with the STS1 and STS0 bits.</p> <p data-bbox="521 1251 591 1278">Notes:</p> <ul data-bbox="521 1289 1468 1619" style="list-style-type: none"> <li data-bbox="521 1289 1468 1356">• Restart disable of each conversion mode (single, continuous and stop) is applied to all starting of timer, external trigger, and software.</li> <li data-bbox="521 1375 1468 1442">• Be sure to rewrite these bits while A/D operation is stopped before the conversion operation is performed.</li> <li data-bbox="521 1461 1468 1619">• When A/D conversion mode selection bit (MD1,MD0) is set to "00<sub>B</sub>", restart can be operating during A/D conversion. Only software start (STS1,STS0=00<sub>B</sub>) can be set in this mode. Please restart according to the following procedure. <ol data-bbox="581 1562 1208 1619" style="list-style-type: none"> <li data-bbox="581 1562 846 1589">1. Clear the INT bit to 0.</li> <li data-bbox="581 1591 1208 1619">2. Write START bit to 1 and INT bit to 0 at the same time.</li> </ol> </li> </ul> <p data-bbox="558 1629 1468 1686">When A/D conversion mode selection bit (MD1,MD0) is set to "01<sub>B</sub>", restart can not be operating during the A/D conversion.</p> <p data-bbox="558 1696 1468 1820">When restarting and termination of the A/D conversion occur at the same time, the A/D conversion is terminated without restarting. And, value of 300<sub>H</sub> is stored in data register (ADCR1/ADCR0). Therefore, please use restart so that neither the A/D conversion restarting nor the terminating may occur at the same time.</p>

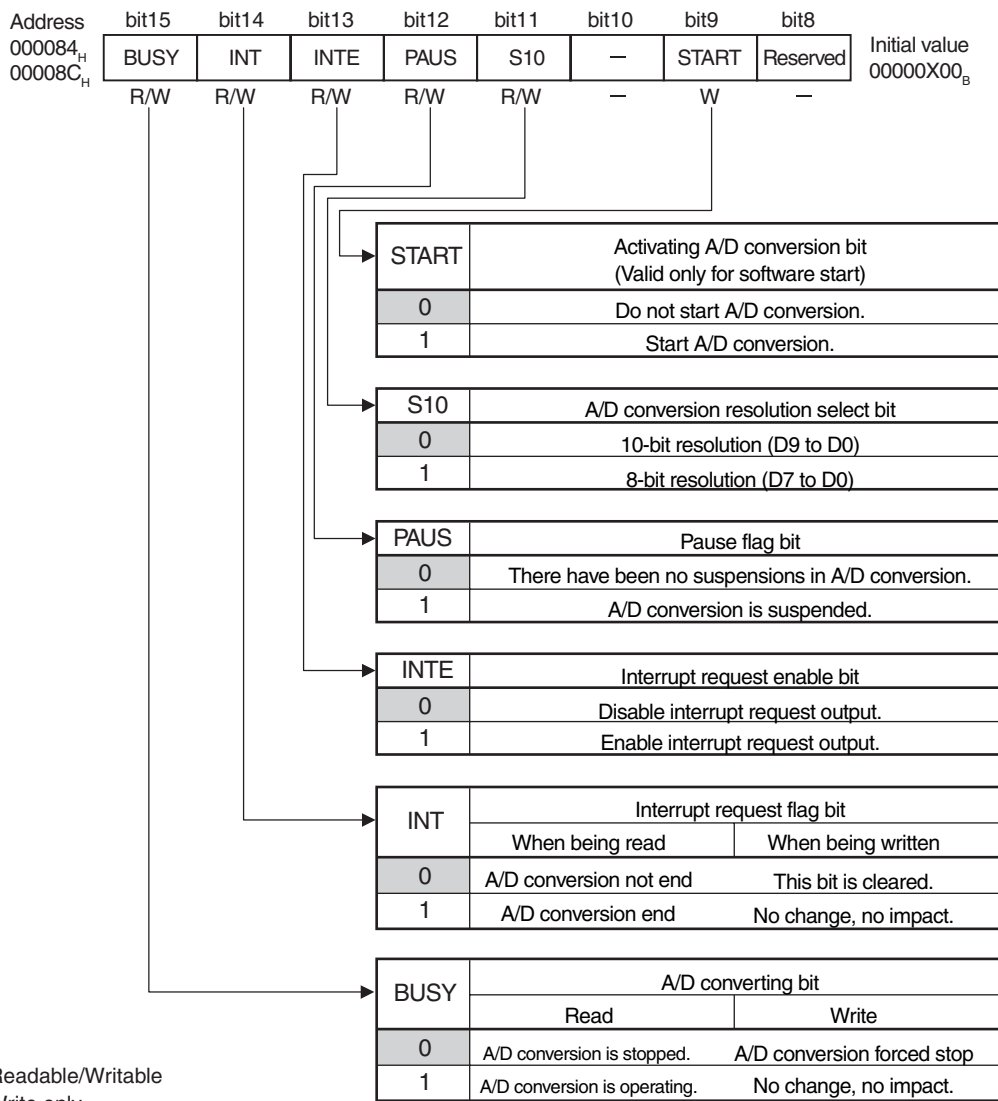
**Table 16.4-2 Description of the Functions of the A/D Mode Setting Register (ADMD) Bits (2 / 2)**

Bit name		Functions
bit5, bit4	STS1, STS0: A/D start trigger select bits	<ul style="list-style-type: none"> <li>These bits select the start trigger of A/D conversion.</li> <li>When more than one start trigger is used, the converter starts with the start trigger that occurred first.</li> </ul> <p>Notes:</p> <ul style="list-style-type: none"> <li>The start trigger is modified at the same time when rewriting it. Therefore, be sure to rewrite it at the status that does not have target start trigger during the A/D conversion operation.</li> <li>In STS1 and STS0 = 11<sub>B</sub>, the timer cannot be activated when the external trigger input is at "L". Also, the external trigger cannot be activated when the timer is at "H".</li> </ul>
bit3, bit2	CT1, CT0: Compare time setting bits	<ul style="list-style-type: none"> <li>These bits select the compare time during the A/D conversion.</li> <li>After the analog input is captured (passage of sampling time), data of the conversion result is determined upon the time set to these bits and stored in the A/D control status register (ADCD).</li> </ul> <p>Notes:</p> <ul style="list-style-type: none"> <li>When CT1 and CT0 = 00<sub>B</sub>, 10<sub>B</sub>, 11<sub>B</sub>, the compare time should be set at least 720 ns and when CT1 and CT0 = 01<sub>B</sub>, the compare time should be set at least 900 ns, so the correct analog conversion value may not be obtained.</li> <li>Be sure to rewrite these bits while the A/D operation is stopped before the conversion operation is performed.</li> </ul>
bit1, bit0	ST1, ST0: Sampling time setting bits	<ul style="list-style-type: none"> <li>These bits select the sampling time during the A/D conversion.</li> <li>When the A/D is started, time set to these bits and analog input are captured.</li> </ul> <p>Notes:</p> <ul style="list-style-type: none"> <li>Sampling time should be set at least 450 ns so the correct analog conversion value may not be obtained.</li> <li>Be sure to rewrite these bits while the A/D operation is stopped before the conversion operation is performed.</li> </ul>

### 16.4.3 A/D Control Status Register (ADCS)

The A/D control status register has features to suspend and confirm conversion, enable/disable interrupt requests, confirm the status of interrupt requests, and select the A/D conversion resolution.

■ A/D Control Status Registers (ADCS: ADCS1, ADCS2)



**Table 16.4-3 Function of Each Bit in A/D Control Status Register (ADCS) (1 / 2)**

Bit name		Function
bit15	BUSY: A/D converting bit	<ul style="list-style-type: none"> <li>This bit indicates whether the A/D converter is operating.</li> <li>When reading, if this bit is "0" it indicates that A/D conversion is stopped. If it is "1", it indicates that A/D conversion is ongoing.</li> <li>When writing, writing "0" to this bit forcibly stops A/D conversion. When "1" is written, the conversion is not changed and no others are affected.</li> <li>"1" is always read during read-modify-write.</li> </ul> <p>Note:</p> <p>Do not perform a forced stop and software activation (BUSY = 0, START = 1) at the same time.</p>
bit14	INT: Interrupt request flag bit	<ul style="list-style-type: none"> <li>This bit is set to "1" if data is set in the A/D data register through A/D conversion.</li> <li>If this bit and the interrupt request enable bit (ADCS: INTE) are "1", an interrupt request is generated.</li> <li>When writing, "0" clears this bit, and with "1" no change is made, and there are no other effects.</li> <li>"1" is always read during read-modify-write.</li> </ul> <p>Note:</p> <p>Clear this bit by writing "0" with A/D off.</p>
bit13	INTE: Interrupt request enable bit	<ul style="list-style-type: none"> <li>This bit is used to enable and disable the interrupt output to CPU.</li> <li>If this bit and the interrupt request flag bit (ADCS: INT) are "1", an interrupt request is generated.</li> </ul>
bit12	PAUS: Pause flag bit	<ul style="list-style-type: none"> <li>It is set to "1" when A/D conversion is suspended.</li> <li>This A/D converter has only 1 A/D data register. For this reason, when in continuous conversion mode, if the old conversion results are not completely read via the CPU, they will be overwritten by the next conversion results and lost. Consequently, when using continuous conversion mode, as a rule you should perform settings so that conversion results are sent to memory each time conversion is completed. However, such case can be assumed that the conversion data transfer is not completed before the next conversion in the multiple interrupts, etc. The function of this bit was created in consideration of this case. After conversion is completed, set this bit to "1" while sending the contents of the data register. During this time, A/D conversion will halt and will not store the next conversion data.</li> <li>This flag can be cleared only by writing "0" to the register.</li> <li>"1" is always read during read-modify-write.</li> </ul>



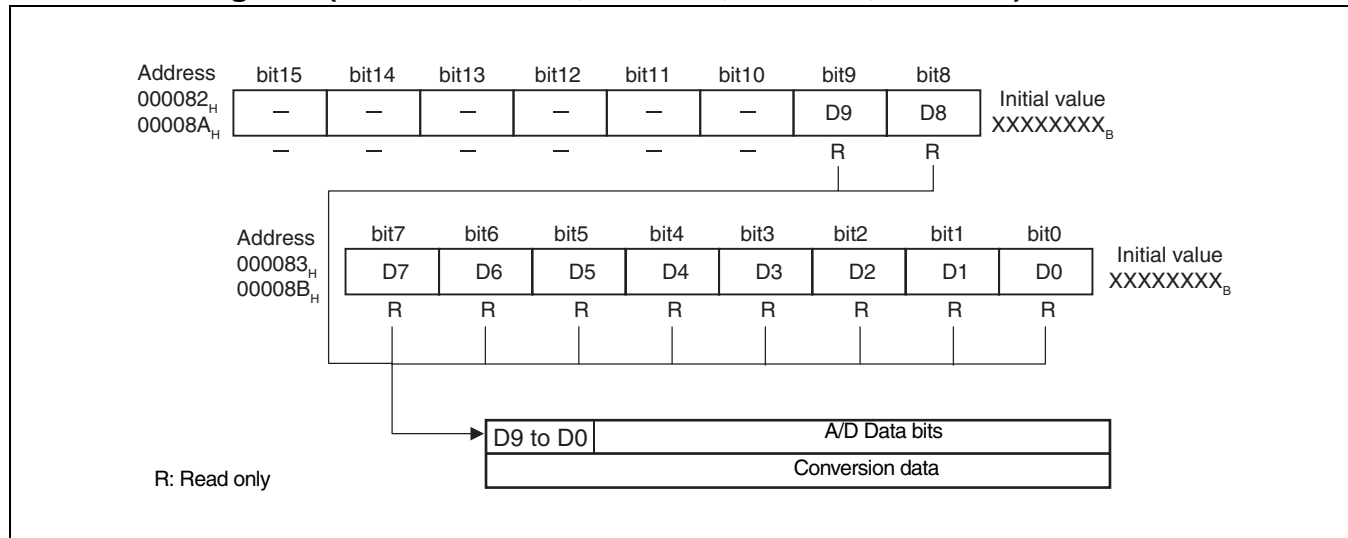
**Table 16.4-3 Function of Each Bit in A/D Control Status Register (ADCS) (2 / 2)**

Bit name		Function
bit11	S10: A/D conversion resolution select bit	<ul style="list-style-type: none"> <li>This bit is used to select the A/D conversion resolution.</li> <li>Write "0" to this bit to select 10-bit resolution. Write "1" to this bit to select 8-bit resolution.</li> </ul> <p>Notes:</p> <ul style="list-style-type: none"> <li>The A/D data bit used are different depending on the resolution.</li> <li>Be sure to update this bit when the A/D operation prior to conversion is off.</li> </ul>
bit10	Undefined bit	<ul style="list-style-type: none"> <li>The read value is indeterminate.</li> <li>Writing to this bit has no effect on the operation.</li> </ul>
bit9	START: Activating A/D conversion bit (Valid only for software start)	<ul style="list-style-type: none"> <li>This bit is used to start the A/D conversion operation by the software.</li> <li>Write "1" to this bit to activate A/D conversion.</li> <li>In stop conversion mode, this bit will not function to re-start conversion.</li> </ul> <p>Note:</p> <p>Do not perform a forced stop and software activation (BUSY = 0, START = 1) at the same time.</p>
bit8	Reserved bit	Be sure to write "0".

## 16.4.4 A/D Data Register (ADCD)

The A/D data register stores A/D conversion results.

### ■ A/D Data Register (ADCD: ADCD10, ADCD11, ADCD20, ADCD21)



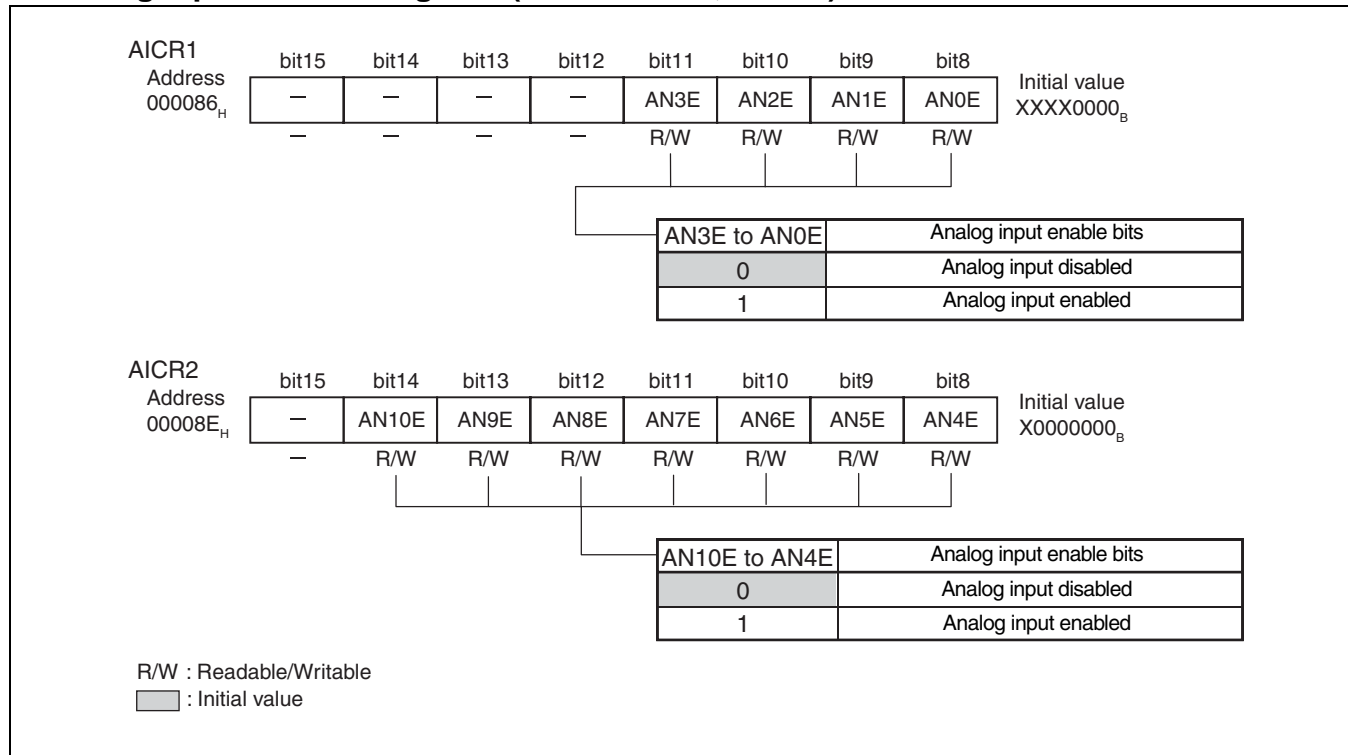
**Table 16.4-4 Functional Explanation of Each Bit in A/D Data Register (ADCD)**

Bit name		Function
bit15 to bit10	Undefined bits	<ul style="list-style-type: none"> <li>The read value is indeterminate.</li> <li>Writing to these bits has no effect on the operation.</li> </ul>
bit9 to bit0	D9 to D0: A/D data bits	<ul style="list-style-type: none"> <li>This register stores the results of A/D conversion, and is overwritten each time conversion is completed.</li> <li>The final conversion value is stored usually.</li> <li>The initial value of this register is indeterminate.</li> </ul> <p>Notes:</p> <ul style="list-style-type: none"> <li>The conversion data protection function is provided.</li> <li>Do not write data to these bits while A/D conversion is ongoing.</li> <li>When 8-bit resolution is selected, "0" is read from D9 and D8.</li> </ul>

## 16.4.5 Analog Input Control Register (AICR)

The analog input control register controls analog input.

### ■ Analog Input Control Register (AICR: AICR1, AICR2)



**Table 16.4-5 The Functions of Each Bits in analog Input Control Register (AICR)**

Bit name		Function
(AICR1) bit15 to bit12 (AICR2) bit15	Undefined bits	<ul style="list-style-type: none"> <li>The read value is indeterminate.</li> <li>Writing to these bits has no effect on the operation.</li> </ul>
(AICR1) bit11 to bit8 (AICR2) bit14 to bit8	AN3E to AN0E, AN10E to AN4E: Analog input enable bits	<ul style="list-style-type: none"> <li>When these bits are "0", analog input is disabled.</li> <li>When these bits are "1", analog input is enabled.</li> <li>Set the AICR register bit corresponding to the pin to be used as the analog input pin to "1". When this is done, the value "0" will be read from the PDR register.</li> </ul>

## 16.5 Interrupt

---

The 8/10-bit A/D converter can generate interrupt requests during A/D conversion by setting data in the A/D data register.

---

### ■ Interrupt of 8/10-bit A/D Converter

See Table 16.5-1 for the interrupt control bits and interrupt triggers of the 8/10-bit A/D converter.

**Table 16.5-1 Interrupt Control Bits and Interrupt Trigger Event for 8/10-bit A/D Converter**

	8/10-bit A/D converter
Interrupt request flag bit	ADCS: INT
Interrupt request enable bit	ADCS: INTE
Interrupt cause	Writing the A/D conversion result to the A/D data register

When A/D conversion activates, and A/D conversion results are set in the A/D data register (ADCD), the INT bit of the A/D control status register (ADCS) is set to "1". At this time, if interrupt requests are enabled (ADCS: INTE = 1), an interrupt request is output to the interrupt controller.

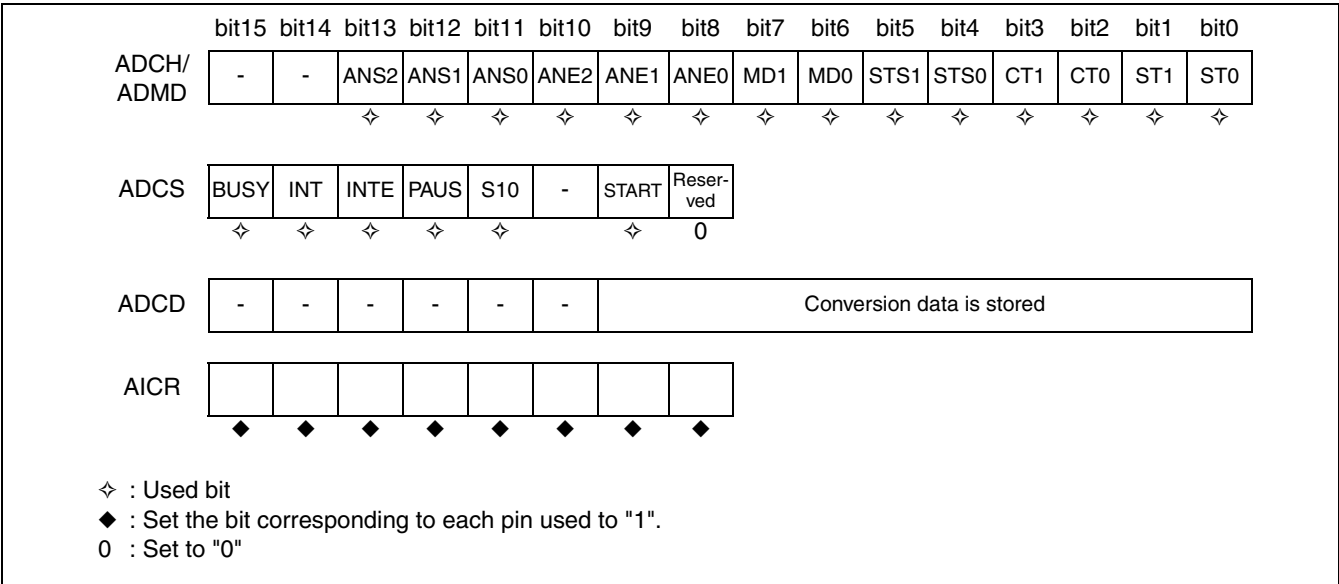
# 16.6 Operation Explanation

Three mode types, the single conversion, continuous conversion, and stop conversion modes are available for the 8/10-bit A/D converter. The operation explanation in each mode is done.

## ■ Operation of Single-shot Conversion Mode

In single conversion mode, it sequentially converts the analog input which has been set by the ANS bit and ANE bit, and when it reaches to the end channel set in ANE bit, it stops the A/D conversion. If the start channel and end channel are the same (ANS=ANE), only one channel specified in the ANS bit will be converted. The settings in Figure 16.6-1 are required in order to operate in single-conversion mode.

Figure 16.6-1 Settings for Single-shot Conversion Mode



Reference:

Given below are examples of sequences for conversion in single-shot conversion mode.

ANS=000<sub>B</sub>, ANE=011<sub>B</sub>: AN0 → AN1 → AN2 → AN3 → End

ANS=011<sub>B</sub>, ANE=011<sub>B</sub>: AN3 → End

---

Notes:

- A/D unit 1 uses the four channels AN0 to AN3, and A/D unit 2 uses the seven channels AN4 to AN10.
- Always write "0" to ANE2 and ANS2 of A/D unit 1.
- For A/D unit 2, neither ANE0 to ANE2=1 nor ANS0 to ANS2=1 can be set.
- Be sure to set A/D units 1 and 2 so that ANS0 is less than or equal to ANE0.
- When A/D conversion mode selection bit (MD1,MD0) is set to "00<sub>B</sub>", restart can be operating during A/D conversion. Only software start (STS1,STS0=00<sub>B</sub>) can be set in this mode. Please restart according to the following procedure.
  1. Clear the INT bit to "0".
  2. Write START bit to "1" and INT bit to "0" at the same time.

When A/D conversion mode selection bit (MD1,MD0) is set to "01<sub>B</sub>", restart can not be operating during the A/D conversion.

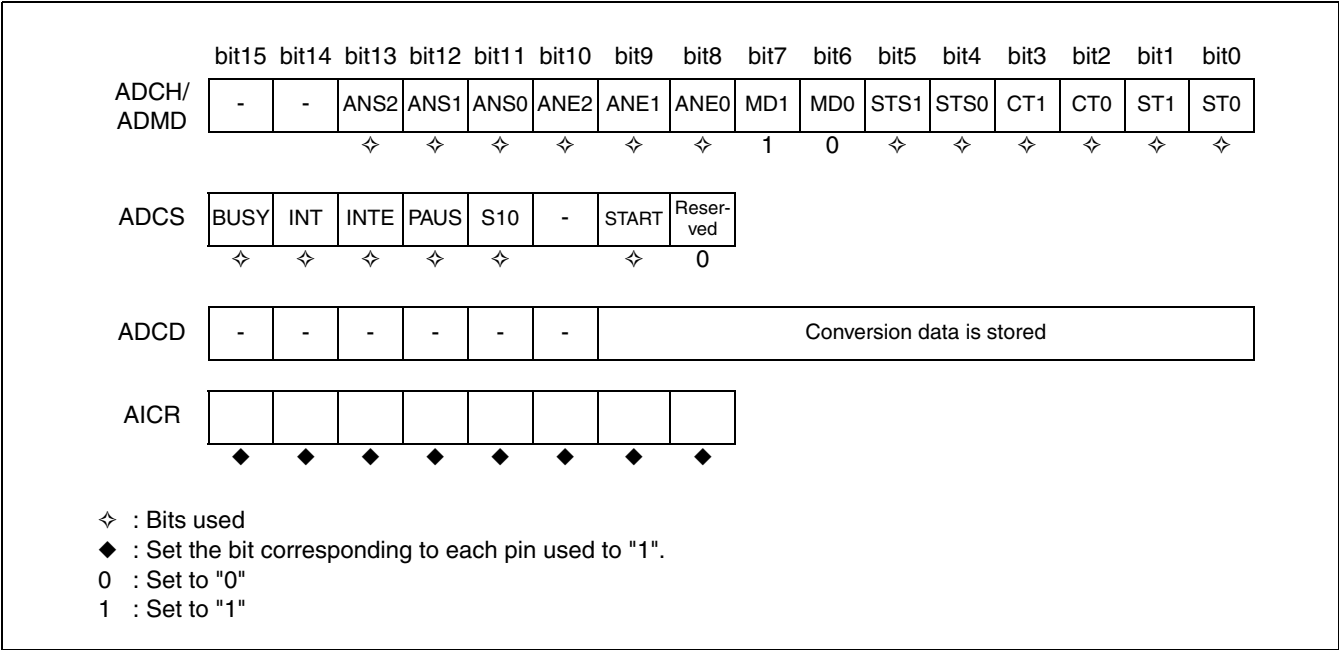
When restarting and termination of the A/D conversion occur at the same time, the A/D conversion is terminated without restarting. And, value of "300<sub>H</sub>" is stored in data register (ADCR1/ADCR0). Therefore, please use restart so that neither the A/D conversion restarting nor the terminating may occur at the same time.

---

■ Operation of Continuous Conversion Mode

In the continuous conversion mode, the analog inputs set by the ANS and ANE bits are sequentially converted, the analog input set by the ANS bit is resumed at the end of conversion of the end channel set by the ANE bit, and the A/D conversion operation is continued. If the start channel and end channel are identical (ANS=ANE), conversion loops on the channel specified by ANS only. The settings shown in Figure 16.6-2 are required in order to operate in continuous conversion mode.

Figure 16.6-2 Settings for Continuous Conversion Mode



Reference:

Given below are examples of sequences for conversion in continuous conversion mode.  
ANS= 000<sub>B</sub>, ANE= 011<sub>B</sub>: AN0 →AN1 →AN2 →AN3 →AN0 →Repeat  
ANS= 011<sub>B</sub>, ANE= 011<sub>B</sub>: AN3 →AN3 →Repeat

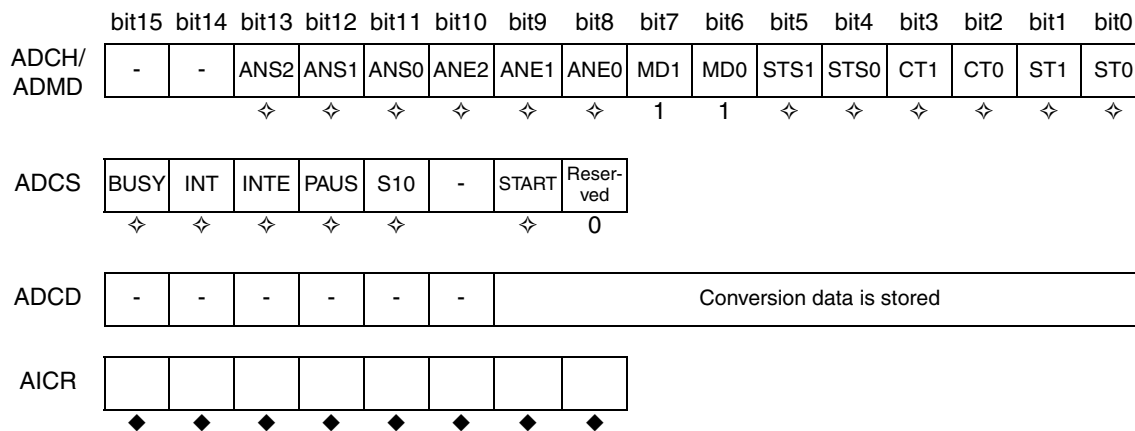
Notes:

- A/D unit 1 uses the four channels AN0 to AN3, and A/D unit 2 uses the seven channels AN4 to AN10.
- Always write "0" to ANE2 and ANS2 of A/D unit 1.
- For A/D unit 2, neither ANE0 to ANE2=1 nor ANS0 to ANS2=1 can be set.
- Be sure to set A/D units 1 and 2 so that ANS0 is less than or equal to ANE0.

## ■ Operation of Pause-conversion Mode

In the stop conversion mode, the analog input set by the ANS and ANE bits is converted by being suspended for every channel, the analog input set by the ANS bit is resumed at the end of conversion of the end channel set by the ANE bit, and the operation of A/D conversion and suspension is continued. If the start channel and end channel are identical (ANS=ANE), conversion loops on the channel specified by the ANS bits only. When the conversion is restarted during the suspension, the start factor specified by the STS1 and STS0 bits is generated. The settings in Figure 16.6-3 are required in order to operate in stop conversion mode.

**Figure 16.6-3 Settings for Stop Conversion Mode**



- ◇ : Used bit  
 ◆ : Set the bit corresponding to each pin used to "1".  
 0 : Set to "0"  
 1 : Set to "1"

### Reference:

Given below are examples of sequences for conversion in stop conversion mode.

ANS= 000<sub>B</sub>, ANE= 011<sub>B</sub>: AN0 → Pause → AN1 → Pause → AN2 → Pause → AN0 → Repeat

ANS= 011<sub>B</sub>, ANE= 011<sub>B</sub>: AN3 → Pause → AN3 → Pause → Repeat

### Notes:

- A/D unit 1 uses the four channels AN0 to AN3, and A/D unit 2 uses the seven channels AN4 to AN10.
- Always write "0" to ANE2 and ANS2 of A/D unit 1.
- For A/D unit 2, neither ANE0 to ANE2=1 nor ANS0 to ANS2=1 can be set.
- Be sure to set A/D units 1 and 2 so that ANS0 is less than or equal to ANE0.



## 16.7 A/D Conversion Data Protection Function

---

**When the A/D conversion is executed in the interrupt enable status, the conversion data protection function works.**

---

### ■ A/D Conversion Data Protection Function

The A/D converter only has one data register for storing conversion data. For this reason, when performing A/D conversion, after conversion is completed, the data stored in the data register is overwritten. Therefore, when the converted data transfer to the memory is delayed, a part of previous data may be lost. To get around this, when interrupts are enabled (INTE = 1), the data-protection feature works as described below.

When conversion data is stored in the A/D data register (ADCD), the INT bit of the A/D control status register (ADCS) is set to "1". While the INT bit is 1, conversion data will not be stored to ADCD after the next conversion. The PAUS bit is set, and A/D conversion becomes suspended. While suspended, the value immediately prior is retained. In order to cancel the suspend, clear the INT bit. After the suspended status is cleared, the conversion data that had been maintained is stored in ADCD, and the next operation is performed.

---

#### Notes:

- The converted data protection function operates only in the interrupt enabled (ADCS: INTE=1) status.
  - When the conversion is restarted during the suspension, the waiting data are destroyed.
-

## 16.8 Notes on Using 8/10-bit A/D Converter

This section summarizes notes on using the 8/10-bit A/D converter.

### ■ Notes on Using 8/10-bit A/D Converter

#### ● Analog input pin

The A/D input pin does double duty as a port I/O pin. The port-direction register (DDR) and analog input enable register (AICR) are switched and used. For the pins used as analog input, set the bits corresponding to DDR to "0" to input the port setting, then set AICR register analog input mode (AICRx = 1). Lock the input gate on the port side. When the intermediate level signal is input in the port input mode (AICRx = 0), the input leak current flows through the gate.

#### ● Cautions for use of internal timers

To activate the A/D converter by an internal timer, set the STS1 and STS0 bits of the A/D control status register (ADMD). When doing so, set the internal timers' input value to the inactive side (for internal timer, this is "L"). If you set it to the active side, the timer may start to operate as soon as you write to the ADMD register.

#### ● Turning-on sequence of power supply to A/D converter and analog inputs

Make sure to apply the A/D converter power source (AVcc, AVRH1, AVRH2, and AVss) and analog input (AN0 to AN10) after or at the same time as applying digital power source (Vcc). When cutting off the power, cut off the digital power source (Vcc) after or at the same time as cutting off the A/D converter power source and analog input.

#### ● Power supply voltage of the A/D converter

To prevent a latch-up from occurring, keep the A/D converter power supply (AVcc) voltage below the digital power supply (Vcc) voltage.

#### ● A/D unit 1 and 2 settings

- A/D unit 1 uses the four channels AN0 to AN3, and A/D unit 2 uses the seven channels AN4 to AN10.
- Always write "0" to ANE2 and ANS2 of A/D unit 1.
- For A/D unit 2, neither ANE0 to ANE2=1 nor ANS0 to ANS2=1 can be set.
- Be sure to set A/D units 1 and 2 so that ANS0 is less than or equal to ANE0.

#### ● Restart of A/D converter

When A/D conversion mode selection bit (MD1,MD0) is set to "00<sub>B</sub>", restart can be operating during A/D conversion. Only software start (STS1,STS0="00<sub>B</sub>") can be set in this mode. Please restart according to the following procedure.

1. Clear the INT bit to 0.
2. Write START bit to 1 and INT bit to 0 at the same time.

When A/D conversion mode selection bit (MD1,MD0) is set to "01<sub>B</sub>", restart can not be operating during the A/D conversion.

When restarting and termination of the A/D conversion occur at the same time, the A/D conversion is terminated without restarting. And, value of "300<sub>H</sub>" is stored in data register (ADCR1/ADCR0). Therefore, please use restart so that neither the A/D conversion restarting nor the terminating may occur at the same time.



# ***CHAPTER 17***

---

## ***16-BIT MAC***

**This chapter explains the overview of 16-BIT MAC, the configuration and functions of registers, and the operation of 16-BIT MAC.**

- 17.1 Features
- 17.2 Defined Instruction
- 17.3 Explanation of Register
- 17.4 Operation Explanation
- 17.5 Instruction Detail Explanation

## 17.1 Features

---

**This section describes features, a register list and a block diagram of 16-BIT MAC.**

---

### ■ Features

- High-speed multiply-accumulate (in one system clock cycle)
- Data format : 16-bit fixed-point ( $16 \times 16 + 40$  bits)
- Instruction area :  $256 \times 6$  bits
- Data area :  $64 \times 16$  bits  $\times$  2 pairs
- Rounding available
- Saturation available
- Number of addition items : Max. 64 items
- Instruction : MAC instruction, STR instruction, JMP instruction
- Delay handling : Capable of arbitrarily transferring data within  $64 \times 16$  bits
- Fixed-point method : Selectable from among Q8 to Q15
- Program execution control : Choice of eight different instructions
- Variable monitor : Capable of monitoring up to  $8 \times 16$ -bit calculation results without terminating a program

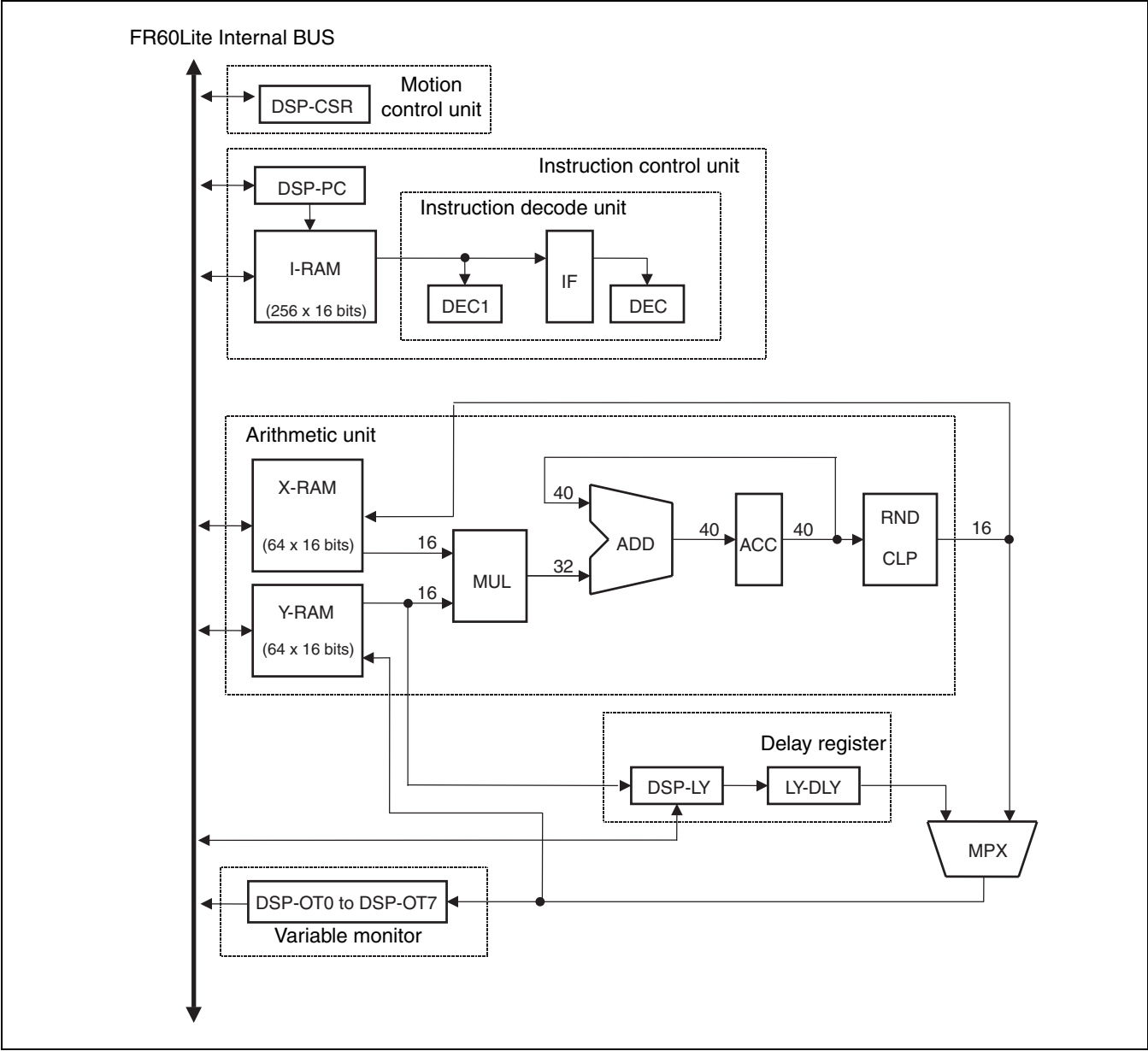
## ■ Register List

	bit15	bit8	bit7	bit0	
Address:00039E <sub>H</sub>	(Reserved area)				Access prohibited
Address:0003A0 <sub>H</sub>	DSP-PC (Program counter)		DSP-CSR (Control/status)		R/W, R, W
Address:0003A2 <sub>H</sub>	DSP-LY (Delay register) upper		DSP-LY (Delay register) lower		R/W
Address:0003A4 <sub>H</sub>	DSP-OT0(Output queue 0) upper		DSP-OT0(Output queue 0) lower		R
Address:0003A6 <sub>H</sub>	DSP-OT1(Output queue 1) upper		DSP-OT1(Output queue 1) lower		R
Address:0003A8 <sub>H</sub>	DSP-OT2(Output queue 2) upper		DSP-OT2(Output queue 2) lower		R
Address:0003AA <sub>H</sub>	DSP-OT3(Output queue 3) upper		DSP-OT3(Output queue 3) lower		R
Address:0003AC <sub>H</sub>	(Reserved area)		(Reserved area)		Access prohibited
Address:0003AE <sub>H</sub>	(Reserved area)		(Reserved area)		Access prohibited
Address:0003B0 <sub>H</sub>	DSP-OT4(Output queue 4) upper		DSP-OT4(Output queue 4) lower		R
Address:0003B2 <sub>H</sub>	DSP-OT5(Output queue 5) upper		DSP-OT5(Output queue 5) lower		R
Address:0003B4 <sub>H</sub>	DSP-OT6(Output queue 6) upper		DSP-OT6(Output queue 6) lower		R
Address:0003B6 <sub>H</sub>	DSP-OT7(Output queue 7) upper		DSP-OT7(Output queue 7) lower		R
Address:				Sum of products operation macro	Access
00C000 <sub>H</sub>	X-RAM (coefficient RAM) ... 64 × 16 bits			00 <sub>H</sub>	R/W
:				:	
00C07E <sub>H</sub>				3F <sub>H</sub>	
Address:				Sum of products operation macro	Access
00C080 <sub>H</sub>	Y-RAM (variable RAM) ... 64 × 16 bits			00 <sub>H</sub>	R/W
:				:	
00C0FE <sub>H</sub>				3F <sub>H</sub>	
Address:				Sum of products operation macro	Access
00C100 <sub>H</sub>	I-RAM (Instruction RAM) ... 256 × 16 bits			00 <sub>H</sub>	R/W
:				:	
00C2FE <sub>H</sub>				FF <sub>H</sub>	

### Notes:

- When writing to the above register/RAM area from the CPU, be sure to use a half-word (or word) transfer instruction to an even-numbered address.
- Although the RAM area can be write-or read-accessed with a byte transfer instruction from the CPU, follow above (1) for use as a multiply-accumulate circuit.

■ Block Diagram of 16-BIT MAC



**Table 17.1-1 Outline Explanation of Block Diagram**

Block	Register	Function
Motion control	DSP-CSR	Operation control register of multiply-and-accumulate calculation macro. This register controls the following operations from the CPU and servo block. <ul style="list-style-type: none"> <li>• Calculation start/end instructions</li> <li>• Interrupt control</li> <li>• Program flow control (used for conditional branching commands of multiply-and-accumulate macro)</li> </ul>
Instruction control	DSP-PC	Program counter Program execution begins from the first address specified by the CPU.
	I-RAM	It is instruction RAM of $256 \times 16$ bits. The CPU can perform Read and Write while the multiply-and-accumulate macro calculation is halted. Load an instruction code from the CPU before starting calculation.
	IF	Instruction fetch register
	DEC1* DEC*	Instruction decoder
Arithmetic unit	X-RAM	It is data RAM of $64 \times 16$ bits. The CPU can perform Read and Write while the multiply-and-accumulate macro calculation is halted. Load a coefficient from the CPU before starting calculation.
	Y-RAM	It is data RAM of $64 \times 16$ bits. The CPU can perform Read and Write while the multiply-and-accumulate macro calculation is halted. Load a variable from the CPU before starting calculation.
	MUL*	It is $16 \times 16$ to 32-bit multiplier.
	ADD*	It is $32 + 40$ to 40-bit adder.
	ACC*	It is 40-bit accumulator.
	CLP* RND* SLQ*	When sending from 40 to 16 bits, if there is out-of-range 16-bit data, saturated to the maximum value. When sending from 40 to 16 bits, the lower-order bits are rounded. When sending from 40 to 16 bits, select the bits to send.
Delay register	DSP-LY LY-DLY*	Delay register This register holds a variable during multiply-accumulate operation, capable of writing it back to Y-RAM.
Variable monitor output	DSP-OT0 to DSP-OT7	Variable monitor output register Stores the same value as addresses 0 to 7 of Y-RAM. During calculation (when Y-RAM access is disabled), you can monitor addresses 0 to 7 of Y-RAM.

\*: Access from CPU disabled.



## 17.2 Defined Instruction

---

The multiply-and-accumulate macro has three main types of command: **MAC/STR/JMP** instruction.

---

### ■ Defined Instruction

This specification uses commands other than these three in notation. The command hierarchy is as follows.

- **MAC Instruction**
  - Multiply-and-accumulate instruction (CLAC bit = 0)
  - Multiplication instruction (CLAC bit = 1)
- **STR Instruction**
  - HLT Instruction (HLT bit = 1)
  - INT Instruction (SIRQ bit = 1)
- **JMP Instruction**
  - Unconditional branching command (COND bit = 0)
  - Conditional branching command (COND bit = 1)
  - HLT Instruction (HLT bit = 1)
  - INT Instruction (SIRQ bit = 1)

### 17.3 Explanation of Register

This section describes the configuration and functions of 16-BIT MAC register.

■ DSP Control/Status Register (DSP-CSR)

The control/status register is an 8-bit register consisting of various flags for switching the multiply-accumulate macro state, controlling interrupts to the CPU, and indicating the multiply-accumulate macro status. This register is also used to set the conditions for conditional branching commands in the multiply-and-accumulate macro.

- The 8-bit register always allows external R/W.

[Control function]

- Multiply-and-accumulate macro state (calculation start/end) transition (**GoDSP** and **HltDSP**)
- Interrupt mask for CPU (**IeDSP**)
- Sets conditions for conditional branching command of multiply-and-accumulate macro (**USR2**, **USR1**, **USR0**).

[Status function]

- Multiply-and-accumulate macro current state acquisition flag (**RunDSP**)
- Interrupt request flag (**IrqDSP**)
- Saturation flag (**SatDSP**)

Figure17.3-1 DSP-CSR

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0		Initial value
Address	–	USR2	USR1	USR0	IrqDSP	IeDSP	HltDSP	GoDSP	Write	00000000 <sub>B</sub>
0003A1 <sub>H</sub>	SatDSP	USR2	USR1	USR0	IrqDSP	IeDSP	–	RunDSP	Read	
	R	R/W	R/W	R/W	R/W	R/W	W	R/W		
R/W: Readable/Writable										
R: Read only										
W: Write only										
–: Undefined										

**[bit7] SatDSP (Saturation flag): Read only**

- This bit serves as the status flag holding the state in which saturation has been performed during calculation.
- This bit is set if saturation is indicated in the **STR** command (**CLP** = 1), and saturation was actually performed. If this bit is set during calculation, the setting is maintained until the next calculation starts.
- This bit is cleared when calculation begins.

Set factor: Set if the **STR** command performed saturation processing during calculation.

- Clear factor: Cleared by start of calculation (initial value).
- Be initialized to "0" when resetting (No saturation).
- This bit is read-only. Writing does not change the bit value.

**[bit6, bit5, bit4] USR2, USR1, USR0 (Set a jump condition bits): Read/Write**

- These bits jump when the bits are referred by the conditional branching command of multiply and accumulate macro (at **COND** bit = 1), and value of these bits and the **UBP** flag for the conditional branching command match (conclusion of the condition). In other words, it is possible to switch between 8 different calculation routines from the CPU by combining this conditional branching command with the calculation commands.
- Be initialized to "000B" when resetting.
- Reading and writing are permitted.

**[bit3] IrqDSP (Interrupt request flag): Read/Write**

- This bit serves as the flag indicating that a multiply-accumulate macro software interrupt request has occurred. When interrupt requests are enabled (**IeDSP** = 1), setting this bit creates an interrupt request for the CPU.
- The multiply-accumulate macro interrupt request is generated by means of software by setting the **SIRQ** bit of the **STR** or **JMP** instruction to "1".

Set factor: Set via the generation of a multiply-and-accumulate macro software interrupt (**STR**/**JMP** command).

Clear factor: Cleared when "0" is written. [Initial value]

- Be initialized to "0" when resetting. (Not generate interrupt request)
- It is possible to read and write, however, that only "0" can be written. Writing "1" will not change the bit value.
- For read-modify-write instructions, the value of "1" is always read, regardless of the value of the bit.

**[bit2] IeDSP (Interrupt request enable bit): Read/Write**

Interrupt requests to the CPU (**IrqDSP** = 1) are controlled as follows.

"0": Interrupt request output disabled (setting **IrqDSP** will not generate an interrupt) [initial value]

"1": Interrupt request output enabled (setting **IrqDSP** will generate an interrupt)

- Be initialized to "0" when resetting. (Interrupt request output is disabled.)
- It is possible to read and write.

[bit1] **HltDSP** (Stop of Calculation): Write only

- This bit forcibly stops calculation.
- Writing "1" to this bit stops calculation, if currently being performed (**RunDSP** = 1), after the instruction being executed terminates (after two cycles if it is a two-cycle instruction), and clears the **RunDSP** flag.
- If calculation is in a stop state, this has not effect.
- If a stop is forced via this bit, DSP-PC points to the command address following the stopped command. It is thus possible to continue with command execution.
  - Writing "1" aborts the instruction.
  - An attempt to write "0" is ignored. Reading the bit always returns "0".
- Be initialized to "0" when resetting.

[bit0] **GoDSP** (Start of calculation): Write only

**RunDSP** (Calculation-in-progress flag): Read only

- Writing "1" to the **GoDSP** bit indicates the start of calculation. When calculation is being stopped (**RunDSP** = 0): Calculation is started and the **RunDSP** flag is set. If calculation is already ongoing (**RunDSP** = 1), this action has no effect.
- The **RunDSP** flag indicates that calculation is being executed. The flag is set when calculation is started; it is cleared either when "1" is written to the **HltDSP** bit or when a multiply-accumulate macro **HLT** instruction is executed.
- When a calculation is being executed (**RunDSP** = 1), DSP-PC, DSP-LY, X-RAM, Y-RAM, and I-RAM cannot be accessed from the CPU. Only DSP-CSR and DSP-OT0 to DSP-OT7 can be monitored.
- To start calculation, the start address of the calculation routine must be loaded into the DSP-PC either upon or before startup.
  - Write-time function (**GoDSP**: start calculation)
    - "0": No function/No effect on operation
    - "1": When calculation has been stopped →Start of calculation
    - When calculation has been executed →No effect
  - Read-time function (**RunDSP**: calculation running flag)
    - "0": Calculation being stopped (Initial value)
      - Clear factor →when HltDSP"1" write or HLT instruction is executed.
    - "1": Calculation being executed
      - Set factor →when calculation starts.
- Be initialized to "0" when resetting. (calculation is being stopped)
- It is possible to read and write, however, that as described above, the significance differs depending on whether reading or writing is being performed.
- For read-modify-write instructions, the value of 0 is always read, regardless of the value of the bit.

■ DSP Program Counter (DSP-PC)

The program counter is 8-bit long. It indicates the memory address (I-RAM) where the command code to be executed by the multiply-and-accumulate macro is stored. Although the program counter is automatically updated when a command is executed, it can be overwritten via a multiply-and-accumulate macro **JMP**.

Additionally, access (R/W) from the CPU is only possible when calculation is stopped. You must store the start address of the calculation routine in DSP-PC before or at the same time as calculations starts.

After executing the **HLT** command, or writing "1" to the DSP-CSRs **HitDSP**, DSP-PC points to the address of the command after the stopped command. Program execution can be continued by again setting **GoDSP**.

	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
Address	—	—	—	—	—	—	—	—	XXXXXXXX <sub>B</sub>
0003A0 <sub>H</sub>	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
R/W: Readable/Writable									
X: Undefined value									
—: Undefined									

- Upon reset: Indeterminate.
- Although read/write is enabled, access is only possible when multiply-and-accumulate macro calculation is halted (DSP-CSR: **RunDSP** = 0).
- When calculation is being executed (DSP-CSR: **RunDSP** = 1), the program counter cannot be accessed by the CPU as it is disconnected from the bus.

■ DSP Delay Register (DSP-LY)

DSP-LY is a 16-bit register used when the delayed write bit (**LDLY**) of the multiply-accumulate macro **MAC** instruction is "1". Access is not possible during calculation (DSP-CSR: **RunDSP** = 1).

- When the **LDLY** bit of the **MAC** instruction is "1", the following two operations are performed in sequence:
  - (1) The contents of the DSP-LY register are sent to the LY-DLY register.
  - (2) The Y-RAM read data selected by means of the **MAC** command is stored in the DSP-LY register.
- When the **STLY** bit of the **MAC** instruction is "1", the LY-DLY register value is written to the Y-RAM address selected by the **MAC** instruction after the **MAC** instruction is executed. At this time, the execution time is two cycles.

	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
Address																	XXXXXXXX <sub>B</sub>
0003A2 <sub>H</sub>	R/W		R/W		R/W		R/W		R/W		R/W		R/W		R/W		
R/W:	Readable/Writable																
X:	Undefined																

- Upon reset: Indeterminate.
- Although read/write is enabled, access is only possible when DSP-LY calculation is halted (DSP-CSR: **RunDSP** = 0). When calculation is being executed (DSP-CSR: **RunDSP** = 1), the program counter cannot be accessed by the CPU as it is disconnected from the bus.

■ DSP Variable Monitor Register (DSP-OT0 to DSP-OT7)

There are 8 registers of 16-bit for use as variable monitor registers. With the exception of the initial status when the power is turned on, addresses 0 to 7 of Y-RAM hold the same value. Only read is always available from the CPU. Even if calculation is ongoing, the contents of addresses 0 to 7 in Y-RAM can be monitored.

Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
DSP-OT0 0003A4 <sub>H</sub>																
DSP-OT1 0003A6 <sub>H</sub>																
DSP-OT2 0003A8 <sub>H</sub>	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R
DSP-OT3 0003AA <sub>H</sub>																
DSP-OT4 0003B0 <sub>H</sub>																
DSP-OT5 0003B2 <sub>H</sub>																
DSP-OT6 0003B4 <sub>H</sub>																
DSP-OT7 0003B6 <sub>H</sub>																

Initial value: XXXXXXXX XXXXXXXX<sub>B</sub>

R: Read only  
X: Undefined

- Upon reset: Indeterminate.
- Only read can always be performed. Read is also possible if multiply-and-accumulate macro program execution is ongoing.

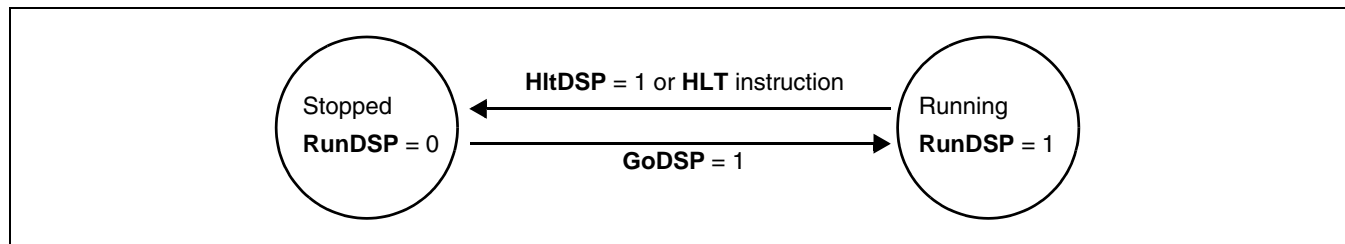
## 17.4 Operation Explanation

This section explains the operation and function of 16-BIT MAC.

### ■ Operating Mode

The action of the multiply-and-accumulate macro is controlled by manipulating the DSP-CSR register.

The multiply-accumulate macro takes either of the two states described below. The multiply-accumulate macro starts program execution either when "1" is written to the **GoDSP** bit or the GODSPSV signal is input from the Servo Block with the macro stopped. Note also that the registers and memory areas accessible from the CPU are different depending on whether calculation is being stopped or executed.



Each of these states is described below.

- **Stopped:** The multiply-and-accumulate macro is stopped. The CPU can access instruction RAM (I-RAM), data RAM (X-RAM, Y-RAM), and all the registers of the multiply-accumulate macro. Transit to this state by writing "1" to **HltDSP**, or executing the **HLT** command. The system is also initialized to this state when it is reset.
- **Running:** The multiply-and-accumulate macro is running. When 1 is written to the **GoDSP** bit with the multiply-accumulate macro inactive, it enters this state and starts program execution from the current DSP-PC (program counter). Transit to the stopped state, and halt program execution, by writing "1" to the **HltDSP** bit, or executing the **HLT** command. Only **DSP-CSR** and the **DSP-OT0** to **DSP-OT7** registers are accessible from the CPU (Other registers and RAM are access-barred \*).

\*: Although access is disabled, R/W operations have the following effect:

Write → No effect (Not written).

Read → Indeterminate

### ■ Instruction Operation

When "1" is written to the **GoDSP** bit of the DSP-CSR register, the multiply-and-accumulate macro begins command execution from the current DSP-PC (program counter) (Operation is in parallel with CPU operation).

Before execution, set I-RAM and DSP-PC values. (DSP-CSR and DSP-PC can be set simultaneously.)

When multiply-and-accumulate macro command execution starts, the following operation control is performed.

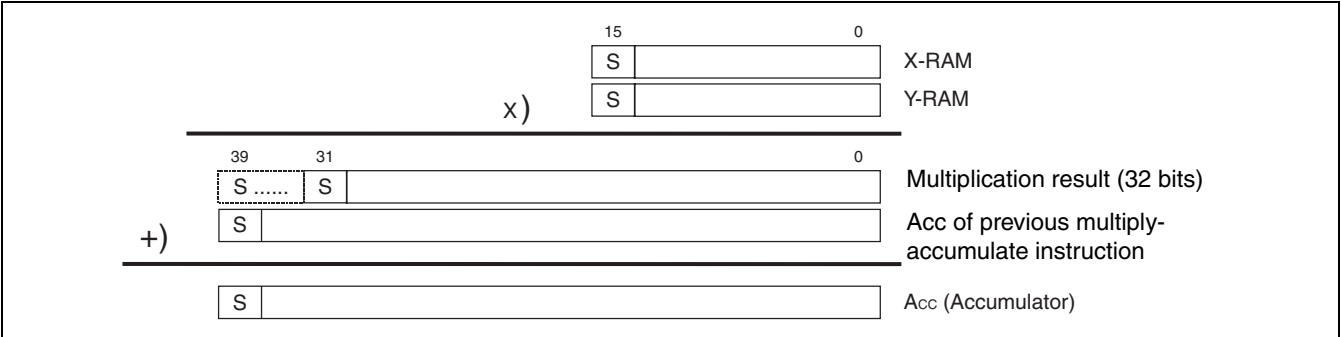
- When the multiply-and-accumulate macro executes an **HLT** command \*, transits to stopped state after the command is executed. At this time, DSP-PC stops pointing to the address following the **HLT** command.



- When a **JMP** command or **STR** command is being executed, interrupt requests can be sent to the CPU. (Interrupt mask enabled)
  - Use a conditional branch instruction which refers to the **USR0** to **USR2** bits of the DSP-CSR register to switch the program flow.
- \*: **HLT** instructions are **JMP** and **STR** instructions with the **HLT** bit set to "1".

■ **Operation Function**

The multiply-accumulate macro has two pairs of 16-bit data RAMs (X-RAM and Y-RAM). When executing a multiply-accumulate (or multiply) instruction, it loads data from each pair of RAMs, performs signed multiply-accumulate (or multiply) operation, and stores the result in a 40-bit accumulator. The data format is as follows.



- Notes:
- In the case of multiplication commands, the multiplication results are stored in the accumulator, extended to a signed 40-bit value (The accumulator contents are zero-cleared immediately beforehand).
  - "S" represents a signed bit.

Repeating a multiply-accumulate instruction many times may yields unpredictable results if it causes the accumulator to overflow.

Do not execute a multiply-accumulate instruction continuously for 512 times or more.

■ **Delayed Write Feature**

When a multiply-accumulate (or multiply) instruction is executed, the following transfer operations can be performed at the same time. Performing this transfer and the calculation in tandem makes it easy to perform delayed data processing with a digital filter.

- The value read from Y-RAM is stored in the DSP-LY register.
- To delayed-write the DSP-LY register value before the execution of the instruction to the read address in Y-RAM via the LY-DLY register.

## ■ Transfer of Operation Results

Although operation results held in the accumulator are transferred to X-RAM or Y-RAM as 16-bit values, the following scaling is performed.

- Output bit selection

The following bit ranges from the 40-bit accumulator can be selected.

bit27 to bit12 (Q12 format)

bit28 to bit13 (Q13 format)

bit29 to bit14 (Q14 format)

bit30 to bit15 (Q15 format)

bit23 to bit8 (Q8 format)

bit24 to bit9 (Q9 format)

bit25 to bit10 (Q10 format)

bit26 to bit11 (Q11 format)

- Rounding

The LSB is set by rounding up the bit below the LSB of the selected output bits (round up if "1", ignore if "0").

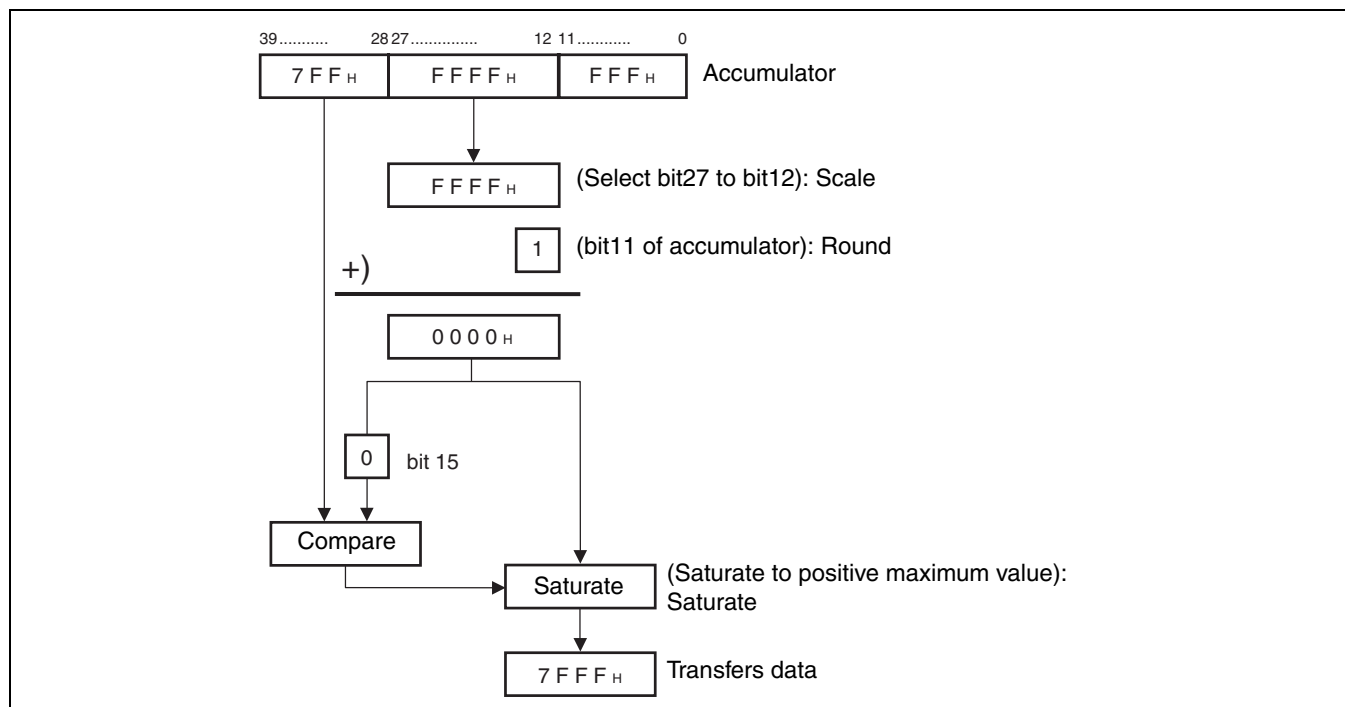
- Saturation

This compares the sign bit (MSB) of the rounded 16-bit data with the upper bits of the accumulator. If a bit with a different value is present, the data value is saturated. The result of saturation depends on the sign bit (MSB) of the accumulator as follows.

If the sign of the accumulator is "0"-saturated to positive maximum value "7FFF<sub>H</sub>"

If the sign of the accumulator is "1"-saturated to negative maximum value "8000<sub>H</sub>"

The following shows an example.



## ■ Variable Monitor Output

The multiply-and-accumulate macro includes registers (DSP-OT0 to DSP-OT7) that always contain the values of addresses 0 to 7 in Y-RAM. Whenever data is written to Y-RAM addresses 0 to 7 (write from CPU, write by **STR** instruction, or delayed write), the same data is written to the DSP-OT0 to DSP-OT7 registers.

Although CPU access to Y-RAM is prohibited during calculation, by using the **STR** instruction to store any results required by the CPU in Y-RAM addresses 0 to 7, the CPU can read calculation results at any time.

---

### Note:

Ensure the following requirements when using the DMA transfer in the sum-of-products macro:

- Set the CPU clock equivalent to or faster than the peripheral clock.
  - If the CPU clock is set slower than the peripheral clock, the DMA transfer does not work properly.
-

## 17.5 Instruction Detail Explanation

This section explains three instructions (**MAC**, **STR**, and **JMP** instructions) used in 16-BIT MAC.

### ■ MAC Instruction

Operation:  $ACC \leftarrow ACC + X \text{ data} \times Y \text{ data}$

LY-DLY  $\leftarrow$  DSP-LY

DSP-LY  $\leftarrow$  Y data (LDLY = 1)

Y-RAM  $\leftarrow$  LY-DLY (STLY = 1)

Explanation: Add the result of multiplying X data from X-RAM and Y data from Y-RAM to the accumulator. Also transfer the contents of the DSP-LY register to the LY-DLY register.

Word count: 1 word (16-bit width)

Cycle count: 1 system clock cycle (2 cycles if STLY = 1)

Operation code:

bit15	bit14	bit13	bit12	bit11.....bit6	bit5.....bit0
1	CLAC	STLY	LDLY	X-Addr	Y-Addr

[bit14] **CLAC** (Clear Acc)

- Setting this bit causes the instruction to act as a multiplication instruction.
- "0":  $ACC \leftarrow ACC + X \text{ data} \times Y \text{ data}$  (multiply-and-accumulate instruction)
- "1":  $ACC \leftarrow 0 + X \text{ data} \times Y \text{ data}$  (multiplication instruction)

[bit13] **STLY** (Store LY)

- When this bit is "1", the following action is performed. When the bit is "0", only the operation is executed.
- After the operation, the instruction stores the LY-DLY register value to the Y-Addr address in Y-RAM.
- The execution time only increases to two cycles when this bit is set.

[bit12] **LDLY** (Load LY)

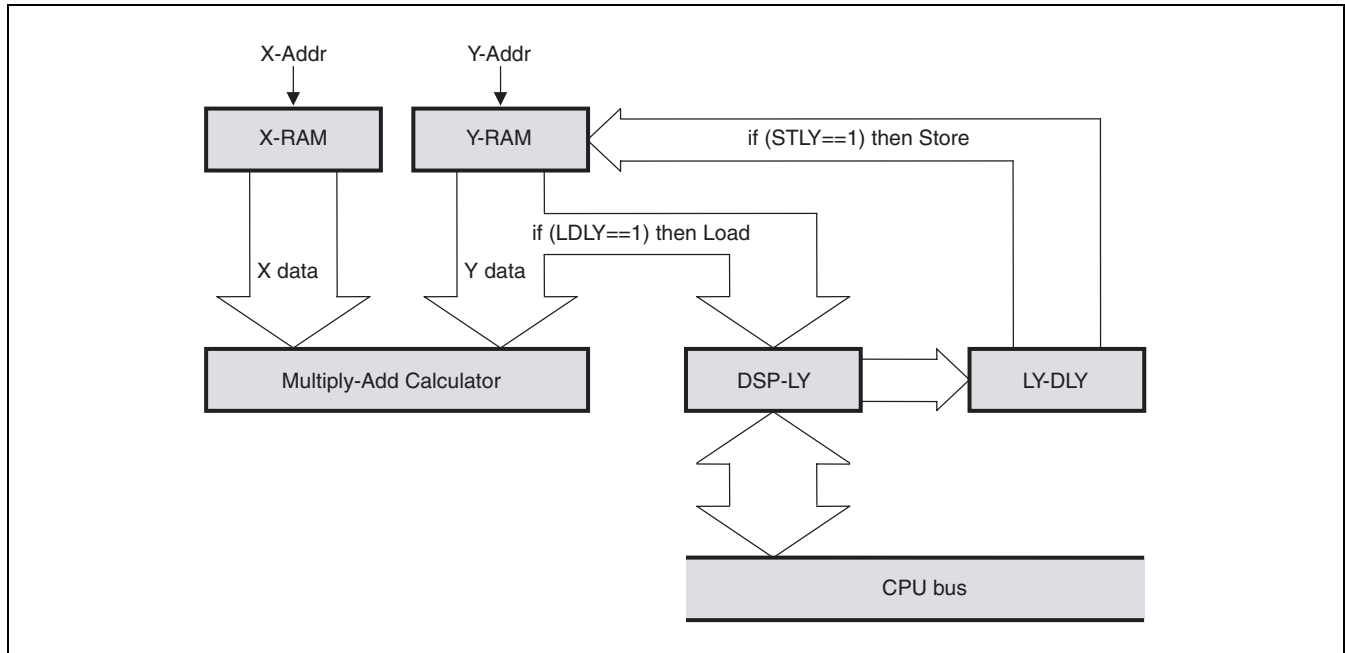
- When this bit is "1", the following action is performed. When the bit is "0", only the operation is executed.
- During the operation, the contents of the Y-Addr address in Y-RAM are stored in the DSP-LY register.

[bit11 to bit6] **X-Addr** (X-RAM Address)

- Address bits for specifying the location of the X data in X-RAM.

[bit5 to bit0] **Y-Addr** (Y-RAM Address)

- Address bits for specifying the location of the Y data in Y-RAM.



## ■ STR Instruction (Transfer Instruction)

Operation: Data RAM  $\leftarrow$  Accumulator

Explanation: Convert the 40-bit accumulator value to a 16-bit value in accordance with the RND, CLP, and SLQ flags. Store the result in the data RAM specified by the SLY flag and X/Y-Addr.

Word count: 1 word (16-bit width)

Cycle count: 1 system clock cycle

Operation code:

bit15	bit14	bit13	bit12	bit11	bit10	bit9.....bit7	bit6	bit5.....bit0
0	1	HLT	SIRQ	RND	CLP	SLQ	SLY	X/Y-Addr

### [bit13] **HLT** (HLT instruction flag)

Setting this bit causes the multiply-and-accumulate macro to halt program execution after instruction execution completes.

Clears the **RunDSP** flag in the DSP-CSR register.

### [bit12] **SIRQ** (INT instruction flag)

Setting this bit causes an interrupt request to be sent to the CPU after instruction execution completes, then sets the **IrqDSP** flag in the DSP-CSR register.

### [bit11] **RND** (Rounding)

This bit specifies whether to perform rounding for 16-bit data specified by the SLQ bits.

Rounding rounds the 16-bit data based on the bit immediately below the LSB (round up if lower bit is 1, ignore if 0).

### [bit10] **CLP** (Clipping)

This bit specifies the saturation of the 16-bit data specified in the SLQ bit if the calculation result of the accumulator is an overflow value of the 16-bit data.

In practice, saturation processing is performed if the MSB of the accumulator (bit39) is different to the MSB of the 16-bit value (determined by the SLQ bits). If rounding is enabled, the comparison is performed using the result after rounding.

The value transferred to data RAM is the maximum positive value (7FFF<sub>H</sub>) if the value of the accumulator was positive before rounding, or the maximum negative value (8000<sub>H</sub>) if negative.

The accumulator sign is saved without being inverted by rounding or saturation.

**[bit9 to bit7] SLQ**

Specifies which bits of the accumulator to transfer to data RAM.

SLQ Bit	Overflow evaluation bits	Transfers 16-bit Data	Rounding Bit	Fixed-point decimal format
0 0 0	bit39 to bit27	bit27 to bit12	bit11	Q12
0 0 1	bit39 to bit28	bit28 to bit13	bit12	Q13
0 1 0	bit39 to bit29	bit29 to bit14	bit13	Q14
0 1 1	bit39 to bit30	bit30 to bit15	bit14	Q15
1 0 0	bit39 to bit23	bit23 to bit8	bit7	Q8
1 0 1	bit39 to bit24	bit24 to bit9	bit8	Q9
1 1 0	bit39 to bit25	bit25 to bit10	bit9	Q10
1 1 1	bit39 to bit26	bit26 to bit11	bit10	Q11

**[bit6] SLY**

Specifies the transfer destination.

"0": X-RAM

"1": Y-RAM

**[bit5 to bit0] X/Y Addr (RAM Address)**

These bits specify direct address in data RAM.

## ■ JMP Instruction (Branch Instruction)

Operation:	[When condition consists]	DSP-PC ← J-Addr8
	[At the condition failure]	DSP-PC ← DSP-PC + 1
Explanation:	Branch if condition is satisfied, do not perform any operation if condition is not satisfied.	
Word count:	1 word (16-bit width)	
Cycle count:	1 system clock cycle	
Operation code:		

bit15	bit14	bit13	bit12	bit11	bit10...bit8	bit7.....bit0
0	0	HLT	SIRQ	COND	UBP2 to UBP0	J-Addr8

### [bit13] **HLT** (HLT instruction flag)

Setting this bit causes the multiply-and-accumulate macro to halt program execution after instruction execution completes.

Clears the **RunDSP** flag in the DSP-CSR register.

### [bit12] **SIRQ** (INT instruction flag)

Setting this bit generates an interrupt request to the CPU after instruction execution completes.

Sets the **IrqDSP** flag in the DSP-CSR register.

### [bit11] **COND** (CONDition)

"0": Unconditional divergence

"1": Conditional branch

### [bit10 to bit8] **UBP2** to **UBP0** (condition specification)

Sets the condition to use for the conditional branch. The condition is established if these bits match the value of the **USR2**, **USR1**, and **USR0** bits in the DSP-CSR register.

These bits must be set to "000<sub>B</sub>" if an unconditional branch is specified.

### [bit7 to bit0] **J-Addr8** (Jump Address)

These bits specify the address of a branch destination.





# **CHAPTER 18**

---

## ***DMAC***

### ***(DMA CONTROLLER)***

**This chapter explains the overview of the DMA controller (DMAC), the configuration and functions of registers, and DMAC operation.**

- 18.1 Overview
- 18.2 Register Details Explanation
- 18.3 Operation of DMAC (DMA Controller)
- 18.4 Operation Flowchart
- 18.5 Data Path

## 18.1 Overview

---

**DMAC is used to implement DMA (Direct Memory Access) transfer. Performing DMA transfer increases the system performance as various types of data can be transferred at high speed without going via the CPU.**

---

### ■ Hardware Configuration

The module consists of the following main circuits and registers.

- Independent DMA channel × 5 channels
- 5-channel independent access control circuit
- 20-bit address registers (Reloading permitted: ch.0 to ch.3)
- 24-bit address registers (Reloading permitted: ch.4)
- 16-bit transfer count registers (Reloading permitted: One per channel)
- 4-bit block count registers (one per channel)
- Two-cycle transfer

### ■ Prime Function

The main data transfer functions supported by this module are as follows.

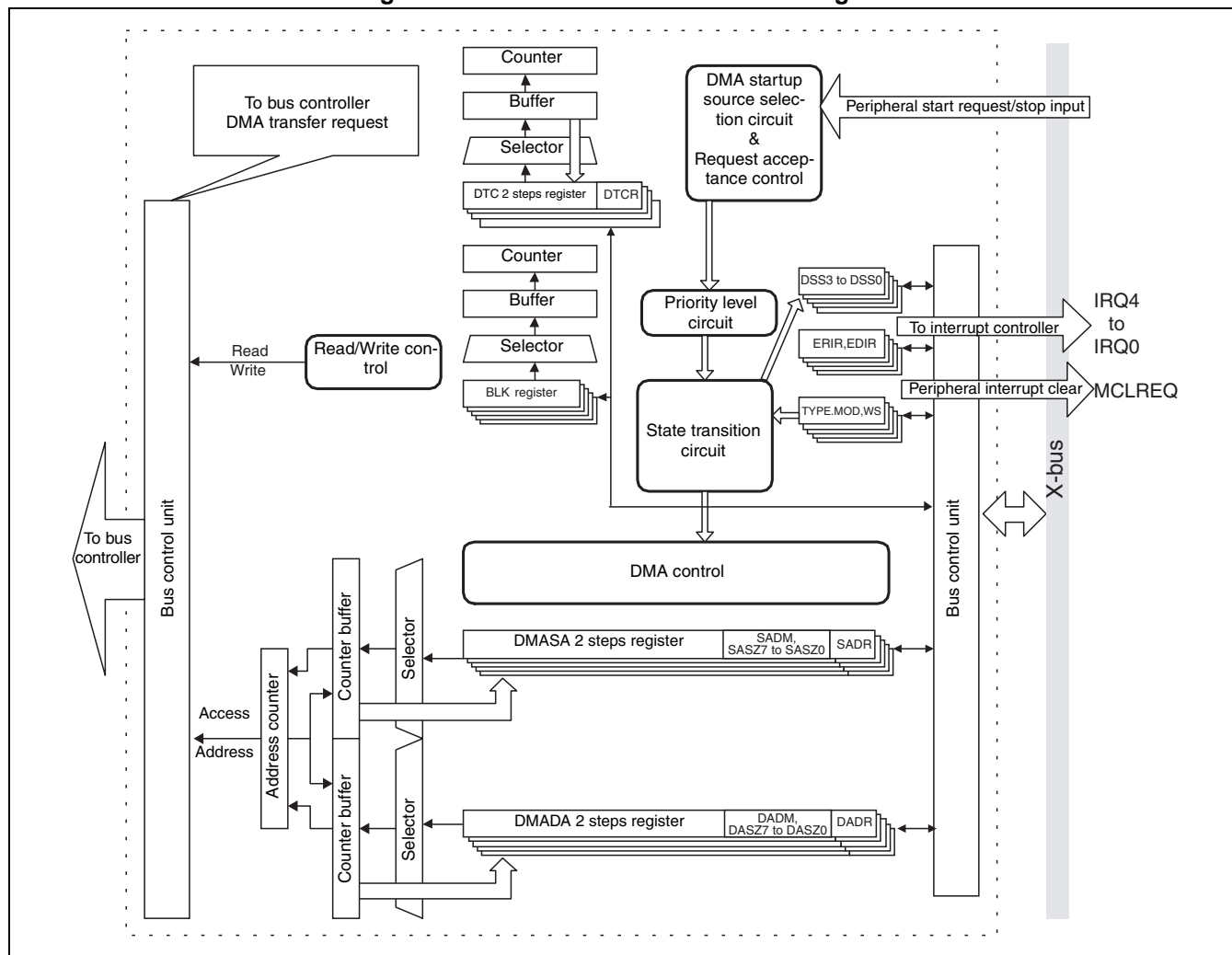
- Independent data transfer can be performed for multiple channels (5 channels)
  - (1) Priority order (ch.0>ch.1>ch.2>ch.3>ch.4)
  - (2) Alternating transfer is supported between ch.0 and ch.1.
  - (3) DMAC startup factor
    - Request from internal peripheral (uses interrupt requests, including the external interrupts)
    - Software request (register write)
  - (4) Transfer mode
    - Burst transfer/step transfer/block transfer
    - Addressing mode 20-bit (24-bit) addressing (incremental/decremental/fixed) (The address increment/decrement is fixed to be  $\pm 1$ ,  $\pm 2$ , or  $\pm 4$ .)
    - Data types byte/half-word/word length
    - Selectable single-shot or reload

## ■ Register Outline

ch.0 Control/Status Register A	DMACA0 000200 <sub>H</sub>	bit 31 24 23 16 15 08 07 00	
ch.0 Control/Status Register B	DMACB0 000204 <sub>H</sub>		
ch.1 Control/Status Register A	DMACA1 000208 <sub>H</sub>		
ch.1 Control/Status Register B	DMACB1 00020C <sub>H</sub>		
ch.2 Control/Status Register A	DMACA2 000210 <sub>H</sub>		
ch.2 Control/Status Register B	DMACB2 000214 <sub>H</sub>		
ch.3 Control/Status Register A	DMACA3 000218 <sub>H</sub>		
ch.3 Control/Status Register B	DMACB3 00021C <sub>H</sub>		
ch.4 Control/Status Register A	DMACA4 000220 <sub>H</sub>		
ch.4 Control/Status Register B	DMACB4 000224 <sub>H</sub>		
Overall DMAC control register	DMACR 000240 <sub>H</sub>		
ch.0 Transfer source address register	DMASA0 001000 <sub>H</sub>	bit 20 19 00	
ch.0 Transfer destination address register	DMADA0 001004 <sub>H</sub>		
ch.1 Transfer source address register	DMASA1 001008 <sub>H</sub>		
ch.1 Transfer destination address register	DMADA1 00100C <sub>H</sub>		
ch.2 Transfer source address register	DMASA2 001010 <sub>H</sub>		
ch.2 Transfer destination address register	DMADA2 001014 <sub>H</sub>		
ch.3 Transfer source address register	DMASA3 001018 <sub>H</sub>		
ch.3 Transfer destination address register	DMADA3 00101C <sub>H</sub>		
ch.4 Transfer source address register	DMASA4 001020 <sub>H</sub>	bit 31 24 23 00	
ch.4 Transfer destination address register	DMADA4 001024 <sub>H</sub>		

## ■ Block Diagram of DMAC

Figure 18.1-1 DMAC 5-channel Block Diagram



## 18.2 Register Details Explanation

This section describes notice when setting a register used in the DMA controller and details of a register.

### ■ Notes on Setting Registers

Some bits in the DMAC may only be set when the DMA is halted. If set during operation (during transfer), correct operation cannot be guaranteed.

An asterisk (\*) indicates bits that will affect operation if set during DMAC transfer. Only modify this bit when the DMAC transfer is halted (set to the disabled or paused state).

Values set while DMA transfer is disabled (DMACR:DMAE=0 or DMACA:DENB=0) become active when DMA is re-enabled.

Values set while DMA transfer is paused (DMACR:DMAH3 to DMAH0) ≠ 0000 or DMACA:PAUS=1) become active when DMA is restarted.

### ■ DMAC-ch.0,ch.1,ch.2,ch.3,ch.4 Control Status Registers A [DMACA: DMACA0 to DMACA4]

These registers control the operations of individual DMAC channels separately.

The function of each bit is as follows

Address	bit31	bit30	bit29	bit28	bit27	bit26	bit25	bit24	bit23	bit22	bit21	bit20	bit19	bit18	bit17	bit16															
ch.0: 000200 <sub>H</sub>	DENB	PAUS	STRG	IS4 to IS0					-				BLK3 to BLK0																		
ch.1: 000208 <sub>H</sub>	R/W	R/W	R/W	R/W					R/W																						
ch.2: 000210 <sub>H</sub>	bit15	bit14	bit13	bit12	bit10	bit10	bit9	bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0															
ch.3: 000218 <sub>H</sub>	DTC15 to DTC0																														
ch.4: 000220 <sub>H</sub>																															
R/W																															
(Initial value: 00000000 00000000 00000000 00000000 <sub>B</sub> )																															
R/W: Readable/Writable																															
-: Undfind																															

[bit31] DENB (Dma ENaBle): DMA Operation Enable Bit

Enables or disables DMA transfer for each transfer channel.

Once a channel is enabled, DMA transfer starts when a transfer request is received.

All transfer requests for disabled channels are ignored.

When data transfer of the started channel has been completed for the specified number of times, this bit is set to "0" and the transfer is stopped.

Writing "0" to this bit aborts DMA transfer. Before aborting it (writing "0"), be sure to set the PAUS bit (DMACA: bit30) to pause DMA. Although aborting DMA without pausing stops it, transferred data is not guaranteed. To check whether DMA has been stopped, use the DSS2 to DSS0 bits (DMACB:bit18 to bit16).

DENB	Function
0	Disable DMA transfer of a given channel. (Initial value)
1	Enable DMA transfer of a given channel.

- At reset, the bit is initialized to "0" upon acceptance of a stop request.
- Reading and writing are permitted.
- When all the channels are disabled by bit15 (DMAE bit) of the DMAC global control register (DMACR), an attempt to write "1" to this bit is ignored and the channels remain off. If all the channels are disabled by the DMAE bit when a channel is enabled with this bit, this bit is set to "0" to interrupt (abort) transfer.

[bit30] PAUS (PAUSE): Pause transfer

Pauses DMA transfer for the corresponding channel. No DMA transfer is performed from the time this bit is set until it is cleared again. (The DSS bits go to "1xx" when DMA is halted.)

If this bit is set before DMA is enabled, DMA remains paused.

New transfer requests are accepted while this bit is set but the transfers do not start until the bit is cleared. (See the "■ Transfer Request Acceptance and Transfer" of Section "18.3.3 General DMA Transfers".)

PAUS	Function
0	Enable DMA transfer of a given channel. (Initial value)
1	Suspend DMA transfer of a given channel.

- At reset: Initialized to "0".
- Reading and writing are permitted.

[bit29] STRG (Software TRiGger): Transfer request

Generates a DMA transfer request for the corresponding channel. When "1" is written to this bit, a transfer request is generated to start transfer a given channel from the point of completion of writing to the register.

However, if the corresponding channel is not enabled, writing to this bit is ignored.

Note: If a transfer request is generated with this bit at the same time as the channels are enabled by setting the DMAE bit, the transfer request is accepted to start transfer. If a transfer request is generated with this bit at the same time as "1" is written to the PAUS bit, the transfer request is accepted but DMA transfer is not started until the PAUS bit is reset to "0".

STRG	Function
0	Invalidity
1	DMA activation request

- At reset: Initialized to "0".
- The value read is always "0".
- Only a value of "1" is valid to be written; an attempt to write "0" has no effect on operation.

[bit28 to bit24] IS4 to IS0 (Input Select) \* : Transfer trigger selection

The transfer request triggers are selected as follows. However, the software transfer request triggered by the STRG bit remains available regardless of this setting.

IS	Function	Transfer Halt Request
00000 <sub>B</sub>	Software transfer request only	None
00001 <sub>B</sub> ↓ 01111 <sub>B</sub>	Setting disabled ↓ Setting disabled	
10000 <sub>B</sub>	UART0 (Reception completion)	
10001 <sub>B</sub>	UART1 (Reception completion)	Provided
10010 <sub>B</sub>	System reservation	
10011 <sub>B</sub>	UART0 (Transmit completion)	
10100 <sub>B</sub>	UART1 (Transmit completion)	None
10101 <sub>B</sub>	System reservation	
10110 <sub>B</sub>	External interrupt 0	
10111 <sub>B</sub>	External interrupt 1	
11000 <sub>B</sub>	Reload timer 0	
11001 <sub>B</sub>	Reload timer 1	
11010 <sub>B</sub>	Reload timer 2	
11011 <sub>B</sub>	Multiply-and-accumulate macro	
11100 <sub>B</sub>	PPG0	
11101 <sub>B</sub>	PPG1	
11110 <sub>B</sub>	PPG2	
11111 <sub>B</sub>	PPG4	

- Be initialized to "00000<sub>B</sub>" when resetting.
- Reading and writing are permitted.

Notes: • When an interrupt from a peripheral function is set as a DMA trigger (IS = 1xxxx), disable the interrupt for the selected function in the ICR register.

- Note also that, if the software transfer request is used to start DMA while triggering DMA transfer from a peripheral function interrupt is enabled, a factor clear signal is sent to the peripheral function when the transfer completes. As this may clear an existing transfer request, do not use the software transfer request function to start DMA while triggering DMA transfer from a peripheral function interrupt is enabled.



[bit23 to bit20] - : Undefined bits

- The value read is fixed to "0000<sub>B</sub>". Writing has no effect.

[bit19 to bit16] BLK3 to BLK0 (BLoCK size): Block size setting

Specifies the size for block transfer on the corresponding channel. The value set in these bits specifies the number of words to transfer for each transfer operation (or, more exactly, the number of times transfer of the specified data width is repeated). Always set "01<sub>H</sub>" (size 1) when not performing block transfer.

BLK3 to BLK0	Function
XXXX <sub>B</sub>	Specify the block size of a given channel.

- At reset: Initialized to "0000<sub>B</sub>".
- Reading and writing are permitted.
- Setting all the bits to "0" specifies a block size of 16 words.
- Reading always returns the block size (reload value).

[bit15 to bit0] DTC15 to DTC0 (Dma Terminal Count register) \* : Transfer count register

This register stores the number of transfers performed. Each register consists of 16 bits.

Each register has its own reload register. On channels that allow the transfer count register to be reloaded, the initial value is automatically reloaded to the register when the transfer completes.

DTC15 to DTC0	Function
XXXX <sub>H</sub>	Specify the number of transfers of a given channel.

When DMA transfer is started, the data in this register is stored to the counter buffer for the DMA dedicated transfer counter to decrement the value by 1 for each transfer unit. When DMA transfer completes, the value of the counter buffer is written back to this register and the DMA operation ends. Accordingly, you cannot use this register to read the specified number of transfers during a DMA operation.

- At reset: Initialized to "00000000 00000000<sub>B</sub>".
- Reading and writing are permitted. Always use half-word or word access to access the DTC register.
- Reading the register returns the count value. You cannot read the reload value.

## ■ DMAC-ch.0,ch.1,ch.2,ch.3,ch.4 Control Status Registers B [DMACB: DMACB0 to DMACB4]

These registers control the operations of individual DMAC channels separately.

The function of each bit is as follows.

Address	bit31	bit30	bit29	bit28	bit27	bit26	bit25	bit24	bit23	bit22	bit21	bit20	bit19	bit18	bit17	bit16
ch.0: 000204 <sub>H</sub>	TYPE1, TYPE0		MOD1, MOD0		WS1, WS0		SADM	DADM	DTCR	SADR	DADR	ERIE	EDIE	DSS2 to DSS0		
ch.1: 00020C <sub>H</sub>	R/W		R/W		R/W		R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		
ch.2: 000214 <sub>H</sub>	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
ch.3: 00021C <sub>H</sub>	SASZ7 to SASZ0								DASZ7 to DASZ0							
ch.4: 000224 <sub>H</sub>	R/W								R/W							

(Initial value: 00000000 00000000 00000000 00000000<sub>B</sub>)

R/W: Readable/Writable

An asterisk (\*) indicates bits that will affect operation if set during DMAC transfer. Only modify this bit when the DMAC is halted (set to the disabled or paused state).

[bit31, bit30] TYPE1, TYPE0 (TYPE) \* : Transfer type setting

Sets the type of operation for the corresponding channel as follows.

Two-cycle transfer mode: This transfer mode sets the transfer source address (DMASA) and transfer destination address (DMADA), then performs the specified number of read and write operations.

TYPE1, TYPE0	Function
00 <sub>B</sub>	Two-cycle transfer (Initial value)
01 <sub>B</sub>	Setting disabled
10 <sub>B</sub>	Setting disabled
11 <sub>B</sub>	Setting disabled

- Be initialized to "00<sub>B</sub>" when resetting.
- Reading and writing are permitted.
- Always set these bits to "00<sub>B</sub>".

[bit29, bit28] MOD1, MOD0 (MODE) \* : Transfer mode setting

Sets the operation mode for the corresponding channel as follows.

MOD1, MOD0	Function
00 <sub>B</sub>	Block/step transfer mode (Initial value)
01 <sub>B</sub>	Burst transfer mode
10 <sub>B</sub>	Setting disabled
11 <sub>B</sub>	Setting disabled

- Be initialized to "00<sub>B</sub>" when resetting.
- Reading and writing are permitted.

**[bit27, bit26] WS1, WS0 (Word Size) : Transfer data size selection**

Sets the transfer data size for the corresponding channel as follows. DMA performs the specified number of transfers using the data size specified in this register.

WS1, WS0	Function
00 <sub>B</sub>	Transfer in BYTE (Initial value)
01 <sub>B</sub>	Transfer in HALF-WORD
10 <sub>B</sub>	Transfer in WORD width
11 <sub>B</sub>	Setting disabled

- Be initialized to "00<sub>B</sub>" when resetting.
- Reading and writing are permitted.

**[bit25] SADM (Source-ADdr. count-Mode select) \* : Transfer source address count mode setting**

This bit specifies the manipulation of the transfer source address of a given channel for each transfer unit.

Address incrementing and decrementing is performed after each transfer in accordance with the "count size for the transfer source address" (SASZ). When the transfer completes, the next access address is written to the corresponding address register (DMASA).

Accordingly, the transfer source address register is not updated until the DMA transfer completes.

To use a fixed address, set this bit to "0" or "1", and set the address count sizes (SASZ and DASZ) to "0".

SADM	Function
0	Increment the transfer source address (Initial value).
1	Decrement the transfer source address.

- Be initialized to "0" when resetting.
- Reading and writing are permitted.

**[bit24] DADM (Destination-ADdr. Count-Mode select)\* :Transfer destination address count mode setting**

This bit specifies the manipulation of the transfer destination address of a given channel for each transfer unit.

Address incrementing and decrementing is performed after each transfer in accordance with the "count size for the transfer destination address" (DASZ). When the transfer completes, the next access address is written to the corresponding address register (DMADA).

Accordingly, the transfer destination address register is not updated until the DMA transfer completes.

To use a fixed address, set this bit to "0" or "1", and set the address count sizes (SASZ and DASZ) to 0.

DADM	Function
0	Increment the transfer destination address (Initial value).
1	Decrement the transfer destination address.

- Be initialized to "0" when resetting.
- Reading and writing are permitted.

**[bit23] DTCR (DTC-reg. Reload)\* : Reload setting for the transfer count register**

Controls the reload function for the transfer count register in the corresponding channel.

When reloading is enabled by this bit, the DMAC stops with the transfer count register value initialized upon completion of transfer and waits for a transfer request (generated with the STRG bit or with start request set by IS bits) (The DENB bit is not cleared when this bit is "1").

Transfer is forcibly halted when DENB=0 or DMAE=0 is set.

Disabling reloading of the transfer counter results in a single-shot transfer. That is, DMA halts when the transfer is complete even if reloading is specified in the address register. In this case, the DENB bit is cleared.

DTCR	Function
0	Disable the reloading of the transfer count register. (Initial value)
1	Enable the reloading of the transfer count register.

- Be initialized to "0" when resetting.
- Reading and writing are permitted.

**[bit22] SADR (Source-ADdr.-reg. Reload)\* : Reload setting for transfer source address register**

Controls the reload function for the transfer source address register in the corresponding channel.

When reloading is enabled by this bit, the transfer source address register is reloaded with its initial value when transfer completes.

Disabling reloading of the transfer counter results in a single-shot transfer. That is, DMA halts when the transfer is complete even if reloading is specified in the address register. In this case, the address register retains its initial value that is reloaded when DMA halts.

When this bit disables reloading, the value of the address register when transfer completes is the next access address after the final address (that is, if incrementing is enabled, it is the incremented address).

SADR	Function
0	Disable the reloading of the transfer source address register. (Initial value)
1	Enable the reloading of the transfer source address register.

- Be initialized to "0" when resetting.
- Reading and writing are permitted.

**[bit21] DADR (Dest.-ADdr.-reg. Reload)\* : Reload setting for transfer destination address register**

Controls the reload function for the transfer destination address register in the corresponding channel.

When reloading is enabled by this bit, the transfer destination address register contains its initial value when transfer completes.

Other details and functions are the same as described for bit22:SADR.

DADR	Function
0	Disable the reloading of the transfer destination address register. (Initial value)
1	Enable the reloading of the transfer destination address register.

- Be initialized to "0" when resetting.
- Reading and writing are permitted.

**[bit20] ERIE (ERror Interrupt Enable)\* : Error interrupt output enable**

This bit controls the generation of an interrupt upon termination in response to an error. The nature of the error is indicated by bits DSS2 to DSS0. Note that this interrupt occurs in response to not all termination events but specific ones only (See the description of the DSS2 to DSS0 bits).

ERIE	Function
0	Disable error interrupt request output. (Initial value)
1	Enable error interrupt request output.

- At reset: Initialized to "0".
- Reading and writing are permitted.

**[bit19] EDIE (EnD Interrupt Enable)\* : Enable end interrupt output**

Controls whether to output an interrupt when transfer ends normally.

EDIE	Function
0	Disable termination interrupt request output. (Initial value)
1	Enable termination interrupt request output.

- At reset: Initialized to "0".
- Reading and writing are permitted.

**[bit18 to bit16] DSS2 to DSS0 (Dma Stop Status)\* : Transfer halt cause indication**

This 3-bit code (termination code) indicates the reason why DMA transfer stopped or halted on the corresponding channel. The termination code meanings are as follows.

DSS2	Function	Interruption generation
0	Initial value	None
1	DMA suspended (DMAH, PAUS bit, interrupt, etc.)	None

DSS1, DSS0	Function	Interruption generation
00 <sub>B</sub>	Initial value	None
01 <sub>B</sub>	-	None
10 <sub>B</sub>	Transfer halt request	Error
11 <sub>B</sub>	Successful completion	End

The transfer stop request is only be set when a request from a peripheral circuit is used.

Note: The "Interruption generation" column indicates the type of interrupt request that could be generated.

- At reset: Initialized to "000<sub>B</sub>".
- Writing "000<sub>B</sub>" clears the bits.
- Although the bits can be either read or written, only "000<sub>B</sub>" is valid as a value written to the bits.

[bit15 to bit8] SASZ7 to SASZ0 (Source Addr count SiZe)\* : Count size setting for the transfer source address

These bits specify the increment and the decrement to the transfer source address (DMASA) of a given channel for each transfer. The value specified by these bits determines by how much the address is incremented or decremented for each transfer. Whether to increment or decrement the address is specified by the transfer source address count mode (SADM).

SASZ7 to SASZ0	Function
00 <sub>H</sub>	Address fixed
01 <sub>H</sub>	Transfer in BYTE
02 <sub>H</sub>	Transfer in HALF-WORD
04 <sub>H</sub>	Transfer in WORD
Excluding the above-mentioned	Setting disabled

- At reset: Initialized to "00000000<sub>B</sub>".
- Reading and writing are permitted.
- If setting other than a fixed address, ensure that the setting matches the transfer data size (WS).

[bit7 to bit0] DASZ7 to DASZ0 (Des Addr count SiZe)\* : Count size setting for the transfer destination address

These bits specify the increment and the decrement to the transfer destination address (DMADA) of a given channel for each transfer. The value specified by these bits determines by how much the address is incremented or decremented for each transfer. Whether to increment or decrement the address is specified by the transfer destination address count mode (DADM).

DASZ7 to DASZ0	Function
00 <sub>H</sub>	Address fixed
01 <sub>H</sub>	Transfer in BYTE
02 <sub>H</sub>	Transfer in HALF-WORD
04 <sub>H</sub>	Transfer in WORD
Excluding the above-mentioned	Setting disabled

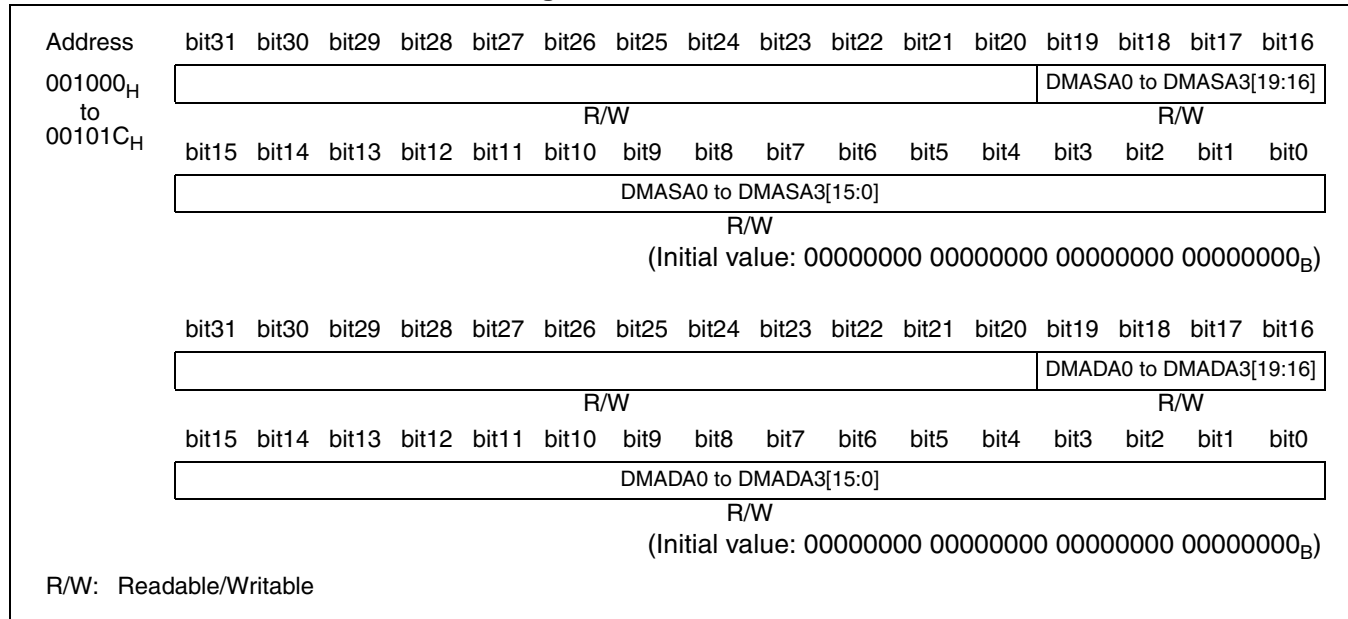
- At reset: Initialized to "00000000<sub>B</sub>".
- Reading and writing are permitted.
- If setting other than a fixed address, ensure that the setting matches the transfer data size (WS).

## ■ DMAC-ch.0,ch.1,ch.2,ch.3,ch.4 Transfer Source/Transfer Destination address Setting Registers [DMASA/DMADA: DMASA0 to DMASA4/DMADA0 to DMADA4]

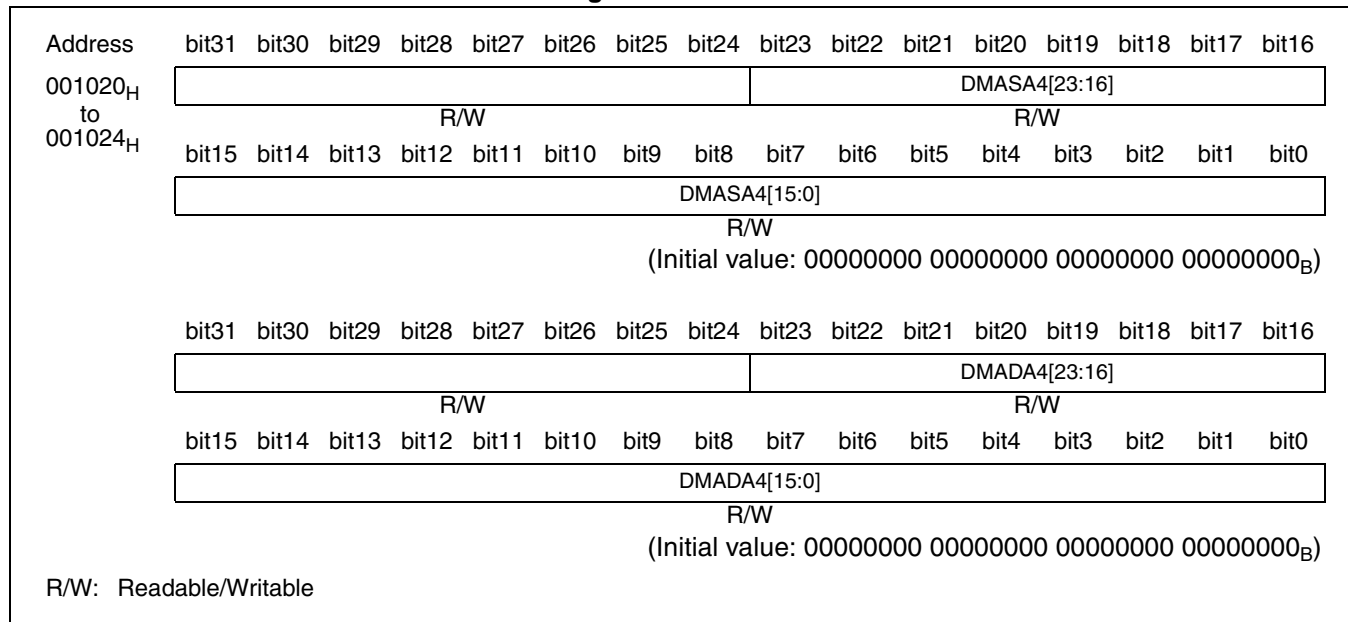
These registers control the operations of individual DMAC channels separately.

The function of each bit is as follows.

**Figure 18.2-1 ch.0 to ch.3**



**Figure 18.2-2 ch.4**



These registers store the transfer source and destination addresses. The registers for ch.0 to ch.3 are 20-bit and the registers for ch.4 are 24-bit.

An asterisk (\*) indicates bits that will affect operation if set during DMAC transfer. Only modify this bit when the DMAC is halted (set to the disabled or paused state).

- [bit19 to bit0] DMASA (DMA Source Addr) \* :  
                   ch.0 to ch.3 : DMASA19 to DMASA0 : Transfer source address setting
- [bit23 to bit0] DMASA (DMA Source Addr) \* :  
                   ch.4 : DMASA23 to DMASA0 : Transfer source address setting

Sets the transfer source address.

- [bit19 to bit0] DMADA (DMA Destination Addr) \* :  
                   ch.0 to ch.3 : DMADA19 to DMADA0 : Transfer destination address setting
- [bit23 to bit0] DMADA (DMA Destination Addr) \* :  
                   ch.4 : DMADA23 to DMADA0 : Transfer destination address setting

Sets the transfer destination address.

When DMA transfer is started, the data in these registers is stored to the counter buffer for the DMA dedicated address counter to count the address for each transfer according to the setting. When DMA transfer completes, the value of the counter buffer is written back to these registers and the DMA operation ends. Accordingly, you cannot read the value of the address counter during DMA operation.

Each register has its own reload register. When used on channels for which reloading the transfer source and transfer destination address registers is enabled, the registers are automatically reloaded with their initial values when transfer completes. However, this has no effect on other address registers.

- At reset: Initialized to "00000000 00000000 00000000 00000000<sub>B</sub>".
- Reading and writing are permitted. Always use 32-bit data access to read or write to these registers.
- During transfer, reading returns the address setting from before transfer started. After transfer completes, reading returns the next access address. You cannot read the reload value. Accordingly, you cannot read the transfer address in real time.
- Set "0" to the non-existent upper bits.

---

Note:

Do not set these registers to locate any register of the DMAC itself. Performing DMA transfer to registers in the DMAC is not permitted.

---



## ■ DMAC-ch.0,ch.1,ch.2,ch.3,ch.4 Overall DMAC Control Register [DMACR]

This register controls the overall operation of all five DMAC channels. Always use byte access to read or write to this register.

The function of each bit is as follows.

Address	bit31	bit30	bit29	bit28	bit27	bit26	bit25	bit24	bit23	bit22	bit21	bit20	bit19	bit18	bit17	bit16
000240 <sub>H</sub>	DMAE	–	–	PM01	DMAH3 to DMAH0				–	–	–	–	–	–	–	–
	R/W			R/W		R/W										
	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

(Initial value: 0XX00000 XXXXXXXX XXXXXXXX XXXXXXXX<sub>B</sub>)

R/W: Readable/Writable  
 -: Undefined

An asterisk (\*) indicates bits that will affect operation if set during DMAC transfer. Only modify this bit when the DMAC is halted (set to the disabled or paused state).

### [bit31] DMAE (DMA Enable) : DMA Enabling Operations

Controls operation for all DMA channels.

When DMA operation is disabled with this bit, all channels' transfer is disabled regardless of the start/stop setting or operating status of each channel. Any requests on channels for which transfer is in progress are cancelled and transfer halts at the next block boundary. When disabled, any operation to start transfer on any channel is ignored.

When DMA operation is enabled with this bit, it is possible to start or stop individual channels. Using this bit to enable DMA operation does not actually start transfer for any channel.

Writing "0" to this bit aborts DMA transfer. Before aborting it (writing "0"), be sure to use the DMAH3 to DMAH0 bits (DMACR: bit27 to bit24) to halt DMA. Although aborting DMA without pausing stops it, transferred data is not guaranteed. To check whether DMA has been stopped, use the DSS2 to DSS0 bits (DMACB:bit18 to bit16).

DMAE	Function
0	Disable DMA operation of all channels. (Initial value)
1	Enable DMA operation of all channels.

- At reset: Initialized to "0".
- Reading and writing are permitted.

### [bit28] PM01 (Priority Mode ch.0, ch.1 robin) : Alternate channel priority

Set this bit to toggle priority between ch.0 and ch.1 for each transfer.

PM01	Function
0	Use fixed priorities (ch.0 > ch.1). (Initial value)
1	Priority level rotation (ch.1 > ch.0)

- At reset: Initialized to "0".
- Reading and writing are permitted.

**[bit27 to bit24] DMAH3 to DMAH0 (DMA Halt) : DMA Suspended**

Pauses DMA for all DMA channels. Setting these bits pauses DMA transfer on all channels until the bits are cleared again.

If these bits are set before enabling DMA, all channels remain paused when DMA is enabled.

Any transfer requests that occur for channels with DMA enabled (DENB=1) while these bits are set are valid but transfer does not start until the bits are cleared.

DMAH3 to DMAH0	Function
0000	Enable DMA operation of all channels. (Initial value)
Other than "0000"	Suspend DMA operation of all channels.

- At reset: Initialized to "0".
- Reading and writing are permitted.

**[bit30, bit29, bit23 to bit0] - : Undefined bits**

- The read value is undefined.

## 18.3 Operation of DMAC (DMA Controller)

---

**This section explains an operation of the DMA controller.**

---

### ■ Main Operation

- The operation of each function can be set independently for each transfer channel.
- Once enabled, a channel does not actually start transfer until the specified transfer request is detected.
- On detecting a transfer request, the DMAC outputs a DMA transfer request to the bus controller and starts transfer on receiving bus access rights from the bus controller.
- The DMAC transfers data in the sequence determined by the mode set independently for each channel.

### ■ Transfer Mode

Each DMA channel operates in accordance with the transfer mode set by the MOD1 and MOD0 bits in its DMACB register.

#### ● Block/step transfer

Only transfers one block of data for each transfer request. Once the transfer completes, DMA removes its transfer request from the bus controller until the next transfer request is received.

1 block transfer unit: Specified block size (DMACA: BLK3 to BLK0)

#### ● Burst transfer

On receiving a transfer request, transfer continues for the specified number of transfers.

Specified number of transfers: Block size  $\times$  transfer count (DMACA: BLK3 to BLK0  $\times$  DMACA: DTC15 to DTC0)

### ■ Transfer Type

#### ● Two-cycle transfer (Normal transfer)

The DMA controller treats a read and write operation as a single step.

The DMAC reads the value from the address set in the transfer source register and then writes it to the address in the transfer destination register.

### ■ Transfer address

The DMAC uses the following type of addressing, in which addresses are set separately for the transfer source and destination of each channel.

#### ● Address setting for two-cycle transfer

Accesses the addresses read from the registers in which the address values have previously been set (DMASA and DMADA).

On receiving a transfer request, DMA stores the addresses from these registers in temporary storage buffers before starting the transfer.

For each transfer (access), the address counter generates the address to be accessed next (by incrementing, decrementing, or using a fixed value selectively) and returns it to a temporary storage buffer. The content of the temporary storage buffer is written back to the registers (DMASA, DMADA) upon completion of each block transfer unit.

Since the values in the address registers (DMASA, DMADA) are therefore updated only for each block transfer unit, the address currently accessed during transfer is not known in real time.

## ■ Number of Transfers and Ending Transfer

### ● Number of transfers

The transfer count register is decremented (by 1) upon completion of each block transfer unit. When the transfer count register reaches "0", indicating that the specified number of transfers have been completed, the DMAC is stopped or restarted with the exit code displayed.

Like the address registers, the transfer count register value is updated only for each block transfer unit.

If reloading the transfer count register is disabled, transfer ends. If enabled, the register is reloaded with its initial value and the DMAC waits for transfer to restart (DMACB:DTCCR).

### ● The end of transfer

Transfer can be ended by the following conditions. The termination code set when transfer ends indicates which condition occurred. (DMACB:DSS2 to DSS0)

- Completion of the specified number of transfers (DMACA:BLK3 to BLK0 × DMACA: DTC15 to DTC0) → Normal termination
- Transfer halt request received from peripheral circuit → error
- Reset occurred → reset

A transfer halt cause code (DSS) is set for each end condition and a transfer complete interrupt or transfer error interrupt can be generated.

## 18.3.1 Setting up Transfer Requests

---

The following two types of transfer requests can be used to start DMA transfer: internal peripheral request and software request. The software request can be used at any time regardless of other request settings.

---

### ■ Internal Peripheral Request

The transfer request is generated by an interrupt from an internal peripheral circuit.

The resource generating an interrupt for a transfer request is specified for each channel (DMACA:IS4 to IS0 = 1xxxx<sub>B</sub>).

---

Note:

As interrupt requests used to generate transfer requests are also passed to the CPU as interrupt requests, always disable these interrupts in the interrupt controller (ICR register).

---

### ■ Software Request

Writing to the trigger bit of a register generates a transfer request (DMACA:STRG).

This is independent of the above transfer requests from peripherals and is always available.

If a software request is set at the same time as transfer is enabled, the DMA transfer request to the bus controller is output immediately and transfer starts.

---

Note:

If a software request is issued to a channel that is also set internal request peripheral, a factor clear signal is sent to the peripheral when the transfer completes. As this may clear an existing transfer request, do not use the software transfer request function in this case.

---

## 18.3.2 Transfer Sequence

For each channel, it is possible to independently set the transfer type and transfer mode which determine the operation sequence to be followed after DMA transfer is started. (Settings for DMACB:TYPE1 and TYPE0, MOD1 and MOD0)

### ■ Selecting the Transfer Sequence

The following sequences can be selected by register settings.

- Burst two-cycle transfer
- Block/step two-cycle transfer

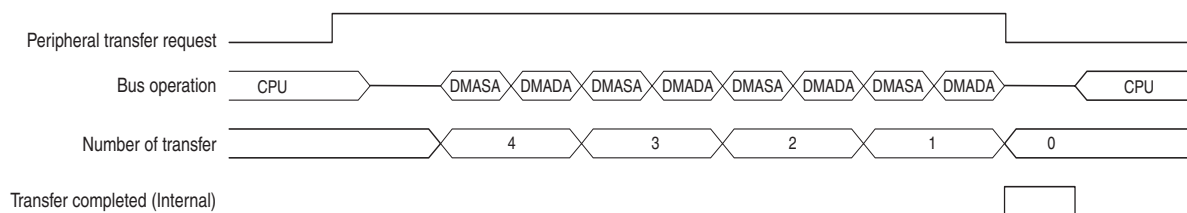
### ■ Burst Two-cycle Transfer

The specified number of transfers is performed each time transfer is invoked. In the case of two-cycle transfer, the transfer source and destination addresses can be specified as 20-bit addresses for ch.0 to ch.3, and as 24-bit addresses for ch.4.

Either a transfer request from a peripheral function or a software transfer request can be used to invoke transfer.

#### ● Burst transfer characteristics

- Each time a transfer request is received, transfer continues until the transfer count register reaches "0".
- The number of transfers is the block size multiplied by the number of blocks to be transferred. (DMACA:BLK3 to BLK0 × DMACA:DTC15 to DTC0)
- If another transfer request is received during a transfer, the request is ignored.
- When the reload function is enabled for the transfer count register, a subsequent transfer request is accepted only after transfer completes.
- If the DMAC accepts a transfer request for a channel during transfer of a different channel of a lower priority level, it switches the target to the higher-priority channel at the end of a block transfer unit and does not return to the previous channel until the transfer request is cleared.



(Example of burst transfer for a peripheral transfer request, number of blocks = 1, number of transfers = 4)

■ Step/block Transfer Two-cycle Transfer

For step/block transfer (transfer of the specified number of blocks for each transfer request), the transfer source and destination addresses can be specified in 20 bits for ch.0 to ch.3 or in 24 bits for ch.4.

● Step transfer

Setting the block size to "1" selects the step transfer sequence.

[Step transfer characteristics]

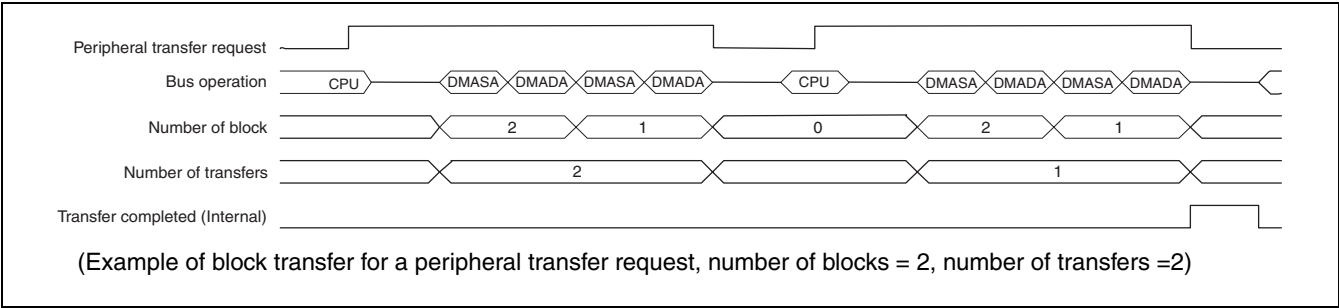
- Each time a transfer request is received, one transfer is performed, and then the transfer request is cleared and transfer halts. (The DMA transfer request is removed from the bus controller)
- If another transfer request is received during a transfer, the request is ignored.
- If the DMAC accepts a transfer request for a channel during transfer of a different channel of a lower priority level, it stops the current transfer and switches to the higher-priority channel to start transfer. For step transfer, priority is only meaningful for the case when transfer requests are generated simultaneously.

● Block transfer

Setting the block size to a value other than "1" selects the block transfer sequence.

[Block transfer characteristics]

Except for the fact that each transfer consists of multiple transfer cycles (specified by the number of blocks), the operation is the same as for step transfer.



### 18.3.3 General DMA Transfers

---

**This section explains a block size of the DMA transfer and a reload operation.**

---

#### ■ Block Size

- The value set in the block size setting register specifies the volume of data for each transfer operation (× data size).
- The size of the data transferred in each transfer cycle is determined by the data size setting, and each transfer consists of the number of transfer cycles specified by the block size.
- During a transfer, if a transfer request for a channel with a higher priority is received or if a transfer pause request is issued, block transfer does not halt until the current block has completed. Although this prevents undesirable splitting or pausing within a data block, it causes the response to be slower if the block size is large.
- In the case of reset, transfer halts immediately and therefore the validity of the transferred data cannot be guaranteed.

#### ■ Reload Operation

This module allows the following three types of reload functions to be set for individual channels:

##### ● Transfer count register reload function

When the specified number of transfers completes, the transfer count register is reloaded with its initial value and the DMAC waits for the next trigger.

Use this setting to perform any of the transfer sequences repeatedly.

If reloading is not enabled, the count register remains at "0" after the specified number of transfers complete and no further transfers are performed.

##### ● Reload function for transfer source address register

The transfer source address register is reloaded with its initial value after the specified number of transfers complete.

Use this setting if repeatedly performing a transfer from a fixed region in the transfer source address range.

If reloading is not enabled, the value of the next transfer address remains set in the transfer source address register after the specified number of transfers complete. Use this setting if the address range is not fixed.

##### ● Reload function for transfer destination address register

The transfer destination address register is reloaded with its initial value after the specified number of transfers complete.

Use this setting if repeatedly performing a transfer to a fixed region in the transfer destination address range.

If reloading is not enabled, the value of the next transfer address remains set in the transfer source address register after the specified number of transfers complete. Use this setting if the address range is not fixed.

- Enabling the reload functions for transfer source and destination address registers does not restart after the specified number of transfers complete. It only causes the address registers to be reloaded with their initial value.



**Note:**

Special operation modes and reload operation cases

- To start over from input detection after stopping upon completion of transfer, do not specify reloading.
  - When burst, block, or step transfer is completed, transfer is suspended after reloading and is not restarted until the input of a new transfer request is detected.
- 

## ■ Addressing Mode

Specify the transfer destination and transfer source addresses independently for each transfer channel.

The following procedure can be used to set the registers. Setup the registers in accordance with the transfer sequence.

### ● Setting the address registers

- In two-cycle transfer mode, set the transfer source address in the transfer source address setting register (DMASA) and set the transfer destination address in the transfer destination address setting register (DMADA).

### ● Features of address registers

- These registers are 20-bit for ch.0 to ch.3 and 24-bit for ch.4.

### ● Functions of address registers

- The registers are read each time an access is performed and output on the address bus.
  - At the same time, the address counter is used to calculate the address for the next access and the result of this calculation is set in the address register.
  - The address calculation is performed independently for each channel, transfer source, and transfer destination. Either incrementing or decrementing can be selected. The size of the address increment or decrement is specified by the address count size setting register (DMACB: SASZ, DASZ).
  - When the reload function is not enabled for an address register, the result of the final address calculation remains in the register after the transfer ends.
  - If the reload function is enabled, the initial value of the address is reloaded.
- 

**Notes:**

- Even when 20-bit or 24-bit full address calculation results in an overflow or underflow, the DMAC continues transfer of the current channel. Set each channel such that neither overflow nor underflow may occur.
  - Do not set the addresses of registers in the DMAC in the address registers.
-

## ■ Kinds of Data

The data length (x data size) transferred in each transfer cycle can be selected from the byte, halfword, word options.

### ● Byte, half-word, word

As word boundaries still apply during DMA transfer, setting a transfer source or destination address setting that conflicts with the data length causes the lower bits to be ignored.

- word: Actually accessed addresses are four bytes with lower two bits beginning with "00<sub>B</sub>".
- half-word: Actually accessed addresses are two bytes with lower one bit beginning with "0".
- byte: The specified address is used as the actual access address.

If the lower bits of the transfer source and destination addresses do conflict, the specified address is output on the internal address bus without modification, but the actual bus access occurs at the address corrected in accordance with the above rules.

## ■ Control of the Number of Transfers

The number of transfers can be set to any 16-bit value (1 to 65536 transfers). Set the number of transfers in the transfer count register (DMACA:DTC).

The register value is copied to a temporary storage buffer when transfer starts, and is decremented by the transfer counter. When the counter value reaches "0", it is detected as the completion of the specified number of transfers, the DMAC waits for a request to stop or restart transfer of the current channel (with reload in specified).

### ● Features of transfer count registers

- Each register is 16-bit.
- Each register has its own reload register.
- When started with a register value of "0", transfer is performed 65536 times.

### ● Reload operation

- Only used for registers with a reload function and for which the reload function is enabled.
- The initial value of the count register is saved in the reload register when transfer starts.
- When the transfer counter reaches to "0", the DMAC reports the completion of transfer, reads the initial value from there reload register, and writes it to the count register.

## ■ CPU Control

When a DMA transfer request is received, DMA issues a transfer request to the bus controller.

The bus controller assigns usage rights for the internal bus to DMA at the next appropriate timing in bus operation, and then DMA transfer starts.

### ● DMA transfer and interrupts

- During DMA transfer, interrupts are generally not received until the transfer ends.
- If a DMA transfer request occurs during interrupt processing, the transfer request is received and interrupt processing halts until the transfer completes.
- Exceptionally, when interrupt requests which are higher level than an NMI request or a hold suppress level set by an interrupt controller generate, DMAC temporarily cancels a transfer request to a bus controller at a boundary of transfer unit (1 block) and suspends transfer until the interrupt request is cleared. The transfer request is held internally during this time. Once the interrupt request is cleared, the DMAC issues another transfer request to the bus controller to obtain bus rights and restarts DMA transfer.

### ● Overriding DMA

- If an interrupt with a higher priority occurs during DMA transfer, DMA transfer halts and control branches to the interrupt routine. This mechanism remains active while the interrupt request is present. The DMA halt is removed as soon as the interrupt is cleared, and so the DMA transfer restarts from inside the interrupt handler routine. Accordingly, in handler routines for interrupts with a level that causes DMA transfer to be halted, use the DMA halt function if you wish to prevent DMA from restarting as soon as the interrupt is cleared. The DMA halt function is invoked by writing a non-zero value to the DMAH3 to DMAH0 bits in the DMA overall control register. The override is cleared by setting the bits back to "0".
- This function is primarily used in interrupt handler routines. Increment the DMA halt register by one in the interrupt handler routine before clearing the interrupt. This prevents any subsequent DMA transfer. When interrupt processing is completed, decrement the DMAH3 to DMAH0 bits by one before exiting the routine. If multiple interrupts have occurred, DMA transfer continues to be suppressed since the DMAH3 to DMAH0 bits are not "0" yet. If a single interrupt has occurred, the DMAH3 to DMAH0 bits become "0". DMA requests are then enabled immediately.

---

#### Notes:

- As the register is four-bit long, this function cannot be used with a multi-level interrupt exceeding 15 levels.
  - Be sure to assign the priority of the DMA tasks at a level that is at least 15 levels higher than other interrupt levels.
-

## ■ Start Operation

Although the start of DMA transfer is controlled independently for each channel, all the channels must be enabled collectively for operation in advance.

### ● All channel operations enable

Before activating each DMAC channel, operation for all channels needs to be enabled in advance with the DMA operation enable bit (DMAE of DMACR). All start settings and transfer requests that occurred before operation is enabled are invalid.

### ● Start transfer

The transfer operation can be started by the operation enable bit of the control register for each channel. If a transfer request to an activated channel is accepted, the DMA transfer operation is started in the specified mode.

### ● Starting from a temporary stop

If a temporary stop occurs before starting with channel-by-channel or all-channel control, the temporary stopped state is maintained even though the transfer operation is started. If transfer requests occur in the meantime, they are accepted and retained. When temporary stopping is released, transfer is started.

## ■ Transfer Request Acceptance and Transfer

This section explains transfer request acceptance and transfer of DMA transfer.

### ● Transfer request acceptance and transfer

- Sampling for transfer requests set for each channel starts after starting.
- If peripheral interrupt start is selected, DMAC continues the transfer until all transfer requests are cleared. When they are cleared, DMAC stops the transfer after one transfer unit (peripheral interrupt start). Since peripheral interrupts are handled as level detection, use interrupt clear by DMA to handle the interrupts.
- Transfer requests are always accepted even while transfer is being performed at an accepted request for other channel. The channel for transfer is determined for each transfer unit by checking the priorities of channels.

## ■ Clearing Peripheral Interrupts by the DMA

This DMA has a function that clears peripheral interrupts. This function works when peripheral interrupt is selected as the DMA start source (At IS4 to IS0 = 1xxxx<sub>B</sub>).

Peripheral interrupts are cleared only for the set start sources. That is, only the peripheral functions set by IS4 to IS0 are cleared.

### ● Interrupt clear signal generation timing

The timing for clearing an interrupt depends on the transfer mode (Refer to Operation flow).

- Block/step transfer

When block transfer is selected, a clear signal is generated for each block (step) transferred.

- Burst transfer

When burst transfer is selected, the clear signal is generated as all of the specified number of transfers have been completed.

## ■ Suspend

The temporary stopping of DMA transfer occurs in the following cases.

- Setting a halt by writing to a control register (to set each channel separately or all the channels at one time)

If temporary stopping is set using the temporary stop bit, transfer on the corresponding channel is stopped until release of temporary stopping is set again. You can check the DSS bits for temporary stopping.

Transfer is restarted when temporary stopping is cancelled.

- NMI/hold suppress level interrupt processing

If an NMI request or an interrupt request with a higher level than the hold suppress level occurs, all channels on which transfer is in progress are stopped at the boundary of the transfer unit and the bus right is opened to give priority to NMI/interrupt. Transfer requests accepted during NMI/interrupt processing are returned, initiating a wait for completion of NMI processing.

Channels for which requests are retained restart transfer after NMI/interrupt processing is completed.

## ■ Operation End/Stopping

Although the termination of DMA transfer is controlled independently for each channel, all the channels can also be disabled collectively for operation.

- The end of transfer

If reloading is disabled, transfer is stopped, "Normal end" is displayed as the end code, and all transfer requests are disabled after the transfer count register becomes "0" (Clear the DMACA:DENB bit).

If reloading is enabled, the initial value is reloaded, "Normal end" is displayed as the end code, and a wait for transfer requests starts after the transfer count register becomes "0" (Do not clear the DMACA:DENB bit).

- Disabling all channels

If the operation of all channels is disabled with the DMA operation enable bit DMAE, all DMAC operations, including operations on active channels, are stopped. From then on, even though all the channels are enabled for DMA operation back again, DMA transfer is not performed unless restarted for each channel. In this case, no interrupt whatever occurs.

## ■ Stopping Due to an Error

In addition to normal end after transfer for the number of times specified, stopping as the result of various types of errors and the forced stopping are provided.

### ● Transfer stop requests from peripheral circuits

Depending on the peripheral circuit that outputs a transfer request, a transfer stop request is issued when an error is detected (Example: Error when data is received at or sent from a communications system peripheral).

The DMAC, when it receives such a transfer stop request, displays "Transfer stop request" as the end code and stops the transfer on the corresponding channel.

---

#### Notes:

- For the availability of a transfer stop request from peripheral circuits, see the description of specifications for transfer source selection bits, which is the bit28 to bit24 (IS4 to IS0) of DMACA register.
  - For details of the conditions under which a transfer stop request is generated, see the specifications for each peripheral circuit.
- 

## ■ DMAC Interrupt Control

The following interrupts can be output for each DMAC channel, separately from peripheral interrupts as transfer requests:

- Transfer end interrupt: Occurs only when operation ends normally.
- Error interrupt: Transfer stop request from peripheral circuit (Peripheral-originated error)

All of these interrupts are output depending on the content of exit code.

An interrupt request is cleared by writing "000<sub>B</sub>" to the DSS2 to DSS0 (exit code) bits of the DMACS register. Note that the exit code must be cleared by writing "000<sub>B</sub>" whenever DMA transfer is restarted.

Although DMA transfer is restarted automatically when reloading is enabled, the exit code is not cleared then and remains preserved until new exit code is written upon completion of transfer.

Since only one end source can be displayed in an end code, the result after considering the order of priority is displayed when multiple sources occur simultaneously. The interrupt that occurs at this point conforms to the displayed end code.

The following shows the priority for displaying end codes (in order of decreasing priority)

- Reset
- Clearing by writing "000<sub>B</sub>"
- Peripheral stop request
- Successful completion
- Channel selection and control

## ■ DMA Transfer during the Sleep Mode

The DMAC can operate even in the sleep mode.

If you anticipate operations during the sleep mode, note the following:

- Since the CPU is stopped, DMAC registers cannot be rewritten. Make settings before the sleep mode is entered.
- The sleep mode is released by an interrupt. Thus, if a peripheral interrupt is selected as the DMAC start source, interrupts must be disabled by the interrupt controller.

Similarly, if you do not want to release the sleep mode with a DMAC end interrupt, disable DMAC end interrupts.

## ■ Channel Selection and Control

Up to five channels can be simultaneously set as transfer channels. In general, an independent function can be set for each channel.

### ● Channel priority

Since DMA transfer is possible only on one channel at a time, priority must be set for the channels.

Channel priority is set in two modes, fixed and toggled, which can be selected for each channel group (described later).

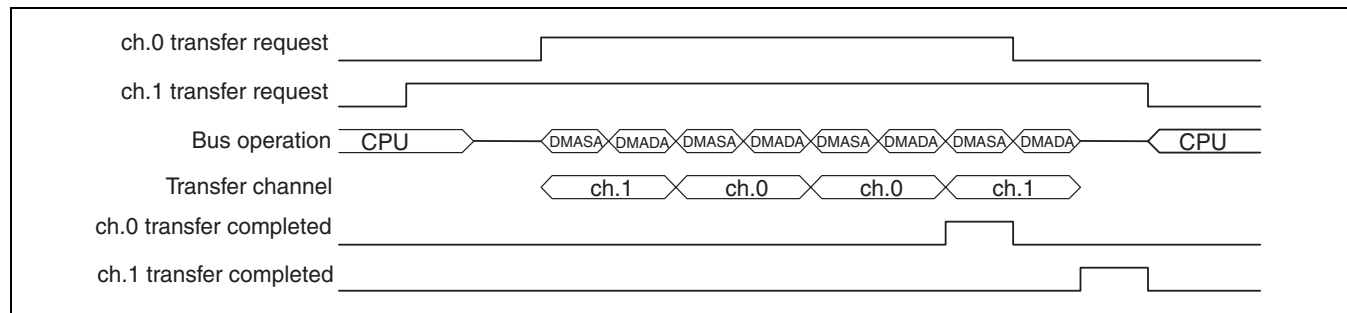
#### (1) Fix mode

The priority is fixed by channel number in ascending order.

(ch.0 > ch.1 > ch.2 > ch.3 > ch.4)

If a transfer request with a higher priority is received during a transfer, the transfer channel becomes the channel with the higher priority when the transfer for the transfer unit (number set in the block size specification register × data width) ends.

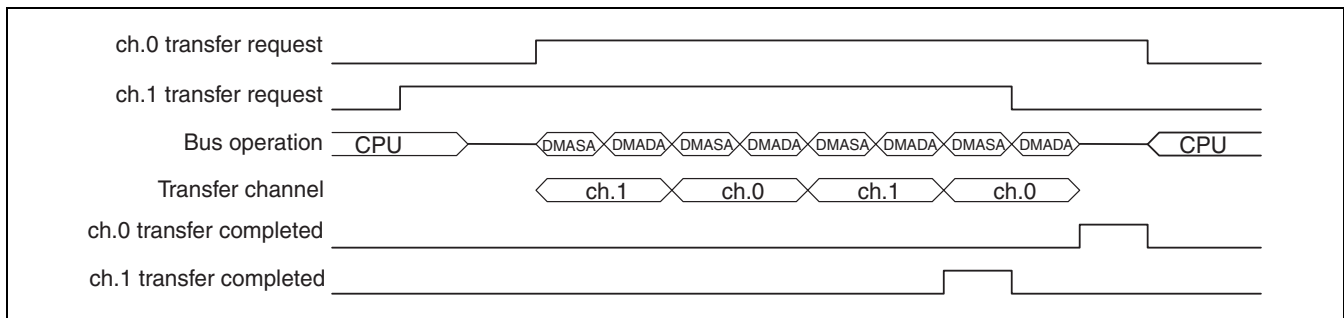
When higher priority transfer is completed, transfer is restarted on the previous channel.



#### (2) Rotation mode (only ch.0 to ch.1)

In the initial state after operation is enabled, the priorities of channels are set in the same order as in (1). The priorities are however inverted upon completion of each transfer. If transfer requests for channels are output, therefore, the channel is switched for each transfer unit.

This mode is effective when continuous or burst transfer is set.



### ● Channel group

Set the selection priority as explained in the table below.

Mode	Priority	Remark
Fixed	ch.0>ch.1	
Rotation	ch.0>ch.1 ↑ ↓ ch.0<ch.1	The upper order is the initial priority. The priority is inverted upon transfer in the upper order.

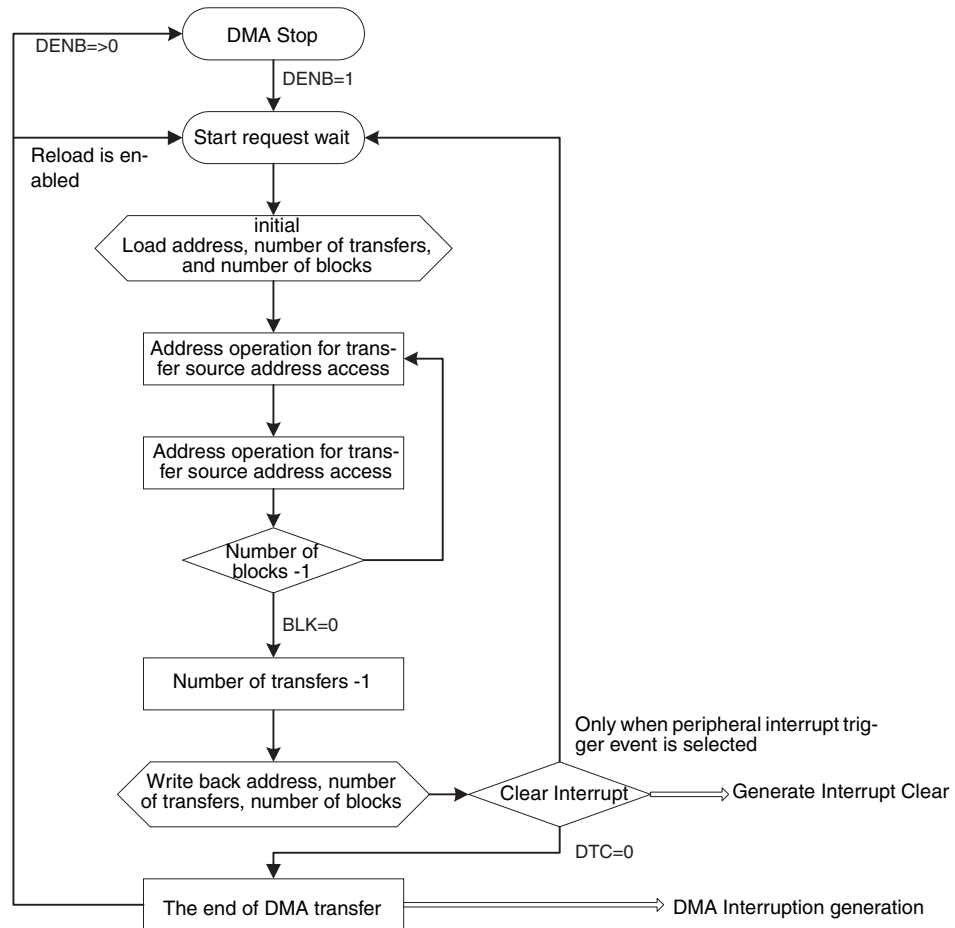


## 18.4 Operation Flowchart

The operation flowchart on the following transfer modes is shown.

- Block transfer
- Burst transfer

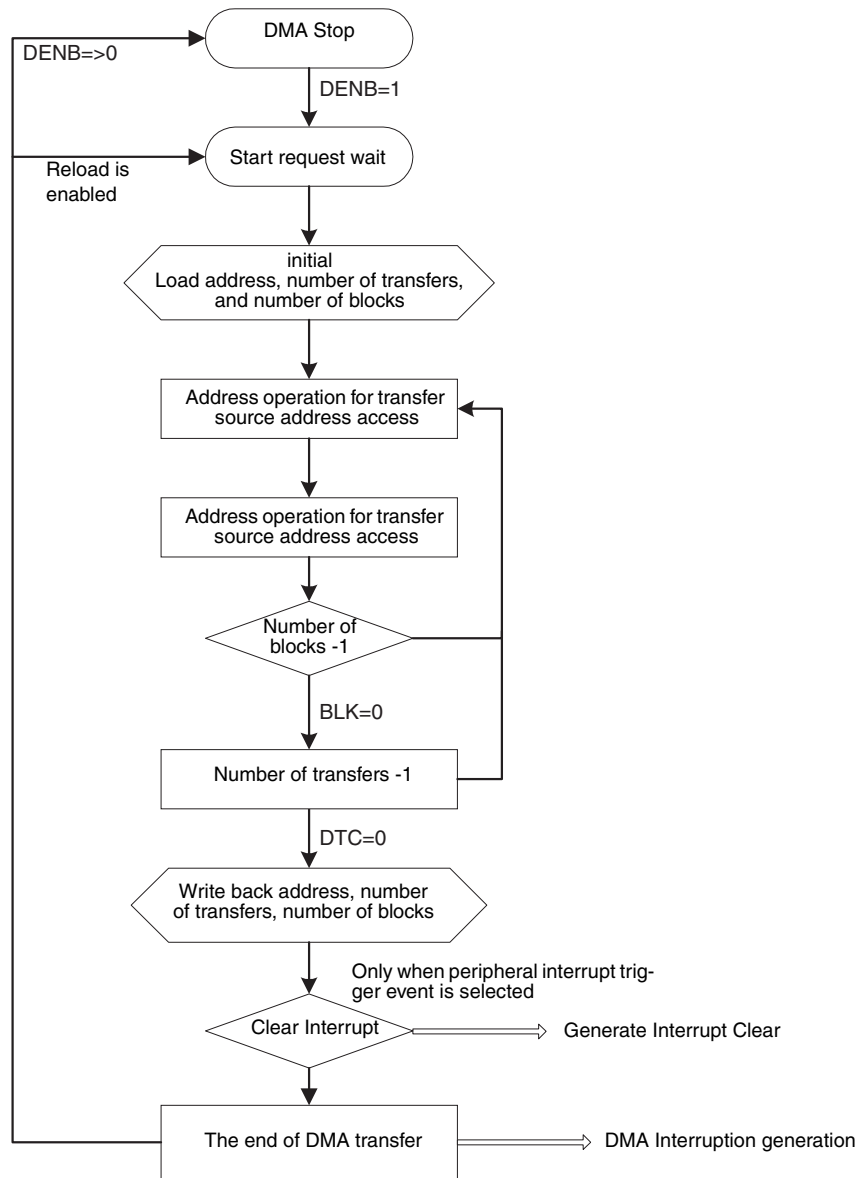
### ■ Block Transfer



#### Block transfer

- Transfer can be started at all trigger events. (Select)
- All areas are accessible.
- The number of blocks can be set.
- An interrupt clear signal is issued upon completion of the number of blocks.
- A DMA interrupt is issued upon completion of the specified number of transfers.

## ■ Burst Transfer



### Burst transfer

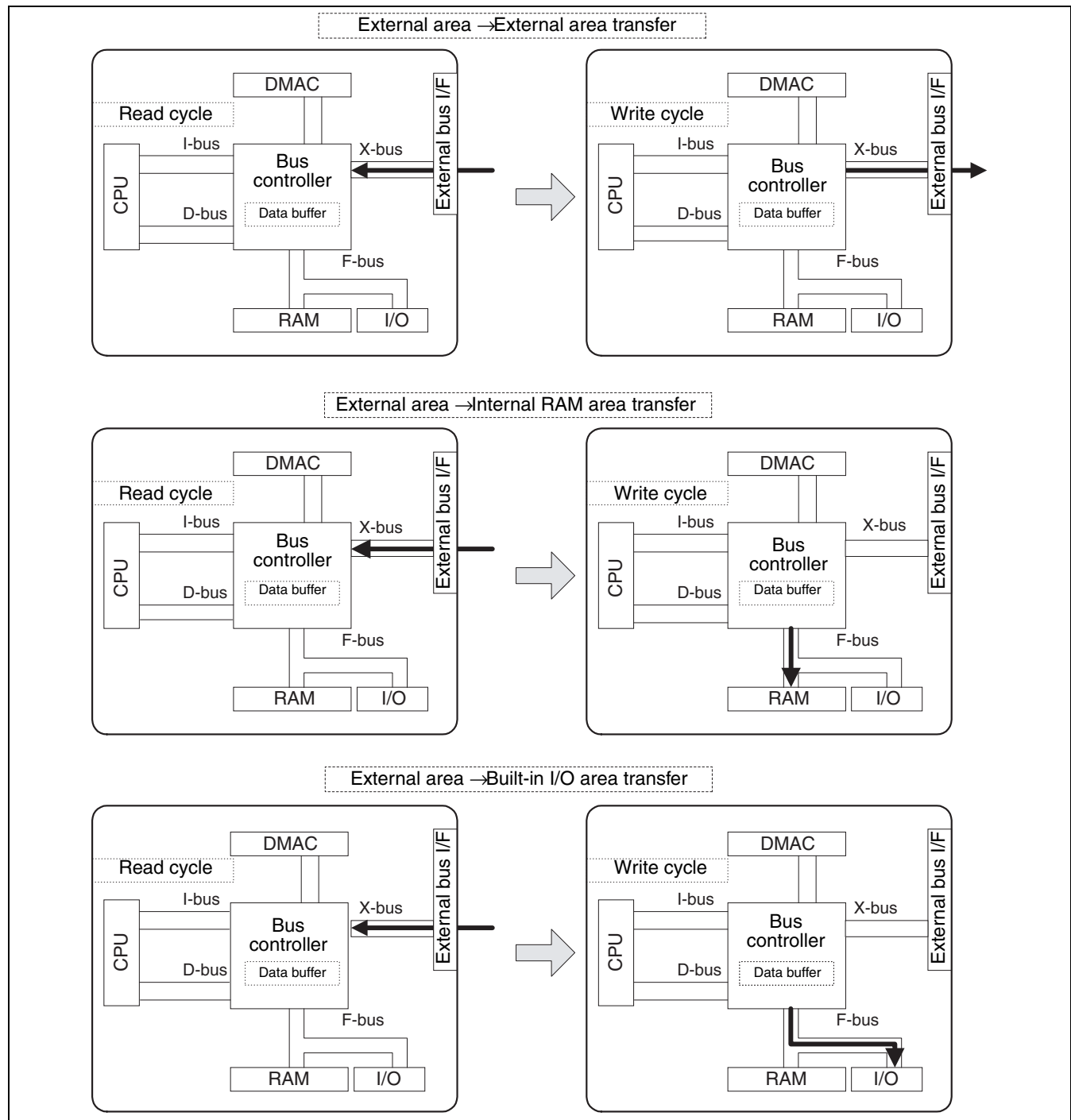
- Transfer can be started at all trigger events. (Select)
- All areas are accessible.
- The number of blocks can be set.
- An interrupt clear signal and DMA interrupt are issued upon completion of the specified number of transfers.

## 18.5 Data Path

Data operation in the two-cycle transfer mode is shown below.

### ■ Operation of Data in Two-cycle Transfer Mode

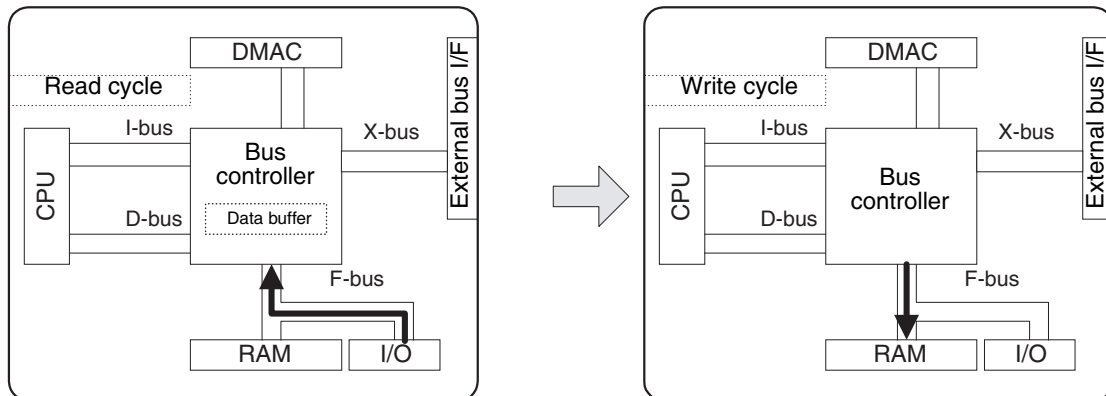
The following examples illustrate six types of transfer (Other combinations are omitted).



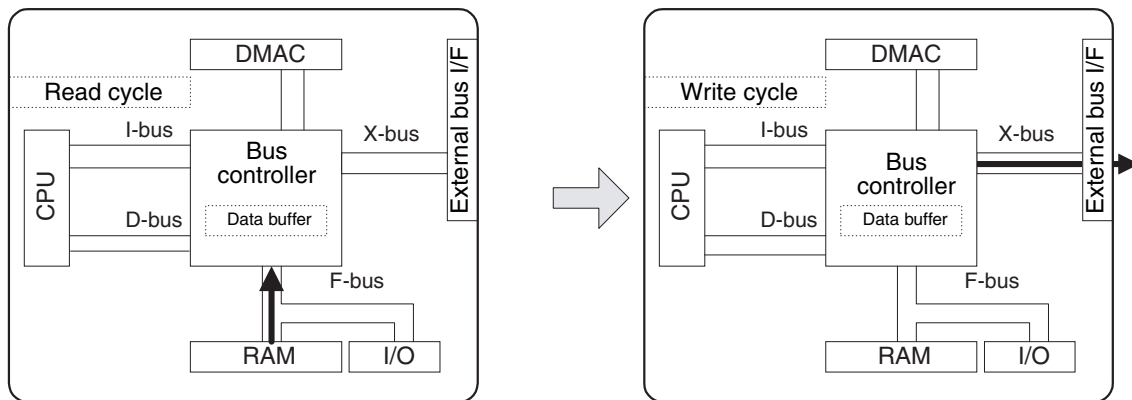
(Continued)

(Continued)

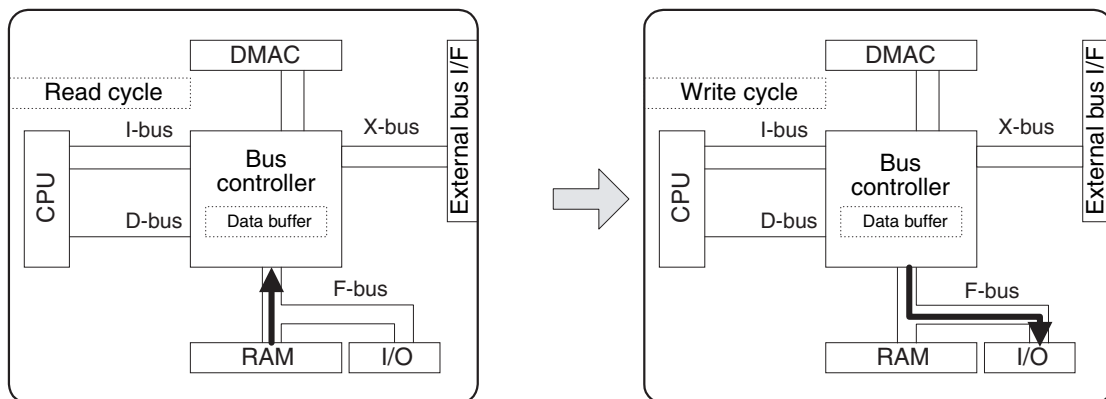
Built-in I/O area → Internal RAM area transfer



Internal RAM area → External area transfer



Internal RAM area → Built-in I/O area transfer





# ***CHAPTER 19***

---

# ***FLASH MEMORY***

**This chapter explains the overview of the flash memory, the configuration and functions of registers, and the flash memory operation.**

- 19.1 Overview
- 19.2 Registers
- 19.3 Access Mode of Flash Memory
- 19.4 Automatic Algorithm Startup Procedures
- 19.5 Automatic Algorithm Execution States
- 19.6 Dual-operations

## 19.1 Overview

---

**The MB91F267A/MB91F267NA contains 128-KB (1-megabit) flash memory which can be erased either for all sectors collectively or sector by sector with a + 3.3 V single power supply and can be programmed by the FR-CPU in half-words or in bytes (in 16 or 8 bits).**

---

### ■ Flash Memory Outline

The flash memory is built-in 128-KB flash memory driven at 3.3 V. It is the same as the MBM29LV400TC, 4-Mbit ( $512\text{-KB} \times 8 / 256\text{-KB} \times 16$ ) flash memory manufactured by Fujitsu (except the capacity and part of the sector configuration) which can be programmed from outside the device by using a ROM programmer.

In addition, the flash memory has MBM29LV400TC - equivalent functions and it allows instructions and data to be read from in words (32 bits), contributing to high-speed operation of the device when used as the built-in ROM for FR-CPU.

You should read the MBM29LV400TC Data Sheet along with this manual.

The combination of the flash memory macro and the FR-CPU interface circuit provides the following features:

- Serving as CPU's memory for storing programs and data
  - Accessible at 32-bit bus width when used as ROM
  - Capable of being read/programmed/erased by the CPU (using the automatic program algorithm \*)
- Functions equivalent to those of a discrete flash memory product, MBM29LV400TC
  - Capable of being read/programmed/erased by a ROM programmer (using the automatic program algorithm \*)

\*: Automatic program algorithm (Embedded Algorithm™)

Embedded Algorithm™ is a trademark of Advanced Micro Devices, Inc.

This section describes the use of the flash memory from the FR-CPU.

For details on using this flash memory with a ROM programmer, refer to the instruction manual for the ROM programmer.

### ■ Automatic Algorithm Execution States

If you start the automatic algorithm in CPU programming mode, you can check the operating state of the automatic algorithm with hardware sequence flags.

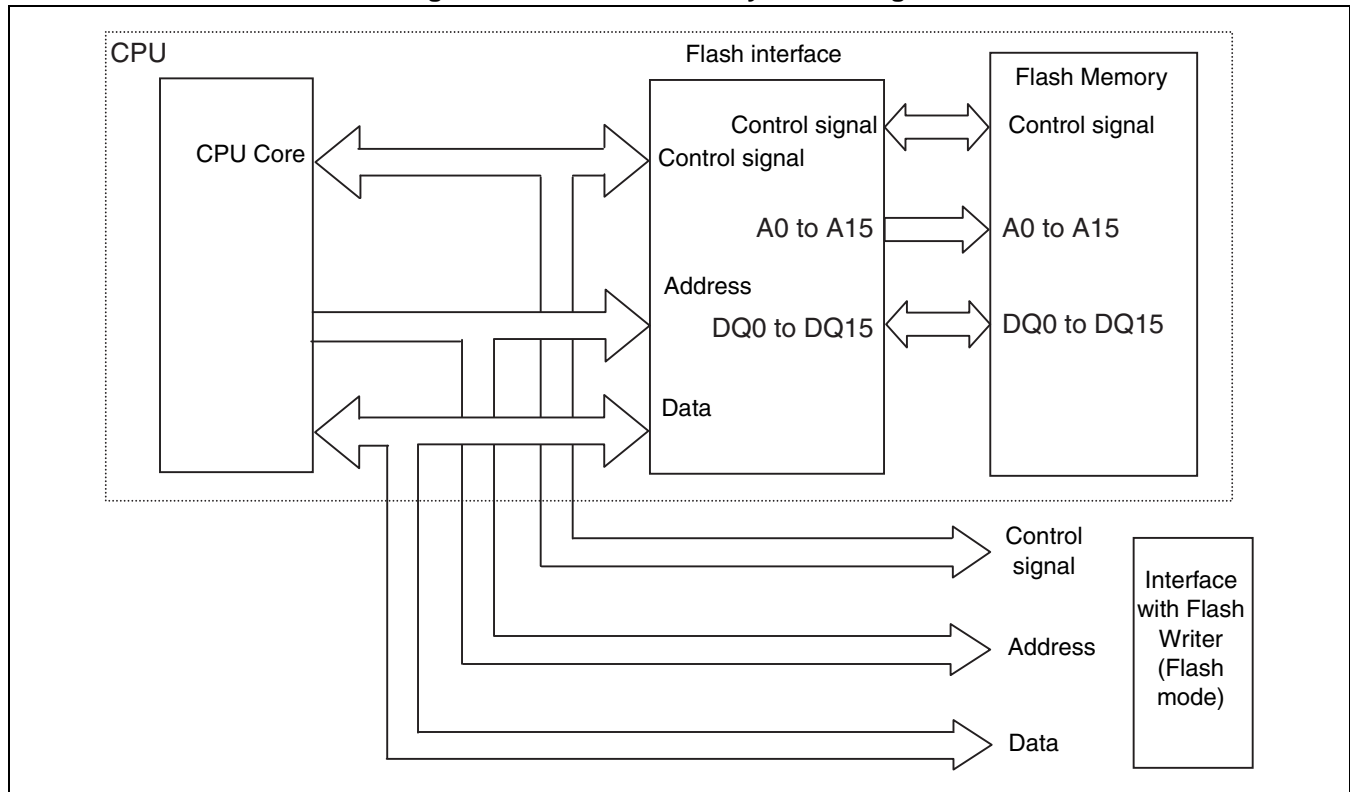
For hardware sequence flags, see Section "19.4 Automatic Algorithm Startup Procedures".

## ■ Programming with a ROM Programmer

This flash memory can be programmed from outside the device by using a ROM programmer. In this state, pin functions equivalent to those of a discrete flash memory product, MBM29LV400TC, are assigned to external pins of the device, where the FR-CPU remains inactive. The connections of address lines are changed from those in CPU mode and the memory areas are mapped differently as well. For details, refer to the "Specifications of Applicable ROM Programmers".

## ■ Flash Memory Block Diagram

**Figure 19.1-1 Flash Memory Block Diagram**



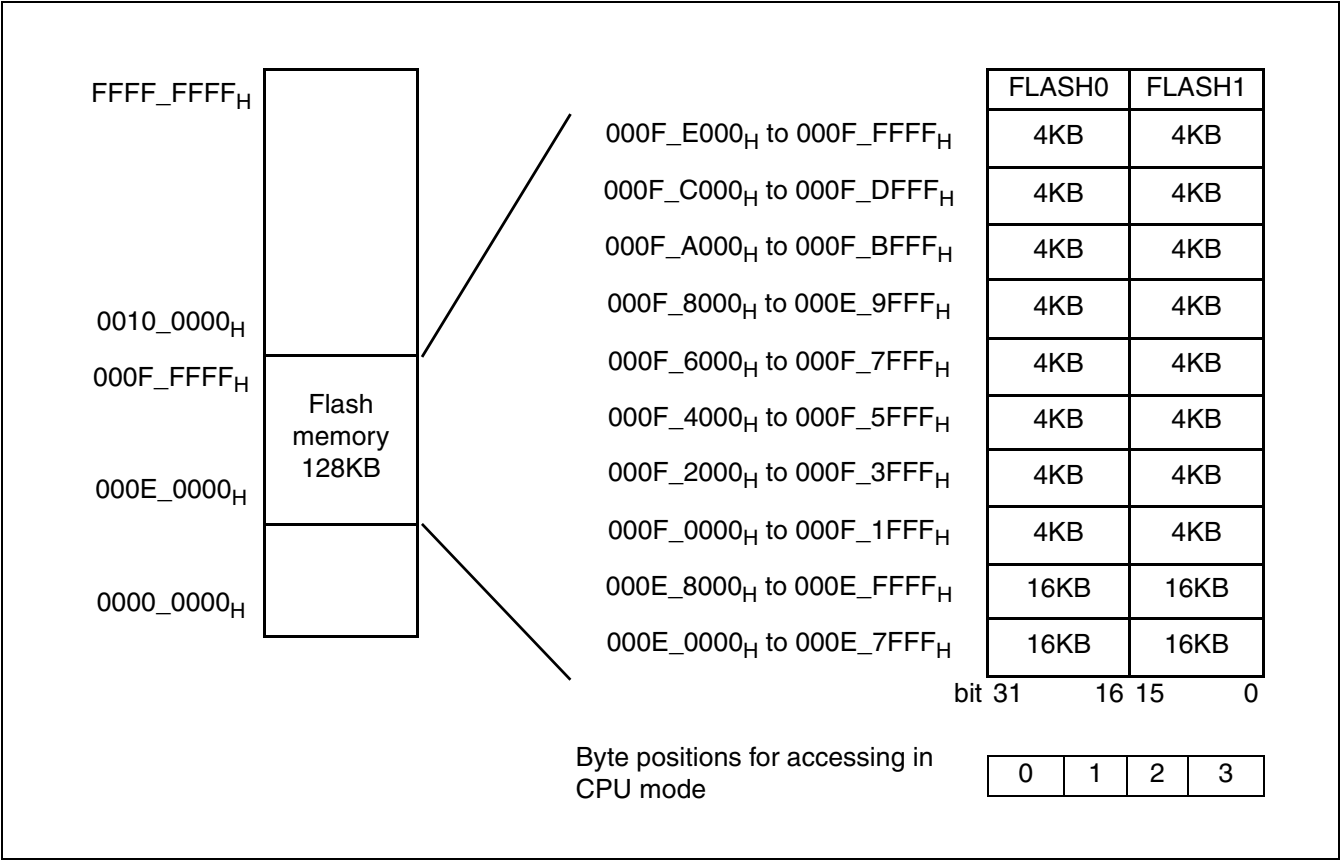


■ Sector Configuration for Flash Memory

The address mapping of flash memory is different between when accessed from the FR-CPU and when accessed from the ROM programmer. Figure 19.1-2 and Table 19.1-1 show memory mapping for access from the FR-CPU; Figure 19.1-3 shows memory mapping for access from the ROM programmer.

● Mapping for Access from FR-CPU

Figure 19.1-2 Mapping for Access from FR-CPU

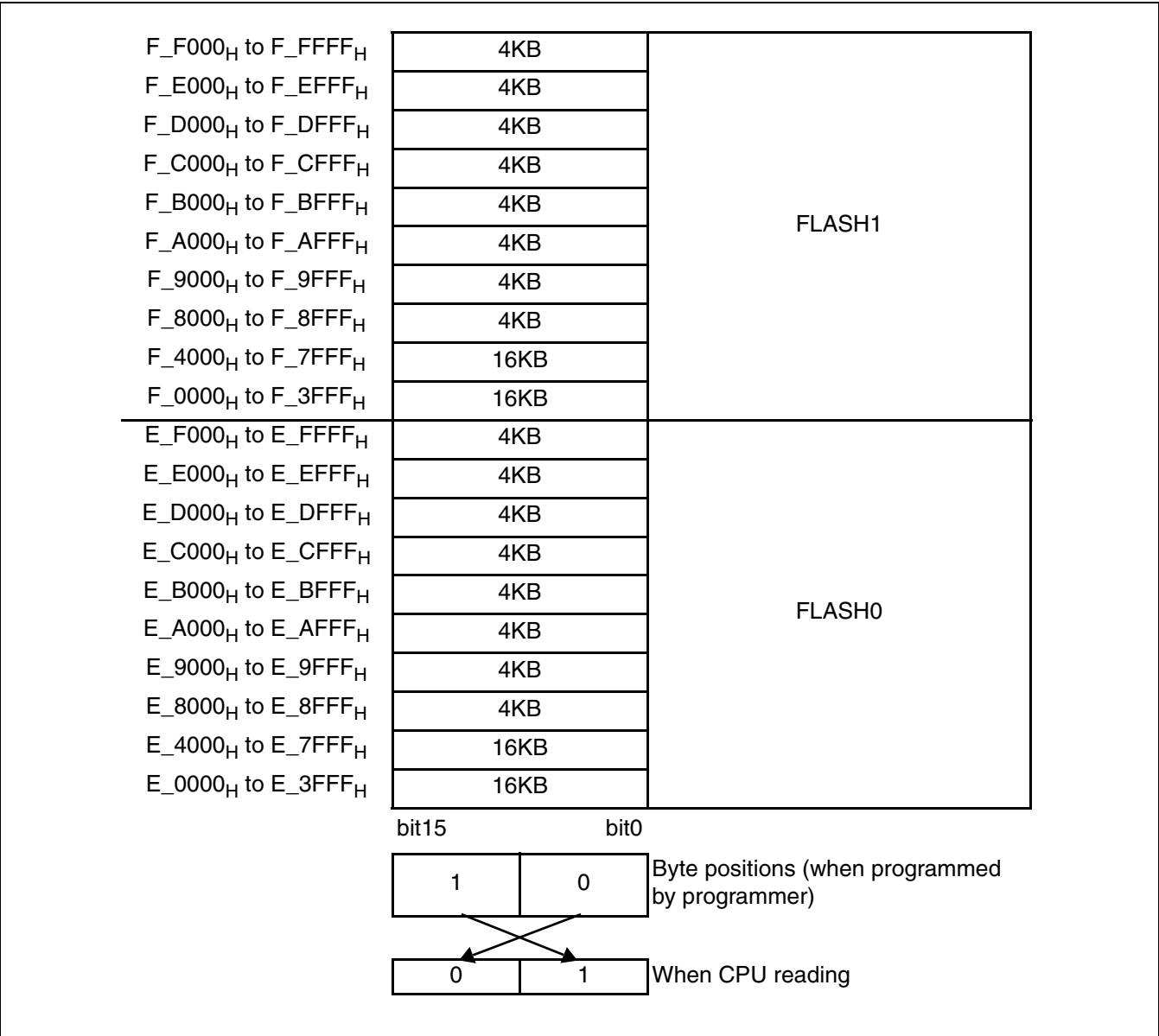


**Table 19.1-1 Sector address List (for Access from FR-CPU)**

Flash	Address area	Corresponding bit position	Sector capacity
FLASH1	F_E002 <sub>H,3H</sub> to F_FFFE <sub>H,FH</sub>	bit15 to bit0	4KB
	F_C002 <sub>H,3H</sub> to F_DFFE <sub>H,FH</sub>	bit15 to bit0	4KB
	F_A002 <sub>H,3H</sub> to F_BFFE <sub>H,FH</sub>	bit15 to bit0	4KB
	F_8002 <sub>H,3H</sub> to F_9FFE <sub>H,FH</sub>	bit15 to bit0	4KB
	F_6002 <sub>H,3H</sub> to F_7FFE <sub>H,FH</sub>	bit15 to bit0	4KB
	F_4002 <sub>H,3H</sub> to F_5FFE <sub>H,FH</sub>	bit15 to bit0	4KB
	F_2002 <sub>H,3H</sub> to F_3FFE <sub>H,FH</sub>	bit15 to bit0	4KB
	F_0002 <sub>H,3H</sub> to F_1FFE <sub>H,FH</sub>	bit15 to bit0	4KB
	E_8002 <sub>H,3H</sub> to E_FFFE <sub>H,FH</sub>	bit15 to bit0	16KB
	E_0002 <sub>H,3H</sub> to E_7FFE <sub>H,FH</sub>	bit15 to bit0	16KB
FLASH0	F_E000 <sub>H,1H</sub> to F_FFFC <sub>H,DH</sub>	bit31 to bit16	4KB
	F_C000 <sub>H,1H</sub> to F_DFFC <sub>H,DH</sub>	bit31 to bit16	4KB
	F_A000 <sub>H,1H</sub> to F_BFFC <sub>H,DH</sub>	bit31 to bit16	4KB
	F_8000 <sub>H,1H</sub> to F_9FFC <sub>H,DH</sub>	bit31 to bit16	4KB
	F_6000 <sub>H,1H</sub> to F_7FFC <sub>H,DH</sub>	bit31 to bit16	4KB
	F_4000 <sub>H,1H</sub> to F_5FFC <sub>H,DH</sub>	bit31 to bit16	4KB
	F_2000 <sub>H,1H</sub> to F_3FFC <sub>H,DH</sub>	bit31 to bit16	4KB
	F_0000 <sub>H,1H</sub> to F_1FFC <sub>H,DH</sub>	bit31 to bit16	4KB
	E_8000 <sub>H,1H</sub> to E_FFFC <sub>H,DH</sub>	bit31 to bit16	16KB
	E_0000 <sub>H,1H</sub> to E_7FFC <sub>H,DH</sub>	bit31 to bit16	16KB

● Mapping for Access from ROM Writer

Figure 19.1-3 Address Mapping for Access from ROM Writer



**Table 19.1-2 Sector Address List (for Access from ROM Programmer)**

Flash	Address area	Corresponding bit position	Sector capacity
FLASH1	F_F000 <sub>H</sub> to F_FFFF <sub>H</sub>	bit15 to bit0	4KB
	F_E000 <sub>H</sub> to F_EFFF <sub>H</sub>	bit15 to bit0	4KB
	F_D000 <sub>H</sub> to F_DFFF <sub>H</sub>	bit15 to bit0	4KB
	F_C000 <sub>H</sub> to F_CFFF <sub>H</sub>	bit15 to bit0	4KB
	F_B000 <sub>H</sub> to F_BFFF <sub>H</sub>	bit15 to bit0	4KB
	F_A000 <sub>H</sub> to F_AFFF <sub>H</sub>	bit15 to bit0	4KB
	F_9000 <sub>H</sub> to F_9FFF <sub>H</sub>	bit15 to bit0	4KB
	F_8000 <sub>H</sub> to F_8FFF <sub>H</sub>	bit15 to bit0	4KB
	E_4000 <sub>H</sub> to E_7FFF <sub>H</sub>	bit15 to bit0	16KB
	E_0000 <sub>H</sub> to E_3FFF <sub>H</sub>	bit15 to bit0	16KB
FLASH0	F_F000 <sub>H</sub> to F_FFFF <sub>H</sub>	bit15 to bit0	4KB
	F_E000 <sub>H</sub> to F_EFFF <sub>H</sub>	bit15 to bit0	4KB
	F_D000 <sub>H</sub> to F_DFFF <sub>H</sub>	bit15 to bit0	4KB
	F_C000 <sub>H</sub> to F_CFFF <sub>H</sub>	bit15 to bit0	4KB
	F_B000 <sub>H</sub> to F_BFFF <sub>H</sub>	bit15 to bit0	4KB
	F_A000 <sub>H</sub> to F_AFFF <sub>H</sub>	bit15 to bit0	4KB
	F_9000 <sub>H</sub> to F_9FFF <sub>H</sub>	bit15 to bit0	4KB
	F_8000 <sub>H</sub> to F_8FFF <sub>H</sub>	bit15 to bit0	4KB
	E_4000 <sub>H</sub> to E_7FFF <sub>H</sub>	bit15 to bit0	16KB
	E_0000 <sub>H</sub> to E_3FFF <sub>H</sub>	bit15 to bit0	16KB

## 19.2 Registers

The flash memory has two registers: flash memory status register (FLCR) and flash memory wait register (FLWC).

### ■ Flash Memory Registers

Figure 19.2-1 lists the registers of the flash memory.

Figure 19.2-1 Flash Memory Registers

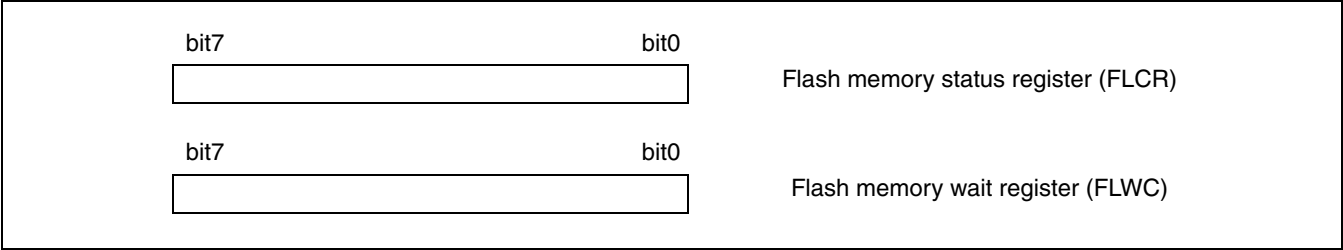


Table 19.2-1 Address Map

Address	Offset				Block
	+0	+1	+2	+3	
FLCR[R/W] B 007000 <sub>H</sub>	01101000	-	-	-	Flash interface
FLWC[R/W] B 007004 <sub>H</sub>	00000011	-	-	-	

## 19.2.1 Flash Memory Status Register (FLCR)

The flash memory status register (FLCR) indicates the operating state of flash memory.

### ■ Flash Memory Status Register (FLCR)

This register controls interrupts to the CPU and programming into flash memory. The register is accessible only by the CPU. The register cannot be accessed when the device is mounted on a programmer.

Do not access this register using a read-modify-write instruction.

Figure 19.2-2 shows the bit structure of the flash memory status register (FLCR).

**Figure 19.2-2 Bit Structure of Flash Memory Status Register (FLCR)**

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
007000 <sub>H</sub>	-	-	-	-	-	-	WE	-	01101000 <sub>B</sub>
	R/W	R/W	R/W	R	R	R/W	R/W	R/W	

R/W: Readable/Writable  
R: Read only

The following describes the function of each bit in the flash memory status register (FLCR):

[bit7] - : Reserved bit

This bit is reserved bit. Always write "0" to the bit.

[bit6, bit5] - : Reserved bit

These bits are reserved bits. Always write "1" to the bits.

[bit4, bit3] - : Reserved bits

These bits are reserved bits. These bits are read-only; an attempt to write to the bits has no effect on operation.

[bit2] - : Reserved bit

This bit is a reserved bit. Always write "0" to the bit.

[bit1] WE: Controls the programming of data and commands into Flash memory in CPU mode.

While this bit contains "0", any attempt to program data or a command into Flash memory is ignored.

While this bit contains "1", the programming of data and commands into Flash memory is valid, where the automatic algorithm can be started.

Note, however, that writing data from Flash memory assumes 16-bit or 8-bit access. Write access to Flash memory must be 16-bit or 8-bit access only. Do not access it in 32 bits.

Before updating this bit, be sure to check the hardware sequence flag to make sure that the automatic algorithm has been stopped.

WE	Function
0	Disable write access to Flash memory.
1	Enable write access to Flash memory.

## ● Restrictions

To update the WE bit of this register, be sure to execute the instruction sequence given below.

When updating this register, do not perform DMA, interrupt, or standby operation.

Instruction sequence:

```
1:  LDI  #(_FLWC),  R0
2:  LDI  #0x01,     R1
3:  STB  R1,         @R0      //set FLWC(WTC=1)

4:  LDI  #(_FLCR),  R0
5:  LDI  #0x62,     R1
6:  STB  r1,         @R0      //set FLCR(WE=1)

7:  LDI  #(_FLWC),  R0
8:  LDI  #0x03,     R1
9:  STB  R1,         @R0      //set FLWC(WTC=3)
```

[bit0] - : Reserved bit

This bit is a reserved bit. Always write "0" to the bit.

## 19.2.2 Flash Memory Wait Register (FLWC)

The flash memory wait register (FLWC) controls wait states of the flash memory in CPU mode.

### ■ Flash Memory Wait Register (FLWC)

Figure 19.2-3 shows the bit structure of the flash memory wait register (FLWC).

**Figure 19.2-3 Bit Structure of Flash Memory Wait Register (FLWC)**

Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
007004 <sub>H</sub>	-	-	FAC1	FAC0	-	WTC2	WTC1	WTC0	00000011 <sub>B</sub>
	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

R/W: Readable/Writable  
R: Read only

The following describes the function of each bit in the flash memory wait register (FLWC):

[bit7, bit6] - : Reserved bits

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

[bit5, bit4] FAC1, FAC0: Pulse width control bits for internal write signal

FAC1	FAC0	ATDIN	EQIN	
0	0	0.5 clock	1 clock	(Initial value)
0	1	1 clock	1.5 clocks	
1	0	1.5 clocks	2 clocks	
1	1	2 clocks	2.5 clocks	

Notes: • ATDIN and EQIN are internal write signals. Use these signals usually with initial value settings.

• For MASK ROM models, always write "00<sub>B</sub>".

[bit3] - : Reserved bit

This bit is a reserved bit. Always write "0" to the bit.



[bit2 to bit0] WTC2 to WTC0: Wait cycle control bits

These bits control the number of wait cycles for access to Flash memory.

WTC2	WTC1	WTC0	Wait Cycle	In Read	In Write
0	0	0	-	Setting disabled	Setting disabled
0	0	1	1	Even 33MHz can be operated.	Setting disabled
0	1	0	2	Even 33MHz can be operated.	Setting disabled
0	1	1	3	Even 33MHz can be operated.	Even 33MHz can be operated. (Initial value)
1	0	0	4	Setting disabled	Setting disabled
1	0	1	5	Setting disabled	Setting disabled
1	1	0	6	Setting disabled	Setting disabled
1	1	1	7	Setting disabled	Setting disabled

- Notes:
- Set these bits so that the number of wait cycles is equal to or greater than that set by the FAC1, FAC0 bits.
  - The initial value is the setting for write access. For read-only applications (FLCR WE = 0), you can set 1 wait cycle (WTC2, WTC1, WTC0 = 001<sub>B</sub>) to read at the maximum speed.
  - Masked models are read with 3 wait cycles inserted (WTC2, WTC1, WTC0 = 011<sub>B</sub>) by default.  
Setting 1 wait cycle (WTC2, WTC1, WTC0 = 001<sub>B</sub>) provides fastest read access.

## 19.3 Access Mode of Flash Memory

---

The following two types of access modes are available to the FR-CPU to access flash memory:

- **ROM word mode**

The CPU can read 32 bits of data collectively but cannot program data into flash memory.

- **Programming word mode**

The CPU cannot access flash memory in 32 bits but can program data into flash memory in half-words (16 bits) or in bytes (8 bits).

---

### ■ FR-CPU ROM Mode (Read only in 32/16/8 Bits)

In this mode, the flash memory serves as internal ROM for the FR-CPU. The CPU can read a word (32 bits) of data collectively but can neither program data into flash memory nor start the automatic algorithm.

- **Specification method of mode**

- This mode is established when the "WE" bit of the flash memory status register is set to "0".
- The flash memory remains in this mode whenever a reset is cancelled during CPU operation.
- This mode cannot be selected not during CPU operation.

- **Operation content**

When reading a flash memory area, the CPU can read a word (32 bits) of data collectively from memory. Read access takes a minimum of two cycles per word (1 wait). This allows an instruction to be supplied to the FR-CPU with no wait required.

- **Restrictions**

The address mapping of flash memory and the endian method in this mode are different from those in ROM programming mode. Note also that, in this mode, you can neither program data into flash memory nor start the automatic algorithm.

## ■ FR-CPU Programming Mode (Read/Write in 16/8 Bits)

In this mode, the CPU can erase or program data. Note, however, that no program can be executed in a bank in which data is being erased or programmed.

### ● Specification method of mode

- This mode is established when the "WE" bit of the flash memory status register (FLCR) is set to "1".
- The "WE" bit is set to "0" after a reset is cancelled during CPU operation. To specify this mode, write "1" to the bit. When "0" is written again or when a reset occurs, the flash memory returns to ROM mode if the bit is set to "0".

### ● Operation content

- When reading a flash memory area, the CPU can read a word (32 bits) of data collectively from flash memory. Note, however, that read access takes four cycles per half-word (three waits) set for write access in the flash memory status register (FLCR).
- You can start the automatic algorithm by programming a command into flash memory. You can erase data from or program data into flash memory by starting the automatic algorithm. For details on the automatic algorithm, see Sections "19.4 Automatic Algorithm Startup Procedures" and "19.5 Automatic Algorithm Execution States".

### ● Restrictions

The address mapping of flash memory and the endian method in this mode are different from those in ROM programming mode.

- When updating the WE bit to switch the programming access mode, follow the restrictions in Section "19.2.1 Flash Memory Status Register (FLCR)".

## 19.4 Automatic Algorithm Startup Procedures

Flash memory can be erased or programmed by starting its own automatic algorithm.

### ■ Command Sequence

To start the automatic algorithm, program a half-word (16 bits) of data into flash memory continuously for once to six times. That is called the commands. The flash memory is reset to the read mode if invalid addresses and data are programmed or addresses and data are programmed in a wrong order.

Table 19.4-1 and Table 19.4-2 list the commands that can be used to write data to or erase data from flash memory.

**Table 19.4-1 FLASH0 Write Command Sequence in CPU Mode**

Command sequence	Bus write cycle	1st bus Write Cycle		2nd bus Write Cycle		3rd bus Write Cycle		4th bus read/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	XXXXX <sub>H</sub>	F0 <sub>H</sub>	—	—	—	—	—	—	—	—	—	—
Read/Reset	4	F5554 <sub>H</sub>	AA <sub>H</sub>	EAAA8 <sub>H</sub>	55 <sub>H</sub>	F5554 <sub>H</sub>	F0 <sub>H</sub>	RA	RD	—	—	—	—
Programming	4	F5554 <sub>H</sub>	AA <sub>H</sub>	EAAA8 <sub>H</sub>	55 <sub>H</sub>	F5554 <sub>H</sub>	A0 <sub>H</sub>	PA	PD	—	—	—	—
Chip Erasing	6	F5554 <sub>H</sub>	AA <sub>H</sub>	EAAA8 <sub>H</sub>	55 <sub>H</sub>	F5554 <sub>H</sub>	80 <sub>H</sub>	F5554 <sub>H</sub>	AA <sub>H</sub>	EAAA8 <sub>H</sub>	55 <sub>H</sub>	F5554 <sub>H</sub>	10 <sub>H</sub>
Sector Erasing	6	F5554 <sub>H</sub>	AA <sub>H</sub>	EAAA8 <sub>H</sub>	55 <sub>H</sub>	F5554 <sub>H</sub>	80 <sub>H</sub>	F5554 <sub>H</sub>	AA <sub>H</sub>	EAAA8 <sub>H</sub>	55 <sub>H</sub>	SA	30 <sub>H</sub>
Sector Erasing being Suspended		Suspend sector erasure by input of address = XXXXX <sub>H</sub> , data = B0 <sub>H</sub> .											
Resume (Sector being Erased)		Restart sector erasure by input of address = XXXXX <sub>H</sub> , data = 30 <sub>H</sub> .											
Auto Select	3	F5554 <sub>H</sub>	AA <sub>H</sub>	EAAA8 <sub>H</sub>	55 <sub>H</sub>	F5554 <sub>H</sub>	90 <sub>H</sub>	—	—	—	—	—	—
Continuous mode	3	F5554 <sub>H</sub>	AA <sub>H</sub>	EAAA8 <sub>H</sub>	55 <sub>H</sub>	F5554 <sub>H</sub>	20 <sub>H</sub>	—	—	—	—	—	—
Continuous writing	2	XXXXX <sub>H</sub>	A0 <sub>H</sub>	PA	PD	—	—	—	—	—	—	—	—
Continuous mode Reset	2	XXXXX <sub>H</sub>	90 <sub>H</sub>	XXXXX <sub>H</sub>	F0 <sub>H</sub> or 00 <sub>H</sub>	—	—	—	—	—	—	—	—

**Table 19.4-2 FLASH1 Write Command Sequence in CPU Mode**

Command sequence	Bus write cycle	1st bus Write Cycle		2nd bus Write Cycle		3rd bus Write Cycle		4th bus read/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	XXXXX <sub>H</sub>	F0 <sub>H</sub>	—	—	—	—	—	—	—	—	—	—
Read/Reset	4	F5556 <sub>H</sub>	AA <sub>H</sub>	EAAAA <sub>H</sub>	55 <sub>H</sub>	F5556 <sub>H</sub>	F0 <sub>H</sub>	RA	RD	—	—	—	—
Programming	4	F5556 <sub>H</sub>	AA <sub>H</sub>	EAAAA <sub>H</sub>	55 <sub>H</sub>	F5556 <sub>H</sub>	A0 <sub>H</sub>	PA	PD	—	—	—	—
Chip Erasing	6	F5556 <sub>H</sub>	AA <sub>H</sub>	EAAAA <sub>H</sub>	55 <sub>H</sub>	F5556 <sub>H</sub>	80 <sub>H</sub>	F5556 <sub>H</sub>	AA <sub>H</sub>	EAAAA <sub>H</sub>	55 <sub>H</sub>	F5556 <sub>H</sub>	10 <sub>H</sub>
Sector Erasing	6	F5556 <sub>H</sub>	AA <sub>H</sub>	EAAAA <sub>H</sub>	55 <sub>H</sub>	F5556 <sub>H</sub>	80 <sub>H</sub>	F5556 <sub>H</sub>	AA <sub>H</sub>	EAAAA <sub>H</sub>	55 <sub>H</sub>	SA	30 <sub>H</sub>
Sector Erasing being Suspended		Suspend sector erasure by input of address = XXXXX <sub>H</sub> , data = B0 <sub>H</sub> .											
Resume (Sector being Erased)		Restart sector erasure by input of address = XXXXX <sub>H</sub> , data = 30 <sub>H</sub> .											
Auto Select	3	F5556 <sub>H</sub>	AA <sub>H</sub>	EAAAA <sub>H</sub>	55 <sub>H</sub>	F5556 <sub>H</sub>	90 <sub>H</sub>	—	—	—	—	—	—
Continuous mode	3	F5556 <sub>H</sub>	AA <sub>H</sub>	EAAAA <sub>H</sub>	55 <sub>H</sub>	F5556 <sub>H</sub>	20 <sub>H</sub>	—	—	—	—	—	—
Continuous writing	2	XXXXX <sub>H</sub>	A0 <sub>H</sub>	PA	PD	—	—	—	—	—	—	—	—
Continuous mode Reset	2	XXXXX <sub>H</sub>	90 <sub>H</sub>	XXXXX <sub>H</sub>	F0 <sub>H</sub> or 00 <sub>H</sub>	—	—	—	—	—	—	—	—

The commands available are the same between word and byte modes. The data in bits not represented are arbitrary.

RA: Read address

PA: Writing address

SA: Sector address (Specify one arbitrary address in a sector)

RD: Read data

PD: Write data

#### ● Read/Reset command

To return to the read mode when the timing limit is exceeded, issue a read/reset command sequence. Data is read from flash memory in the read cycle. The flash memory remains in the read state until another command is input.

The flash memory is set to the read/reset state automatically when the power is turned on. In this case, no command is required to read data.

#### ● Programming

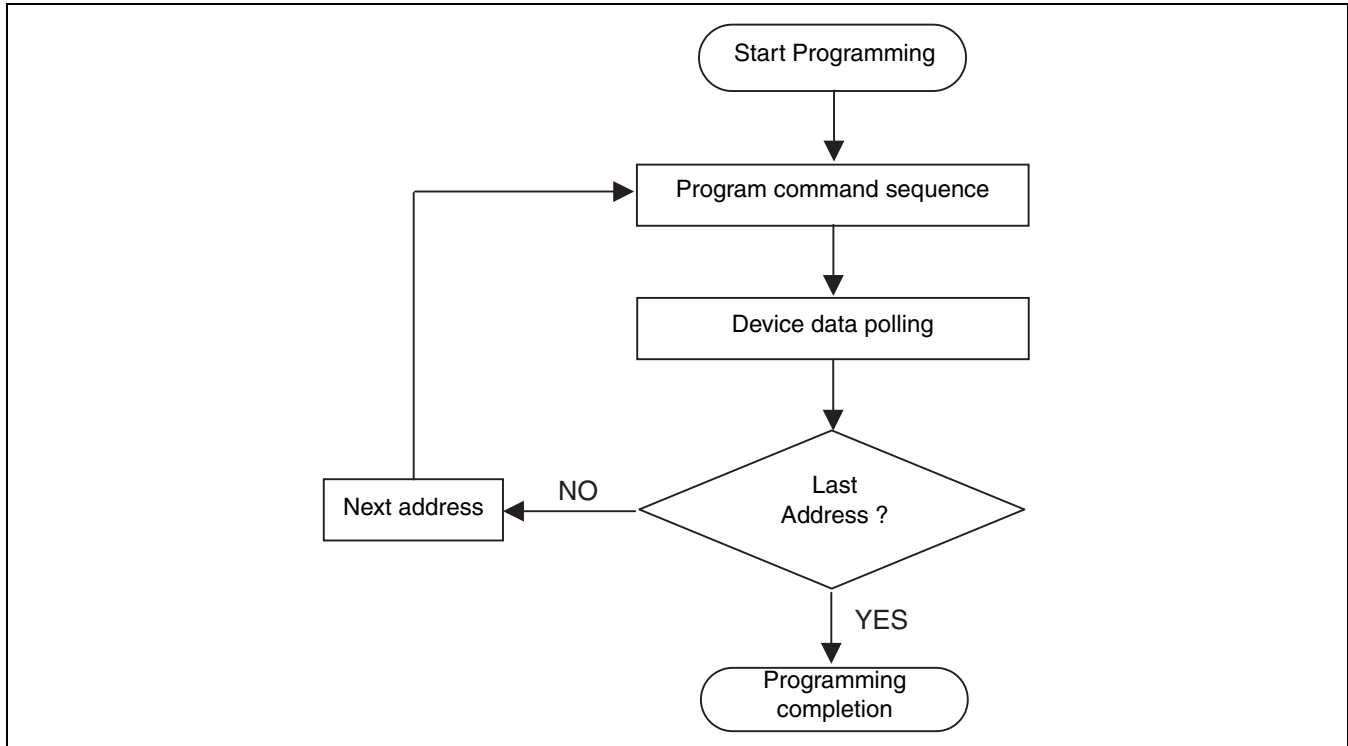
In CPU programming mode, programming is performed in half-words. Programming requires four bus operations. The command sequence has two "unlock" cycles, followed by a programming setup command and a programming data cycle. Programming into memory is started in the last programming cycle.

Once the automatic programming algorithm command sequence is executed, the flash memory does not require further external control. The flash memory generates appropriate programming pulse produced internally to verify the margins of programmed cells. The automatic programming operation finishes when bit7 data matches the data written into this bit by the data polling function (see ■Hardware sequence flag in Section "19.5 Automatic Algorithm Execution States"). Afterward, the operation returns to the read mode and does not accept writing addresses any more. Accordingly, the flash memory demands the next valid address. In this way, data polling indicates that memory is being programmed.

During programming, any command programmed into flash memory is ignored. If a hardware reset is activated during programming at an address, the data at that address is not guaranteed. Programming is allowed in any order of addresses and beyond the boundaries of sectors. Data "0" cannot be returned to data "1" by writing. If data "1" is programmed over data "0", either the data polling algorithm determines that the element is defective or data "1" apparently looks as if it were programmed.

When the data is read in reset/read mode, however, it remains as "0". Data "0" can be updated to data "1" only by erasing. Figure 19.4-1 illustrates the procedure of programming using the program command sequence.

**Figure 19.4-1 Procedure of Programming Using the Program Command**



### ● Chip Erasing

Chip erasure (erasing all of the sectors collectively) is performed by accessing flash memory six times. Two "unlock" cycles come first, then a "setup" command is programmed soon. Another two "unlock" cycles are inserted prior to the chip erase command.

Before chip erasing, the user need not perform programming to flash memory. During execution of the automatic erase algorithm, the flash memory automatically verifies its cells by programming patterns of 0s (preprogramming) before erasing all the cells. During preprogramming, the flash memory requires no external control.

Automatic erasure is started by programming in the command sequence and terminates when bit7 is set to "1", when the flash memory returns to the read mode. The chip erase time is "sector erase time  $\times$  the number of all sectors + chip program time (preprogram)".

Figure 19.4-2 illustrates the procedure of chip erasure using the chip erase command sequence.

### ● Sector Erasing

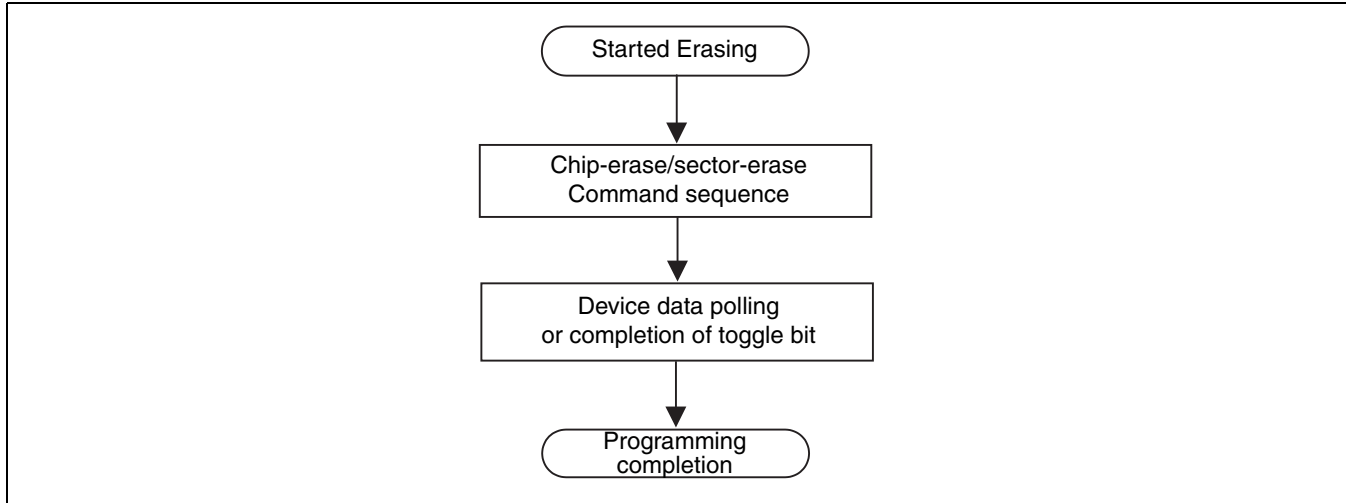
Sector erasure is performed by accessing flash memory six times. Two "unlock" cycles come first, then a "setup" command is programmed soon. Another two "unlock" cycles are inserted, then the sector erase command is input in the sixth cycle to start erasing a sector. During a time-out period of 50  $\mu$ s after the last sector erase command is programmed, the next sector erase command can be accepted.

The erasing of multiple sectors can be accepted at the same time by programming six bus cycles as described above. This sequence is executed by programming the sector erase command (30<sub>H</sub>) continuously at the addresses of the sectors to be erased at the same time. When the 50  $\mu$ s time-out after the last sector erase command is programmed expires, the flash memory starts erasing the sectors. To erase multiple sectors at the same time, therefore, the sector erase command for each of the sectors must be input within a time-out period of 50  $\mu$ s, or the command may not be accepted if it expires. You can check whether the successive sector erase command is valid by monitoring bit3 (see "Hardware sequence flag" in Section "19.5 Automatic Algorithm Execution States").

If any command other than the sector erase command and erasure pause command is input in a time-out period, the flash memory is reset to the read mode and ignores the preceding command sequence. In this case, the relevant sector is erased completely by erasing it again. Any combination and number (0 to 6) of sector addresses can be input to the sector erase buffer. For erasing sectors, the user does not have to program into flash memory in advance. The flash memory automatically programs (preprograms) into all the cells in the sectors to be erased. Note also that the erasing of sectors has no effect on any other sector. During these operations, the flash memory requires no external control.

Automatic sector erasure is started after a time-out period of 50  $\mu$ s after the last sector erase command is programmed and terminates when bit7 is set to "1" (see "Hardware sequence flag" in Section "19.5 Automatic Algorithm Execution States"). The flash memory returns to the read mode. Any other command is ignored. Data polling works for any address in the sectors erased. The multiple-sector erase time is "(sector erase time + sector program time (preprogram))  $\times$  the number of sectors erased".

Figure 19.4-2 illustrates the procedure of chip erasure using the chip erase command sequence.

**Figure 19.4-2 Procedure of Chip Erasure Using the Chip Erase Command**

● **Temporary deletion stop**

The erasure pause command allows the user to suspend the flash memory's automatic algorithm during erasure of sectors in order to read data from or program data into other sectors. This command is valid only during sector erasure; it is ignored during chip erasure or programming. The erasure pause command (B0<sub>H</sub>) is valid only during the period of sector erasure, including the sector erasure time-out period after the sector erase command (30<sub>H</sub>). When this command is input during the time-out period, the flash memory terminates the time-out and suspends erasure. The flash memory restarts erasure when the erasure restart command is programmed. The erasure pause and restart commands can be input with any address.

When the erasure pause command is input during sector erasure, it takes a maximum of 20 μs for the flash memory to pause erasure. When the flash memory enters the erasure pause mode, the ready/busy output and bit7 output "1", and bit6 stops toggling. You can check whether erasure is suspended by inputting the sector address being erased to monitor the values read from bit6 and bit7. An attempt to program another erasure pause command is ignored. When erasure is paused, the flash memory enters the erasure pause read mode. Data reading in this mode is the same as typical data reading, except that it is effective for sectors containing data not being erasure-paused. In erasure pause read mode, bit2 toggles for continuous reading from the sector being erasure-paused (see "Hardware sequence flag" in Section "19.5 Automatic Algorithm Execution States").

In erasure pause read mode, the user can program into flash memory by programming the program command sequence. This program mode is the erasure pause program mode. Programming in this mode is the same as normal writing in bytes, except that it is effective for sectors containing data not being erasure-paused. In erasure pause program mode, bit2 toggles for continuous reading from the sector being erasure-paused. The erasure pause state can be detected by checking the erasure pause bit (bit6).

---

**Note:**

Bit7 must be read for the address being programmed while bit6 can be read for any address. To restart sector erasure, input the restart command (30<sub>H</sub>). Another restart command is ignored if input at this point of time. In contrast, the erasure pause command can be input after the flash memory restarts erasure.

---



## 19.5 Automatic Algorithm Execution States

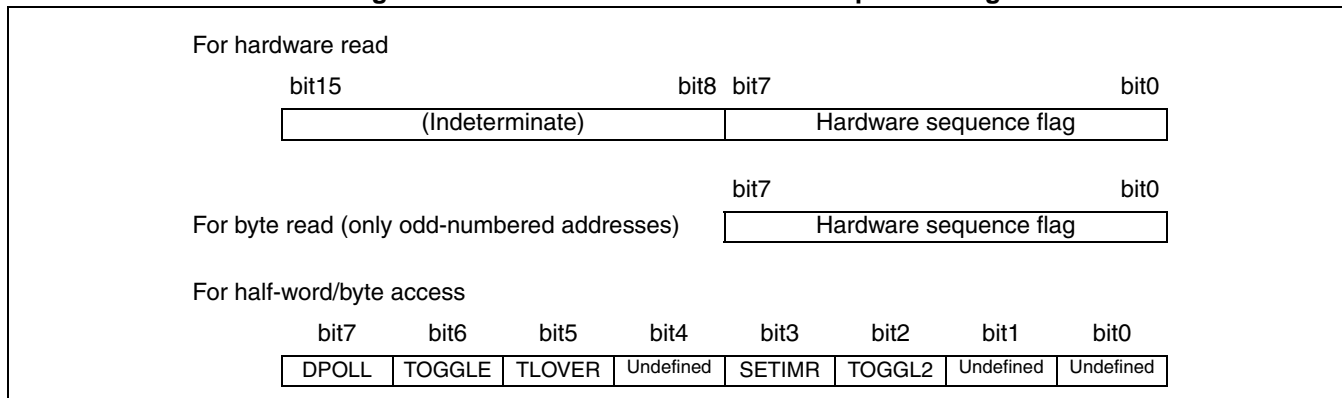
As this flash memory uses the automatic algorithm for program/erase flow, it has a piece of hardware to inform the internal operating state of the flash memory or the completion of its operation to the outside of flash memory.

### ■ Hardware Sequence Flag

A hardware sequence flag is obtained as data by reading an address (an odd-numbered address during byte access) of flash memory during execution of the automatic algorithm. The obtained data contains five effective bits, each of which indicates a state of the automatic algorithm. Figure 19.5-1 shows the structure of the hardware sequence flag.

Table 19.5-1 lists hardware sequence flag states.

**Figure 19.5-1 Structure of Hardware Sequence Flag**



**Table 19.5-1 Hardware Sequence Flag Status List**

State		DPOLL (bit7)	TOGLLE (bit6)	TLOVER (bit5)	SETIMR (bit3)	TOGGL2 (bit2)
During execution	Automatic Programming Operation	Inverted data	Toggle	0	0	1
	Programming/erasing during automatic erasure	0	Toggle	0	1	Toggle
	Erasing pause	Reading (Sector being erased)	1	1	0	0
		Reading (Sector unerased)	Data	Data	Data	Data
		Programming (Sector unerased)	Inverted data	Toggle	0	0
Time limit is exceeded	Automatic Programming Operation	Inverted data	Toggle	1	0	1
	Programming/erasing during automatic erasure	0	Toggle	1	1	*2

\*1: In erasure pause program mode, bit2 outputs logic "1" for a read from the address being programmed. bit2 toggles for a continuous read from the sector being erasure-paused.

\*2: When bit5 is "1" (time limit exceeded), bit2 toggles for a continuous read to the sector being programmed/erased but does not toggle for a read to any other sector.

The following describes the function of each bit:

**[bit7] DPOLL: Data polling**

- During automatic programming  
When the flash memory is read during execution of the automatic program algorithm, it outputs the inverted version of data finally written to bit7. When the flash memory is read-accessed upon completion of the automatic program algorithm, it outputs bit7 of the data read from the address located by the address signal.
- During automatic erasure  
When the flash memory is read during execution of the automatic erase algorithm, it outputs "0" regardless of the address located by the address signal. In the same way, 1 is output when it ends.
- During sector erasure suspended  
When the flash memory is read during suspended sector erasure, it outputs "1" if the address located by the address signal belongs to the sector being erased. If the address does not belong to the sector being erased, the flash memory outputs bit7 of read data at the address located by the address signal. By referring this bit along with bit6 described below as the toggle bit, you can check whether sector erasure is currently being suspended and which sector is being erased.

---

**Note:**

When the automatic algorithm comes close to termination, bit7 (data polling) changes asynchronously during read access. This means that the flash memory sends information on the operating status to bit7 and then outputs finalized data. When the flash memory terminates the automatic algorithm or when bit7 outputs finalized data, other bits still remain indeterminate. Finalized data in other bits are read by execution of a continuous read.

---

**[bit6] TOGGLE: Toggle bit**

- During automatic programming/erasure  
When the flash memory is continuously read during execution of the automatic program or erase algorithm, it outputs the result of toggling between "1" and "0" to bit6. When the automatic program or erase algorithm terminates, the flash memory stops toggling bit6 for continuous read access and outputs valid data. The toggle bit takes effect after the last program cycle of each command sequence.  
Note that, if the sector attempted to be programmed is an update-guaranteed sector, the toggle bit toggles for about 2  $\mu$ s and stops toggling without updating the sector with data. If all of the selected sectors to be erased are program-guaranteed sectors, the toggle bit toggles for about 100  $\mu$ s and returns to the read mode without updating their data.
- During sector erasure suspended  
When the flash memory is read during suspended sector erasure, it outputs "1" if the address located by the address signal belongs to the sector being erased. If the address does not belong to the sector being erased, the flash memory outputs bit6 of read data at the address located by the address signal.

## ● Restrictions

To read the toggle bit, be sure to execute the instruction sequence shown below.

Instruction sequence:

1:	LDI	#(Read address),	R0	
2:	NOP			//requisite nop
3:	LDUH	@R0,	R1	//read a toggle data

### [bit5] TLOVER: Timing limit exceeded

- During automatic programming/erasure

Bit5 indicates that the execution of the automatic algorithm has exceeded the time (the number of internal pulses) specified internally to the flash memory. Bit5 outputs 1 under this status. If this flag outputs "1" during execution of the automatic algorithm, the flag indicates that the current write or erase operation has failed. Bit5 will also indicate failure if attempting to write to non-blank areas without prior deletion. In this case, defined data cannot be read from bit7 (data polling), and bit6 (toggle bit) will continuously toggle. 1 will be output to bit5 if the time limit is exceeded under this status. This indicates that the flash memory was not used correctly rather than any defect with the flash memory. If this event occurs, the reset command should be executed.

### [bit3] SETIMR: Sector erase timer

- During sector erasure

Bit3 indicates the sector erasure wait period after execution of the first sector erase command sequence. Bit3 outputs 0 during this period, or 1 if the waiting period for sector deletion has been exceeded. The data polling and toggle bits are validated after executing the first sector deletion command sequence.

If this flag is "1" when the data polling or toggle bit function indicates that the erase algorithm is being executed, internally controlled erasure has been started and any command write is ignored until the data polling or toggle bit function indicates the completion of erasure (only the erasure pause code input is accepted). If this flag is "0", the flash memory accepts the additional sector deletion code to be written. To confirm this, it is advisable to check the state in this flag by software before programming succeeding sector erase code. If 1 is shown at the 2nd status check, the additional sector deletion code may not have been accepted. When reading is carried out while sector deletion is suspended, the flash memory outputs 1 if the address indicated by the address signal belongs to the sector during deletion. If the address does not belong to the sector being erased, the flash memory outputs bit3 of read data at the address located by the address signal.

### [bit2] TOGGL2: Toggle bit2

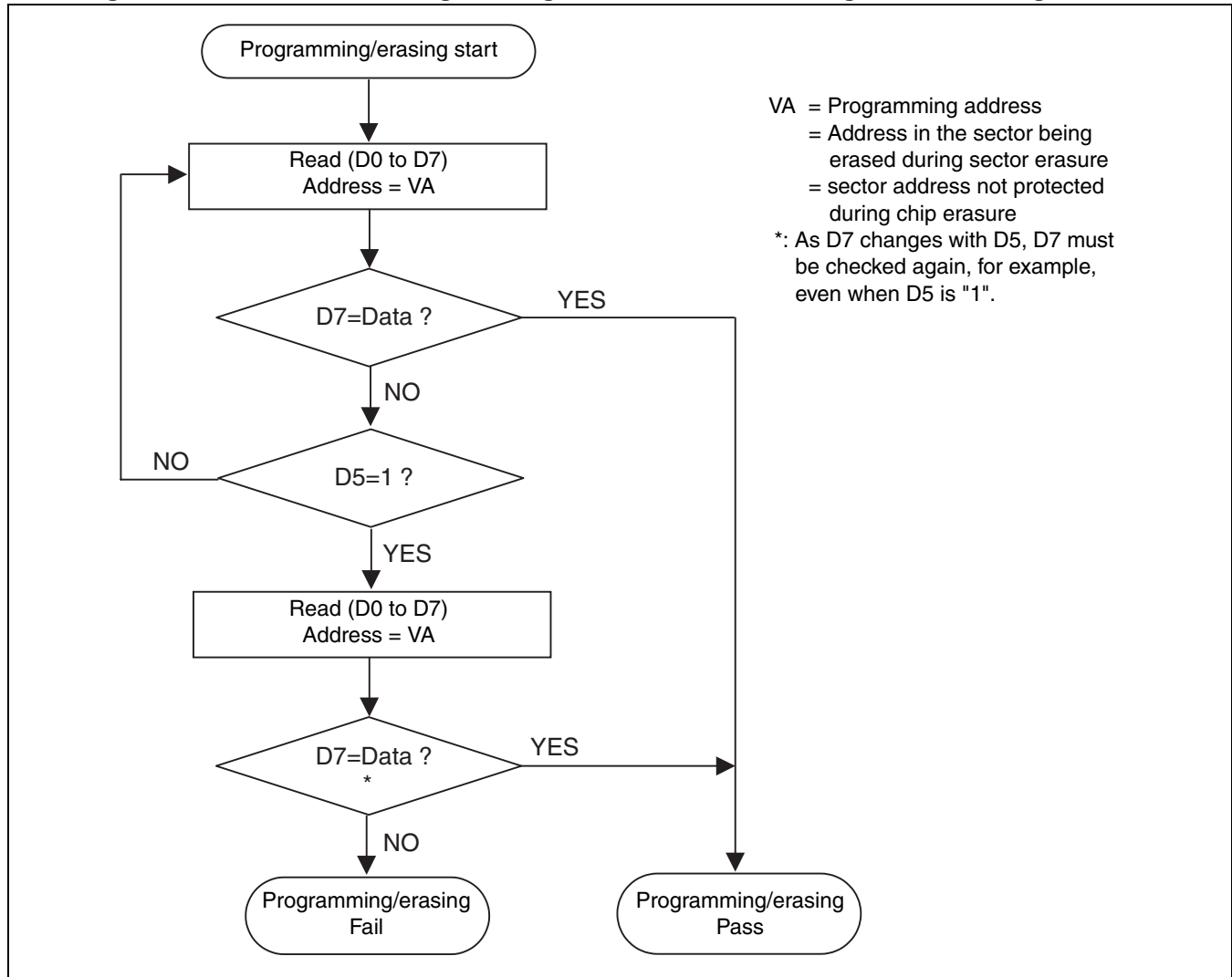
- During sector erasure

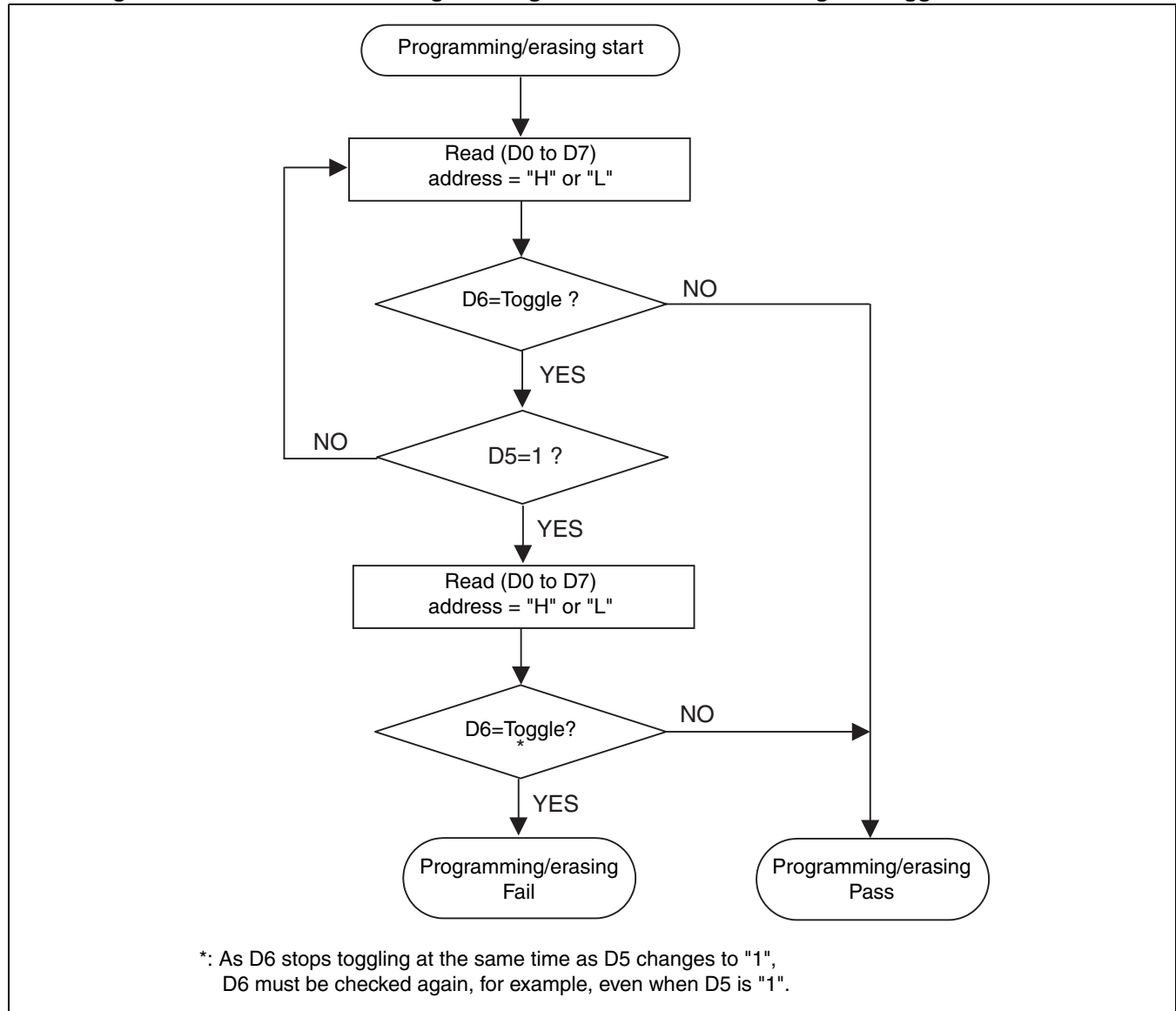
This toggle bit is used, along with another toggle bit of bit6, to detect whether the flash memory is executing automatic erasure and whether erasure is currently being suspended. Bit2 toggles when the sector being erased is continuously read during automatic erasure. If the flash memory is under deletion suspension reading mode, bit2 operates the toggle by continuously reading addresses from the sector in which deletion is suspended. If the flash memory is under deletion suspension programming mode, 1 is read by bit2 by continuously reading addresses from the sector in which deletion is not suspended. Unlike bit2, bit6 toggles only during normal programming, erasure, or erasure pause programming. Bit2 and bit6 are used together, for example, to detect the erasure pause read mode (bit6 does not toggle while bit2 does). Bit2 is also used to detect the deleted sectors. During erasure of flash memory, bit2 toggles for a read from the sector being erased.

## ■ Examples of Using the Hardware Sequence Flag

By using the hardware sequence flag, you can check the status of the automatic algorithm in flash memory. See Figure 19.5-2 and Figure 19.5-3 for programming/erase evaluation flowcharts as examples using the data polling and toggle bit functions, respectively.

**Figure 19.5-2 Flowchart of Programming/Erase Evaluation Using the Data Polling Function**



**Figure 19.5-3 Flowchart of Programming/Erase Evaluation Using the Toggle Bit Function**

## 19.6 Dual-operations

The Flash memory of the MB91F267A/MB91F267NA supports dual-operations. Unlike conventional flash memory products, the MB91F267A/MB91F267NA flash memory can simultaneously erase/program and read data separately in the upper bank (32Kbytes × 2 + 8Kbytes × 4) and lower bank (8Kbytes × 4).

### ■ Features of Dual-operation Flash Memory

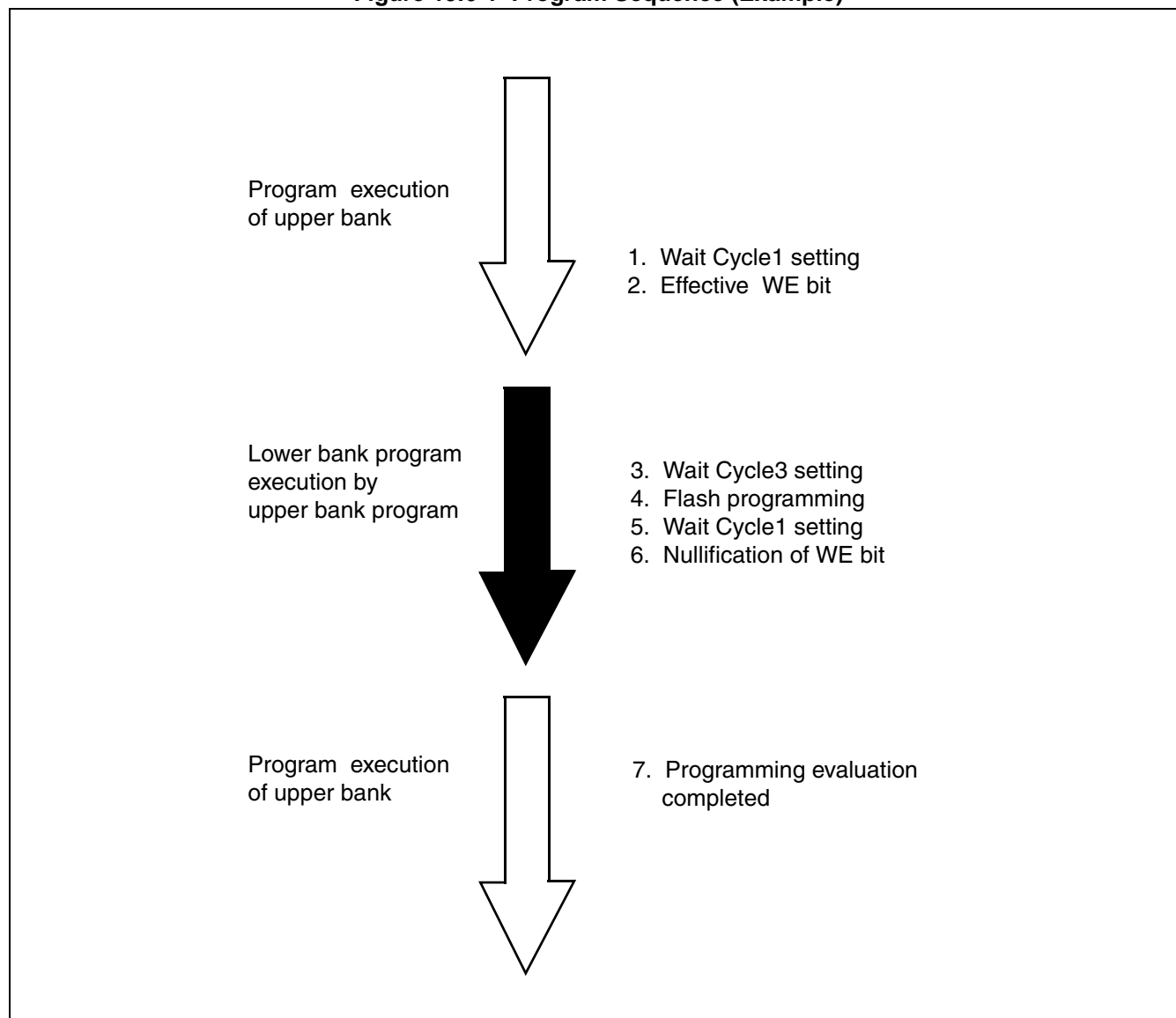
- The flash memory consists of two banks, capable of simultaneous execution of erase/program and read.

	FLASH0	FLASH1
000F_E000 <sub>H</sub> to 000F_FFFF <sub>H</sub>	4KB	4KB
000F_C000 <sub>H</sub> to 000F_DFFF <sub>H</sub>	4KB	4KB
000F_A000 <sub>H</sub> to 000F_BFFF <sub>H</sub>	4KB	4KB
000F_8000 <sub>H</sub> to 000E_9FFF <sub>H</sub>	4KB	4KB
000F_6000 <sub>H</sub> to 000F_7FFF <sub>H</sub>	4KB	4KB
000F_4000 <sub>H</sub> to 000F_5FFF <sub>H</sub>	4KB	4KB
000F_2000 <sub>H</sub> to 000F_3FFF <sub>H</sub>	4KB	4KB
000F_0000 <sub>H</sub> to 000F_1FFF <sub>H</sub>	4KB	4KB
000E_8000 <sub>H</sub> to 000E_FFFF <sub>H</sub>	16KB	16KB
000E_0000 <sub>H</sub> to 000E_7FFF <sub>H</sub>	16KB	16KB
	bit 31	16 15 0

- The upper and lower banks can be used for the following four combinations of operations:

	Upper bank	Lower bank
1	Read	Read
2	Read	Program/Sector-erase
3	Program/Sector-erase	Read
4	Chip erasing	Chip erasing

**Figure 19.6-1 Program Sequence (Example)**



# ***CHAPTER 20***

---

# ***SERIAL PROGRAMMING CONNECTION***

This chapter explains the basic configuration for serial programming, pins used for serial on-board programming, the connection example for serial programming, and system configuration for flash microcontroller programmer.

## 20.1 Overview



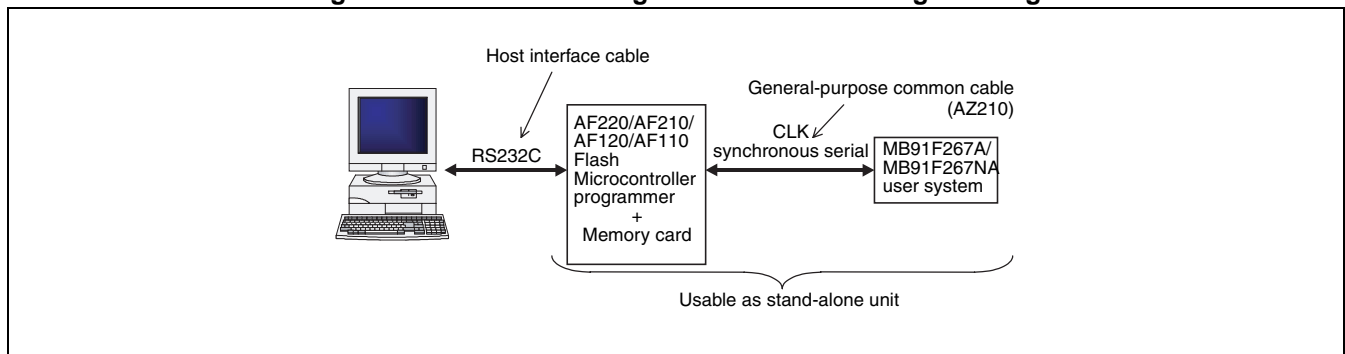
## 20.1 Overview

The MB91F267A/MB91F267NA supports serial on-board programming (Fujitsu standard) into flash ROM. This section summarizes its specifications.

### ■ Basic Configuration for Serial Programming

Fujitsu-standard serial on-board programming uses the AF220/AF210/AD120/AF110 flash microcontroller programmer manufactured by Yokogawa Digital Computer Corporation. Write operation can be done by program that operates in the single-chip mode. Figure 20.1-1 illustrates the basic configuration for MB91F267A/MB91F267NA serial programming connection

**Figure 20.1-1 Basic Configuration for Serial Programming.**



**Note:**

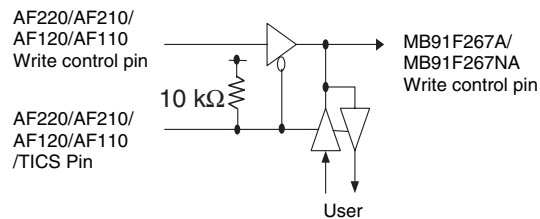
For the functions and usage of the AF210 flash microcontroller programmer or its general-purpose common cable (AZ210) for connection and connector, contact Yokogawa Digital Computer Corporation.

## ■ Pins Used for Fujitsu-standard Serial On-board Programming

Pin	Function	Supplementary Information
MD2,MD1,MD0	Mode Pin	Controlled for program mode Flash serial program mode: MD2,MD1,MD0=1,0,0 Reference: Single-chip mode: MD2,MD1,MD0=0,0,0
P30,P31	Write program startup pins	Input "L" level to P30, "H" level to P31.
INIT	Reset pin	-
SIN0	Serial data input pin	Uses the UART ch.0 resource as clock synchronous mode.
SOT0	Serial data output pin	
SCK0	Serial clock input pin	
V <sub>CC</sub>	Power voltage supply pin	Supply a program voltage from the user system. Do not connect to the power supply of the user side when connecting.
V <sub>SS</sub>	GND pin	GND pin is common to the GND of the flash microcontroller programmer.

### Notes:

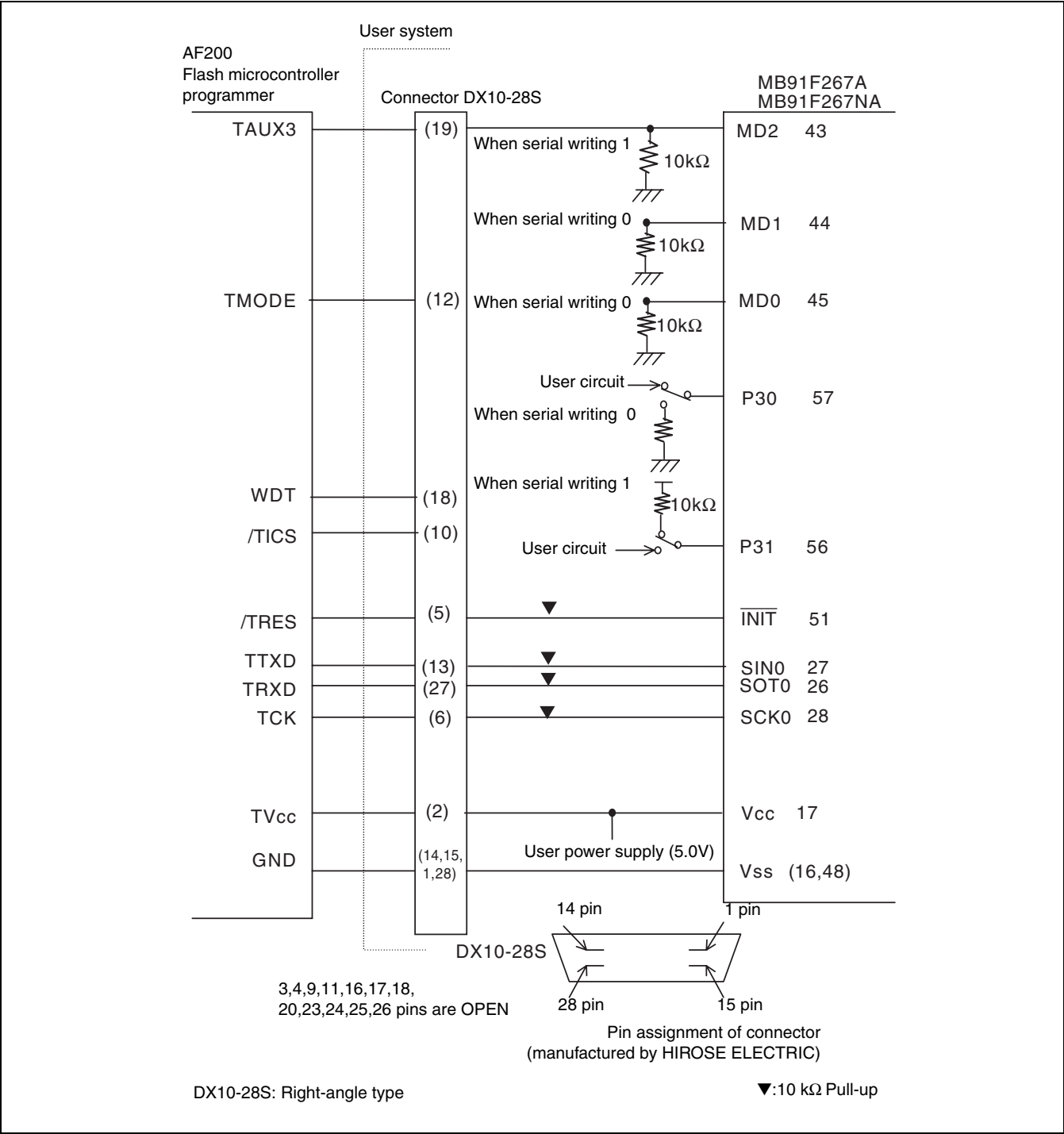
- The control circuit shown below is required for the user system to use the P30, P31, SIN0, SOT0, and SCK0 pins. (During serial programming, the user circuit can be isolated by the/TICS signal of the flash microcontroller programmer.)



- The AF200 must be connected with the user power supply off.

■ Connection Example for Serial Programming

Figure 20.1-2 Connection Example for MB91F267A/MB91F267NA Serial Programming



## ■ System Configuration for AF200 Flash Microcontroller Programmer (Manufactured by Yokogawa Digital Computer Corporation)

Model		Function
Unit	AF220 /AC4P	Ethernet interface model /100V to 220V power supply adapter
	AF210 /AC4P	Standard model /100 V to 220 V power supply adapter
	AF120 /AC4P	Single key Ethernet interface model /100V to 220V power supply adapter
	AF110 /AC4P	Single key model /100 V to 220 V power supply adapter
AZ221		PC/AT RS232C cable for writer
AZ210		Standard target probe (a) length: 1 m
FF201		Control module for Fujitsu FR flash microcontroller
AZ290		Remote controller
/P4		4MB PC Card (Option) Flash memory capacity is up to 512 KB.

Contact: Yokogawa Digital Computer Corporation

Phone: + 81-42-333-6224

## ■ Oscillation Clock Frequency

The oscillation clock frequency available for programming into flash memory is 4.0 MHz.

## ■ Other Notes

The port state during programming into flash memory using a serial programmer is the same as the reset state, except for the pin being used for programming.



# APPENDIX

---

**The appendix describes pin states in each CPU state, notes on using the little-endian areas, a list of FR family instructions, and notes on using MB91265A series.**

APPENDIX A I/O Map

APPENDIX B Interrupt Vector

APPENDIX C Pin States in Each CPU State

APPENDIX D Notes when Little Endian Region is used

APPENDIX E Instruction List

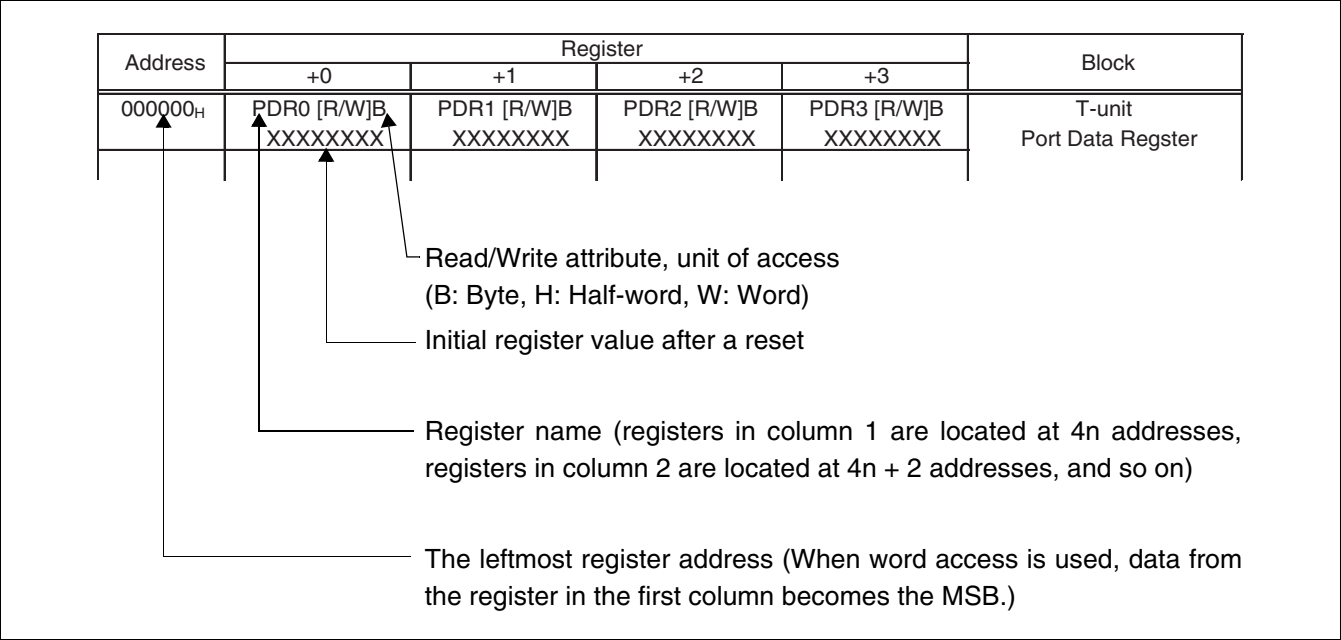
APPENDIX F Notes on Using

# APPENDIX A I/O Map

This appendix shows the correspondence between the various peripheral resource registers and the memory space area.

## I/O Map

Figure A-1 View in Table



Note:

The bit value of the register shows the initial value as follows.

"1" : Initial value "1"

"0" : Initial value "0"

"X" : Initial value "X"

- : There is physically no register in the position.

The access by the data access attribute not described is disabled.

Table A-1 I/O Map (1 / 9)

Address	Register				Block
	+0	+1	+2	+3	
000000 <sub>H</sub>	PDR0 [R/W] B,H,W XXXXXXXXXX	PDR1 [R/W] B,H,W XXXXXXXXXX	PDR2 [R/W] B,H,W XXXXXXXXXX	PDR3 [R/W] B,H,W XXXXXXXXXX	Port Data Register
000004 <sub>H</sub>	PDR4 [R/W] B,H,W XXXXXXXXXX	PDR5 [R/W] B,H,W XXXXXXXXXX	—	—	
000008 <sub>H</sub>	—	—	—	—	
00000C <sub>H</sub>	—	—	—	—	
000010 <sub>H</sub>	PDRG [R/W] B,H,W -----X-	—	—	—	
000014 <sub>H</sub> to 00003C <sub>H</sub>	—				Reserved
000040 <sub>H</sub>	EIRR0 [R/W] B,H,W 00000000	ENIR0 [R/W] B,H,W 00000000	ELVR0 [R/W] B,H,W 00000000 00000000		External Interrupt (INT0 to INT7)
000044 <sub>H</sub>	DICR [R/W] B,H,W -----0	HRCL [R/W,R] B,H,W 0--11111	—	—	Delay Interrupt/ Hold Request
000048 <sub>H</sub>	TMRLR0 [W] H,W XXXXXXXXXX XXXXXXXXXX		TMR0 [R] H,W XXXXXXXXXX XXXXXXXXXX		Reload Timer 0
00004C <sub>H</sub>	—	—	TMCSR0 [R/W,R] B,H,W ---00000 00000000		
000050 <sub>H</sub>	TMRLR1 [W] H,W XXXXXXXXXX XXXXXXXXXX		TMR1 [R] H,W XXXXXXXXXX XXXXXXXXXX		Reload Timer 1
000054 <sub>H</sub>	—	—	TMCSR1 [R/W,R] B,H,W ---00000 00000000		
000058 <sub>H</sub>	TMRLR2 [W] H,W XXXXXXXXXX XXXXXXXXXX		TMR2 [R] H,W XXXXXXXXXX XXXXXXXXXX		Reload Timer 2
00005C <sub>H</sub>	—		TMCSR2 [R/W,R] B,H,W ---00000 00000000		
000060 <sub>H</sub>	SSR0 [R/W,R] B,H,W 00001000	SIDR0[R]/SODR0[W] B,H,W XXXXXXXXXX	SCR0 [R/W,W] B,H,W 00000100	SMR0 [R/W,W] B,H,W 00--0-0-	UART0
000064 <sub>H</sub>	UTIM0 [R] H / UTIMR0 [W] H 00000000 00000000		DRCL0 -----*1	UTIMC0 [R/W] B 0--00001	U-Timer 0
000068 <sub>H</sub>	SSR1 [R/W,R] B,H,W 00001000	SIDR1[R]/SODR1[W] B,H,W XXXXXXXXXX	SCR1 [R/W,W] B,H,W 00000100	SMR1 [R/W] B,H,W 00--0-0-	UART1
00006C <sub>H</sub>	UTIM1 [R] H / UTIMR1 [W] H 00000000 00000000		DRCL1 -----*1	UTIMC1 [R/W] B 0--00001	U-Timer 1
000070 <sub>H</sub>	—	—	—	—	Reserved
000074 <sub>H</sub>	—	—	—	—	
000078 <sub>H</sub>	—	—	—	—	
00007C <sub>H</sub>	—	—	—	—	



**Table A-1 I/O Map (2 / 9)**

Address	Register				Block
	+0	+1	+2	+3	
000080 <sub>H</sub>	ADCH1 [R/W] B,H,W XX000000	ADMD1 [R/W] B,H,W 00001111	ADCD11 [R] B,H,W XXXXXXXXXX	ADCD10 [R] B,H,W XXXXXXXXXX	A/D Converter 1 / Analog Input Control 1
000084 <sub>H</sub>	ADCS1[R/W,W] B,H,W 00000X00	—	AICR1 [R/W] B,H,W ----0000	—	
000088 <sub>H</sub>	ADCH2 [R/W] B,H,W XX000000	ADMD2 [R/W] B,H,W 00001111	ADCD21 [R] B,H,W XXXXXXXXXX	ADCD20 [R] B,H,W XXXXXXXXXX	A/D Converter 2 / Analog Input Control 2
00008C <sub>H</sub>	ADCS2[R/W,W] B,H,W 00000X00	—	AICR2 [R/W] B,H,W -0000000	—	
000090 <sub>H</sub>	OCCPBH0, OCCPBL0[W]/OCCPH0, OCCPL0 [R] H,W 00000000 00000000		OCCPBH1, OCCPBL1[W]/OCCPH1, OCCPL1 [R] H,W 00000000 00000000		16-bit OCU
000094 <sub>H</sub>	OCCPBH2, OCCPBL2[W]/OCCPH2, OCCPL2 [R] H,W 00000000 00000000		OCCPBH3, OCCPBL3[W]/OCCPH3, OCCPL3 [R] H,W 00000000 00000000		
000098 <sub>H</sub>	OCCPBH4, OCCPBL4[W]/OCCPH4, OCCPL4 [R] H,W 00000000 00000000		OCCPBH5, OCCPBL5[W]/OCCPH5, OCCPL5 [R] H,W 00000000 00000000		
00009C <sub>H</sub>	OCSH1 [R/W] B,H,W X1100000	OCSL0 [R/W] B,H,W 00001100	OCSH3 [R/W] B,H,W X1100000	OCSL2 [R/W] B,H,W 00001100	
0000A0 <sub>H</sub>	OCSH5 [R/W] B,H,W X1100000	OCSL4 [R/W] B,H,W 00001100	OCMOD [R/W] B,H,W XX000000	—	
0000A4 <sub>H</sub>	CPCLRBH0, CPCLRBL0[W]/CPCLRH0, CPCLRL0[R] H,W 11111111 11111111		TCDTH0, TCDTL0 [R/W] H,W 00000000 00000000		16-bit Free-run Timer 0
0000A8 <sub>H</sub>	TCCSH0 [R/W] B,H,W 00000000	TCCSL0 [R/W] B,H,W 01000000	—	ADTRGC [R/W] B,H,W XXXXX0000	
0000AC <sub>H</sub>	IPCPH0, IPCPL0 [R] H,W XXXXXXXXXX XXXXXXXXXX		IPCPH1, IPCPL1 [R] H,W XXXXXXXXXX XXXXXXXXXX		16-bit ICU
0000B0 <sub>H</sub>	IPCPH2, IPCPL2 [R] H,W XXXXXXXXXX XXXXXXXXXX		IPCPH3, IPCPL3 [R] H,W XXXXXXXXXX XXXXXXXXXX		
0000B4 <sub>H</sub>	PICSH01 [W] B,H,W 000000--	PICSL01 [R/W] B,H,W 00000000	ICSH23 [R] B,H,W XXXXXX00	ICSL23 [R/W] B,H,W 00000000	
0000B8 <sub>H</sub>	—	—	—	—	Reserved
0000BC <sub>H</sub>	TMRRH0, TMRRL0 [R/W] H,W XXXXXXXXXX XXXXXXXXXX		TMRRH1, TMRRL1 [R/W] H,W XXXXXXXXXX XXXXXXXXXX		Waveform Generator
0000C0 <sub>H</sub>	TMRRH2, TMRRL2 [R/W] H,W XXXXXXXXXX XXXXXXXXXX		—	—	
0000C4 <sub>H</sub>	DTCCR0 [R/W] B,H,W 00000000	DTCCR1 [R/W] B,H,W 00000000	DTCCR2 [R/W] B,H,W 00000000	—	
0000C8 <sub>H</sub>	—	SIGCR1 [R/W] B,H,W 10000000	—	SIGCR2 [R/W] B,H,W XXXXXXXXX1	

Table A-1 I/O Map (3 / 9)

Address	Register				Block
	+0	+1	+2	+3	
0000CC <sub>H</sub>	—	—	ADCOMP1 [R/W] H,W 00000000 00000000		A/D COMP
0000D0 <sub>H</sub>	ADCOMP2 [R/W] H,W 00000000 00000000		ADCOMP2 [R/W] B,H,W XX0000XX	ADCOMP1 [R/W] B,H,W XXXXX00X	
0000D4 <sub>H</sub>	—	—	—	—	Reserved
0000D8 <sub>H</sub>	—	—	—	—	
0000DC <sub>H</sub>	—	—	—	—	
0000E0 <sub>H</sub>	PWCSR0 [R/W,R] B,H,W 00000000 00000000		PWCR0 [R] H,W 00000000 00000000		16-bit PWC Timer
0000E4 <sub>H</sub>	—	—	—	—	
0000E8 <sub>H</sub>	—	PDIVR0 [R/W] B,H,W XXXXX000	—	—	
0000EC <sub>H</sub>	—	—	—	—	Reserved
0000F0 <sub>H</sub>	—	—	—	—	
0000F4 <sub>H</sub> to 0000EC <sub>H</sub>	—	—	—	—	
000100 <sub>H</sub>	PRLH0 [R/W] B,H,W XXXXXXXXXX	PRL0 [R/W] B,H,W XXXXXXXXXX	PRLH1 [R/W] B,H,W XXXXXXXXXX	PRL1 [R/W] B,H,W XXXXXXXXXX	8/16-bit PPG Timer 0 to 7
000104 <sub>H</sub>	PRLH2 [R/W] B,H,W XXXXXXXXXX	PRL2 [R/W] B,H,W XXXXXXXXXX	PRLH3 [R/W] B,H,W XXXXXXXXXX	PRL3 [R/W] B,H,W XXXXXXXXXX	
000108 <sub>H</sub>	PPGC0 [R/W] B,H,W 0000000X	PPGC1 [R/W] B,H,W 0000000X	PPGC2 [R/W] B,H,W 0000000X	PPGC3 [R/W] B,H,W 0000000X	
00010C <sub>H</sub>	PRLH4 [R/W] B,H,W XXXXXXXXXX	PRL4 [R/W] B,H,W XXXXXXXXXX	PRLH5 [R/W] B,H,W XXXXXXXXXX	PRL5 [R/W] B,H,W XXXXXXXXXX	
000110 <sub>H</sub>	PRLH6 [R/W] B,H,W XXXXXXXXXX	PRL6 [R/W] B,H,W XXXXXXXXXX	PRLH7 [R/W] B,H,W XXXXXXXXXX	PRL7 [R/W] B,H,W XXXXXXXXXX	
000114 <sub>H</sub>	PPGC4 [R/W] B,H,W 0000000X	PPGC5 [R/W] B,H,W 0000000X	PPGC6 [R/W] B,H,W 0000000X	PPGC7 [R/W] B,H,W 0000000X	
000118 <sub>H</sub> to 00012C <sub>H</sub>	—	—	—	—	Reserved
000130 <sub>H</sub>	TRG [R/W] B,H,W 00000000 00000000		—	GATEC [R/W] B,H,W XXXXXX00	8/16-bit PPG Timer 0 to 7
000134 <sub>H</sub>	REVC [R/W] B,H,W 00000000 00000000		—	—	
000138 <sub>H</sub>	—	—	—	—	Reserved
00013C <sub>H</sub>	—	—	—	—	
000140 <sub>H</sub>	—	—	—	—	

**Table A-1 I/O Map (4 / 9)**

Address	Register				Block
	+0	+1	+2	+3	
000144 <sub>H</sub>	TTCR [R/W] B,H,W 11110000	—	—	TSTPR0 [R] B,H,W 00000000	Timing Generator
000148 <sub>H</sub>	COMP0 [R/W] B,H,W 00000000	COMP2 [R/W] B,H,W 00000000	COMP4 [R/W] B,H,W 00000000	COMP6 [R/W] B,H,W 00000000	
00014c <sub>H</sub>	—	—	—	—	
000150 <sub>H</sub>	—	—	—	—	
000154 <sub>H</sub>	CPCLRBH1, CPCLRBL1[W]/CPCLRH0, CPCLRL0[R] H,W 11111111 11111111		TCDTH1, TCDTL1 [R/W] H,W 00000000 00000000		16-bit Free-run Timer 1
000158 <sub>H</sub>	TCCSH1 [R/W] B,H,W 00000000	TCCSL1 [R/W] B,H,W 01000000	—	—	
00015C <sub>H</sub>	CPCLRBH2, CPCLRBL2[W]/CPCLRH0, CPCLRL0[R] H,W 11111111 11111111		TCDTH2, TCDTL2 [R/W] H,W 00000000 00000000		16-bit Free-run Timer 2
000160 <sub>H</sub>	TCCSH2 [R/W] B,H,W 00000000	TCCSL2 [R/W] B,H,W 01000000	—	—	
000164 <sub>H</sub>	—	—	—	—	Reserved
000168 <sub>H</sub>	—	FSR2 [R/W] B,H,W 00000000	FSR1 [R/W] B,H,W XXXX0000	FSR0 [R/W] B,H,W 00000000	FRT Selector
00016C <sub>H</sub> to 0001A4 <sub>H</sub>	—				Reserved
0001A8 <sub>H</sub>	CANPRE [R,R/W] B,H,W 00000000	—	—	—	C-CAN <sup>*2</sup> Prescaler
0001AC <sub>H</sub> to 0001FC <sub>H</sub>	—				Reserved

Table A-1 I/O Map (5 / 9)

Address	Register				Block
	+0	+1	+2	+3	
000200 <sub>H</sub>	DMACA0 [R/W] B,H,W <sup>*3</sup> 00000000 00000000 00000000 00000000				DMAC
000204 <sub>H</sub>	DMACB0 [R/W] B,H,W 00000000 00000000 00000000 00000000				
000208 <sub>H</sub>	DMACA1 [R/W] B,H,W <sup>*3</sup> 00000000 00000000 00000000 00000000				
00020C <sub>H</sub>	DMACB1 [R/W] B,H,W 00000000 00000000 00000000 00000000				
000210 <sub>H</sub>	DMACA2 [R/W] B,H,W <sup>*3</sup> 00000000 00000000 00000000 00000000				
000214 <sub>H</sub>	DMACB2 [R/W] B,H,W 00000000 00000000 00000000 00000000				
000218 <sub>H</sub>	DMACA3 [R/W] B,H,W <sup>*3</sup> 00000000 00000000 00000000 00000000				
00021C <sub>H</sub>	DMACB3 [R/W] B,H,W 00000000 00000000 00000000 00000000				
000220 <sub>H</sub>	DMACA4 [R/W] B,H,W <sup>*3</sup> 00000000 00000000 00000000 00000000				
000224 <sub>H</sub>	DMACB4 [R/W] B,H,W 00000000 00000000 00000000 00000000				
000228 <sub>H</sub> to 00023C <sub>H</sub>	—				Reserved
000240 <sub>H</sub>	DMACR [R/W] B 0XX00000 XXXXXXXX XXXXXXXX XXXXXXXX				DMAC
000244 <sub>H</sub> to 00024C <sub>H</sub>	—				Reserved
000250 <sub>H</sub>	—	—	—	—	Reserved
000254 <sub>H</sub> to 000398 <sub>H</sub>	—				Reserved
00039C <sub>H</sub>	—	—	—	—	16-bit MAC
0003A0 <sub>H</sub>	DSP-PC [R/W] XXXXXXXX	DSP-CSR [R/W,R,W] 00000000	DSP-LY [R/W] XXXXXXXX XXXXXXXX		
0003A4 <sub>H</sub>	DSP-OT0 [R] XXXXXXXX XXXXXXXX		DSP-OT1 [R] XXXXXXXX XXXXXXXX		
0003A8 <sub>H</sub>	DSP-OT2 [R] XXXXXXXX XXXXXXXX		DSP-OT3 [R] XXXXXXXX XXXXXXXX		
0003AC <sub>H</sub>	—	—	—	—	
0003B0 <sub>H</sub>	DSP-OT4 [R] XXXXXXXX XXXXXXXX		DSP-OT5 [R] XXXXXXXX XXXXXXXX		
0003B4 <sub>H</sub>	DSP-OT6 [R] XXXXXXXX XXXXXXXX		DSP-OT7 [R] XXXXXXXX XXXXXXXX		
0003B8 <sub>H</sub>	—	—	—	—	

# APPENDIX

**Table A-1 I/O Map (6 / 9)**

Address	Register				Block
	+0	+1	+2	+3	
0003BC <sub>H</sub> to 0003EC <sub>H</sub>	—				Reserved
0003F0 <sub>H</sub>	BSD0 [W] W XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX				Bit Search Module
0003F4 <sub>H</sub>	BSD1 [R/W] W XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX				
0003F8 <sub>H</sub>	BSDC [W] W XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX				
0003FC <sub>H</sub>	BSRR [R] XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX				
000400 <sub>H</sub>	DDR0 [R/W] B,H,W 00000000	DDR1 [R/W] B,H,W 00000000	DDR2 [R/W] B,H,W 00000000	DDR3 [R/W] B,H,W 00000000	Data Direction Register
000404 <sub>H</sub>	DDR4 [R/W] B,H,W -0000000	DDR5 [R/W] B,H,W 00000000	—	—	
000408 <sub>H</sub>	—	—	—	—	
00040C <sub>H</sub>	—	—	—	—	
000410 <sub>H</sub>	DDRG [R/W] B,H,W -----0-	—	—	—	
000414 <sub>H</sub> to 00041C <sub>H</sub>	—				Reserved
000420 <sub>H</sub>	PFR0 [R/W] B,H,W 00-----	PFR1 [R/W] B,H,W --0-00-0	—	—	Port Function Register
000424 <sub>H</sub>	—	—	—	—	
000428 <sub>H</sub>	—	—	—	—	
00042C <sub>H</sub>	—	—	—	—	
000430 <sub>H</sub>	—	—	—	PTFR0 [R/W] B,H,W 00000000	
000434 <sub>H</sub> to 00043C <sub>H</sub>	—				Reserved

Table A-1 I/O Map (7 / 9)

Address	Register				Block
	+0	+1	+2	+3	
000440 <sub>H</sub>	ICR00 [R/W,R] B,H,W ---11111	ICR01 [R/W,R] B,H,W ---11111	ICR02 [R/W,R] B,H,W ---11111	ICR03 [R/W,R] B,H,W ---11111	Interrupt Control Unit
000444 <sub>H</sub>	ICR04 [R/W,R] B,H,W ---11111	ICR05 [R/W,R] B,H,W ---11111	ICR06 [R/W,R] B,H,W ---11111	ICR07 [R/W,R] B,H,W ---11111	
000448 <sub>H</sub>	ICR08 [R/W,R] B,H,W ---11111	ICR09 [R/W,R] B,H,W ---11111	ICR10 [R/W,R] B,H,W ---11111	ICR11 [R/W,R] B,H,W ---11111	
00044C <sub>H</sub>	ICR12 [R/W,R] B,H,W ---11111	ICR13 [R/W,R] B,H,W ---11111	ICR14 [R/W,R] B,H,W ---11111	ICR15 [R/W,R] B,H,W ---11111	
000450 <sub>H</sub>	ICR16 [R/W,R] B,H,W ---11111	ICR17 [R/W,R] B,H,W ---11111	ICR18 [R/W,R] B,H,W ---11111	ICR19 [R/W,R] B,H,W ---11111	
000454 <sub>H</sub>	ICR20 [R/W,R] B,H,W ---11111	ICR21 [R/W,R] B,H,W ---11111	ICR22 [R/W,R] B,H,W ---11111	ICR23 [R/W,R] B,H,W ---11111	
000458 <sub>H</sub>	ICR24 [R/W,R] B,H,W ---11111	ICR25 [R/W,R] B,H,W ---11111	ICR26 [R/W,R] B,H,W ---11111	ICR27 [R/W,R] B,H,W ---11111	
00045C <sub>H</sub>	ICR28 [R/W,R] B,H,W ---11111	ICR29 [R/W,R] B,H,W ---11111	ICR30 [R/W,R] B,H,W ---11111	ICR31 [R/W,R] B,H,W ---11111	
000460 <sub>H</sub>	ICR32 [R/W,R] B,H,W ---11111	ICR33 [R/W,R] B,H,W ---11111	ICR34 [R/W,R] B,H,W ---11111	ICR35 [R/W,R] B,H,W ---11111	
000464 <sub>H</sub>	ICR36 [R/W,R] B,H,W ---11111	ICR37 [R/W,R] B,H,W ---11111	ICR38 [R/W,R] B,H,W ---11111	ICR39 [R/W,R] B,H,W ---11111	
000468 <sub>H</sub>	ICR40 [R/W,R] B,H,W ---11111	ICR41 [R/W,R] B,H,W ---11111	ICR42 [R/W,R] B,H,W ---11111	ICR43 [R/W,R] B,H,W ---11111	
00046C <sub>H</sub>	ICR44 [R/W,R] B,H,W ---11111	ICR45 [R/W,R] B,H,W ---11111	ICR46 [R/W,R] B,H,W ---11111	ICR47 [R/W,R] B,H,W ---11111	
000470 <sub>H</sub> to 00047C <sub>H</sub>	—				Reserved
000480 <sub>H</sub>	RSRR [R,R/W] B,H,W 10000000	STCR [R/W] B,H,W 00110011	TBCR [R/W] B,H,W 00XXXX00	CTBR [W] B,H,W XXXXXXXXXX	Clock Control Unit
000484 <sub>H</sub>	CLKR [R/W] B,H,W 00000000	WPR -----*1	DIVR0 [R/W] B,H,W 00000011	DIVR1 [R/W] B,H,W 00000000	
000488 <sub>H</sub>	—	—	—	—	
00048C <sub>H</sub>	—	—	—	—	
000490 <sub>H</sub>	—	—	—	—	
000494 <sub>H</sub> to 0005FC <sub>H</sub>	—				Reserved
000600 <sub>H</sub>	PCR0 [R/W] B,H,W 00000000	PCR1 [R/W] B,H,W 00000000	PCR2 [R/W] B,H,W 00000000	PCR3 [R/W] B,H,W 00-----	Pull Up Control
000604 <sub>H</sub>	PCR4 [R/W] B,H,W ----0000	—	—	—	
000608 <sub>H</sub>	—	—	—	—	
00060C <sub>H</sub>	—	—	—	—	
000610 <sub>H</sub>	PCRG [R/W] B,H,W -----0-	—	—	—	

**Table A-1 I/O Map (8 / 9)**

Address	Register				Block
	+0	+1	+2	+3	
000614 <sub>H</sub> to 000FFC <sub>H</sub>	—				Reserved
001000 <sub>H</sub>	DMASA0 [R/W] W 00000000 00000000 00000000 00000000				DMAC
001004 <sub>H</sub>	DMADA0 [R/W] W 00000000 00000000 00000000 00000000				
001008 <sub>H</sub>	DMASA1 [R/W] W 00000000 00000000 00000000 00000000				
00100C <sub>H</sub>	DMADA1 [R/W] W 00000000 00000000 00000000 00000000				
001010 <sub>H</sub>	DMASA2 [R/W] W 00000000 00000000 00000000 00000000				
001014 <sub>H</sub>	DMADA2 [R/W] W 00000000 00000000 00000000 00000000				
001018 <sub>H</sub>	DMASA3 [R/W] W 00000000 00000000 00000000 00000000				
00101C <sub>H</sub>	DMADA3 [R/W] W 00000000 00000000 00000000 00000000				
001020 <sub>H</sub>	DMASA4 [R/W] W 00000000 00000000 00000000 00000000				
001024 <sub>H</sub>	DMADA4 [R/W] W 00000000 00000000 00000000 00000000				
001028 <sub>H</sub> to 006FFC <sub>H</sub>	—				Reserved
007000 <sub>H</sub>	FLCR [R/W,R] B 01101000	—	—	—	Flash
007004 <sub>H</sub>	FLWC [R/W,R] B 00000011	—	—	—	
007008 <sub>H</sub>	—	—	—	—	
00700C <sub>H</sub>	—	—	—	—	
007010 <sub>H</sub>	—	—	—	—	
007014 <sub>H</sub> to 00BFFC <sub>H</sub>	—				Reserved
00C000 <sub>H</sub> to 00C07C <sub>H</sub>	X-RAM (coefficient RAM) [R/W] 64 × 16-bit				16-bit MAC
00C080 <sub>H</sub> to 00C0FC <sub>H</sub>	Y-RAM (variable RAM) [R/W] 64 × 16-bit				
00C100 <sub>H</sub> to 00C2FC <sub>H</sub>	I-RAM (instruction RAM) [R/W] 256 × 16-bit				

Table A-1 I/O Map (9 / 9)

Address	Register				Block
	+0	+1	+2	+3	
00C300 <sub>H</sub> to 00FFFC <sub>H</sub>	—				Reserved
020000 <sub>H</sub>	CTRLR0 [R,R/W] 00000000 00000001		STATR0 [R,R/W] 00000000 00000000		C-CAN *2
020004 <sub>H</sub>	ERRCNT0 [R] 00000000 00000000		BTR0 [R,R/W] 00100011 00000001		
020008 <sub>H</sub>	INTR0 [R] 00000000 00000000		TESTR0 [R,R/W] 00000000 X0000000		
02000C <sub>H</sub>	BRPER0 [R,R/W] 00000000 00000000		—	—	
020010 <sub>H</sub>	IF1CREQ0 [R,R/W] 00000000 00000000		IF1CMSK0 [R,R/W] 00000000 00000000		
020014 <sub>H</sub>	IF1MSK20 [R,R/W] 11111111 11111111		IF1MSK10 [R/W] 11111111 11111111		
020018 <sub>H</sub>	IF1ARB20 [R/W] 00000000 00000000		IF1ARB10 [R/W] 00000000 00000000		
02001C <sub>H</sub>	IF1MCTR0 [R,R/W] 00000000 00000000		—	—	
020020 <sub>H</sub>	IF1DTA10 [R/W] 00000000 00000000		IF1DTA20 [R/W] 00000000 00000000		
020024 <sub>H</sub>	IF1DTB10 [R/W] 00000000 00000000		IF1DTB20 [R/W] 00000000 00000000		
020030 <sub>H</sub>	Reserved (IF1 data mirror, little endian byte ordering)				
020040 <sub>H</sub>	IF2CREQ0 [R,R/W] 00000000 00000000		IF2CMSK0 [R,R/W] 00000000 00000000		
020044 <sub>H</sub>	IF2MSK20 [R,R/W] 11111111 11111111		IF2MSK10 [R/W] 11111111 11111111		
020048 <sub>H</sub>	IF2ARB20 [R/W] 00000000 00000000		IF2ARB10 [R/W] 00000000 00000000		
02004C <sub>H</sub>	IF2MCTR0 [R,R/W] 00000000 00000000		—	—	
020050 <sub>H</sub>	IF2DTA10 [R/W] 00000000 00000000		IF2DTA20 [R/W] 00000000 00000000		
020054 <sub>H</sub>	IF2DTB10 [R/W] 00000000 00000000		IF2DTB20 [R/W] 00000000 00000000		
020060 <sub>H</sub>	Reserved (IF2 data mirror, little endian byte ordering)				
020080 <sub>H</sub>	TREQR20 [R] 00000000 00000000		TREQR10 [R] 00000000 00000000		
020090 <sub>H</sub>	NEWDT20 [R] 00000000 00000000		NEWDT10 [R] 00000000 00000000		
0200A0 <sub>H</sub>	INTPND20 [R] 00000000 00000000		INTPND10 [R] 00000000 00000000		
0200B0 <sub>H</sub>	MESVAL20 [R] 00000000 00000000		MESVAL10 [R] 00000000 00000000		C-CAN *2



## APPENDIX

\*1: These are reserved registers. These access are prohibited.

\*2: C-CAN is loaded in MB91F267NA/MB91267NA.

\*3: The lower 16 bits (DTC15 to DTC0) between DMACA0 to DMACA4 cannot be accessed as bytes.

Notes: • The initial value of FLWC (7004<sub>H</sub>) is "00010011<sub>B</sub>" on the tool for the evaluation product. There is no change in operation even if "00000011<sub>B</sub>" is written with the evaluation product.

- Do not use RMW instructions on registers with write-only bits.
- Data in areas marked as "Reserved" or (-) is undefined.

## APPENDIX B Interrupt Vector

This appendix shows the vector table of the MB91265A series.

### ■ Interrupt Vector

Table B-1 Vector Table (1 / 3)

Interrupt Factor	Interrupt Number		Interrupt Level	Offset	TBR Default Address
	Decimal	Hexa-decimal			
Reset	0	00	-	3FC <sub>H</sub>	000FFFC <sub>H</sub>
Mode vector	1	01	-	3F8 <sub>H</sub>	000FFF8 <sub>H</sub>
System reservation	2	02	-	3F4 <sub>H</sub>	000FFF4 <sub>H</sub>
System reservation	3	03	-	3F0 <sub>H</sub>	000FFF0 <sub>H</sub>
System reservation	4	04	-	3EC <sub>H</sub>	000FFEC <sub>H</sub>
System reservation	5	05	-	3E8 <sub>H</sub>	000FFE8 <sub>H</sub>
System reservation	6	06	-	3E4 <sub>H</sub>	000FFE4 <sub>H</sub>
Coprocessor absent trap	7	07	-	3E0 <sub>H</sub>	000FFE0 <sub>H</sub>
Coprocessor error trap	8	08	-	3DC <sub>H</sub>	000FFDC <sub>H</sub>
INTE instruction	9	09	-	3D8 <sub>H</sub>	000FFD8 <sub>H</sub>
System reservation	10	0A	-	3D4 <sub>H</sub>	000FFD4 <sub>H</sub>
System reservation	11	0B	-	3D0 <sub>H</sub>	000FFD0 <sub>H</sub>
Step trace trap	12	0C	-	3CC <sub>H</sub>	000FFCC <sub>H</sub>
NMI demand (tool)	13	0D	-	3C8 <sub>H</sub>	000FFC8 <sub>H</sub>
Undefined instruction exception	14	0E	-	3C4 <sub>H</sub>	000FFC4 <sub>H</sub>
NMI demand	15	0F	15(F <sub>H</sub> ) fixed	3C0 <sub>H</sub>	000FFC0 <sub>H</sub>
External interrupt 0	16	10	ICR00	3BC <sub>H</sub>	000FFBC <sub>H</sub>
External interrupt 1	17	11	ICR01	3B8 <sub>H</sub>	000FFB8 <sub>H</sub>
External interrupt 2	18	12	ICR02	3B4 <sub>H</sub>	000FFB4 <sub>H</sub>
External interrupt 3	19	13	ICR03	3B0 <sub>H</sub>	000FFB0 <sub>H</sub>
External interrupt 4	20	14	ICR04	3AC <sub>H</sub>	000FFAC <sub>H</sub>
External interrupt 5	21	15	ICR05	3A8 <sub>H</sub>	000FFA8 <sub>H</sub>
External interrupt 6/C-CAN wake up*	22	16	ICR06	3A4 <sub>H</sub>	000FFA4 <sub>H</sub>
External interrupt 7	23	17	ICR07	3A0 <sub>H</sub>	000FFA0 <sub>H</sub>
Reload timer 0	24	18	ICR08	39C <sub>H</sub>	000FF9C <sub>H</sub>
Reload timer 1	25	19	ICR09	398 <sub>H</sub>	000FF98 <sub>H</sub>
Reload timer 2	26	1A	ICR10	394 <sub>H</sub>	000FF94 <sub>H</sub>
UART0 (receive completed)	27	1B	ICR11	390 <sub>H</sub>	000FF90 <sub>H</sub>

**Table B-1 Vector Table (2 / 3)**

Interrupt Factor	Interrupt Number		Interrupt Level	Offset	TBR Default Address
	Decimal	Hexa-decimal			
UART0 (transmit completed)	28	1C	ICR12	38C <sub>H</sub>	000FFF8C <sub>H</sub>
DTTI pin	29	1D	ICR13	388 <sub>H</sub>	000FFF88 <sub>H</sub>
DMAC0 (end and error)	30	1E	ICR14	384 <sub>H</sub>	000FFF84 <sub>H</sub>
DMAC1 (end and error)	31	1F	ICR15	380 <sub>H</sub>	000FFF80 <sub>H</sub>
DMAC2/DMAC3/DMAC4 (end and error)	32	20	ICR16	37C <sub>H</sub>	000FFF7C <sub>H</sub>
UART1 (receive completed)	33	21	ICR17	378 <sub>H</sub>	000FFF78 <sub>H</sub>
UART1 (transmit completed)	34	22	ICR18	374 <sub>H</sub>	000FFF74 <sub>H</sub>
C-CAN 0*	35	23	ICR19	370 <sub>H</sub>	000FFF70 <sub>H</sub>
System reservation	36	24	ICR20	36C <sub>H</sub>	000FFF6C <sub>H</sub>
Sum of product	37	25	ICR21	368 <sub>H</sub>	000FFF68 <sub>H</sub>
PPG0/PPG1	38	26	ICR22	364 <sub>H</sub>	000FFF64 <sub>H</sub>
PPG2/PPG3	39	27	ICR23	360 <sub>H</sub>	000FFF60 <sub>H</sub>
PPG4/PPG5/PPG6/PPG7	40	28	ICR24	35C <sub>H</sub>	000FFF5C <sub>H</sub>
System reservation	41	29	ICR25	358 <sub>H</sub>	000FFF58 <sub>H</sub>
Waveform (underflow) 0/1/2	42	2A	ICR26	354 <sub>H</sub>	000FFF54 <sub>H</sub>
Free-run timer 1 (compare clear)	43	2B	ICR27	350 <sub>H</sub>	000FFF50 <sub>H</sub>
Free-run timer 1 (0-detect)	44	2C	ICR28	34C <sub>H</sub>	000FFF4C <sub>H</sub>
Free-run timer 2 (compare clear)	45	2D	ICR29	348 <sub>H</sub>	000FFF48 <sub>H</sub>
Free-run timer 2 (0-detect)	46	2E	ICR30	344 <sub>H</sub>	000FFF44 <sub>H</sub>
Time-base timer overflow	47	2F	ICR31	340 <sub>H</sub>	000FFF40 <sub>H</sub>
Free-run timer 0 (compare clear)	48	30	ICR32	33C <sub>H</sub>	000FFF3C <sub>H</sub>
Free-run timer 0 (0-detect)	49	31	ICR33	338 <sub>H</sub>	000FFF38 <sub>H</sub>
System reservation	50	32	ICR34	334 <sub>H</sub>	000FFF34 <sub>H</sub>
A/D1	51	33	ICR35	330 <sub>H</sub>	000FFF30 <sub>H</sub>
A/D2	52	34	ICR36	32C <sub>H</sub>	000FFF2C <sub>H</sub>
PWC0 (measurement finished)	53	35	ICR37	328 <sub>H</sub>	000FFF28 <sub>H</sub>
System reservation	54	36	ICR38	324 <sub>H</sub>	000FFF24 <sub>H</sub>
PWC0 (overflow)	55	37	ICR39	320 <sub>H</sub>	000FFF20 <sub>H</sub>
System reservation	56	38	ICR40	31C <sub>H</sub>	000FFF1C <sub>H</sub>
ICU 0 (fetch)	57	39	ICR41	318 <sub>H</sub>	000FFF18 <sub>H</sub>
ICU 1 (fetch)	58	3A	ICR42	314 <sub>H</sub>	000FFF14 <sub>H</sub>
ICU 2/ICU 3 (fetch)	59	3B	ICR43	310 <sub>H</sub>	000FFF10 <sub>H</sub>
OCU0/OCU1 (match)	60	3C	ICR44	30C <sub>H</sub>	000FFF0C <sub>H</sub>
OCU2/OCU3 (match)	61	3D	ICR45	308 <sub>H</sub>	000FFF08 <sub>H</sub>
OCU4/OCU5 (match)	62	3E	ICR46	304 <sub>H</sub>	000FFF04 <sub>H</sub>
Delay interrupt trigger bit	63	3F	ICR47	300 <sub>H</sub>	000FFF00 <sub>H</sub>

**Table B-1 Vector Table (3 / 3)**

Interrupt Factor	Interrupt Number		Interrupt Level	Offset	TBR Default Address
	Decimal	Hexa-decimal			
System reservation (used for REALOS)	64	40	-	2FC <sub>H</sub>	000FFEFC <sub>H</sub>
System reservation (used for REALOS)	65	41	-	2F8 <sub>H</sub>	000FFE8 <sub>H</sub>
System reservation	66	42	-	2F4 <sub>H</sub>	000FFE4 <sub>H</sub>
System reservation	67	43	-	2F0 <sub>H</sub>	000FFE0 <sub>H</sub>
System reservation	68	44	-	2EC <sub>H</sub>	000FEEC <sub>H</sub>
System reservation	69	45	-	2E8 <sub>H</sub>	000FEE8 <sub>H</sub>
System reservation	70	46	-	2E4 <sub>H</sub>	000FEE4 <sub>H</sub>
System reservation	71	47	-	2E0 <sub>H</sub>	000FEE0 <sub>H</sub>
System reservation	72	48	-	2DC <sub>H</sub>	000FEDC <sub>H</sub>
System reservation	73	49	-	2D8 <sub>H</sub>	000FED8 <sub>H</sub>
System reservation	74	4A	-	2D4 <sub>H</sub>	000FED4 <sub>H</sub>
System reservation	75	4B	-	2D0 <sub>H</sub>	000FED0 <sub>H</sub>
System reservation	76	4C	-	2CC <sub>H</sub>	000FECC <sub>H</sub>
System reservation	77	4D	-	2C8 <sub>H</sub>	000FEC8 <sub>H</sub>
System reservation	78	4E	-	2C4 <sub>H</sub>	000FEC4 <sub>H</sub>
System reservation	79	4F	-	2C0 <sub>H</sub>	000FEC0 <sub>H</sub>
Used in INT instruction	80 to 255	50 to FF	-	2BC <sub>H</sub> to 000 <sub>H</sub>	000FEBC <sub>H</sub> to 000FFC00 <sub>H</sub>

\*: Interrupt of C-CAN is function loaded in MB91F267NA/MB91267NA.

## APPENDIX C Pin States in Each CPU State

---

**This appendix defines the following terms related to pin states:**

---

### ■ Pin States in Each CPU State

Input enabled

Means that an input function can be used.

Input fixed to "0"

A state of a pin, in which "0" is transmitted to internal circuitry, with the external input shut off by the input gate adjacent to the pin.

Output Hi-Z

Means to place a pin in a high impedance state by disabling the pin driving transistor from driving.

Output storage

Means to output the state existing immediately prior to entering this mode.

That is, to output according to an internal resource with an output when it is operating or to preserve an output when the output is provided, for example, as a port.

Preserving the previous state

Means to be able to output or input the state existing immediately prior to entering this mode.

**Table C-1 Single-chip Mode**

Pin No. (LQFP)	Pin Name	Function	At initialization		At sleep	In stop mode	
			$\overline{\text{INIT}}=\text{L}^{*1}$	$\overline{\text{INIT}}=\text{H}^{*2}$		HIZ=0	HIZ=1
3 to 10	P50 to P57	AN0 to AN7	Output Hi-Z/ Input not ready	Output Hi-Z/ Input ready	Retention of the immediately prior state	Retention of the immediately prior state	Output Hi-Z/ input 0 fixed
11 to 13	P44 to P46	AN8 to AN10					
14	$\overline{\text{NMI}}$	$\overline{\text{NMI}}$	Input not ready	Input ready	Input ready	Input ready	Input ready
18	P00	PPG1/INT4	Output Hi-Z/ Input not ready	Output Hi-Z/ Input ready	Retention of the immediately prior state	Retention of the immediately prior state	Output Hi-Z/ Input 0 fixed
19	P01	PPG2			Input ready	Input ready	Input ready
20	P02	PPG3/INT5			Retention of the immediately prior state	Retention of the immediately prior state	Output Hi-Z/ Input 0 fixed
21 to 23	P03 to P05	TIN0 to TIN2			Input ready	Input ready	Input ready
24, 25	P06, P07	TOT1, TOT2			Retention of the immediately prior state	Retention of the immediately prior state	Output Hi-Z/ Input 0 fixed
26	P10	SOT0					
27	P11	SIN0					
28	P12	SCK0					
29	P13	SOT1					
30	P14	SIN1					
31	P15	SCK1					
32	P16	PPG5/INT6/ RX0 <sup>*3</sup>			Input ready	Input ready	Input ready
33	P17	PPG6/TX0 <sup>*3</sup>			Retention of the immediately prior state	Retention of the immediately prior state	Output Hi-Z/ Input 0 fixed
34	P20	ADTG1/IC2					
35	P21	ADTG2/IC3					
36	P22	PWI0					
37	P23	DTTI					
38	P24	CKI					
39	P25	IC0					
40	P26	IC1					
41	P27	Port					
42	PG1	PPG0					
49	P37	PPG4					
50	P36	PPG7/INT7			Input ready	Input ready	Input ready
52 to 57	P35 to P30	RTO5 to RTO0			Retention of the immediately prior state	Retention of the immediately prior state	Output Hi-Z/ Input 0 fixed
58 to 61	P40 to P43	INT0 to INT3			Input ready	Input ready	Input ready

\* 1:  $\overline{\text{INIT}} = \text{"L"}:$  Indicates the pin state with  $\overline{\text{INIT}}$  remaining at "L" level.

\* 2:  $\overline{\text{INIT}} = \text{"H"}:$  Indicates the pin state immediately after  $\overline{\text{INIT}}$  transition from "L" to "H".

\* 3: C-CAN pin is loaded in MB91F267NA/MB91267NA.

## APPENDIX D Notes when Little Endian Region is used

---

Here, notes of each the following items when the little endian region is used are explained.

- C Compiler
  - Assembler
  - Linker
  - Debugger
- 

### ■ C Compiler (fcc911)

Care must be taken as operation cannot be guaranteed if the following are performed on the little endian area when programming in C language.

- Arrangement of variable with initial value
- Structure substitution
- Operations other than character type array which uses character-string handling function
- Specifying the -K lib option when using a character-string handling function
- Use of double type and long double type
- Arrangement in little endian region of stack

#### ● Arrangement of variable with initial value

The variable with the initial value cannot be arranged in the little endian region.

The compiler does not generate the initial value of the little endian. The initial value cannot be set though the variable can be arranged in the little endian region.

Please do processing by which the initial value is set at the head of the program.

Example: When you set the initial value in variable `little_data` of the Little endian region

```
extern int little_data;

void little_init(void) {
    little_data = Initial value;
}

void main(void) {
    little_init();
    ...
}
```

## ● Structure substitution

When structures are substituted, the compiler selects the optimal transfer method, and transfers are executed per byte, half-word, or word. Thus, errors will result if structure substitution is performed between a structure variable allocated to the ordinary area and another allocated to the little endian area.

Please substitute the member of structure respectively.

Example: When you substitute structure for structure variable `little_st` of the Little endian region

```
struct tag { char c; int i; } normal_st;
extern struct tag little_st;

#define STRMOVE(DEST, SRC) DEST.c=SRC.c; DEST.i=SRC.i;

void main(void) {
    STRMOVE(little_st, normal_st);
}
```

As the layout of structure members differs per compiler, it should be assumed that the member layout is different from structures compiled by other compilers. A correct result is not obtained in the above-mentioned method this time.

When the layout of structure members is unmatched, do not allocate structure variables to the little endian area.

## ● Operations other than character type array which uses character-string handling function

The character-string handling functions provided in the standard library handle strings in bytes. Thus, errors will result if processing using character string operation functions is performed on areas with types other than the "char", "unsigned char", and "signed char" types that are allocated in the little endian area.

Please do not use such processing.

[Example of trouble] Forwarding of word data by `memcpy`

```
int big = 0x01020304;    /* Big endian region    */
extern int little;        /* Little endian region    */
memcpy(&little, &big, 4); /* Forwarding by memcpy    */
```

The above-mentioned execution result



Where, word data transfer results in an error.

(Correct result)

04	03	02	01
----	----	----	----



### ● Specifying the -K lib option when using a character-string handling function

When a -K lib option is specified, the compiler performs inline development on some character string operation functions. At this time, the compiler may change the processing of the function into half-words or words to select the optimum process.

Therefore, processing to the little endian region is not correctly executed.

While processing using character string operation functions is performed on the little endian area, do not specify the -K lib option.

Similarly, do not specify the -O 4 or -K speed options that include the -K lib option either.

### ● Use of double type and long double type

Accessing double or long double forms can be performed by accessing the upper or lower 1 word respectively. Thus, errors will result if accessing the double or long double form variables allocated to the little endian area.

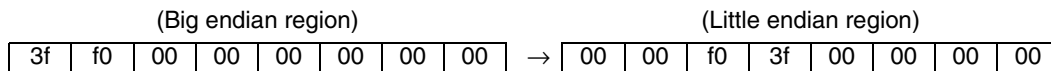
Substitution between the same type of little endian-allocated variables is possible, but as a result of optimization, such substitutions may be replaced by substitution of constant numbers.

Do not place a double or long-double type variable in the little endian area.

[Example of trouble] Forwarding of double type data

```
double big = 1.0;      /* Big endian region      */
extern int little;     /* Little endian region    */
little = big;          /* Forwarding of double type data*/
```

The above-mentioned execution result



Where, double-type data transfer results in an error.

(Correct result) 

00	00	00	00	00	00	f0	3f
----	----	----	----	----	----	----	----

### ● Arrangement in Little endian region of stack

Placing part or whole of the stack in the little endian area yields unpredictable results of operation.

## ■ Assembler (fasm911)

Points to note regarding the little endian area when programming in FR assembler language are described as follows.

### ● About the section

The little endian region has aimed to exchange data with little endian system CPU chiefly. The little endian area should therefore be defined as a data section with no initial value.

If a code, stack, or data section with an initial value is specified for the little endian area, access operations under this model cannot be guaranteed.

[Example]

```
/* Section definition of correct Little endian region */
.SECTION Little_Area, DATA, ALIGN=4

Little_Word:
    .RES.W 1

Little_Half:
    .RES.H 1

Little_Byte:
    .RES.B 1
```

### ● About the access of data

When data access is performed on the little endian area, the data value can be coded without consideration for the endian status. However, to access little endian area data, access must be performed on the same size as the data size.

[Example]

```
LDI    #0x01020304, r0
LDI    #Little_Word, r1

LDI    #0x0102, r2
LDI    #Little_Half, r3

LDI    #0x01, r4
LDI    #Little_Byte, r5

/* Access 32-bit data using the ST instruction (or LD instruction, etc.).*/
ST     r0, @r1

/* Access 16-bit data using the STH instruction (or LDH instruction, etc.).*/
STH    r2, @r3

/* Access 8-bit data using the STB instruction (or LDB instruction, etc.).*/
```

STB r4, @r5

If the little endian area is accessed in a size different from the data size on this model, the resulting value is not guaranteed. For example, if two sequential blocks of 16-bit data are simultaneously accessed using a 32-bit access command, the data value cannot be guaranteed.

## ■ Linker (flnk911)

Points to note regarding the section layout while linking when programs using the little endian area are created are described as follows.

### ● Limitation of section type

In the little endian area, only data sections with no initial value can be placed.

When a data section with initial value, stack section, or code section is allocated in the little endian area, program operation cannot be guaranteed since the arithmetic processing, such as address solutions, is performed on the big endian area within the linker.

### ● Undetection of error

As the linker does not recognize the little endian area, no error message will be sent even if an allocation that violates the above restrictions is performed. Sufficiently check the contents of the sections placed in the little endian area before use.

## ■ Debugger (sim911, eml911, mon911)

### ● Simulator debugger

There is no memory space specification command by which the little endian region is shown.

Memory manipulation commands and instructions which involve memory manipulation are therefore handled as those using the big endian method.

### ● Emulator debugger and monitor debugger

Care must be taken that data will not be treated as having a normal value when the little endian area is accessed using the following command.

- set memory / show memory / enter / examine / set watch Command

When floating point (single/double) data is handled, specific values cannot be set or displayed.

- search memory Command

→ When searching data of half-word or word, the search cannot be executed with specific values.

- The line/the disassembly (The disassembly display of the source window is included).

A normal instruction code cannot be set and displayed. (Do not place instruction code in the little endian area.)

- call / show call Command

→ Arrangement of the stack area in the little endian area does not lead to normal operation. (Do not place the stack area in the little endian area.)

## APPENDIX E Instruction List

It is an instruction table of the FR family.

### ■ Instruction List of FR Family

[How to Read the Instruction List]

Mnemonic	Type	OP	CYC	NZVC	Operation	Remark
ADD Rj, Rj	A	AG	1	CCCC	$R_i + R_j \rightarrow R_j$	
* ADD #s5, Rj	C	A4	1	CCCC	$R_i + s5 \rightarrow R_i$	
,	,	,	,	,	,	
,	,	,	,	,	,	

(1)
(2)
(3)
(4)
(5)
(6)
(7)

(1) The instruction name is shown.

Instructions marked with \* are extended instructions implemented either by extending existing instructions or by coding from scratch using the assembler, which are not native to the CPU.

(2) A specifiable Addressing mode is shown in the operand by the sign.

Please refer to the sign of the Addressing mode (next item) for the meaning of the sign.

(3) The instruction format is shown.

(4) The hexadecimal number is displayed to the instruction code.

(5) The number of machine cycles is shown.

a: It is a memory access cycle, and there is a possibility to postpone by the Ready function.

b: It is a memory access cycle, and there is a possibility to postpone by the Ready function. When the immediately succeeding instruction references the register to be subject to LD operation, however, an interlock is applied and the number of execution cycles is incremented by 1.

c: If the succeeding instruction reads or writes to R15, SSP, or USP or if the instruction is in instruction format A, an interlock is applied and the number of execution cycles is incremented by 1 to become 2.

d: If the succeeding instruction references MDH/MDL, an interlock is applied and the number of execution cycles is incremented to become 2. a, b, c, d, and the minimum are one cycle.

(6) The flag change is shown.

Flag change
C : Change
- : No change
0 : Clear
1 : Set

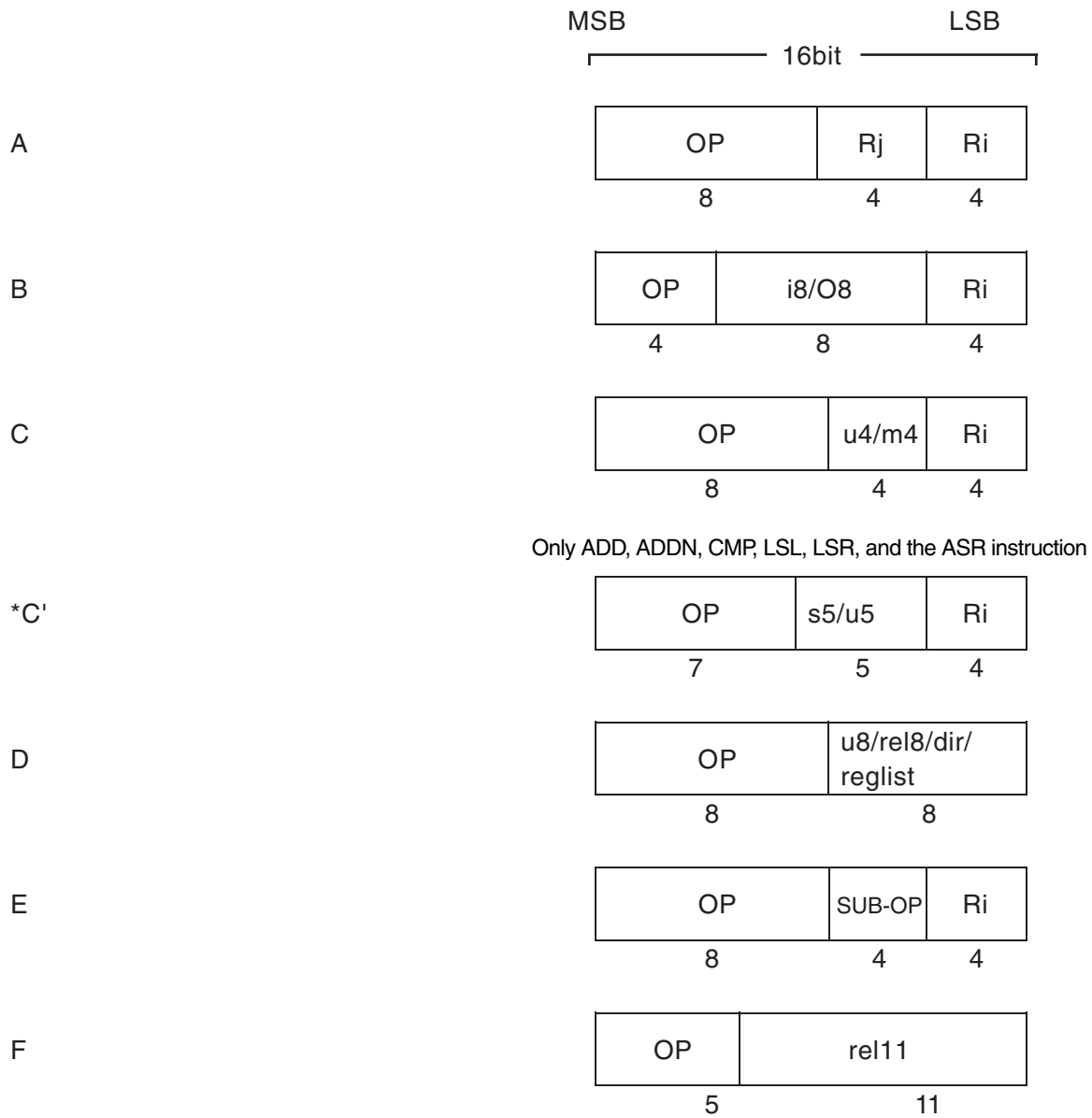
Meaning of flag
N : Negative flag
Z : Zero flag
V : Overflow flag
C : Carry flag

(7) The instruction operation is written.

● Addressing mode symbols

Ri :	register direct (R0 to R15,AC,FP,SP)
Rj :	register direct (R0 to R15,AC,FP,SP)
R13 :	register direct (R13,AC)
Ps :	Register (direct program status register)
Rs:	register direct(TBR, RP, SSP, USP, MDH, MDL)
Cri :	register direct(CR0 to CR15)
CRj :	register direct(CR0 to CR15)
#i8 :	Unsigned 8 bit value immediately(-128 to 255) Attention: - 128 to -1 is treated as 128 to 255.
#i20 :	Unsigned 20 bit value immediately(-0X80000 to 0XFFFFFF) Attention -0X7FFFF to -1 is treated as 0X7FFFF to 0XFFFFFF.
#i32 :	Unsigned 32 bit value immediately(-0X80000000 to 0xFFFFFFFF) Attention: 0X80000000 to -1 is treated as .
#s5 :	Signed 5-bit immediate value (-16 to 15)
#s10 :	Signed 10-bit immediate value (only multiples of - 512 to 5084)
#u4 :	Unsigned 4 bit value immediately (0 to 15)
#u5 :	Unsigned 5 bit value immediately (0 to 31)
#u8 :	Unsigned 8 bit value immediately (0 to 255)
#u10 :	Unsigned ten bit value immediately (Only the multiple of 0 to 10204)
@dir8 :	Unsigned 8 bit direct address (0 to 0XFF)
@dir9 :	Unsigned 8 bit direct address (Only the multiple of 0 to 0X1FE 2)
@dir10 :	Unsigned ten bit direct address (Only the multiple of 0 to 0X3FC4)
label9 :	Signed 9-bit branch address (only multiples of -0X100 to 0XFC 2)
label12 :	Signed 12-bit branch address (only multiples of -0X800 to 0X7FC 2)
label20:	Divergence address of signed 20 bits (-0X80000 to 0X7FFFF)
label32:	Divergence address of signed 32 bits (-0X80000000 to 0X7FFFFFFF)
@Ri :	Register indirect (R0 to R15, AC, FP, SP)
@Rj :	Register indirect (R0 to R15, AC, FP, SP)
@(R13,Rj) :	Relativity is register indirect (Rj: R0 to R15, AC, FP, SP)
@(R14,disp10) :	Relative indirectly register (Only the multiple of disp10: -0X200 to 0X1FC 4)
@(R14,disp9) :	Relative indirectly register (Only the multiple of disp9: -0X100 to 0XFE 2)
@(R14,disp8) :	Relativity is register indirect (disp8: -0X80 to 0X7F)
@(R15,udisp6) :	Relative indirectly register (Only the multiple of udisp6: 0 to 60 4)
@Ri+:	Register indirect with post increment (R0 to R15, AC, FP, SP)
@R13+:	Register indirect with post increment (R13, AC)
@SP+:	Stack pop
@-SP:	Stack push
(reglist) :	Register list

## ● Instruction format



**Table E-1 Addition and Subtraction**

Mnemonic	Type	OP	CYCLE	NZVC	Operation	Remark
ADD Rj, Ri	A	A6	1	CCCC	Ri+Rj→ Ri	I think high rank 1bit to be a sign in the assembler.
*ADD #s5, Ri	C'	A4	1	CCCC	Ri+s5→ Ri	
ADD #u4, Ri	C	A4	1	CCCC	Ri+extu(i4)→ Ri	Byte 0 expansions
ADD2 #u4, Ri	C	A5	1	CCCC	Ri+extu(i4)→ Ri	Minus expansion
ADDN Rj, Ri	A	A7	1	CCCC	Ri+Rj+c→ Ri	Addition with carry
ADDN Rj, Ri	A	A2	1	----	Ri+Rj→ Ri	I think high rank 1bit to be a sign in the assembler.
*ADDN #s5, Ri	C'	A0	1	----	Ri+s5→ Ri	
ADDN #u4, Ri	C	A0	1	----	Ri+extu(i4)→ Ri	Byte 0 expansions
ADDN2 #u4, Ri	C	A1	1	----	Ri+extu(i4)→ Ri	Minus expansion
SUB Rj, Ri	A	AC	1	CCCC	Ri-Rj→ Ri	
SUBC Rj, Ri	A	AD	1	CCCC	Ri-Rj-c→ Ri	Reduction with carry
SUBN Rj, Ri	A	AE	1	----	Ri-Rj→ Ri	

**Table E-2 Comparison Operation**

Mnemonic	Type	OP	CYCLE	NZVC	Operation	Remark
CMP Rj, Ri	A	AA	1	CCCC	Ri-Rj	I think high rank 1bit to be a sign in the assembler.
*CMP #s5, Ri	C'	A8	1	CCCC	Ri-s5	
CMP #u4, Ri	C	A8	1	CCCC	Ri-extu(i4)	Byte 0 expansions
CMP2 #u4, Ri	C	A9	1	CCCC	Ri-extu(i4)	Minus expansion

**Table E-3 Logical Operation**

Mnemonic	Type	OP	CYCLE	NZVC	Operation	RMW	Remark
AND Rj, Ri	A	82	1	CC--	Ri &= Rj	–	word
AND Rj, @Ri	A	84	1+2a	CC--	(Ri) &= Rj	○	word
ANDH Rj, @Ri	A	85	1+2a	CC--	(Ri) &= Rj	○	half-word
ANDB Rj, @Ri	A	86	1+2a	CC--	(Ri) &= Rj	○	6 KB
OR Rj, Ri	A	92	1	CC--	Ri  = Rj	–	word
OR Rj, @Ri	A	94	1+2a	CC--	(Ri)  = Rj	○	word
ORH Rj, @Ri	A	95	1+2a	CC--	(Ri)  = Rj	○	half-word
ORB Rj, @Ri	A	96	1+2a	CC--	(Ri)  = Rj	○	6 KB
EOR Rj, Ri	A	9A	1	CC--	Ri ^= Rj	–	word
EOR Rj, @Ri	A	9C	1+2a	CC--	(Ri) ^= Rj	○	word
EORH Rj, @Ri	A	9D	1+2a	CC--	(Ri) ^= Rj	○	half-word
EORB Rj, @Ri	A	9E	1+2a	CC--	(Ri) ^= Rj	○	6 KB

**Table E-4 Bit Manipulation Instructions**

Mnemonic	Type	OP	CYCLE	NZVC	Operation	RMW	Remark
BANDL #u4, @Ri	C	80	1+2a	----	(Ri)&=(0xF0+u4)	○	The subordinate position four bits are operated. The high rank four bits are operated.
BANDH #u4, @Ri	C	81	1+2a	----	(Ri)&=((u4<<4)+0x0F)	○	
*BAND #u8, @Ri * <sup>1</sup>				----	(Ri)&=u8	–	
BORL #u4, @Ri	C	90	1+2a	----	(Ri)  = u4	○	The subordinate position four bits are operated. The high rank four bits are operated.
BORH #u4, @Ri	C	91	1+2a	----	(Ri)  = (u4<<4)	○	
*BOR #u8, @Ri * <sup>2</sup>				----	(Ri)  = u8	–	
BEORL #u4, @Ri	C	98	1+2a	----	(Ri) ^= u4	○	The subordinate position four bits are operated. The high rank four bits are operated.
BEORH #u4, @Ri	C	99	1+2a	----	(Ri) ^= (u4<<4)	○	
*BEOR #u8, @Ri * <sup>3</sup>				----	(Ri) ^= u8	–	
BTSTL #u4, @Ri	C	88	2+a	0C--	(Ri) & u4	–	The subordinate position four bits are tested. The high rank four bits are tested.
BTSTH #u4, @Ri	C	89	2+a	CC--	(Ri) & (u4<<4)	–	

\*1: The assembler generates BANDL or BANDH when the bit is set at u8&0x0F or u8&0xF0, respectively. Both BANDL and BANDH are occasionally generated.

\*2: The assembler generates BORL or BORH when the bit is set at u8&0x0F or u8&0xF0, respectively. Both BORL and BORH are occasionally generated.

\*3: The assembler generates BEORL or BEORH when the bit is set at u8&0x0F or u8&0xF0, respectively. Both BEORL and BEORH are occasionally generated.



**Table E-5 Multiplication and Division**

Mnemonic	Type	OP	CYCLE	NZVC	Operation	Remark
MUL Rj,Ri	A	AF	5	CCC-	Ri * Rj →MDH,MDL	32bit*32bit=64bit
MULU Rj,Ri	A	AB	5	CCC-	Ri * Rj →MDH,MDL	Unsigned
MULH Rj,Ri	A	BF	3	CC--	Ri * Rj →MDL	16bit*16bit=32bit
MULUH Rj,Ri	A	BB	3	CC--	Ri * Rj →MDL	Unsigned
DIV0S Ri	E	97-4	1	----		Step operation
DIV0U Ri	E	97-5	1	----		32bit/32bit=32bit
DIV1 Ri	E	97-6	d	-C-C		
DIV2 Ri	E	97-7	1	-C-C		
DIV3	E	9F-6	1	----		
DIV4S	E	9F-7	1	----		
*DIV Ri *1			36	-C-C	MDL / Ri →MDL , MDL % Ri →MDH	
*DIVU Ri *2				-C-C	MDL / Ri →MDL , MDL % Ri →MDH	

**Table E-6 Shift**

Mnemonic	Type	OP	CYCLE	NZVC	Operation	Remark
LSL Rj, Ri	A	B6	1	CC-C	Ri << Rj →Ri	Logical shift
*LSL #u5, Ri(u5:0 to 31)	C'	B4	1	CC-C	Ri << u5 →Ri	
LSL #u4, Ri	C	B4	1	CC-C	Ri << u4 →Ri	
LSL2 #u4, Ri	C	B5	1	CC-C	Ri <<(u4+16) →Ri	
LSR Rj, Ri	A	B2	1	CC-C	Ri >> Rj →Ri	Logical shift
*LSR #u5, Ri(u5:0 to 31)	C'	B0	1	CC-C	Ri >> u5 →Ri	
LSR #u4, Ri	C	B0	1	CC-C	Ri >> u4 →Ri	
LSR2 #u4, Ri	C	B1	1	CC-C	Ri >>(u4+16) →Ri	
ASR Rj, Ri	A	BA	1	CC-C	Ri >> Rj →Ri	Arithmetic shift
*ASR #u5, Ri (u5:0 to 31)	C'	B8	1	CC-C	Ri >> u5 →Ri	
ASR #u4, Ri	C	B8	1	CC-C	Ri >> u4 →Ri	
ASR2 #u4, Ri	C	B9	1	CC-C	Ri >>(u4+16) →Ri	

**Table E-7 Value Move Operation of Value Sets/16 Bits/32 Bits Immediately**

Mnemonic	Type	OP	CYCLE	NZVC	Operation	Remark
LDI:32 #i32, Ri	E	9F-8	3	----	i32 →Ri	0 expansions of high rank 12bit.
LDI:20 #i20, Ri	C	9B	2	----	i20 →Ri	0 expansions of high rank 12bit.
LDI:8 #i8, Ri	B	C0	1	----	i8 →Ri	
*LDI # {i8 i20 i32} ,Ri *3					{i8 i20 i32} →Ri	

\*1: DIV0S, DIV1x32, DIV2, DIV3, and DIV4S are generated. The instruction code length becomes 72 bytes.

\*2: DIV0S, DIV1x32 are generated. The instruction code length becomes 66 bytes.

\*3: When the immediate value is an absolute value, i8, i20, or i32 is selected automatically by the assembler. When the immediate value is a relative value or includes an externally referenced symbol, i32 is selected.

**Table E-8 Memory Loading**

Mnemonic	Type	OP	CYCLE	NZVC	Operation	Remark
LD @Rj, Ri	A	04	b	----	(Rj)→ Ri	
LD @(R13,Rj), Ri	A	00	b	----	(R13+Rj)→ Ri	
LD @(R14,disp10), Ri	B	2	b	----	(R14+disp10)→ Ri	
LD @(R15,udisp6), Ri	C	03	b	----	(R15+udisp6)→ Ri	
LD @R15+, Ri	E	07-0	b	----	(R15)→ Ri, R15+=4	
LD @R15+, Rs	E	07-8	b	----	(R15)→ Rs, R15+=4	Rs: Special register
LD @R15+, PS	E	07-9	1+a+b	CCCC	(R15)→ PS, R15+=4	*
LDUH @Rj, Ri	A	05	b	----	(Rj)→ Ri	Byte 0 expansions
LDUH @(R13,Rj), Ri	A	01	b	----	(R13+Rj)→ Ri	Byte 0 expansions
LDUH @(R14,disp9), Ri	B	4	b	----	(R14+disp9)→ Ri	Byte 0 expansions
LDUB @Rj, Ri	A	06	b	----	(Rj)→ Ri	Byte 0 expansions
LDUB @(R13,Rj), Ri	A	02	b	----	(R13+Rj)→ Ri	Byte 0 expansions
LDUB @(R14,disp8), Ri	B	6	b	----	(R14+disp8)→ Ri	Byte 0 expansions

\* : In the hard-spec o8 and o4 fields, the assembler sets values through the following calculation:

disp10/4→ o8, disp9/2→ o8, disp8→ o8, disp10, disp9, disp8: signed. udisp6/4→ o4udisp6: unsigned.

**Table E-9 Memory Store**

Mnemonic	Type	OP	CYCLE	NZVC	Operation	Remark
ST Ri, @Rj	A	14	a	----	Ri→(Rj)	word
ST Ri, @(R13,Rj)	A	10	a	----	Ri→(R13+Rj)	word
ST Ri, @(R14,disp10)	B	3	a	----	Ri→(R14+disp10)	word
ST Ri, @(R15,udisp6)	C	13	a	----	Ri→(R15+udisp6)	
ST Ri, @-R15	E	17-0	a	----	R15-=4, Ri→(R15)	
ST Rs, @-R15	E	17-8	a	----	R15-=4, Rs→(R15)	Rs: Special register
ST PS, @-R15	E	17-9	a	----	R15-=4, PS→(R15)	*
STH Ri, @Rj	A	15	a	----	Ri→(Rj)	half-word
STH Ri, @(R13,Rj)	A	11	a	----	Ri→(R13+Rj)	half-word
STH Ri, @(R14,disp9)	B	5	a	----	Ri→(R14+disp9)	half-word
STB Ri, @Rj	A	16	a	----	Ri→(Rj)	6 KB
STB Ri, @(R13,Rj)	A	12	a	----	Ri→(R13+Rj)	6 KB
STB Ri, @(R14,disp8)	B	7	a	----	Ri→(R14+disp8)	6 KB

\*: In the hard-spec o8 and o4 fields, the assembler sets values through the following calculation:

disp10/4→ o8, disp9/2→ o8, disp8→ o8, disp10, disp9, disp8: signed. udisp6/4→ o4udisp6: unsigned.

**Table E-10 Transfer Between Registers**

Mnemonic	Type	OP	CYCLE	NZVC	Operation	Remark
MOV Rj, Ri	A	8B	1	----	Rj →Ri	Transfer between general-purpose registers
MOV Rs, Ri	A	B7	1	----	Rs →Ri	
MOV Ri, Rs	E	B3	1	----	Ri →Rs	Rs: Special register
MOV PS, Ri	E	17-1	1	----	PS →Ri	Rs: Special register
MOV Ri, PS	E	07-1	c	CCCC	Ri →PS	*

\* : Special register Rs: TBR,RP,USP,SSP,MDH,MDL

**Table E-11 The Divergence (There is No Delay) Usually.**

Mnemonic	Type	OP	CYCLE	NZVC	Operation	Remark
JMP @Ri	E	97-0	2	----	Ri → PC	
CALL label12	E	D0	2	----	PC+2 → RP, PC+2+(label12-PC-2) → PC	
CALL @Ri	F	97-1	2	----	PC+2 → RP, Ri → PC	
RET	E	97-2	2	----	RP → PC	Return
INT #u8	D	1F	3+3a	----	SSP-=4, PS- $\times$ (SSP), SSP-=4, PC+2- $\times$ (SSP), 0 → I flag, 0 → S flag, (TBR+0x3FC-u8 $\times$ 4) → PC	
INTE	E	9F-3	3+3a	----	SSP-=4, PS- $\times$ (SSP), SSP-=4, PC+2- $\times$ (SSP), 0 → S Flag, (TBR+0x3D8) → PC	
RETI	E	97-3	2+2A	CCCC	(R15)-PC, R15-=4, (R15)-PS, R15-=4	for emulator
BRA label9	D	E0	2	----	PC+2+(label9-PC-2) → PC	
BNO label9	D	E1	1	----	Point-to-point	
BEQ label9	D	E2	2/1	----	if(Z==1) then PC+2+(label9-PC-2) → PC	
BNE label9	D	E3	2/1	----	↑ s/Z==0	
BC label9	D	E4	2/1	----	↑ s/C==1	
BNC label9	D	E5	2/1	----	↑ s/C==0	
BN label9	D	E6	2/1	----	↑ s/N==1	
BP label9	D	E7	2/1	----	↑ s/N==0	
BV label9	D	E8	2/1	----	↑ s/V==1	
BNV label9	D	E9	2/1	----	↑ s/V==0	
BLT label9	D	EA	2/1	----	↑ s/V xor N==1	
BGE label9	D	EB	2/1	----	↑ s/V xor N==0	
BLE label9	D	EC	2/1	----	↑ s/(V xor N) or Z==1	
BGT label9	D	ED	2/1	----	↑ s/(V xor N) or Z==0	
BLS label9	D	EE	2/1	----	↑ s/C or Z==1	
BHI label9	D	EF	2/1	----	↑ s/C or Z==0	

Notes: • "2/1" in the CYCLE column means "2" when a branch occurs or "1" when no branch occurs.

- In the hard-spec rel11 and rel8 fields, the assembler sets values through the following calculation:  
(label12-PC-2)/2 → rel11, (label9-PC-2)/2 → rel8, label12, label9 are signed.

- The execution of the RETI instruction assumes that the S flag is "0".

**Table E-12 Delay Divergence**

Mnemonic	Type	OP	CYCLE	NZVC	Operation	Remark
JMP:D @Ri	E	9F-0	1	----	Ri →PC	
CALL:D label12	F	D8	1	----	PC+4→RP, PC+2+(label12-PC-2)→PC	
CALL:D @Ri	E	9F-1	1	----	PC+4→RP, Ri→PC	
RET:D	E	9F-2	1	----	RP →PC	Return
BRA:D label9	D	F0	1	----	PC+2+(label9-PC-2)→PC	
BNO:D label9	D	F1	1	----	Point-to-point	
BEQ:D label9	D	F2	1	----	if(Z==1) then PC+2+(label9-PC-2)→PC	
BNE:D label9	D	F3	1	----	↑ s/Z==0	
BC:D label9	D	F4	1	----	↑ s/C==1	
BNC:D label9	D	F5	1	----	↑ s/C==0	
BN:D label9	D	F6	1	----	↑ s/N==1	
BP:D label9	D	F7	1	----	↑ s/N==0	
BV:D label9	D	F8	1	----	↑ s/V==1	
BNV:D label9	D	F9	1	----	↑ s/V==0	
BLT:D label9	D	FA	1	----	↑ s/V xor N==1	
BGE:D label9	D	FB	1	----	↑ s/V xor N==0	
BLE:D label9	D	FC	1	----	↑ s/(V xor N) or Z==1	
BGT:D label9	D	FD	1	----	↑ s/(V xor N) or Z==0	
BLS:D label9	D	FE	1	----	↑ s/C or Z==1	
BHI:D label9	D	FF	1	----	↑ s/C or Z==0	

- Notes:
- In the hard-spec rel11 and rel8 fields, the assembler sets values through the following calculation:  
(label12-PC-2)/2→rel11, (label9-PC-2)/2→rel8, label12, label9 are signed.
  - A delayed branch always takes place after the instruction that follows (delay slot) is executed.
  - All of 1-cycle, a-, b-, c-, and d-cycle instructions can be placed in the delay slot.  
Two or more-cycle instruction cannot be put.

**Table E-13 The Others**

Mnemonic	Type	OP	CYCLE	NZVC	Operation	RMW	Remark
NOP	E	9F-A	1	----	Anything does not change either.	—	
ANDCCR #u8	D	83	c	CCCC	CCR and u8 $\rightarrow$ CCR	—	
ORCCR #u8	D	93	c	CCCC	CCR or u8 $\rightarrow$ CCR	—	
STILM #u8	D	87	1	----	i8 $\rightarrow$ ILM	—	Value set of ILM immediately
ADDSP #s10 <sup>*1</sup>	D	A3	1	----	R15 += s10	—	ADD SP instruction
EXTSB Ri	E	97-8	1	----	Sign extension 8 $\rightarrow$ 32bit	—	
EXTUB Ri	E	97-9	1	----	Byte 0 expansions 8 $\rightarrow$ 32bit	—	
EXTSH Ri	E	97-A	1	----	Sign extension 16 $\rightarrow$ 32bit	—	
EXTUH Ri	E	97-B	1	----	Byte 0 expansions 16 $\rightarrow$ 32bit	—	
LDM0 (reglist)	D	8C		----	(R15) $\rightarrow$ reglist, R15 increment	—	loading multi R0-R7
LDM1 (reglist)	D	8D		----	(R15) $\rightarrow$ reglist, R15 increment	—	loading multi R8-R15
*LDM (reglist) <sup>*2</sup>				----	(R15) $\rightarrow$ reglist, R15 increment	—	loading multi R0-R15
STM0 (reglist)	D	8E		----	R15 decrement, reglist $\rightarrow$ (R15)	—	store multi R0-R7
STM1 (reglist)	D	8F		----	R15 decrement, reglist $\rightarrow$ (R15)	—	store multi R8-R15
*STM (reglist) <sup>*3</sup>				----	R15 decrement, reglist $\rightarrow$ (R15)	—	store multi R0-R15
ENTER #u10 <sup>*4</sup>	D	0F	1+a	----	R14 $\rightarrow$ (R15 - 4), R15 - 4 $\rightarrow$ R14, R15 - u10 $\rightarrow$ R15	—	Entrance processing of function
LEAVE	E	9F-9	b	----	R14 + 4 $\rightarrow$ R15, (R15 - 4) $\rightarrow$ R14	—	Exit processing of function
XCHB @Rj, Ri	A	8A	2a	----	Ri $\rightarrow$ TEMP (Rj) $\rightarrow$ Ri TEMP $\rightarrow$ (Rj)	○	For semaphore control Byte data

- Notes:
- The number of execution cycles for LDM0 (reglist) and LDM1 (reglist) is  $a \times (n-1) + b + 1$ , where the specified number of registers is n.
  - The number of execution cycles for STM0(reglist) and STM1(reglist) is  $a \times n + 1$ , where the specified number of registers is n.

\*1: S10 calculates s10/4 by the assembler, makes to s8, and sets the value. s10 is signed.

\*2: If reglist specifies any of R0 to R7, LDM0 is generated. If it specifies any of R8 to R15, LDM1 is generated. Both LDM0 and LDM1 are occasionally generated.

\*3: If reglist specifies any of R0 to R7, STM0 is generated. If it specifies any of R8 to R15, STM1 is generated. Both STM1 and STM0 are occasionally generated.

\*4: u10 is obtained by calculating u10/4 and the value is set as u8. u10 is unsigned.

**Table E-14 Usual Divergence Macroinstruction of 20bi**

Mnemonic	Operation	Remark
*CALL20 label20,Ri	Address of the following instruction→ RP, label20→PC	Ri: Temporary register (See Note1.)
*BRA20 label20,Ri	label20→PC	Ri: Temporary register (Note2)
*BEQ20 label20,Ri	if(Z==1) then label20→PC	Ri: Temporary register (Note3)
*BNE20 label20,Ri	↑ s/Z==0	↑
*BC20 label20,Ri	↑ s/C==1	↑
*BNC20 label20,Ri	↑ s/C==0	↑
*BN20 label20,Ri	↑ s/N==1	↑
*BP20 label20,Ri	↑ s/N==0	↑
*BV20 label20,Ri	↑ s/V==1	↑
*BNV20 label20,Ri	↑ s/V==0	↑
*BLT20 label20,Ri	↑ s/V xor N==1	↑
*BGE20 label20,Ri	↑ s/V xor N==0	↑
*BLE20 label20,Ri	↑ s/(V xor N) or Z==1	↑
*BGT20 label20,Ri	↑ s/(V xor N) or Z==0	↑
*BLS20 label20,Ri	↑ s/C or Z==1	↑
*BHI20 label20,Ri	↑ s/C or Z==0	↑

**Reference 1: CALL20**

(1) When label20-PC-2 is -0x800 to +0x7fe, the instruction is generated as follows.

```
CALL    label12
```

(2) When label20-PC-2 falls outside the range in (1) or contains an externally referenced symbol, the following instructions are generated:

```
LDI:20  #label20,Ri
CALL    @Ri
```

**Reference 2: BRA20**

(1) When label20-PC-2 is -0x100 to +0xfe, the instruction is generated as follows.

```
BRA     label9
```

(2) When label20-PC-2 falls outside the range in (1) or contains an externally referenced symbol, the following instructions are generated:

```
LDI:20  #label20,Ri
JMP     @Ri
```

**Reference 3: Bcc20**

(1) When label20-PC-2 is -0x100 to +0xfe, the instruction is generated as follows.

```
Bcc     label9
```

(2) When label20-PC-2 falls outside the range in (1) or contains an externally referenced symbol, the following instructions are generated:

```
Bxcc    false      xcc is a contradiction condition of cc.
LDI:20  #label20,Ri
JMP     @Ri
false:
```

**Table E-15 20bit Delay Divergence Macroinstruction**

Mnemonic	Operation	Remark
*CALL20:D label20,Ri	Address of the following instruction+2→RP, label20→PC	Ri: Temporary register (See Note1.)
*BRA20:D label20,Ri	label20→PC	Ri: Temporary register (See Note2.)
*BEQ20:D label20,Ri	if(Z==1) then label20→PC	Ri: Temporary register (See Note3.)
*BNE20:D label20,Ri	↑ s/Z==0	↑
*BC20:D label20,Ri	↑ s/C==1	↑
*BNC20:D label20,Ri	↑ s/C==0	↑
*BN20:D label20,Ri	↑ s/N==1	↑
*BP20:D label20,Ri	↑ s/N==0	↑
*BV20:D label20,Ri	↑ s/V==1	↑
*BNV20:D label20,Ri	↑ s/V==0	↑
*BLT20:D label20,Ri	↑ s/V xor N==1	↑
*BGE20:D label20,Ri	↑ s/V xor N==0	↑
*BLE20:D label20,Ri	↑ s/(V xor N) or Z==1	↑
*BGT20:D label20,Ri	↑ s/(V xor N) or Z==0	↑
*BLS20:D label20,Ri	↑ s/C or Z==1	↑
*BHI20:D label20,Ri	↑ s/C or Z==0	↑

**Reference 1: CALL20:D**

(1) When label20-PC-2 is -0x800 to +0x7fe, the instruction is generated as follows.

CALL:D label12

(2) When label20-PC-2 falls outside the range in (1) or contains an externally referenced symbol, the following instructions are generated:

LDI:20 #label20,Ri

CALL:D @Ri

**Reference 2: BRA20:D**

(1) When label20-PC-2 is -0x100 to +0xfe, the instruction is generated as follows.

BRA:D label9

(2) When label20-PC-2 falls outside the range in (1) or contains an externally referenced symbol, the following instructions are generated:

LDI:20 #label20,Ri

JMP:D @Ri

**Reference 3: Bcc20:D**

(1) When label20-PC-2 is -0x100 to +0xfe, the instruction is generated as follows.

Bcc:D label9

(2) When label20-PC-2 falls outside the range in (1) or contains an externally referenced symbol, the following instructions are generated:

Bxcc false xcc is a contradiction condition of cc.

LDI:20 #label20,Ri

JMP:D @Ri

false:

**Table E-16 Usual Divergence Macroinstruction of 32bit**

Mnemonic	Operation	Remark
*CALL32 label32,Ri	Address of the following instruction—RP, label32→PC	Ri: Temporary register (See Note1.)
*BRA32 label32,Ri	label32→PC	Ri: Temporary register (See Note 2.)
*BEQ32 label32,Ri	if(Z==1) then label32→PC	Ri: Temporary register (See Note 3.)
*BNE32 label32,Ri	↑ s/Z==0	↑
*BC32 label32,Ri	↑ s/C==1	↑
*BNC32 label32,Ri	↑ s/C==0	↑
*BN32 label32,Ri	↑ s/N==1	↑
*BP32 label32,Ri	↑ s/N==0	↑
*BV32 label32,Ri	↑ s/V==1	↑
*BNV32 label32,Ri	↑ s/V==0	↑
*BLT32 label32,Ri	↑ s/V xor N==1	↑
*BGE32 label32,Ri	↑ s/V xor N==0	↑
*BLE32 label32,Ri	↑ s/(V xor N) or Z==1	↑
*BGT32 label32,Ri	↑ s/(V xor N) or Z==0	↑
*BLS32 label32,Ri	↑ s/C or Z==1	↑
*BHI32 label32,Ri	↑ s/C or Z==0	↑

**Reference 1: CALL32**

(1) When label32-PC-2 is -0x800 to +0x7fe, the instruction is generated as follows.

CALL label12

(2) When label32-PC-2 falls outside the range in (1) or contains an externally referenced symbol, the following instructions are generated:

LDI:32 #label32,Ri

CALL @Ri

**Reference 2: BRA32**

(1) When label32-PC-2 is -0x100 to +0xfe, the instruction is generated as follows.

BRA label9

(2) When label32-PC-2 falls outside the range in (1) or contains an externally referenced symbol, the following instructions are generated:

LDI:32 #label32,Ri

JMP @Ri

**Reference 3: Bcc32**

(1) When label32-PC-2 is -0x100 to +0xfe, the instruction is generated as follows.

Bcc label9

(2) When label32-PC-2 falls outside the range in (1) or contains an externally referenced symbol, the following instructions are generated:

Bxcc false      xcc is a contradiction condition of cc.

LDI:32 #label32,Ri

JMP @Ri

false:



**Table E-17 32bit Delay Divergence Macroinstruction**

Mnemonic	Operation	Remark
*CALL32D label32,Ri	Address of the following instruction+2- <del>RP</del> , label32→PC	Ri: Temporary register (See Note 1.)
*BRA32:D label32,Ri	label32→PC	Ri: Temporary register (See Note 2.)
*BEQ32:D label32,Ri	if(Z==1) then label32→PC	Ri: Temporary register (See Note 3.)
*BNE32:D label32,Ri	↑ s/Z==0	↑
*BC32:D label32,Ri	↑ s/C==1	↑
*BNC32:D label32,Ri	↑ s/C==0	↑
*BN32:D label32,Ri	↑ s/N==1	↑
*BP32:D label32,Ri	↑ s/N==0	↑
*BV32:D label32,Ri	↑ s/V==1	↑
*BNV32:D label32,Ri	↑ s/V==0	↑
*BLT32:D label32,Ri	↑ s/V xor N==1	↑
*BGE32:D label32,Ri	↑ s/V xor N==0	↑
*BLE32:D label32,Ri	↑ s/(V xor N) or Z==1	↑
*BGT32:D label32,Ri	↑ s/(V xor N) or Z==0	↑
*BLS32:D label32,Ri	↑ s/C or Z==1	↑
*BHI32:D label32,Ri	↑ s/C or Z==0	↑

**Reference 1: CALL32:D**

(1) When label32-PC-2 is -0x800 to +0x7fe, the instruction is generated as follows.

CALL:D label12

(2) When label32-PC-2 falls outside the range in (1) or contains an externally referenced symbol, the following instructions are generated:

LDI:32 #label32,Ri

CALL:D @Ri

**Reference 2: BRA32:D**

(1) When label32-PC-2 is -0x100 to +0xfe, the instruction is generated as follows.

BRA:D label9

(2) When label32-PC-2 falls outside the range in (1) or contains an externally referenced symbol, the following instructions are generated:

LDI:32 #label32,Ri

JMP:D @Ri

**Reference 3: Bcc32:D**

(1) When label32-PC-2 is -0x100 to +0xfe, the instruction is generated as follows.

Bcc:D label9

(2) When label32-PC-2 falls outside the range in (1) or contains an externally referenced symbol, the following instructions are generated:

Bxcc false xcc is a contradiction condition of cc.

LDI:32 #label32,Ri

JMP:D @Ri

false:

**Table E-18 Direct addressing**

Mnemonic	Type	OP	CYCLE	NZVC	Operation	Remark
DMOV @dir10, R13	D	08	b	----	(dir10)→R13	word
DMOV R13, @dir10	D	18	a	----	R13 →(dir10)	word
DMOV @dir10, @R13+	D	0C	2a	----	(dir10)→(R13),R13+=4	word
DMOV @R13+, @dir10	D	1C	2a	----	(R13)→(dir10),R13+=4	word
DMOV @dir10, @-R15	D	0B	2a	----	R15-=4,(R15)→(dir10)	word
DMOV @R15+, @dir10	D	1B	2a	----	(R15)→(dir10),R15+=4	word
DMOVH @dir9, R13	D	09	b	----	(dir9)→R13	half-word
DMOVH R13, @dir9	D	19	a	----	R13 →(dir9)	half-word
DMOVH @dir9, @R13+	D	0D	2a	----	(dir9)→(R13),R13+=2	half-word
DMOVH @R13+, @dir9	D	1D	2a	----	(R13)→(dir9),R13+=2	half-word
DMOV B @dir8, R13	D	0A	b	----	(dir8)→R13	6 KB
DMOV B R13, @dir8	D	1A	a	----	R13 →(dir8)	6 KB
DMOV B @dir8, @R13+	D	0E	2a	----	(dir8)→(R13),R13++	6 KB
DMOV B @R13+, @dir8	D	1E	2a	----	(R13)→(dir8),R13++	6 KB

Note: The assembler calculates in dir8, dir9, and the dir10 field as follows and the value is set.

Dir8 →dir, dir9/2 →dir, dir10/4 →dir, dir8, dir9, and dir10 are unsigned.

**Table E-19 Resource Instruction**

Mnemonic	Type	OP	CYCLE	NZVC	Operation	Remark
LDRES @Ri+, #u4	C	BC	a	----	(Ri)→Resource of u4 Ri+=4	u4: Channel number
STRES #u4, @Ri+	C	BD	a	----	Resource of u4 →(Ri) Ri+=4	u4: Channel number

Note: Not available to this mode as it has no channel-numbered resource.

**Table E-20 Coprocessor Control Instruction**

{CRi|CRj} := CR0|CR1|CR2|CR3|CR4|CR5|CR6|CR7|CR8|CR9|CR10|CR11|CR12|CR13|CR14|CR15|

u4: := channel specification

u8: := command specification

Mnemonic	Type	OP	CYCLE	NZVC	Operation	Remark
COPOP #u4, #u8, CRj, Cri	E	9F-C	2+a	----	Operation instruction	Error trap none
COPLD #u4, #u8, Rj, Cri	E	9F-D	1+2a	----	Rj →CRi	
COPST #u4, #u8, CRj, Ri	E	9F-E	1+2a	----	CRj →Ri	
COPSV #u4, #u8, CRj, Ri	E	9F-F	1+2a	----	CRj →Ri	

Note: Not available to this model as it contains no coprocessor.

## APPENDIX F Notes on Using

This appendix provides notes on using the MB91265A series.

### ■ Common Matter

#### ● Clock control block

For "L" input to  $\overline{\text{INIT}}$ , reserve the oscillation stabilization wait time.

#### ● Switching shared ports between functions

Switching between a port and a shared pin relies on the PFR (port function register). Note, however, that bus pins are switched depending on the external bus setting.

#### ● D-bus Memory

Do not set a code area in memory on the D-bus.

No instruction fetch applies to the D-bus. Applying an instruction fetch to the D-bus area causes wrong data to be interpreted as code, involving the risk of running out of control.

#### ● Low-power Consumption Mode

(1) To place the device in standby mode, use the synchronous standby mode (set with bit8 (SYNCS bit) of the time-base counter control register, TBCR) and be sure to use the following sequence:

```
/* Writing STCR */
ldi    #_STCR, R0            ; STCR register (0x0481)
ldi    #Val_of_Stby, r1       ; Val_of_Stby is the write data to STCR.
stb     r1, @r0               ; Writing to STCR

/* Writing STBR */
ldi     #_CTBR, r2            ; CTBR register (0x0483)
ldi     #0xA5, r1              ; Clear command (1)
stb     r1, @r2               ; Writing A5 to CTBR
ldi     #0xA5, r1              ; Clear command (2)
stb     r1, @r2               ; Writing A5 to CTBR

/* Clearing the time base counter in here */
ldub    @r0, r1                ; Reading STCR

/* Starting synchronous standby transition */
ldub    @r0, r1                ; Reading dummy STCR
nop                                           ; NOP × 5 for timing adjustment
nop
nop
nop
nop
```

(2) Please do not perform the following when the monitor debugger is used.

- Please do not set the break point to the above-mentioned instruction row.
- Moreover, please do not execute the step for the above-mentioned instruction row.

### ● Notes of PS register

As the PS register is processed speculatively for some instructions, the interrupt service routine may break or the PS flag display may be updated when the debugger is used if exceptions are handled as follows: The device is designed such that processing is performed again correctly after recovery from EIT in either case. Appropriate actions are therefore taken before and after EIT as specification.

- The following operations may occur when (a) user interrupt/NMI is received, (b) step execution is performed, (c) break occurs in a data event or emulator menu in an instruction immediately preceding DIV0U/DIV0S instruction.
  - (1) D0 and D1 flags precede and are renewed.
  - (2) EIT processing routine (user interruption, NMI or emulator) is executed.
  - (3) After returning from EIT, DIV0U/DIV0S instructions are executed and the D0 and D1 flags are updated to the same value as (1).
- When each ORCCR/STILM/MOV Ri and PS instruction is executed to permit interrupting with the user interruption and the NMI factor generated, the following operations are done.
  - (1) The PS register precedes and is updated.
  - (2) Execute an EIT processing routine (user interrupt or NMI).
  - (3) After returning from EIT, the above instructions are executed and the PS register is updated to the same value as (1).

### ● About watchdog timer function

The watchdog timer integrated in this model monitors the program to check that it delays a reset within a certain period of time and, if the program runs out of control and fails to delay the reset, resets the CPU in place. Once the watchdog timer is enabled, it keeps running until reset.

As an exception, the watchdog timer delays the reset automatically when a condition which stops program execution by the CPU develops. For those conditions which correspond to this exception, see "[Temporarily halting the watchdog timer (automatically postpone generation)]" of "● Watchdog timer" for "■ Time-base Counter" in the Section "3.11.8 Peripheral Circuit Functions in the Clock Controller".

## ■ Notes on Debugging

### ● Single stepping of RETI instructions

In an environment where interrupts frequently occur during single-step execution, only the relevant interrupt processing routines are executed repeatedly during single-step execution of the RETI instruction. As a result, a low program of the main routine and interrupt levels is not executed.

To avoid it, do not single-step RETI instructions

When the relevant interrupt routine no longer requires being debugged, disable the relevant interrupt and perform debugging.

### ● Operand break

Do not apply a data even break to access of the area containing the address of a system stack pointer.

### ● Execution of an unused area of Flash memory

If an unused area (data at "FFFF<sub>H</sub>") of Flash memory is executed accidentally, no break can be accepted. To prevent this, it is recommended to use the code event address mask feature of the debugger to break at instruction access to the unused area.

### ● Power-on debugging

When turning the power off during power-on debugging, satisfy all of the following three conditions:

(1) The time for the user power supply to fall from 0.9V<sub>CC</sub> to 0.5V<sub>CC</sub> is 25 μs or longer.

Note: When two power supplies are provided, V<sub>CC</sub> indicates the external I/O power supply voltage.

(2) CPU operating frequency must be higher than 1 MHz.

(3) The user program is running.

### ● Interrupt handler for NMI request (tool)

Add the following program to the interrupt handler to prevent the device from malfunctioning when the source flag is set accidentally with no ICE connected, for example, due to noise to the DSU pin, which is to be set only at the break request of the ICE. Note that the ICE can be used normally with this program added.

Added place

Next interrupt handler

Interrupt source	: NMI request (tool)
Interrupt number	: 13 (decimal), 0D (Hexadecimal)
Offset	: 3C8H
TBR default address	: 000FFFC8H

Added program

```

STM    (R0, R1)
LDI    #B00H,R0    ; B00H is the address of the DSUs break source register.
LDI    #0, R1
STB    R1,@R0      ; Clear the break source register.
LDM    (R0, R1)
RETI

```

# Index

## Numerics

16/8 Bits	
FR-CPU Programming Mode	
(Read/Write in 16/8 Bits) .....	504
16-bit Bus	
32-bit<->16-bit Bus Converter .....	26
16-bit Dead Timer Control Register	
16-bit Dead Timer Control Register, Lower Byte	
(DTCR1) .....	251
16-bit Dead Timer Control Register, Upper Byte	
(DTCR0) .....	248
16-bit Dead Timer Control Register, Upper Byte	
(DTCR2) .....	254
16-bit Dead Timer Register	
16-bit Dead Timer Register (TMRRH0 to	
TMRRH2, TMRRL0 to TMRRL2) .....	247
16-bit Free-run Timer	
16-bit Free-run Timer Interrupt .....	264
Block Diagram of 16-bit Free-run Timer .....	205
Example Program for the 16-bit Free-run Timer	
.....	303
Notes on Using the 16-bit Free-run Timers .....	301
16-bit Free-run Timer Register	
16-bit Free-run Timer Register .....	212
16-bit Input Capture	
16-bit Input Capture Operation .....	287
Block Diagram of the 16-bit Input Capture .....	207
Cautions for Use of 16-bit Input Capture .....	301
Free-run Timer Selection for the 16-bit Input Capture	
.....	286
Input Timing for 16-bit Input Capture .....	287
16-bit Input Capture Interrupt	
16-bit Input Capture Interrupt .....	265
16-bit Input Capture Register	
16-bit Input Capture Register .....	215
16-BIT MAC	
Block Diagram of 16-BIT MAC .....	436
16-bit Output Compare	
16-bit Output Compare Operation (Inversion	
Mode, MOD1x=0) .....	276
16-bit Output Compare Operation	
(Set/Reset Mode, MOD1x=1) .....	279
16-bit Output Compare Timing .....	280
Block Diagram of the 16-bit Output Compare	
.....	206
Example Program for the 16-bit Output Compare	
.....	304
Free-run Timer Selection for the 16-bit Output	
Compare .....	275
Notes on Using the 16-bit Output Compare .....	301
Operation of 16-bit Output Compare and Free-run	
Timer .....	281
16-bit Output Compare Interrupt	
16-bit Output Compare Interrupt .....	265
16-bit Output Compare Register	
16-bit Output Compare Register .....	214
16-bit Reload Register	
TMRLR Register (16-bit Reload Register) .....	149
16-bit Timer Register	
TMR Register (16-bit Timer Register) .....	149
32/16/8 Bits	
FR-CPU ROM Mode (Read only in 32/16/8 Bits)	
.....	503
32-bit Bus	
32-bit<->16-bit Bus Converter .....	26
8/10-bit A/D Converter	
Block Diagram of 8/10-bit A/D Converter .....	410
Block Diagram of 8/10-bit A/D Converter Pin	
.....	106
Block Diagram of 8/10-bit A/D Converter Pins	
.....	413
Functions of the 8/10-bit A/D Converter .....	408
Interrupt of 8/10-bit A/D Converter .....	425
Pins of 8/10-bit A/D Converter .....	412
Registers of 8/10-bit A/D Converter .....	414
8-bit PPG	
Block Diagram of the 8-bit PPG (ch.0, ch.2, ch.4, ch.6)	
.....	165
Block Diagram of the 8-bit PPG (ch.1, ch.5) .....	166
Block Diagram of the 8-bit PPG (ch.3, ch.7) .....	167

<b>A</b>	
A/D	
A/D Activation.....	299
A/D Compare Activation Enabled.....	299
A/D Compare Activation Mode .....	299
A/D Activation Compare	
Block Diagram of the A/D Activation Compare	
.....	209
A/D Activation Compare Register	
A/D Activation Compare Register.....	217
A/D Activation Via Free-run Timer	
A/D Activation Via Free-run Timer 0.....	274
A/D Channel Control Register	
A/D Channel Control Register	
(ADCH : ADCH1,ADCH2).....	415
A/D Control Status Registers	
A/D Control Status Registers	
(ADCS : ADCS1,ADCS2) .....	420
A/D Conversion Data Protection	
A/D Conversion Data Protection Function .....	430
A/D Converter	
Block Diagram of 8/10-bit A/D Converter .....	410
Block Diagram of 8/10-bit A/D Converter Pin	
.....	106
Block Diagram of 8/10-bit A/D Converter Pins	
.....	413
Functions of the 8/10-bit A/D Converter .....	408
Interrupt of 8/10-bit A/D Converter.....	425
Pins of 8/10-bit A/D Converter .....	412
Registers of 8/10-bit A/D Converter.....	414
A/D Data Register	
A/D Data Register (ADCD : ADCD10,ADCD11,	
ADCD20,ADCD21) .....	423
A/D Mode Setting Register	
A/D Mode Setting Register	
(ADMD : ADMD1,ADMD2) .....	417
A/D Trigger Control Register	
A/D Trigger Control Register (ADTRGC) .....	226
ADCD	
A/D Data Register (ADCD : ADCD10,ADCD11,	
ADCD20,ADCD21) .....	423
ADCH	
A/D Channel Control Register	
(ADCH : ADCH1,ADCH2).....	415
ADCOMP	
Compare Register 1,2 (ADCOMP1,ADCOMP2)	
.....	260
ADCOMPC	
Control Register 1,2 (ADCOMPC1,ADCOMPC2)	
.....	261
ADCS	
A/D Control Status Registers	
(ADCS : ADCS1,ADCS2) .....	420
Addressing Mode	
Addressing Mode.....	478
ADMD	
A/D Mode Setting Register	
(ADMD : ADMD1,ADMD2).....	417
ADTRGC	
A/D Trigger Control Register (ADTRGC).....	226
AF200	
System Configuration for AF200 Flash	
Microcontroller Programmer (Manufactured	
by Yokogawa Digital Computer	
Corporation) .....	521
AICR	
Analog Input Control Register	
(AICR : AICR1,AICR2) .....	424
Analog Input Control Registers	
(AICR : AICR1,AICR2) .....	107
Analog Input Control Register	
Analog Input Control Register	
(AICR : AICR1,AICR2) .....	424
Analog Input Control Registers	
Analog Input Control Registers	
(AICR : AICR1,AICR2) .....	107
Application Example	
Application Example.....	330
Assembler	
Assembler (fasm911) .....	543
Asynchronous	
Asynchronous (start-stop Synchronization) Mode	
.....	325
Automatic Algorithm Execution States	
Automatic Algorithm Execution States .....	492
<b>B</b>	
Backup	
Backup/Restore Processing.....	142
Basic Block Diagram	
Basic Block Diagram of the I/O Port.....	98
Basic Clock Dividing Frequency Set Register	
DIVR0:Basic Clock Dividing Frequency	
Set Register 0 .....	84
DIVR1:Basic Clock Dividing Frequency	
Set Register 1 .....	85
Basic Configuration	
Basic Configuration for Serial Programming	
.....	518
Basic Mode	
Basic Mode .....	401
Basic Programming	
Basic Programming Model .....	29
Baud Rate	
Calculation of Baud Rate .....	312

Bit Ordering	
Bit Ordering .....	36
Bit Search Module	
Block Diagram of Bit Search Module .....	138
Bit Search Module Registers	
Bit Search Module Registers .....	138
Block Diagram	
Basic Block Diagram of the I/O Port .....	98
Block Diagram .....	4, 127, 181, 404
Block Diagram of 16-bit Free-run Timer .....	205
Block Diagram of 8/10-bit A/D Converter .....	410
Block Diagram of 8/10-bit A/D Converter Pin .....	106
Block Diagram of 8/10-bit A/D Converter Pins .....	413
Block Diagram of Bit Search Module .....	138
Block Diagram of Clock Generation Control Unit .....	73
Block Diagram of Delayed Interrupt Module .....	136
Block Diagram of DMAC .....	458
Block Diagram of Gate Function .....	168
Block Diagram of the 16-bit Input Capture .....	207
Block Diagram of the 16-bit Output Compare .....	206
Block Diagram of the 8-bit PPG (ch.0,ch.2,ch.4,ch.6) .....	165
Block Diagram of the 8-bit PPG (ch.1,ch.5) .....	166
Block Diagram of the 8-bit PPG (ch.3,ch.7) .....	167
Block Diagram of the A/D Activation Compare .....	209
Block Diagram of the Free-run Timer Selector .....	210
Block Diagram of the Interrupt Controller .....	113
Block Diagram of the Multifunction Timer .....	204
Block Diagram of the Reload Timer .....	145
Block Diagram of the Waveform Generator .....	208
Block Diagram of Timing Generator .....	157
Block Diagram of UART .....	316
Block Diagram of U-TIMER .....	308
Block Diagram of 16-BIT MAC .....	436
Flash Memory Block Diagram .....	493
Block Size	
Block Size .....	477
Block Transfer	
Block Transfer .....	486
Branch Instruction	
JMP Instruction (Branch Instruction) .....	453
BRPER	
CAN Prescaler Extended Register (BRPER) .....	355
BTR	
CAN Bit Timing Register (BTR) .....	351
Burst Transfer	
Burst Transfer .....	487
Burst Two-cycle Transfer	
Burst Two-cycle Transfer .....	475
Bus Mode	
Bus Mode .....	56
Byte Ordering	
Byte Ordering .....	36
<b>C</b>	
C Compiler	
C Compiler (fcc911) .....	540
Calculation	
Calculation of Baud Rate .....	312
CAN	
CAN Clock Prescaler Setting .....	406
CAN Controller .....	335
CAN Controller	
CAN Controller .....	335
CAN_TX	
Software Control of Pin CAN_TX .....	402
Cancel a Hold Request	
Example of Using the Function to Generate a Request to Cancel a Hold Request (HRCR) .....	122
Cascade Mode	
Cascade Mode .....	312
Cautions	
Cautions for Use of 16-bit Input Capture .....	301
C-CAN	
Features of C-CAN .....	334
CCR	
CCR (Condition Code Register) .....	31
Changing Clock	
Procedure of Changing Clock .....	405
Channel Selection and Control	
Channel Selection and Control .....	484
Clear	
Compare Clear Buffer .....	269
Compare Clear Buffer Register (CPCLRBH0 to CPCLRBH2, CPCLRBL0 to CPCLRBL2) .....	218
Compare Clear Register (CPCLRHO to CPCLRHO2, CPCLRL0 to CPCLRL2) .....	219
Counter Clear .....	193
Timer Clear .....	268
Clearing Peripheral Interrupts	
Clearing Peripheral Interrupts by the DMA .....	482
CLKB	
CPU Clock (CLKB) .....	71
CLKP	
Clock in Surrounding (CLKP) .....	71
CLKR	
CLKR:Clock Source Control Register .....	82



CLKT	
External Bus Clock (CLKT) .....	71
Clock	
Clock in Surrounding (CLKP) .....	71
Clock Divider	
Clock Divider .....	72
Clock Generation	
Overview of Clock Generation Control .....	66
Clock Generation Control Unit	
Block Diagram of Clock Generation Control Unit .....	73
Clock Prescaler Register	
Clock Prescaler Register .....	341
Clock Selection	
Clock Selection of UART .....	324
Clock Source Control Register	
CLKR:Clock Source Control Register .....	82
Clock Synchronous Mode	
Clock Synchronous Mode .....	326
Combination	
Combination of Silent Mode and Loop Back Mode .....	401
Command Sequence	
Command Sequence .....	505
Common	
Common Matter .....	560
Compare Clear Buffer	
Compare Clear Buffer .....	269
Compare Clear Buffer Register	
Compare Clear Buffer Register (CPCLRBH0 to CPCLRBH2, CPCLRBL0 to CPCLRBL2) .....	218
Compare Clear Register	
Compare Clear Register (CPCLRH0 to CPCLRH2, CPCLRL0 to CPCLRL2) .....	219
Compare Control Register	
Compare Control Register, Lower Byte (OCSL0, OCSL2, OCSL4) .....	233
Compare Control Register, Upper Byte (OCSH1, OCSH3, OCSH5) .....	230
Compare Mode Control Register	
Compare Mode Control Register (OCMOD) .....	236
Compare Register	
Compare Register 1,2 (ADCOMP1, ADCOMP2) .....	260
Condition Code Register	
CCR (Condition Code Register) .....	31
Configuration	
Configuration of Message Object .....	369
Connection Example	
Connection Example for Serial Programming .....	520
Continuous Conversion Mode	
Operation of Continuous Conversion Mode .....	428
Control	
Control of the Number of Transfers .....	479
Control Register	
Control Register 1,2 (ADCOMP1, ADCOMP2) .....	261
Control Status Register	
Control Status Register (TMCSR : TMCSR0 to TMCSR2) .....	146
Converter	
32-bit<->16-bit Bus Converter .....	26
Harvard<->Princeton Bus Converter .....	26
Count Clock	
Count Clock Selection .....	190
Counter	
CAN Error Counter (ERRCNT) .....	350
Counter Clear .....	193
Counter Operation	
Counter Operation States .....	153
CPCLRBH	
Compare Clear Buffer Register (CPCLRBH0 to CPCLRBH2, CPCLRBL0 to CPCLRBL2) .....	218
CPCLRBL	
Compare Clear Buffer Register (CPCLRBH0 to CPCLRBH2, CPCLRBL0 to CPCLRBL2) .....	218
CPCLRH	
Compare Clear Register (CPCLRH0 to CPCLRH2, CPCLRL0 to CPCLRL2) .....	219
CPCLRL	
Compare Clear Register (CPCLRH0 to CPCLRH2, CPCLRL0 to CPCLRL2) .....	219
CPU	
CPU .....	26
CPU Control .....	480
CPU Interface .....	335
Pin States in Each CPU State .....	538
CPU Clock	
CPU Clock (CLKB) .....	71
CTBR	
CTBR:Time-base Counter Clear Register .....	81
CTRLR	
CAN Control Register (CTRLR) .....	344

## D

### Data Direction Registers

Data Direction Registers (DDR : DDR0 to DDR5,DDRG) .....	101
---	-----

### Data Frame

Data Frame Reception .....	390
----------------------------	-----

### Data Transmission/Reception

Message RAM and Data Transmission/Reception .....	386
--	-----

### DDR

Data Direction Registers (DDR : DDR0 to DDR5,DDRG) .....	101
---	-----

### Dead Time Timer Mode

Operation during Dead Time Timer Mode .....	293
---	-----

### Debugger

Debugger (sim911,eml911,mon911) .....	544
---------------------------------------	-----

### Debugging

Notes on Debugging.....	562
-------------------------	-----

### Defined Instruction

Defined Instruction .....	438
---------------------------	-----

### Delay Slot

Operation with Delay Slot.....	39
Operation without Delay Slot .....	41

### Delayed Interrupt

Block Diagram of Delayed Interrupt Module .....	136
--	-----

### Delayed Interrupt Module Registers

Delayed Interrupt Module Registers.....	136
---	-----

### Delayed Write Feature

Delayed Write Feature.....	446
----------------------------	-----

### Detailed Explanation

Detailed Explanation of Register .....	127
--	-----

### Details

Details of Pulse Width Count Operation.....	193
---	-----

### Determining

Determining the Priority .....	116
--------------------------------	-----

### Device

State of Device and Each Transition .....	90
---	----

### Direct Addressing

Direct Addressing Area .....	22, 38
------------------------------	--------

### Dividing Frequency Control Register

Ratio of Dividing Frequency Control Register (PDIVR : PDIVR0) .....	188
--	-----

### DIVR0

DIVR0:Basic Clock Dividing Frequency Set Register 0 .....	84
--	----

### DIVR1

DIVR1:Basic Clock Dividing Frequency Set Register 1 .....	85
--	----

### DMA

Clearing Peripheral Interrupts by the DMA .....	482
DMA Transfer during the Sleep Mode .....	484

### DMA Transfer

DMA Transfer during the Sleep Mode .....	484
--	-----

### DMAC

Block Diagram of DMAC .....	458
DMAC Interrupt Control .....	483
DMAC-ch.0,ch.1,ch.2,ch.3,ch.4 Control Status Registers A [DMACA : DMACA0 to DMACA4] .....	459
DMAC-ch.0,ch.1,ch.2,ch.3,ch.4 Control Status Registers B [DMACB : DMACB0 to DMACB4] .....	463
DMAC-ch.0,ch.1,ch.2,ch.3,ch.4 Overall DMAC Control Register[DMACR] .....	470
DMAC-ch.0,ch.1,ch.2,ch.3,ch.4 Transfer Source/ Transfer Destination address Setting Registers [DMASA/DMADA : DMASA0 to DMASA4/DMADA0 to DMADA4] .....	468

### DMACA

DMAC-ch.0,ch.1,ch.2,ch.3,ch.4 Control Status Registers A [DMACA : DMACA0 to DMACA4] .....	459
--	-----

### DMACB

DMAC-ch.0,ch.1,ch.2,ch.3,ch.4 Control Status Registers B [DMACB : DMACB0 to DMACB4] .....	463
--	-----

### DMACR

DMAC-ch.0,ch.1,ch.2,ch.3,ch.4 Overall DMAC Control Register[DMACR] .....	470
---	-----

### DMASA/DMADA

DMAC-ch.0,ch.1,ch.2,ch.3,ch.4 Transfer Source/ Transfer Destination address Setting Registers [DMASA/DMADA :DMASA0 toDMASA4/ DMADA0 toDMADA4].....	468
--	-----

### DSP Control/Status Register

DSP Control/Status Register (DSP-CSR) .....	439
---	-----

### DSP Delay Register

DSP Delay Register (DSP-LY) .....	443
-----------------------------------	-----

### DSP Program Counter

DSP Program Counter (DSP-PC) .....	442
------------------------------------	-----

### DSP Variable Monitor Register

DSP Variable Monitor Register (DSP-OT0 to DSP-OT7) .....	444
---	-----

### DSP-CSR

DSP Control/Status Register (DSP-CSR) .....	439
---	-----

### DSP-LY

DSP Delay Register (DSP-LY) .....	443
-----------------------------------	-----

### DSP-OT

DSP Variable Monitor Register (DSP-OT0 to DSP-OT7) .....	444
---	-----

DSP-PC	
DSP Program Counter (DSP-PC) .....	442
DTCR	
16-bit Dead Timer Control Register, Lower Byte (DTCR1) .....	251
16-bit Dead Timer Control Register, Upper Byte (DTCR0) .....	248
16-bit Dead Timer Control Register, Upper Byte (DTCR2) .....	254
DTTI	
DTTI Bit Operation of Waveform Control Register 2 (SIGCR2) .....	298
DTTI Pin Input Operation .....	297
DTTI Pin Interrupt .....	298
DTTI Pin Noise Cancellation Feature .....	298
Dual-operation	
Features of Dual-operation Flash Memory .....	515
<b>E</b>	
EIT	
EIT Factor .....	42
EIT Operation .....	52
EIT Vector Table .....	46
Feature of EIT .....	42
Multiple EIT Processing .....	50
Priority Level of EIT Factor .....	50
Return from EIT .....	42
eml911	
Debugger (sim911, eml911, mon911) .....	544
End	
Operation End/Stopping .....	482
ERRCNT	
CAN Error Counter (ERRCNT) .....	350
Error	
Stopping Due to an Error .....	483
Example	
Example of Using the Function to Generate a Request to Cancel a Hold Request (HRCR) .....	122
Examples of Using the Hardware Sequence Flag .....	513
Example Program	
Example Program for the 16-bit Free-run Timer .....	303
Example Program for the 16-bit Output Compare .....	304
Explanation	
Explanation of Operation .....	129
External Bus Clock	
External Bus Clock (CLKT) .....	71
External Count Clock	
Selected External Count Clock .....	273

<b>F</b>	
fasm911	
Assembler (fasm911) .....	543
fcc911	
C Compiler (fcc911) .....	540
Feature	
Feature of EIT .....	42
Features .....	2, 434
Features of C-CAN .....	334
Features of Dual-operation Flash Memory .....	515
Features of Internal Architecture .....	24
Features of UART .....	314
FIFO Buffer	
Configuration of FIFO Buffer .....	393
Message Reception by FIFO Buffer .....	393
Read from FIFO Buffer .....	394
Flag	
Examples of Using the Hardware Sequence Flag .....	513
Hardware Sequence Flag .....	510
Interrupt Generation and Flag Setting Timings .....	327
Flash Memory	
Features of Dual-operation Flash Memory .....	515
Flash Memory Block Diagram .....	493
Flash Memory Outline .....	492
Flash Memory Registers .....	498
Flash Memory Status Register (FLCR) .....	499
Flash Memory Wait Register (FLWC) .....	501
Sector Configuration for Flash Memory .....	494
Flash Memory Registers	
Flash Memory Registers .....	498
Flash Memory Status Register	
Flash Memory Status Register (FLCR) .....	499
Flash Memory Wait Register	
Flash Memory Wait Register (FLWC) .....	501
Flash Microcontroller Programmer	
System Configuration for AF200 Flash Microcontroller Programmer (Manufactured by Yokogawa Digital Computer Corporation) .....	521
FLCR	
Flash Memory Status Register (FLCR) .....	499
flnk911	
Linker (flnk911) .....	544
FLWC	
Flash Memory Wait Register (FLWC) .....	501
FPT-64P-M23	
Package Dimension of FPT-64P-M23 .....	6
FR Family	
Instruction List of FR Family .....	545
FR-CPU Programming Mode	
FR-CPU Programming Mode (Read/Write in 16/8 Bits) .....	504

FR-CPU ROM Mode	
FR-CPU ROM Mode (Read only in 32/16/8 Bits)	503
Free-run Timer	
16-bit Free-run Timer Interrupt	264
16-bit Free-run Timer Register	212
A/D Activation Via Free-run Timer 0	274
Block Diagram of 16-bit Free-run Timer	205
Block Diagram of the Free-run Timer Selector	210
Example Program for the 16-bit Free-run Timer	303
Free-run Timer Selection for the 16-bit Input Capture	286
Free-run Timer Selection for the 16-bit Output Compare	275
Free-run Timer Selector Register	217
Free-run Timer Selector Register (FSR0 to FSR2)	262
Notes on Using the 16-bit Free-run Timers	301
Operation of 16-bit Output Compare and Free-run Timer	281
Free-run Timer Selection	
Free-run Timer Selection for the 16-bit Output Compare	275
Free-run Timer Selector	
Block Diagram of the Free-run Timer Selector	210
Free-run Timer Selector Register	
Free-run Timer Selector Register	217
Free-run Timer Selector Register (FSR0 to FSR2)	262
FSR	
Free-run Timer Selector Register (FSR0 to FSR2)	262
Function	
Function of Message Object	369
Function of Registers	344, 347, 350, 351, 352, 353, 355, 357, 359, 368, 377, 379, 381, 383
Functions of the 8/10-bit A/D Converter	408
Functions of the PPG	162
Functions of the PWC	180
Function of Registers	384
<b>G</b>	
GATE	
Output Status of RTO0 to RTO5 and GATE	288
Gate Function	
Block Diagram of Gate Function	168
GATE Function Control Register	
GATE Function Control Register (GATEC)	173
GATEC	
GATE Function Control Register (GATEC)	173
General-purpose Register	
General-purpose Register	30
Generate a Request	
Example of Using the Function to Generate a Request to Cancel a Hold Request (HRCR)	122
<b>H</b>	
Handling Device	
Precautions on Handling Device	18
Hardware	
Hardware Configuration	456
Hardware Configuration	
Hardware Configuration of the Interrupt Controller	110
Hardware Sequence Flag	
Examples of Using the Hardware Sequence Flag	513
Hardware Sequence Flag	510
Harvard Bus	
Harvard<->Princeton Bus Converter	26
Hold Request	
Example of Using the Function to Generate a Request to Cancel a Hold Request (HRCR)	122
Hold Request Cancel Level Register	
Hold Request Cancel Level Register (HRCL)	115
Hold Request Cancel Request	
Hold Request Cancel Request	120
HRCL	
Hold Request Cancel Level Register (HRCL)	115
HRCR	
Example of Using the Function to Generate a Request to Cancel a Hold Request (HRCR)	122
<b>I</b>	
I Flag	
I Flag	44
I/O	
Basic Block Diagram of the I/O Port	98
I/O Circuit Type	13
I/O Map	524
I/O Port Mode	99
I/O Circuit	
I/O Circuit Type	13
I/O Map	
I/O Map	524



Interrupt Stack .....	46	List of Message Interface Register.....	338
NMI (Non Maskable Interrupt).....	120	List of Overall Control Register .....	337
Precautions when Returning from STOP State		List of Pin Functions .....	7
Using External Interrupt .....	132	Loop Back Mode	
Timer Interrupt .....	270	Combination of Silent Mode and Loop Back Mode	
Waveform Generator Interrupts .....	266	.....	401
Interrupt Control Register		Loop Back Mode .....	400
ICR (Interrupt Control Register).....	44	Low-power Consumption Mode	
Interrupt Control Register (ICR).....	114	Low-power Consumption Mode .....	93
Interrupt Controller		LQFP-64	
Block Diagram of the Interrupt Controller .....	113	LQFP-64 (MB91F267A/MB91F267NA/MB91267A/	
Hardware Configuration of the Interrupt Controller		MB91267NA) .....	5
.....	110		
Interrupt Controller Registers		<b>M</b>	
Interrupt Controller Registers .....	111	MAC	
Interrupt Level		Block Diagram of 16-BIT MAC.....	436
Interrupt Level .....	43	MAC Instruction .....	449
Interrupt Vector		Main Operation	
Interrupt Vector .....	535	Main Operation .....	472
Interruption		Major Functions	
Level Mask to Interruption and NMI.....	44	Major Functions .....	110
INTPND		MB91267A	
CAN Interrupt Pending Registers		LQFP-64 (MB91F267A/MB91F267NA/MB91267A/	
(INTPND1, INTPND2) .....	380	MB91267NA) .....	5
INTR		MB91F267A/MB91F267NA/MB91267A/	
CAN Interrupt Register (INTR).....	352	MB91267NA Memory Map.....	23
Inversion Mode		MB91267NA	
16-bit Output Compare Operation		LQFP-64 (MB91F267A/MB91F267NA/MB91267A/	
(Inversion Mode,MOD1x=0).....	276	MB91267NA) .....	5
IPCPH		MB91F267A/MB91F267NA/MB91267A/	
Input Capture Data Register		MB91267NA Memory Map.....	23
(IPCPH0 to IPCPH3,IPCPL0 to IPCPL3)		MB91F267A	
.....	238	LQFP-64 (MB91F267A/MB91F267NA/MB91267A/	
IPCPL		MB91267NA) .....	5
Input Capture Data Register		MB91F267A/MB91F267NA/MB91267A/	
(IPCPH0 to IPCPH3,IPCPL0 to IPCPL3)		MB91267NA Memory Map.....	23
.....	238	MB91F267NA	
		LQFP-64 (MB91F267A/MB91F267NA/MB91267A/	
		MB91267NA) .....	5
		MB91F267A/MB91F267NA/MB91267A/	
		MB91267NAA Memory Map .....	23
<b>J</b>		Memory Map	
JMP		MB91F267A/MB91F267NA/MB91267A/	
JMP Instruction (Branch Instruction) .....	453	MB91267NA Memory Map.....	23
		Memory Map .....	38
<b>K</b>		Message	
Kinds of Data		Message Reception by FIFO Buffer.....	393
Kinds of Data .....	479	Message Handler	
		Message Handler .....	335
<b>L</b>		Message Handler Register	
Level Mask		List of Message Handler Register.....	340
Level Mask to Interruption and NMI.....	44	Message Handler Register .....	342, 375
Linker		Message Interface Register	
Linker (flnk911) .....	544	List of Message Interface Register.....	338
List			
List of Message Handler Register .....	340		

Message Interface Register.....	342
<b>Message Object</b>	
Configuration of Message Object.....	369
Function of Message Object .....	369
Message Object.....	386
Setting of Reception Message Object .....	391
Setting of Transmission Message Object .....	388
Updating a Transmission Message Object.....	389
<b>Message RAM</b>	
Message RAM .....	335
Message RAM and Data Transmission/Reception .....	386
<b>Message Transmission</b>	
Message Transmission .....	388
<b>MOD</b>	
16-bit Output Compare Operation (Inversion Mode,MOD1x=0) .....	276
16-bit Output Compare Operation (Set/Reset Mode,MOD1x=1) .....	279
<b>Mode Setting</b>	
Mode Setting .....	56
<b>mon911</b>	
Debugger (sim911,eml911,mon911) .....	544
<b>MSGVAL</b>	
CAN Message Enable Registers (MSGVAL1, MSGVAL2).....	382
<b>Multifunction Timer</b>	
Block Diagram of the Multifunction Timer .....	204
Operation of the Multifunction Timer.....	267
Pins of the Multifunction Timer .....	211
Structure of the Multifunction Timer .....	202
<b>Multiple EIT Processing</b>	
Multiple EIT Processing .....	50
<b>Multiplication and Division Register</b>	
Multiplication and Division Register (Multiply & Divide Register).....	35
<b>Multiply &amp; Divide Register</b>	
Multiplication and Division Register (Multiply & Divide Register).....	35
<b>N</b>	
<b>NEWDT</b>	
CAN Data Renewal Registers (NEWDT1, NEWDT2) .....	378
<b>NMI</b>	
Level Mask to Interruption and NMI .....	44
NMI (Non Maskable Interrupt) .....	120
<b>Non Maskable Interrupt</b>	
NMI (Non Maskable Interrupt) .....	120
<b>Normal Reset</b>	
Normal Reset Operation.....	65
<b>Note</b>	
Note .....	55
Notes .....	198

Notes on Debugging.....	562
Notes on Setting Registers .....	459
Notes on Using the 16-bit Free-run Timers .....	301
Notes on Using the 16-bit Output Compare .....	301
Notes on Using the Waveform Generator .....	302
Other Notes .....	521

<b>Number</b>	
Number of Transfers and Ending Transfer .....	473

## O

### OCCPBH

Output Compare Buffer Register (OCCPBH0 to OCCPBH5, OCCPBL0 to OCCPBL5) .....	228
---	-----

### OCCPBL

Output Compare Buffer Register (OCCPBH0 to OCCPBH5, OCCPBL0 to OCCPBL5) .....	228
---	-----

### OCCPH

Output Compare Register (OCCPH0 to OCCPH5,OCCPL0 to OCCPL5) .....	229
--	-----

### OCCPL

Output Compare Register (OCCPH0 to OCCPH5,OCCPL0 to OCCPL5) .....	229
--	-----

### OCMOD

Compare Mode Control Register (OCMOD) .....	236
--	-----

### OCSH

Compare Control Register,Upper Byte (OCSH1,OCSH3,OCSH5) .....	230
--	-----

### OCSL

Compare Control Register,Lower Byte (OCSL0,OCSL2,OCSL4) .....	233
--	-----

### Operating Mode

Operating Mode.....	56, 324, 445
---------------------	--------------

### Operation

16-bit Input Capture Operation .....	287
16-bit Output Compare Operation (Inversion Mode,MOD1x=0) .....	276
16-bit Output Compare Operation (Set/Reset Mode,MOD1x=1) .....	279
Details of Pulse Width Count Operation .....	193
DTTI Bit Operation of Waveform Control Register 2 (SIGCR2).....	298
DTTI Pin Input Operation .....	297
EIT Operation.....	52
Explanation of Operation .....	129
Instruction Operation.....	445
Internal Clock Operation .....	150
Main Operation.....	472
Operation during Dead Time Timer Mode.....	293
Operation End/Stopping .....	482
Operation Explanation.....	137, 140, 174

Operation Function .....	446	Output Compare Register	
Operation of 16-bit Output Compare and Free-run Timer .....	281	Output Compare Register (OCCPH0 to OCCPH5,OCCPL0 to OCCPL5) .....	229
Operation of Continuous Conversion Mode .....	428	Output Inversion Register	
Operation of Data in Two-cycle Transfer Mode .....	488	Output Inversion Register (REVC) .....	172
Operation of Pause-conversion Mode .....	429	Output Pin Function	
Operation of Single-shot Conversion Mode .....	426	Output Pin Function .....	152
Operation of the Multifunction Timer .....	267	Output Status	
Operation of Timer Mode .....	292	Output Status of RTO0 to RTO5 and GATE .....	288
Operation with Delay Slot .....	39	Overall Control Register	
Operation without Delay Slot .....	41	List of Overall Control Register .....	337
PLL Enabling Operation .....	68	Overall Control Register .....	342, 343
Reload Operation .....	477	Overview	
Start Operation .....	481	Operation Overview of Timing Generator .....	159
Transfer of Operation Results .....	447	Overview of Clock Generation Control .....	66
Wait Time after Enabling PLL Operation .....	69	Overview of Instructions .....	27
Overview of Reset Operation .....		Overview of Reset Operation .....	58
Operation Initialization Reset		<b>P</b>	
Operation Initialization Reset (RST) .....	59	Package Dimension	
Operation Initialization Reset (RST) Release Sequence .....	62	Package Dimension of FPT-64P-M23 .....	6
Operation Mode		Pause-conversion Mode	
Select Operation Mode .....	191	Operation of Pause-conversion Mode .....	429
Operation Overview		PC	
Operation Overview of Timing Generator .....	159	PC (Program Counter) .....	34
Oscillation Clock Frequency		PCR	
Oscillation Clock Frequency .....	521	Pull-up Control Registers (PCR : PCR0 to PCR4,PCRG) .....	102
Oscillation Stabilization Wait		PDIVR	
Triggers for the Oscillation Stabilization Wait .....	63	Ratio of Dividing Frequency Control Register (PDIVR : PDIVR0) .....	188
Oscillation Stabilization Wait Time		PDR	
Select Oscillation Stabilization Wait Time .....	64	Port Data Registers (PDR : PDR0 to PDR5,PDRG) .....	100
Other Notes		PFR	
Other Notes .....	521	Port Function Registers (PFR: PFR0,PFR1,PTFR0) .....	103
Output Compare		PICSH	
16-bit Output Compare Interrupt .....	265	PPG Output Control Register Upper Byte (PICSH01) .....	243
16-bit Output Compare Operation (Inversion Mode,MOD1x=0) .....	276	PICSL	
16-bit Output Compare Operation (Set/Reset Mode,MOD1x=1) .....	279	Input Capture State Control Register (ch.0,ch.1),Lower Byte (PICSL01) .....	245
16-bit Output Compare Timing .....	280	Pin Functions	
Example Program for the 16-bit Output Compare .....	304	List of Pin Functions .....	7
Free-run Timer Selection for the 16-bit Output Compare .....	275	Pin States	
Notes on Using the 16-bit Output Compare .....	301	Pin States in Each CPU State .....	538
Operation of 16-bit Output Compare and Free-run Timer .....	281	Pins	
Output Compare Buffer Register		Pins of 8/10-bit A/D Converter .....	412
Output Compare Buffer Register (OCCPBH0 to OCCPBH5, OCCPBL0 to OCCPBL5) .....	228	Pins of the Multifunction Timer .....	211



Pins Used for Fujitsu-standard Serial On-board Programming .....	519	PRL/PRH Register	
PLL		PRL/PRH Register (Reload Register : PRL0 to PRL7/PRH0 to PRH7) .....	171
PLL Enabling Operation .....	68	Procedure	
PLL Multiplication Rate .....	68	Procedure of Changing Clock.....	405
Wait Time after Changing the PLL Multiplier Ratio .....	69	Process	
Wait Time after Enabling PLL Operation .....	69	Process of Reception Message.....	392
Port Data Registers		Program Counter	
Port Data Registers (PDR : PDR0 to PDR5,PDRG) .....	100	PC (Program Counter) .....	34
Port Function Registers		Program Status	
Port Function Registers (PFR: PFR0,PFR1,PTFR0) .....	103	PS (Program Status) .....	30
Power Supply		Programming	
Wait Time after Power Supply is Turned on.....	69	Programming with a ROM Programmer .....	493
PPG		PS	
Block Diagram of the 8-bit PPG (ch.0,ch.2,ch.4,ch.6) .....	165	PS (Program Status) .....	30
Block Diagram of the 8-bit PPG (ch.1,ch.5) .....	166	PTFR	
Block Diagram of the 8-bit PPG (ch.3,ch.7) .....	167	Port Function Registers (PFR: PFR0,PFR1,PTFR0) .....	103
Functions of the PPG .....	162	Pull-up Control Registers	
PPG Start Register (TRG) .....	172	Pull-up Control Registers (PCR : PCR0 to PCR4,PCRG) .....	102
PPG0 Output Control.....	290	Pulse Width	
PPG0 Output Via Gate Trigger .....	290	Pulse Width Measurement Function.....	189
Registers of the PPG Timer .....	163	Pulse Width Count	
PPG Output Control Register		Details of Pulse Width Count Operation.....	193
PPG Output Control Register Upper Byte (PICSH01) .....	243	Starting and Stopping of the Pulse Width Count .....	192
PPG Start Register		Pulse Width Counter	
PPG Start Register (TRG) .....	172	Registers of the Pulse Width Counter.....	180
PPGCn Register		PWC	
PPGCn Register (PPGn Operation Mode Control Register) n=0,1,2,3,4,5,6,7 .....	169	Functions of the PWC .....	180
PPGn Operation Mode Control Register		PWC Control	
PPGCn Register (PPGn Operation Mode Control Register) n=0,1,2,3,4,5,6,7 .....	169	PWC Control/Status Register (PWCSR : PWCSR0) .....	182
Precautions		PWC Data Buffer Register	
Precautions on Handling Device .....	18	PWC Data Buffer Register (PWCR : PWCR0) .....	187
Precautions when Using .....	329	PWCR	
Prescaler Register		PWC Data Buffer Register (PWCR : PWCR0) .....	187
Prescaler Register.....	342	PWCSR	
Prime Function		PWC Control/Status Register (PWCSR : PWCSR0) .....	182
Prime Function.....	456	R	
Princeton Bus		RAM	
Harvard<->Princeton Bus Converter .....	26	Message RAM and Data Transmission/Reception .....	386
Priority		Ratio	
Determining the Priority .....	116	Ratio of Dividing Frequency Control Register (PDIVR : PDIVR0).....	188
Priority Level		Reception	
Priority Level of EIT Factor .....	50	Data Frame Reception .....	390
Reception Priority Level .....	390		
Transmission Priority Level .....	388		

Message RAM and Data Transmission/Reception .....	386	Compare Clear Buffer Register (CPCLRBH0 to CPCLRBH2, CPCLRBL0 to CPCLRBL2) .....	218
Message Reception by FIFO Buffer .....	393	Compare Clear Register (CPCLRH0 to CPCLRH2, CPCLRL0 to CPCLRL2) .....	219
Process of Reception Message .....	392	Compare Control Register, Lower Byte (OCSL0, OCSL2, OCSL4) .....	233
Reception Filter of Reception Message .....	390	Compare Control Register, Upper Byte (OCSH1, OCSH3, OCSH5) .....	230
Reception Priority Level .....	390	Compare Mode Control Register (OCMOD) .....	236
Setting of Reception Message Object .....	391	Compare Register 1,2 (ADCOMP1, ADCOMP2) .....	260
Recovery		Control Register 1,2 (ADCOMPC1, ADCOMPC2) .....	261
Recovery from a Standby Mode (Stop or Sleep) .....	121	Control Status Register (TMCSR : TMCSR0 to TMCSR2) .....	146
Register		CTBR: Time-base Counter Clear Register .....	81
16-bit Dead Timer Control Register, Lower Byte (DTCR1) .....	251	Data Direction Registers (DDR : DDR0 to DDR5, DDRG) .....	101
16-bit Dead Timer Control Register, Upper Byte (DTCR0) .....	248	Delayed Interrupt Module Registers .....	136
16-bit Dead Timer Control Register, Upper Byte (DTCR2) .....	254	Detailed Explanation of Register .....	127
16-bit Dead Timer Register (TMRRH0 to TMRRH2, TMRRL0 to TMRRL2) .....	247	DIVR0: Basic Clock Dividing Frequency Set Register 0 .....	84
16-bit Free-run Timer Register .....	212	DIVR1: Basic Clock Dividing Frequency Set Register 1 .....	85
16-bit Input Capture Register .....	215	DMAC-ch.0, ch.1, ch.2, ch.3, ch.4 Overall DMAC Control Register [DMACR] .....	470
16-bit Output Compare Register .....	214	DMAC-ch.0, ch.1, ch.2, ch.3, ch.4 Transfer Source/ Transfer Destination address Setting Registers [DMASA/DMADA : DMASA0 to DMASA4/DMADA0 to DMADA4] .....	468
A/D Activation Compare Register .....	217	DSP Control/Status Register (DSP-CSR) .....	439
A/D Channel Control Register (ADCH : ADCH1, ADCH2) .....	415	DSP Delay Register (DSP-LY) .....	443
A/D Control Status Registers (ADCS : ADCS1, ADCS2) .....	420	DSP Variable Monitor Register (DSP-OT0 to DSP-OT7) .....	444
A/D Data Register (ADCD : ADCD10, ADCD11, ADCD20, ADCD21) .....	423	DTTI Bit Operation of Waveform Control Register 2 (SIGCR2) .....	298
A/D Mode Setting Register (ADMD : ADMD1, ADMD2) .....	417	Flash Memory Registers .....	498
A/D Trigger Control Register (ADTRGC) .....	226	Flash Memory Status Register (FLCR) .....	499
Analog Input Control Register (AICR : AICR1, AICR2) .....	424	Flash Memory Wait Register (FLWC) .....	501
Analog Input Control Registers (AICR : AICR1, AICR2) .....	107	Free-run Timer Selector Register .....	217
Bit Search Module Registers .....	138	Free-run Timer Selector Register (FSR0 to FSR2) .....	262
CAN Bit Timing Register (BTR) .....	351	Function of Registers .....	344, 347, 350, 351, 352, 353, 355, 357, 359, 368, 377, 379, 381, 383, 384
CAN Clock Prescaler Register .....	384	GATE Function Control Register (GATEC) .....	173
CAN Control Register (CTRLR) .....	344	General-purpose Register .....	30
CAN Data Renewal Registers (NEWDT1, NEWDT2) .....	378	Hold Request Cancel Level Register (HRCL) .....	115
CAN Interrupt Pending Registers (INTPND1, INTPND2) .....	380		
CAN Interrupt Register (INTR) .....	352		
CAN Message Enable Registers (MSGVAL1, MSGVAL2) .....	382		
CAN Prescaler Extended Register (BRPER) .....	355		
CAN Status Register (STATR) .....	347		
CAN Test Register (TESTR) .....	353		
CAN Transmission Request Registers (TREQR1, TREQR2) .....	376		
CCR (Condition Code Register) .....	31		
CLKR: Clock Source Control Register .....	82		
Clock Prescaler Register .....	341		

ICR (Interrupt Control Register) .....	44
IFx Arbitration Registers 1 and 2 (IFxARB1, IFxARB2) .....	365
IFx Command Mask Register (IFxCMSK) .....	359
IFx Command Request Register (IFxCREQ) .....	357
IFx Data Registers A1, A2, B1 and B2 (IFxDTA1, IFxDTA2, IFxDTB1, IFxDTB2) .....	367
IFx Mask Registers 1 and 2 (IFxMSK1, IFxMSK2) .....	364
IFx Message Control Register (IFxMCTR) .....	366
Input Capture Data Register (IPCPL0 to IPCPL3, IPCPL0 to IPCPL3) .....	238
Input Capture State Control Register (ch.0,ch.1),Lower Byte (PICSLO1) .....	245
Input Capture State Control Register (ch.2,ch.3),Lower Byte (ICSL23).....	241
Input Capture State Control Register (ch.2,ch.3),Upper Byte (ICSH23).....	239
Interrupt Control Register (ICR) .....	114
Interrupt Controller Registers .....	111
List of Message Handler Register .....	340
List of Message Interface Register .....	338
List of Overall Control Register .....	337
Message Handler Register .....	342, 375
Message Interface Register.....	342
Multiplication and Division Register (Multiply & Divide Register).....	35
Notes on Setting Registers.....	459
Output Compare Buffer Register (OCCPBH0 to OCCPBH5, OCCPBL0 to OCCPBL5) .....	228
Output Compare Register (OCCPH0 to OCCPH5,OCCPL0 to OCCPL5).....	229
Output Inversion Register (REVC).....	172
Overall Control Register .....	342, 343
Port Data Registers (PDR : PDR0 to PDR5,PDRG) .....	100
Port Function Registers (PFR: PFR0,PFR1,PTFR0) .....	103
PPG Output Control Register Upper Byte (PICSH01) .....	243
PPG Start Register (TRG) .....	172
PPGn Register (PPGn Operation Mode Control Register) n=0,1,2,3,4,5,6,7 .....	169
Prescaler Register.....	342
PRL/PLH Register (Reload Register : PRL0 to PRL7/PRLH0 to PRLH7) .....	171
Pull-up Control Registers (PCR : PCR0 to PCR4,PCRG).....	102
PWC Control/Status Register (PWCSR : PWCSR0) .....	182

PWC Data Buffer Register (PWCR : PWCR0) .....	187
Ratio of Dividing Frequency Control Register (PDIVR : PDIVR0).....	188
Register Details Explanation .....	137, 139
Register Explanation .....	309
Register Group .....	335
Register List .....	126, 156, 308, 315, 435
Register Outline.....	457
Registers of 8/10-bit A/D Converter .....	414
Registers of the PPG Timer.....	163
Registers of the Pulse Width Counter .....	180
Registers of Timing Generator .....	158
Reload Timer Registers .....	144
RSRR: Reset Source Register/Watchdog Timer Control Register.....	74
SCR (System Condition Code Register) .....	32
Serial Control Register (SCR : SCR0,SCR1) .....	319
Serial Mode Register (SMR : SMR0,SMR1) .....	317
Serial Input Data Register (SIDR : SIDR0,SIDR1)/ Serial Output Data Register (SODR : SODR0,SODR1) .....	321
Serial Status Register (SSR : SSR0,SSR1).....	322
STCR : Standby Control Register .....	76
TBCR : Time-base Counter Control Register .....	79
TBR (Table Base Register) .....	34, 46
Timer Data Register (TCDTH0 to TCDTH2, TCDTL0 to TCDTL2) .....	220
Timer State Control Register,Lower Byte (TCCSL0 to TCCSL2) .....	224
Timer State Control Register,Upper Byte (TCCSH0 to TCCSH2).....	221
TMR Register (16-bit Timer Register) .....	149
TMRLR Register (16-bit Reload Register) .....	149
Waveform Control Register 1 (SIGCR1) .....	257
Waveform Control Register 2 (SIGCR2) .....	259
Waveform Generator Register .....	216
Release Sequence Operation Initialization Reset (RST) Release Sequence.....	62
Set Initialization Reset (INIT) Release Sequence .....	62
Reload Operation Reload Operation .....	477
Reload Register PRL/PLH Register (Reload Register : PRL0 to PRL7/PRLH0 to PRLH7) .....	171
Reload Timer Block Diagram of the Reload Timer .....	145
Reload Timer Registers Reload Timer Registers .....	144

Remote Frame	
Remote Frame .....	391
Request	
Example of Using the Function to Generate a Request to Cancel a Hold Request (HRCR) .....	122
Reset	
16-bit Output Compare Operation (Set/Reset Mode,MOD1x=1).....	279
$\overline{\text{INIT}}$ Pin Input (Set Initialization Reset Pin) .....	60
Normal Reset Operation .....	65
Operation Initialization Reset (RST) .....	59
Operation Initialization Reset (RST) Release Sequence .....	62
Overview of Reset Operation .....	58
RSRR: Reset Source Register/Watchdog Timer Control Register.....	74
Set Initialization Reset (INIT) .....	59
Set Initialization Reset (INIT) Release Sequence .....	62
STCR: SRST Bit Writing (Software Reset) .....	60
Synchronous Reset Operation.....	65
Watchdog Reset.....	61
Reset Factor	
Reset Factor .....	60
Reset Operation	
Normal Reset Operation .....	65
Overview of Reset Operation .....	58
Synchronous Reset Operation.....	65
Reset Source Register	
RSRR: Reset Source Register/Watchdog Timer Control Register.....	74
Restore	
Backup/Restore Processing .....	142
Return	
Return from EIT .....	42
Return Pointer	
RP (Return Pointer).....	34
REVC	
Output Inversion Register (REVC) .....	172
ROM	
FR-CPU ROM Mode (Read only in 32/16/8 Bits) .....	503
Programming with a ROM Programmer .....	493
RP	
RP (Return Pointer).....	34
RSRR	
RSRR: Reset Source Register/Watchdog Timer Control Register.....	74
RST	
Operation Initialization Reset (RST) .....	59
Operation Initialization Reset (RST) Release Sequence.....	62
RTO	
Output Status of RTO0 to RTO5 and GATE .....	288
<b>S</b>	
SCR	
SCR (System Condition Code Register).....	32
Serial Control Register (SCR : SCR0,SCR1) .....	319
Sector Configuration	
Sector Configuration for Flash Memory .....	494
Select	
Select Oscillation Stabilization Wait Time .....	64
Select Operation Mode	
Select Operation Mode.....	191
Selected External Count Clock	
Selected External Count Clock.....	273
Selecting	
Selecting the Transfer Sequence.....	475
Selection	
Selection of the Source Clock .....	67
Serial Control Register	
Serial Control Register (SCR : SCR0,SCR1) .....	319
Serial Input Data Register	
Serial Input Data Register (SIDR : SIDR0,SIDR1)/ Serial Output Data Register (SODR : SODR0,SODR1).....	321
Serial Mode Register	
Serial Mode Register (SMR : SMR0,SMR1) .....	317
Serial On-board Programming	
Pins Used for Fujitsu-standard Serial On-board Programming .....	519
Serial Output Data Register	
Serial Input Data Register (SIDR : SIDR0,SIDR1)/ Serial Output Data Register (SODR : SODR0,SODR1).....	321
Serial Programming	
Basic Configuration for Serial Programming .....	518
Connection Example for Serial Programming .....	520
Serial Status Register	
Serial Status Register (SSR : SSR0,SSR1) .....	322
Set Initialization Reset	
$\overline{\text{INIT}}$ Pin Input (Set Initialization Reset Pin) .....	60
Set Initialization Reset (INIT).....	59
Set Initialization Reset (INIT) Release Sequence .....	62
Set/Reset	
16-bit Output Compare Operation (Set/Reset Mode,MOD1x=1) .....	279

Setting	
Setting of Reception Message Object .....	391
Setting of Transmission Message Object .....	388
SIDR	
Serial Input Data Register (SIDR : SIDR0,SIDR1)/	
Serial Output Data Register	
(SODR : SODR0,SODR1) .....	321
SIGCR	
DTTI Bit Operation of Waveform Control Register 2	
(SIGCR2) .....	298
Waveform Control Register 1 (SIGCR1) .....	257
Waveform Control Register 2 (SIGCR2) .....	259
Silent Mode	
Combination of Silent Mode and Loop Back Mode	
.....	401
Silent Mode .....	399
sim911	
Debugger (sim911,eml911,mon911) .....	544
Single-shot Conversion Mode	
Operation of Single-shot Conversion Mode	
.....	426
Sleep	
Recovery from a Standby Mode (Stop or Sleep)	
.....	121
Sleep Mode	
DMA Transfer during the Sleep Mode.....	484
SMR	
Serial Mode Register (SMR : SMR0,SMR1)	
.....	317
SODR	
Serial Input Data Register (SIDR : SIDR0,SIDR1)/	
Serial Output Data Register	
(SODR : SODR0,SODR1) .....	321
Software	
Software Control of Pin CAN_TX .....	402
Software Request	
Software Request .....	474
Software Reset	
STCR: SRST Bit Writing (Software Reset) .....	60
Source Clock	
Selection of the Source Clock .....	67
SRST Bit Writing	
STCR: SRST Bit Writing (Software Reset) .....	60
SSP	
SSP (System Stack Pointer).....	34, 45
SSR	
Serial Status Register (SSR : SSR0,SSR1) .....	322
Standby Control Register	
STCR : Standby Control Register .....	76
Standby Mode	
Recovery from a Standby Mode (Stop or Sleep)	
.....	121
Start Operation	
Start Operation.....	481

Starting and Stopping	
Starting and Stopping of the Pulse Width Count	
.....	192
start-stop Synchronization	
Asynchronous (start-stop Synchronization) Mode	
.....	325
State	
Input Capture State Control Register	
(ch.0,ch.1),Lower Byte (PICSL01) .....	245
Input Capture State Control Register	
(ch.2,ch.3),Lower Byte (ICSL23) .....	241
Input Capture State Control Register	
(ch.2,ch.3),Upper Byte (ICSH23) .....	239
Pin States in Each CPU State .....	538
State of Device and Each Transition .....	90
Timer State Control Register,Lower Byte	
(TCCSL0 to TCCSL2).....	224
Timer State Control Register,Upper Byte	
(TCCSH0 to TCCSH2).....	221
STATR	
CAN Status Register (STATR).....	347
Status	
A/D Control Status Registers	
(ADCS : ADCS1,ADCS2) .....	420
Control Status Register (TMCSR :	
TMCSR0 to TMCSR2).....	146
DMAC-ch.0,ch.1,ch.2,ch.3,ch.4 Control Status	
Registers A	
[DMACA : DMACA0 to DMACA4]	
.....	459
DMAC-ch.0,ch.1,ch.2,ch.3,ch.4 Control Status	
Registers B	
[DMACB : DMACB0 to DMACB4]	
.....	463
DSP Control/Status Register (DSP-CSR) .....	439
Flash Memory Status Register (FLCR) .....	499
Output Status of RTO0 to RTO5 and GATE	
.....	288
PS (Program Status) .....	30
PWC Control/Status Register (PWCSR : PWCSR0)	
.....	182
Serial Status Register (SSR : SSR0,SSR1).....	322
Status Register	
PWC Control/Status Register (PWCSR : PWCSR0)	
.....	182
STCR	
STCR : Standby Control Register .....	76
STCR: SRST Bit Writing (Software Reset) .....	60
Step/block Transfer	
Step/block Transfer Two-cycle Transfer.....	476
STOP	
Precautions when Returning from STOP State	
Using External Interrupt .....	132

Stop	
Recovery from a Standby Mode (Stop or Sleep)	121
Stop Mode	
Wait Time after Recovering from Stop Mode	70
Stopping	
Operation End/Stopping	482
Stopping Due to an Error	483
STR	
STR Instruction (Transfer Instruction)	451
Structure	
Structure of the Multifunction Timer	202
Suspend	
Suspend	482
Synchronous Reset	
Synchronous Reset Operation	65
System Condition Code Register	
SCR (System Condition Code Register)	32
System Configuration	
System Configuration for AF200 Flash	
Microcontroller Programmer	
(Manufactured by Yokogawa Digital	
Computer Corporation)	521
System Stack Pointer	
SSP (System Stack Pointer)	34, 45
<b>T</b>	
Table Base Register	
TBR (Table Base Register)	34, 46
Tansfer	
Tansfer Type	472
TBCR	
TBCR : Time-base Counter Control Register	79
TBR	
TBR (Table Base Register)	34, 46
TCCSH	
Timer State Control Register,Upper Byte	
(TCCSH0 to TCCSH2)	221
TCCSL	
Timer State Control Register,Lower Byte	
(TCCSL0 to TCCSL2)	224
TCDTH	
Timer Data Register (TCDTH0 to TCDTH2,	
TCDTL0 to TCDTL2)	220
TCDTL	
Timer Data Register (TCDTH0 to TCDTH2,	
TCDTL0 to TCDTL2)	220
Test Mode Setting	
Test Mode Setting	399
TESTR	
CAN Test Register (TESTR)	353
Time-base Counter	
Time-base Counter	87
Time-base Counter Clear Register	
CTBR:Time-base Counter Clear Register	81
Time-base Counter Control Register	
TBCR : Time-base Counter Control Register	79
Timer	
Timer Clear	268
Timer Interrupt	270
Timer Data Register	
Timer Data Register (TCDTH0 to TCDTH2,	
TCDTL0 to TCDTL2)	220
Timer Mode	
Operation of Timer Mode	292
Timer Mode	268
Timer State Control Register	
Timer State Control Register,Lower Byte	
(TCCSL0 to TCCSL2)	224
Timer State Control Register,Upper Byte	
(TCCSH0 to TCCSH2)	221
Timing Generator	
Block Diagram of Timing Generator	157
Operation Overview of Timing Generator	159
Registers of Timing Generator	158
TMCSR	
Control Status Register (TMCSR :	
TMCSR0 to TMCSR2)	146
TMR Register	
TMR Register (16-bit Timer Register)	149
TMRLR Register	
TMRLR Register (16-bit Reload Register)	149
TMRRH	
16-bit Dead Timer Register (TMRRH0 to	
TMRRH2,TMRRL0 to TMRRL2)	
	247
TMRRL	
16-bit Dead Timer Register	
(TMRRH0 to TMRRH2,	
TMRRL0 to TMRRL2)	247
Transfer	
Block Transfer	486
Burst Transfer	487
Burst Two-cycle Transfer	475
Control of the Number of Transfers	479
DMAC-ch.0,ch.1,ch.2,ch.3,ch.4 Transfer Source/	
Transfer Destination address Setting	
Registers	
[DMASA/DMADA : DMASA0 to	
DMASA4/DMADA0 to DMADA4]	
	468
Number of Transfers and Ending Transfer	473
Operation of Data in Two-cycle Transfer Mode	
	488
Selecting the Transfer Sequence	475

Step/block Transfer Two-cycle Transfer .....	476
Transfer address .....	472
Transfer Mode .....	472
Transfer of Operation Results .....	447
Transfer Request Acceptance and Transfer .....	481
<b>Transfer Instruction</b>	
STR Instruction (Transfer Instruction) .....	451
<b>Transition</b>	
State of Device and Each Transition .....	90
<b>Transmission</b>	
Message RAM and Data Transmission/Reception .....	386
Setting of Transmission Message Object .....	388
Transmission Priority Level .....	388
Updating a Transmission Message Object .....	389
<b>TREQR</b>	
CAN Transmission Request Registers (TREQR1, TREQR2) .....	376
<b>TRG</b>	
PPG Start Register (TRG) .....	172
<b>Triggers</b>	
Triggers for the Oscillation Stabilization Wait .....	63
<b>Two-cycle Transfer Mode</b>	
Operation of Data in Two-cycle Transfer Mode .....	488
<b>U</b>	
<b>UART</b>	
Block Diagram of UART .....	316
Clock Selection of UART .....	324
Features of UART .....	314
<b>Underflow Operation</b>	
Underflow Operation .....	151
<b>Updating</b>	
Updating a Transmission Message Object .....	389
<b>User Stack Pointer</b>	
USP (User Stack Pointer) .....	35
<b>USP</b>	
USP (User Stack Pointer) .....	35

<b>U-TIMER</b>	
Block Diagram of U-TIMER .....	308
<b>V</b>	
<b>Variable Monitor Output</b>	
Variable Monitor Output .....	448
<b>Vector Table</b>	
EIT Vector Table .....	46
Vector Table Initial Area .....	38
<b>W</b>	
<b>Wait Time</b>	
Wait Time after Changing the PLL Multiplier Ratio .....	69
Wait Time after Enabling PLL Operation .....	69
Wait Time after Power Supply is Turned on .....	69
Wait Time after Recovering from Stop Mode .....	70
Wait Time after Setting is Initialized .....	69
<b>Watchdog Reset</b>	
Watchdog Reset .....	61
<b>Watchdog Timer Control Register</b>	
RSRR: Reset Source Register/Watchdog Timer Control Register .....	74
<b>Waveform Control Register</b>	
DTTI Bit Operation of Waveform Control Register 2 (SIGCR2) .....	298
Waveform Control Register 1 (SIGCR1) .....	257
Waveform Control Register 2 (SIGCR2) .....	259
<b>Waveform Generator</b>	
Block Diagram of the Waveform Generator .....	208
Notes on Using the Waveform Generator .....	302
Waveform Generator Interrupts .....	266
<b>Waveform Generator Register</b>	
Waveform Generator Register .....	216
<b>Word Alignment</b>	
Word Alignment .....	37

CM71-10130-4E

---

**FUJITSU SEMICONDUCTOR • CONTROLLER MANUAL**  
FR60Lite  
32-BIT MICROCONTROLLER  
MB91265A Series  
HARDWARE MANUAL

---

November 2007 the fourth edition

Published **FUJITSU LIMITED** Electronic Devices  
Edited Strategic Business Development Dept

---



