

To our customers,

---

## Old Company Name in Catalogs and Other Documents

---

On April 1<sup>st</sup>, 2010, NEC Electronics Corporation merged with Renesas Technology Corporation, and Renesas Electronics Corporation took over all the business of both companies. Therefore, although the old company name remains in this document, it is a valid Renesas Electronics document. We appreciate your understanding.

Renesas Electronics website: <http://www.renesas.com>

April 1<sup>st</sup>, 2010  
Renesas Electronics Corporation

Issued by: Renesas Electronics Corporation (<http://www.renesas.com>)

Send any inquiries to <http://www.renesas.com/inquiry>.

## Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.
  - “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.
  - “High Quality”: Transportation equipment (automobiles, trains, ships, etc.); traffic control systems; anti-disaster systems; anti-crime systems; safety equipment; and medical equipment not specifically designed for life support.
  - “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

(Note 1) “Renesas Electronics” as used in this document means Renesas Electronics Corporation and also includes its majority-owned subsidiaries.

(Note 2) “Renesas Electronics product(s)” means any product developed or manufactured by or for Renesas Electronics.

# User's Manual

**Phase-out/Discontinued**

# V821™

## 32-/16-Bit Microprocessor

## Hardware

---

### μPD70741

[MEMO]

**SUMMARY OF CONTENTS**

<b>CHAPTER 1</b>	<b>OVERVIEW .....</b>	<b>23</b>
<b>CHAPTER 2</b>	<b>PIN FUNCTIONS.....</b>	<b>31</b>
<b>CHAPTER 3</b>	<b>CPU FUNCTIONS .....</b>	<b>45</b>
<b>CHAPTER 4</b>	<b>INTERRUPT/EXCEPTION HANDLING FUNCTIONS .....</b>	<b>55</b>
<b>CHAPTER 5</b>	<b>BUS CONTROL FUNCTION .....</b>	<b>75</b>
<b>CHAPTER 6</b>	<b>WAIT CONTROL FUNCITONS .....</b>	<b>109</b>
<b>CHAPTER 7</b>	<b>MEMORY ACCESS CONTROL FUNCTIONS .....</b>	<b>119</b>
<b>CHAPTER 8</b>	<b>DMA FUNCTIONS (DMA CONTROLLER).....</b>	<b>139</b>
<b>CHAPTER 9</b>	<b>SERIAL INTERFACE FUNCTION .....</b>	<b>155</b>
<b>CHAPTER 10</b>	<b>TIMER/COUNTER FUNCTIONS (REAL-TIME PULSE UNIT) .....</b>	<b>183</b>
<b>CHAPTER 11</b>	<b>WATCHDOG TIMER FUNCTIONS .....</b>	<b>213</b>
<b>CHAPTER 12</b>	<b>PORT FUNCTIONS .....</b>	<b>217</b>
<b>CHAPTER 13</b>	<b>CLOCK GENERATION FUNCTIONS .....</b>	<b>221</b>
<b>CHAPTER 14</b>	<b>STANDBY FUNCTIONS .....</b>	<b>225</b>
<b>CHAPTER 15</b>	<b>RESET FUNCTIONS .....</b>	<b>237</b>
<b>APPENDIX A</b>	<b>REGISTER INDEX .....</b>	<b>243</b>
<b>APPENDIX B</b>	<b>GENERAL INDEX .....</b>	<b>247</b>

**NOTES FOR CMOS DEVICES****① PRECAUTION AGAINST ESD FOR SEMICONDUCTORS**

Note:

Strong electric field, when exposed to a MOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop generation of static electricity as much as possible, and quickly dissipate it once, when it has occurred. Environmental control must be adequate. When it is dry, humidifier should be used. It is recommended to avoid using insulators that easily build static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work bench and floor should be grounded. The operator should be grounded using wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions need to be taken for PW boards with semiconductor devices on it.

**② HANDLING OF UNUSED INPUT PINS FOR CMOS**

Note:

No connection for CMOS device inputs can be cause of malfunction. If no connection is provided to the input pins, it is possible that an internal input level may be generated due to noise, etc., hence causing malfunction. CMOS devices behave differently than Bipolar or NMOS devices. Input levels of CMOS devices must be fixed high or low by using a pull-up or pull-down circuitry. Each unused pin should be connected to  $V_{DD}$  or GND with a resistor, if it is considered to have a possibility of being an output pin. All handling related to the unused pins must be judged device by device and related specifications governing the devices.

**③ STATUS BEFORE INITIALIZATION OF MOS DEVICES**

Note:

Power-on does not necessarily define initial status of MOS device. Production process of MOS does not define the initial operation status of the device. Immediately after the power source is turned ON, the devices with reset function have not yet been initialized. Hence, power-on does not guarantee out-pin levels, I/O settings or contents of registers. Device is not initialized until the reset signal is received. Reset operation must be executed immediately after power-on for devices having reset function.

V821 Family and V810 Family are trademarks of NEC Corporation.

UNIX is a registered trademark licensed by X/Open Company Limited in the US and other countries.

Windows is either a registered trademark or a trademark of Microsoft Corporation in the United States and/or other countries.

The application circuits and their parameters are for reference only and are not intended for use in actual design-ins.

**The information in this document is subject to change without notice.**

No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Corporation. NEC Corporation assumes no responsibility for any errors which may appear in this document.

NEC Corporation does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from use of a device described herein or any other liability arising from use of such device. No license, either express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Corporation or others.

While NEC Corporation has been making continuous effort to enhance the reliability of its semiconductor devices, the possibility of defects cannot be eliminated entirely. To minimize risks of damage or injury to persons or property arising from a defect in an NEC semiconductor device, customers must incorporate sufficient safety measures in its design, such as redundancy, fire-containment, and anti-failure features.

NEC devices are classified into the following three quality grades:

"Standard", "Special", and "Specific". The Specific quality grade applies only to devices developed based on a customer designated "quality assurance program" for a specific application. The recommended applications of a device depend on its quality grade, as indicated below. Customers must check the quality grade of each device before using it in a particular application.

Standard: Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment and industrial robots

Special: Transportation equipment (automobiles, trains, ships, etc.), traffic control systems, anti-disaster systems, anti-crime systems, safety equipment and medical equipment (not specifically designed for life support)

Specific: Aircrafts, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems or medical equipment for life support, etc.

The quality grade of NEC devices is "Standard" unless otherwise specified in NEC's Data Sheets or Data Books. If customers intend to use NEC devices for applications other than those specified for Standard quality grade, they should contact an NEC sales representative in advance.

Anti-radioactive design is not implemented in this product.

## Regional Information

Some information contained in this document may vary from country to country. Before using any NEC product in your application, please contact the NEC office in your country to obtain a list of authorized representatives and distributors. They will verify:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, registered trademarks, export restrictions, and other legal issues may also vary from country to country.

**NEC Electronics Inc. (U.S.)**

Santa Clara, California  
Tel: 408-588-6000  
800-366-9782  
Fax: 408-588-6130  
800-729-9288

**NEC Electronics (Germany) GmbH**

Duesseldorf, Germany  
Tel: 0211-65 03 02  
Fax: 0211-65 03 490

**NEC Electronics (UK) Ltd.**

Milton Keynes, UK  
Tel: 01908-691-133  
Fax: 01908-670-290

**NEC Electronics Italiana s.r.l.**

Milano, Italy  
Tel: 02-66 75 41  
Fax: 02-66 75 42 99

**NEC Electronics (Germany) GmbH**

Benelux Office  
Eindhoven, The Netherlands  
Tel: 040-2445845  
Fax: 040-2444580

**NEC Electronics (France) S.A.**

Velizy-Villacoublay, France  
Tel: 01-30-67 58 00  
Fax: 01-30-67 58 99

**NEC Electronics (France) S.A.**

Spain Office  
Madrid, Spain  
Tel: 01-504-2787  
Fax: 01-504-2860

**NEC Electronics (Germany) GmbH**

Scandinavia Office  
Taeby, Sweden  
Tel: 08-63 80 820  
Fax: 08-63 80 388

**NEC Electronics Hong Kong Ltd.**

Hong Kong  
Tel: 2886-9318  
Fax: 2886-9022/9044

**NEC Electronics Hong Kong Ltd.**

Seoul Branch  
Seoul, Korea  
Tel: 02-528-0303  
Fax: 02-528-4411

**NEC Electronics Singapore Pte. Ltd.**

United Square, Singapore 1130  
Tel: 65-253-8311  
Fax: 65-250-3583

**NEC Electronics Taiwan Ltd.**

Taipei, Taiwan  
Tel: 02-2719-2377  
Fax: 02-2719-5951

**NEC do Brasil S.A.**

Electron Devices Division  
Rodovia Presidente Dutra, Km 214  
07210-902-Guarulhos-SP Brasil  
Tel: 55-11-6465-6810  
Fax: 55-11-6465-6829



## Major Changes

Page	Description
P. 24	The $\mu$ PD70741GC-25-7EA has been deleted from <b>Section 1.2</b> .
P. 25	The $\mu$ PD70741GC-25-7EA has been deleted from <b>Section 1.3</b> .
P. 27	<b>Section 1.4.1</b> has been modified.
P. 123	The description has been added to <b>Section 7.1.5</b> .
P. 151	<b>Figure 8-9</b> has been modified.
P. 221	The clock generation function block diagram in <b>Section 13.2</b> has been modified.
P. 223	The description has been added to <b>Section 13.3.2 (2)</b> .
P. 231	The description has been added to <b>Section 14.5.1</b> .

The mark ★ shows major revised points.

[MEMO]

## PREFACE

**Intended readers** : This manual is aimed at those users who wish to become familiar with the functions of the V821 ( $\mu$ PD70741), and those involved in designing systems based on the V821.

**Purpose** : The purpose of this manual is to assist users in understanding the hardware functions of the V821, as listed in "Configuration" below.

**Configuration** : The V821 User's Manual is provided as two volumes, Hardware (this manual) and Architecture (V810 Family<sup>TM</sup> User's Manual, Architecture).

Hardware
----------

- Pin functions
- CPU functions
- Interrupt functions
- Bus control functions
- Built-in peripheral functions
- Reset functions

Architecture
--------------

- Register set
- Data types
- Address space
- Instruction format and instruction set
- Interrupts and exceptions

**How to use this manual:** Readers are assumed to be familiar with electronics, logic circuits, and micro-computers.

For an explanation of the CPU functions

-> Read **Chapter 3**.

For an explanation of the instructions

-> Refer to the **V810 Family User's Manual, Architecture** (the companion volume to this manual).

To gain an overall understanding of the functions of the V821

-> Read this manual in its entirety.

To know the electrical characteristics

-> Refer to the **V821 Data Sheet** provided separately.

In this manual, 2-byte data is referred to as halfword data, while 4-byte data is referred to as word data.

## Legend

- : Significance of data representation : Left high, right low
- Representation of active low :  $\overline{\text{xxx}}$  [bar above a pin or signal name)
- Memory map address : Top upper, bottom lower
- Note : Explanation of Note marked in text
- Caution : Note to which particular attention must be afforded
- Remark : Supplementary explanation of text
- Representation of numbers : xxxx or xxxxB for a binary number  
xxxx for a decimal number  
xxxxH for a hexadecimal number
- Prefixes indicating powers of two (address space, memory capacity):  
K (kilo) :  $2^{10} = 1,024$   
M (mega) :  $2^{20} = 1,024^2$   
G (giga) :  $2^{30} = 1,024^3$

**Related document** : Some related documents may be preliminary editions; Note, however, that whether a related document is preliminary is not indicated in this manual.

### ○ Documents related to devices

Document name	Document number
V821 User's Manual, Hardware	This manual
V810 Family User's Manual, Architecture	U10082E
μPD70741 Data Sheet	U11678E

### ○ Documents related to development tools (user's manual)

Document name	Document number
CA732 (C compiler)	Operation (UNIX™-based)
	Operation (Windows™-based)
	Assembly language
	C
	Project manager
RX732 (Real-time OS)	Basic
	Technical
	Nucleus installation

## CONTENTS

<b>CHAPTER 1</b>	<b>OVERVIEW .....</b>	<b>23</b>
1.1	FEATURES .....	23
1.2	ORDERING INFORMATION .....	24
1.3	PIN CONFIGURATION (TOP VIEW) .....	25
1.4	FUNCTIONAL BLOCK CONFIGURATION .....	27
1.4.1	Internal Block Diagram .....	27
1.4.2	Internal Units .....	28
<b>CHAPTER 2</b>	<b>PIN FUNCTIONS.....</b>	<b>31</b>
2.1	PIN FUNCTIONS .....	31
2.1.1	Port Pins .....	31
2.1.2	Non-Port Pins .....	32
2.2	PIN STATES .....	34
2.3	PIN FUNCTIONS .....	35
2.3.1	Address Bus .....	35
2.3.2	Data Bus .....	35
2.3.3	Bus Control Signals .....	35
2.3.4	System Control Signals.....	37
2.3.5	Interrupt Control Signals .....	38
2.3.6	DRAM Control Signals .....	38
2.3.7	DMA Control Signals .....	39
2.3.8	Real-Time Pulse Control Signals .....	40
2.3.9	Serial Control Signals .....	40
2.3.10	Watchdog Timer Control Signal .....	41
2.3.11	Port Control Signal .....	41
2.4	PIN I/O CIRCUITS AND PROCESSING OF UNUSED PINS .....	42
2.5	PIN I/O CIRCUITS .....	43
<b>CHAPTER 3</b>	<b>CPU FUNCTIONS .....</b>	<b>45</b>
3.1	FEATURES .....	45
3.2	ADDRESS SPACE .....	45
3.2.1	Memory Map .....	46
3.2.2	I/O Map .....	47
3.2.3	Images .....	47
3.2.4	Differences between the Memory and I/O Maps .....	48
3.3	CPU REGISTER SET .....	49
3.3.1	Program Register Set .....	50

3.3.2	System Register Set .....	51
3.4	BUILT-IN PERIPHERAL I/O REGISTERS .....	52
<b>CHAPTER 4</b>	<b>INTERRUPT/EXCEPTION HANDLING FUNCTIONS .....</b>	<b>55</b>
4.1	FEATURES .....	55
4.2	NONMASKABLE INTERRUPTS .....	58
4.2.1	Operation .....	58
4.3	MASKABLE INTERRUPTS .....	60
4.3.1	Block Diagram .....	60
4.3.2	Operation .....	61
4.3.3	Maskable Interrupt Priority .....	63
4.4	CONTROL REGISTERS .....	64
4.4.1	Interrupt Group Priority Register (IGP) .....	64
4.4.2	Interrupt Clear Register (ICR) .....	66
4.4.3	Interrupt Request Register (IRR) .....	66
4.4.4	Interrupt Request Mask Register (IMR) .....	67
4.4.5	ICU Mode Register (IMOD) .....	68
4.5	EXCEPTION HANDLING (SOFTWARE EXCEPTION AND EXCEPTION TRAP) .....	69
4.5.1	Operation .....	70
4.5.2	Return from an Exception or Interrupt .....	72
4.6	PRIORITY CONTROL .....	73
4.6.1	Priorities of Interrupts and Exceptions .....	73
4.6.2	Priorities of Floating-Point Exceptions .....	74
<b>CHAPTER 5</b>	<b>BUS CONTROL FUNCTION .....</b>	<b>75</b>
5.1	CPU BUS STATES .....	75
5.2	DMAC BUS STATE .....	77
5.3	BUS PRIORITY .....	80
5.4	DATA FLOW WHEN USING AN ADDITIONAL BUS CYCLE .....	81
5.5	RELATIONSHIP BETWEEN EXTERNAL ACCESSES AND THE DATA BUS .....	82
5.5.1	Relationship between External Access and Byte Enable Signal .....	82
5.5.2	Operand Read .....	83
5.5.3	Operand Write .....	84
5.5.4	Notes on Bit Strings .....	85
5.6	EXTERNAL I/O ACCESS .....	86
5.6.1	External I/O Read Cycle .....	86
5.6.2	External I/O Write Cycle .....	87
5.7	INTERNAL I/O ACCESS .....	88
5.7.1	Internal I/O Read Cycle .....	88

5.7.2	Internal I/O Write Cycle.....	89
5.8	SRAM (ROM) ACCESS.....	90
5.8.1	SRAM (ROM) Read Cycle .....	90
5.8.2	SRAM Write Cycle .....	91
5.9	PAGE-ROM ACCESS .....	92
5.9.1	Page-ROM Read Cycle .....	92
5.10	DRAM ACCESS.....	93
5.10.1	DRAM Read Cycle .....	93
5.10.2	DRAM Write Cycle .....	95
5.10.3	CBR Refresh Cycle .....	98
5.10.4	Self-Refresh Cycle .....	99
5.10.5	Bus Cycle during Fly-by Transfer .....	100
5.11	EXCEPTION HANDLING CYCLES .....	104
5.11.1	Machine Fault Cycle .....	104
5.11.2	Halt Acknowledge Cycle .....	105
5.12	CONTROL SIGNAL TIMING .....	106
5.12.1	Bus Hold .....	106
5.12.2	Bus Lock .....	108
<b>CHAPTER 6</b>	<b>WAIT CONTROL FUNCTIONS .....</b>	<b>109</b>
6.1	FEATURES .....	109
6.2	ADDRESS SPACES AND BLOCKS .....	109
6.3	CHIP SELECT CREATION AND BUS CYCLE SELECTION FUNCTIONS.....	110
6.4	PROGRAMMABLE WAIT FUNCTION .....	110
6.4.1	Wait Control Using the $\overline{\text{READY}}$ Pin.....	110
6.4.2	Wait Control during DMA Transfer .....	111
6.4.3	Bus Cycles during Which the Wait Function Is Effective .....	111
6.5	CONTROL REGISTERS .....	114
6.5.1	Bus Cycle Type Control Register (BCTC) .....	114
6.5.2	Programmable Wait Control Register 0 (PWC0) .....	115
6.5.3	Programmable Wait Control Register 1 (PWC1) .....	116
6.5.4	Programmable Wait Control Register 2 (PWC2) .....	117
<b>CHAPTER 7</b>	<b>MEMORY ACCESS CONTROL FUNCTIONS .....</b>	<b>119</b>
7.1	DRAM CONTROLLER (DRAMC) .....	119
7.1.1	Features.....	119
7.1.2	Connecting DRAM.....	119
7.1.3	Address Multiplexing Function.....	121
7.1.4	DRAM Configuration Register (DRC).....	122
7.1.5	DRAM Read/Write Cycles.....	123

7.1.6	Wait Control for DRAM Cycle .....	127
7.1.7	Refresh Function .....	129
7.1.8	Self-Refresh Function .....	132
7.2	ROM CONTROLLER (ROMC) .....	134
7.2.1	on-page/off-page Decision .....	134
7.2.2	Page-ROM Access .....	135
7.2.3	Page-ROM Configuration Register (PRC) .....	137
<b>CHAPTER 8</b>	<b>DMA FUNCTIONS (DMA CONTROLLER) .....</b>	<b>139</b>
8.1	FEATURES .....	139
8.2	CONFIGURATION .....	140
8.3	DMA CONTROL REGISTERS .....	140
8.3.1	DMA Source Address Registers 0 and 1 (DSA0 and DSA1) .....	140
8.3.2	DMA Destination Address Registers 0 and 1 (DDA0 and DDA1) .....	141
8.3.3	DMA Byte Count Registers 0 and 1 (DBC0 and DBC1) .....	143
8.3.4	DMA Channel Control Registers 0 and 1 (DCHC0 and DCHC1) .....	144
8.4	TRANSFER MODES .....	146
8.4.1	Single Transfer Mode .....	146
8.4.2	Single-Step Transfer Mode .....	146
8.4.3	Block Transfer Mode .....	147
8.5	DMA TRANSFER TYPES AND TRANSFER OBJECTS .....	148
8.5.1	Two-Cycle Transfer .....	148
8.5.2	Fly-by Transfer .....	149
8.5.3	Transfer Objects .....	149
8.6	DMA CHANNEL PRIORITIES .....	150
8.7	DMA TRANSFER REQUESTS .....	150
8.8	DMA TRANSFER END INTERRUPTS .....	150
8.9	DMA TRANSFER END OUTPUT .....	151
8.10	FORCIBLE INTERRUPTION .....	152
8.11	DATA FLOW DURING DMA TRANSFER .....	152
<b>CHAPTER 9</b>	<b>SERIAL INTERFACE FUNCTION .....</b>	<b>155</b>
9.1	FEATURES .....	155
9.2	ASYNCHRONOUS SERIAL INTERFACE (UART) .....	155
9.2.1	Features .....	155
9.2.2	Configuration .....	156
9.2.3	UART Control Registers .....	158
9.2.4	Interrupt Requests .....	164
9.2.5	Operation .....	165
9.3	SYNCHRONOUS SERIAL INTERFACE (CSI) .....	169



9.3.1	Features .....	169
9.3.2	Configuration .....	170
9.3.3	CSI Control Registers .....	171
9.3.4	Basic Operation .....	172
9.3.5	Transmission in Three-Wire Serial I/O Mode .....	174
9.3.6	Reception in Three-Wire Serial I/O Mode .....	175
9.3.7	Transmission and Reception in Three-Wire Serial I/O Mode .....	176
9.3.8	Example System Configuration .....	177
9.4	BAUD RATE GENERATOR (BRG) .....	178
9.4.1	Configuration and Function .....	178
9.4.2	Baud Rate Generator Register (BRG) .....	181
9.4.3	Baud Rate Generator Prescaler Mode Register (BPRM) .....	181
<b>CHAPTER 10</b>	<b>TIMER/COUNTER FUNCTIONS (REAL-TIME PULSE UNIT) .....</b>	<b>183</b>
10.1	FEATURES .....	183
10.2	BASIC CONFIGURATION .....	184
10.2.1	Timer 0 .....	186
10.2.2	Timer 1 .....	188
10.3	CONTROL REGISTERS .....	189
10.4	OPERATION OF TIMER 0 .....	195
10.4.1	Counting .....	195
10.4.2	Count Clock Selection .....	195
10.4.3	Overflow .....	196
10.4.4	Clearing/Starting the Timer by the TCLR Input .....	197
10.4.5	Capturing .....	198
10.4.6	Comparison .....	199
10.5	OPERATION OF TIMER 1 .....	201
10.5.1	Counting .....	201
10.5.2	Input Clock Selection .....	201
10.5.3	Overflow .....	201
10.5.4	Comparison .....	202
10.6	EXAMPLES OF APPLICATIONS .....	204
10.7	CAUTIONS .....	211
<b>CHAPTER 11</b>	<b>WATCHDOG TIMER FUNCTIONS .....</b>	<b>213</b>
11.1	FEATURES .....	213
11.2	CONFIGURATION .....	213
11.3	OPERATION .....	214
11.4	WDT MODE REGISTER (WDTM) .....	215

<b>CHAPTER 12</b>	<b>PORT FUNCTIONS .....</b>	<b>217</b>
12.1	FEATURES .....	217
12.2	CONFIGURATION .....	217
12.3	PIN FUNCTIONS OF PORT 0 .....	218
12.4	CONTROL REGISTER .....	219
12.4.1	Port Mode Control Register 0 (PMC0) .....	219
12.4.2	Port Mode Register 0 (PM0) .....	219
<b>CHAPTER 13</b>	<b>CLOCK GENERATION FUNCTIONS .....</b>	<b>221</b>
13.1	FEATURES .....	221
13.2	CONFIGURATION .....	221
13.3	INPUT CLOCK SELECTION .....	222
13.3.1	Direct Mode .....	222
13.3.2	PLL Mode .....	222
13.4	CLOCK OUTPUT CONTROL .....	223
13.5	CLOCK CONTROL REGISTER (CGC) .....	224
<b>CHAPTER 14</b>	<b>STANDBY FUNCTIONS .....</b>	<b>225</b>
14.1	FEATURES .....	225
14.2	STANDBY MODE .....	225
14.3	STANDBY CONTROL REGISTER (STBC) .....	228
14.4	HALT MODE .....	229
14.4.1	Placing the System in the HALT Mode and Operation State .....	229
14.4.2	Releasing the System from the HALT Mode .....	230
14.5	IDLE MODE .....	231
14.5.1	Placing the System in the IDLE Mode and Operation State .....	231
14.5.2	Releasing the System from the IDLE Mode .....	233
14.6	STOP MODE .....	234
14.6.1	Placing the System in the STOP Mode and Operation State .....	234
14.6.2	Releasing the System from the STOP Mode .....	235
14.7	SECURING OSCILLATION SETTLING TIME .....	235
<b>CHAPTER 15</b>	<b>RESET FUNCTIONS .....</b>	<b>237</b>
15.1	FEATURES .....	237
15.2	PIN FUNCTIONS .....	237
15.3	INITIALIZATION .....	239
<b>APPENDIX A</b>	<b>REGISTER INDEX .....</b>	<b>243</b>
A.1	REGISTER NAMES .....	243
A.2	REGISTER SYMBOLS .....	245
<b>APPENDIX B</b>	<b>GENERAL INDEX .....</b>	<b>247</b>

## LIST OF FIGURES

Figure No.	Title	Page
3-1.	Memory Map .....	46
3-2.	I/O Map .....	47
3-3.	Images in an Address Space .....	48
4-1.	Noise Elimination of the $\overline{\text{NMI}}$ Pin Introducing an Analog Delay .....	58
4-2.	Nonmaskable Interrupt Handling Procedure .....	59
4-3.	Accepting a Nonmaskable Interrupt Request .....	59
4-4.	Maskable Interrupts .....	60
4-5.	Maskable Interrupt Handling Procedure .....	62
5-1.	CPU Bus Cycle State Transition .....	76
5-2.	DMAC Bus Cycle State Transition .....	78
5-3.	Data Flow When an Additional Bus Cycle Is Used .....	81
5-4.	Read Cycle .....	83
5-5.	Write Cycle .....	84
5-6.	Write Cycles for Bit Strings .....	85
5-7.	External I/O Read Cycle .....	86
5-8.	External I/O Write Cycle .....	87
5-9.	Internal I/O Read Cycle .....	88
5-10.	Internal I/O Write Cycle .....	89
5-11.	SRAM (ROM) Read Cycle .....	90
5-12.	SRAM Write Cycle .....	91
5-13.	Page-ROM Read Cycle .....	92
5-14.	DRAM Read Cycle .....	94
5-15.	DRAM Write Cycle .....	96
5-16.	CBR Refresh Cycle .....	98
5-17.	CBR Self-Refresh Cycle .....	99
5-18.	Fly-by Read Cycle during DMA Block Transfer (DRAM to External I/O Device) .....	100
5-19.	Fly-by Read Cycle during DMA Single Transfer and Single-Step Transfer .....	101
	(DRAM to External I/O Device) .....	101
5-20.	Fly-by Write Cycle during DMA Block Transfer (External I/O Device to DRAM) .....	102
5-21.	Fly-by Write Cycle during DMA Single Transfer and Single-Step Transfer .....	103
	(External I/O Device to DRAM) .....	103
5-22.	Machine Fault Cycle .....	105
5-23.	Halt Acknowledge Cycle .....	105
5-24.	Bus Hold Cycle .....	107

Figure No.	Title	Page
5-25.	Bus Lock Cycle When the CAXI Instruction Is Executed .....	108
6-1.	Memory and I/O Maps .....	109
7-1.	Connection of 16-Mbyte (1-Mbyte x 16) DRAM.....	120
7-2.	Connection of 4-Mbyte (1-Mbyte x 4) DRAM.....	120
7-3.	Output of Row and Column Addresses.....	121
7-4.	DRAM Read Cycle .....	124
7-5.	DRAM Write Cycle .....	125
7-6.	Insertion of Wait State (1 Wait) during DRAM Read for .....	127
	Fly-by DMA Transfer (DRAM to External I/O) .....	127
7-7.	Insertion of Wait State (1 Wait) during DRAM Write for .....	128
	Fly-by DMA Transfer (External I/O to DRAM) .....	128
7-8.	CBR Refresh Cycle .....	131
7-9.	Cancellation of CBR Self Refresh with $\overline{\text{NMI}}$ Input .....	132
7-10.	Cancellation of CBR Self-Refresh with $\overline{\text{HLDRQ}}$ Input (in IDLE mode).....	133
7-11.	on-page/off-page Decision When ROM Having a Page Access Function Is Connected .....	134
7-12.	Timing of the Page-ROM Cycle (16-Mbit (1-Mbit x 16) Page-ROM) .....	135
7-13.	Connection of 16-Mbit (1-Mbit x 16) Page-ROM .....	135
7-14.	Connection of 16-Mbit (2-Mbit x 8) Page-ROM .....	136
8-1.	Block Diagram of DMAC .....	140
8-2.	Single Transfer Example 1 .....	146
8-3.	Single Transfer Example 2 .....	146
8-4.	Single-Step Transfer Example 1 .....	146
8-5.	Single-Step Transfer Example 2 .....	147
8-6.	Block Transfer Example .....	147
8-7.	Timing of Two-Cycle, Single-Step DMA Transfer (External I/O to DRAM (on-page)) ...	148
8-8.	Timing of Fly-by DMA Transfer (DRAM (on-page) to External I/O) .....	149
8-9.	Timing of DMA Transfer End Output .....	151
8-10.	Data Flow during DMA Transfer (Two-Cycle Byte Transfer) .....	152
8-11.	Data Flow during DMA Transfer (Two-Cycle, 16-Bit Transfer) .....	153
8-12.	Data Flow during DMA Transfer (Fly-by Transfer) .....	154
9-1.	Asynchronous Serial Interface .....	157
9-2.	Format of Transmission and Reception Data Transferred .....	165
	via Asynchronous Serial Interface .....	165

Figure No.	Title	Page
9-3.	Timing of Transmission Completion Interrupt via Asynchronous Serial Interface .....	166
9-4.	Timing of Reception Completion Interrupt via Asynchronous Serial Interface .....	167
9-5.	Timing of Reception Error .....	168
9-6.	Timing of Three-Wire Serial I/O Mode (Transmission) .....	174
9-7.	Timing of Three-Wire Serial I/O Mode (Reception) .....	175
9-8.	Timing of Three-Wire Serial I/O Mode (Transmission and Reception) .....	177
9-9.	Example System Configuration Using CSI .....	177
9-10.	Block Diagram .....	178
10-1.	Basic Operation of Timer 0 .....	195
10-2.	Operation after an Overflow (When ECLR0 = 0 and OST = 1) .....	196
10-3.	Clearing/Starting TM0 by a TCLR Input (When ECLR0 = 1 and OST = 0) .....	197
10-4.	Clearing/Starting TM0 by a TCLR Input and the Related Overflow (When ECLR0 = 0 and OST = 1) .....	197
10-5.	Example of TM0 Capturing (When Both Rising and Falling Edges Are Specified) .....	198
10-6.	Example of TM0 Capturing .....	199
10-7.	Example of Comparison .....	200
10-8.	Example of TM0 Comparison (Set/Reset Output Mode) .....	200
10-9.	Basic Operation of Timer 1 .....	201
10-10.	Operation from When CM1 Is 1 until It Is FFFFH .....	202
10-11.	Operation with CM1 Cleared to 0 .....	203
10-12.	Interval Timer Operation Timing (Timer 1) .....	204
10-13.	Interval Timer Operation Setting Procedure (Timer 1) .....	204
10-14.	Pulse Width Measurement Timing (Timer 0) .....	205
10-15.	Pulse Width Measurement Setting Procedure (Timer 0) .....	206
10-16.	Pulse Width Calculation Interrupt Request Handling Routine (Timer 0) .....	206
10-17.	PWM Output Timing (TM0) .....	207
10-18.	PWM Output Setting Procedure (Timer 0) .....	208
10-19.	Compare Value Rewrite Interrupt Request Handling Routine (Timer 0) .....	209
10-20.	Pulse Width Measurement Timing (TM0) .....	210
10-21.	Period Measurement Setting Procedure (Timer 0) .....	210
10-22.	Period Calculation Interrupt Request Handling Routine (Timer 0) .....	211
11-1.	Watchdog Timer Block Diagram .....	213
14-1.	Bus Hold in IDLE Mode and Restart of Self-Refresh .....	233

## LIST OF TABLES

Table No.	Title	Page
3-1.	Program Registers .....	50
3-2.	System Register Numbers .....	51
3-3.	Built-in Peripheral I/O Registers .....	52
4-1.	Interrupts .....	56
4-2.	Correspondence between Priorities, Exception Codes, Handler Addresses, and Interrupt Levels (When the IGP Register Is Set to 4EH) .....	65
4-3.	Correspondence between Interrupt Control Register Bits and Interrupt Request Signals .....	67
4-4.	Priorities of Interrupts and Exceptions .....	73
4-5.	Priorities of Floating-Point Exceptions .....	74
5-1.	Bus Priority .....	80
5-2.	Relationship between Addresses, Data Length, $\overline{UBE}$ , and Address Pins.....	82
5-3.	Correspondence between the Address Bus and Data Bus in the Machine Fault Cycle .....	104
6-1.	Bus Cycles during Which the Wait Function Is Effective .....	112
6-2.	Cycles during Which Programmable Waits Can Be Set, and .....	113
	Corresponding Control Registers .....	113
6-3.	Number of Waits to Be Inserted during Fly-by DMA Transfer (at off-page Time during DRAM Cycle) .....	118
7-1.	DRAM Control Pins .....	119
7-2.	Examples of DRAM and Address Multiplexing Width .....	121
7-3.	Differences between on-page and off-page Cycles .....	123
7-4.	DRAM Refresh Intervals .....	130
7-5.	Examples of Setting the Interval Factor .....	130
8-1.	Transfer Types vs. Transfer Objects .....	149
9-1.	Generated Interrupts and Default Priorities .....	164
9-2.	BRG Setting Data .....	180
10-1.	Configuration of the RPU .....	184
10-2.	Capture Trigger Signal for the 16-Bit Capture Register (TM0) .....	198

Table No.	Title	Page
10-3.	Interrupt Request Signal from the 16-Bit Compare Register (TM0) .....	199
12-1.	Port Operation .....	220
14-1.	Clock Generator Operation under Standby Control .....	226
14-2.	Operations during the HALT Mode .....	229
14-3.	Operations during the IDLE Mode .....	232
14-4.	Operations during the STOP Mode .....	234
14-5.	Count Time Examples .....	236
15-1.	Output State of Each Pin during a Reset .....	238
15-2.	Initial Value in Each Register after a Reset .....	240

[MEMO]



## CHAPTER 1 OVERVIEW

The V821 is a 32-/16-bit RISC microprocessor that uses, as its processor core, the high-performance 32-bit V810™ (μPD70732) microprocessor designed for built-in control applications. It incorporates peripheral functions such as a DRAM/ROM controller, 2-channel DMA controller, real-time pulse unit, serial interface, and interrupt controller.

The V821, which offers quick real-time response, high-speed integer instructions, bit string instructions, and floating-point instructions, is ideally suited to use in OA equipment such as printers and facsimiles, image processing devices such as those used in navigation units, portable devices, and other devices demanding excellent cost performance.

### 1.1 FEATURES

- 32-bit CPU core
  - Compatible with V810 instructions
  - Built-in 1-Kbyte instruction cache memory
  - 1-clock-pulse pitch pipeline
  - 16-bit data bus
  - Internal 4-Gbyte linear addresses
  - General-purpose registers: 32 bits x 32
  - Instruction group that can be applied to a wide range of fields (floating-point instructions, bit string manipulation instructions)
- Interrupts/exceptions
  - Nonmaskable : 1 external input
  - Maskable : 8 external inputs
  - 11 types of internal sources
  - Priorities can be specified in units of four groups.
- Wait control unit
  - Capable of CS control over four blocks in both memory and I/O spaces.
  - Linear address space of each block: 16 Mbytes
  - Automatic insertion of 0-7 waits per block
  - Bus cycle (page-ROM, DRAM) selection function
- Memory access control functions
  - Supports DRAM high-speed page mode.
  - Supports page-ROM page mode.
- DMA controller
  - 2 channels
  - Maximum transfer count: 65,536
  - Two transfer types (fly-by (1-cycle) transfer and 2-cycle transfer)
  - Three transfer modes (single transfer, single-step transfer, and block transfer)
  - Programmable wait function

- Serial interfaces
  - Asynchronous serial interface (UART): 1 channel
  - Synchronous serial interface (CSI): 1 channel
  - Built-in dedicated baud rate generator
- Real-time pulse unit
  - 16-bit timer/event counter: 1 channel
  - Timer output: 2
  - 16-bit capture/compare register: 4
  - 16-bit interval timer: 1 channel
- Watchdog timer functions
- Clock generator
  - Multiplication-by-5 function with a PLL clock synthesizer
- Standby functions
  - HALT, IDLE, and STOP modes
  - Clock output control

## ★ 1.2 ORDERING INFORMATION

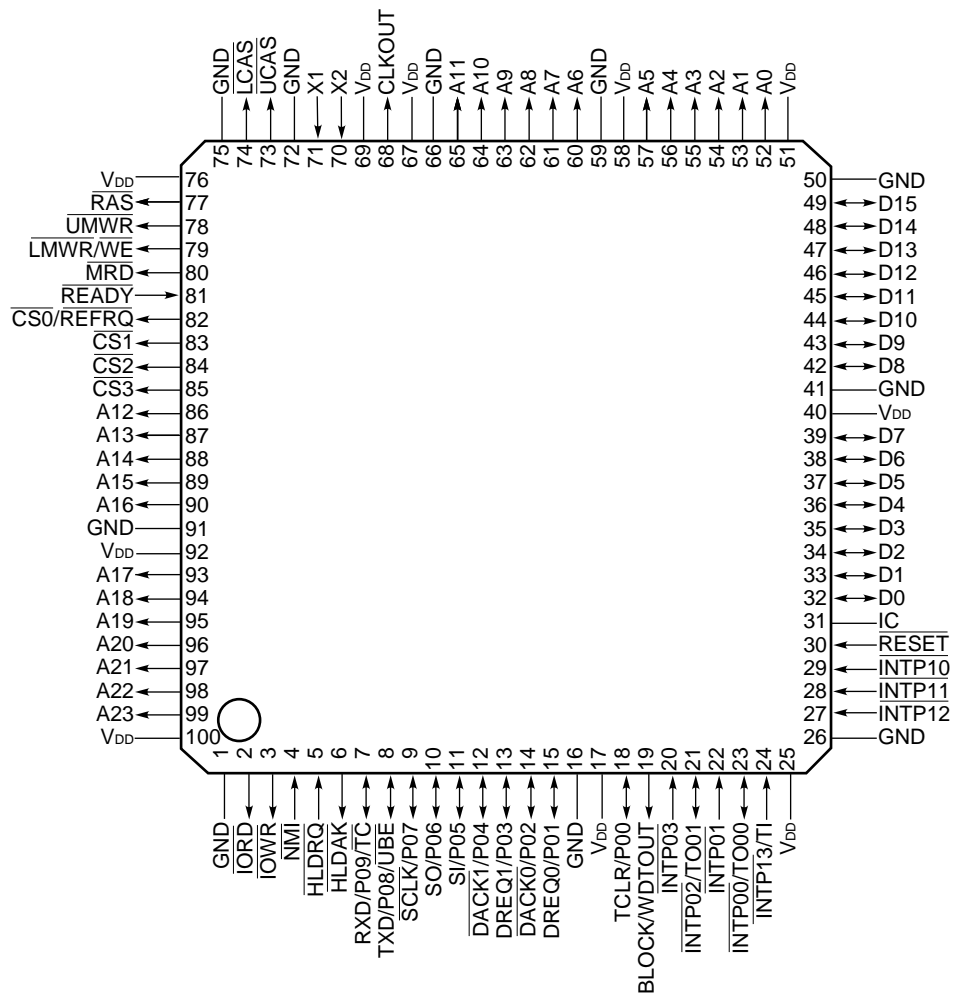
Part number	Package	Maximum operating frequency (MHz)
μPD70741GC-25-8EU	100-pin plastic LQFP (fine pitch) (14 x 14 x 1.40 mm)	25

Not all devices/types available in every country. Please check with local NEC representative for availability and additional information.

## ★ 1.3 PIN CONFIGURATION (TOP VIEW)

100-pin plastic LQFP (fine pitch) (14 x 14 mm)

μPD70741GC-25-8EU



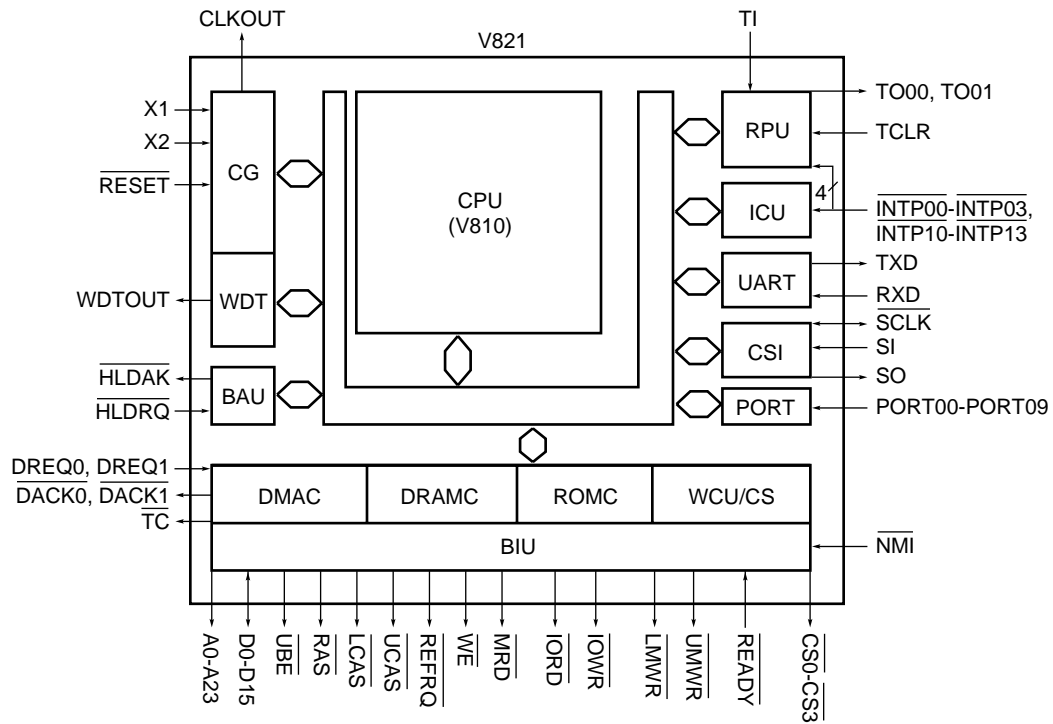
**Caution** Connect pin IC to GND via a resistor.

**Pin names**

A0-A23	: Address Bus
BLOCK	: Bus Lock
CLKOUT	: System Clock Out
$\overline{\text{CS0-CS3}}$	: Chip Select
D0-D15	: Data Bus
$\overline{\text{DACK0}}, \overline{\text{DACK1}}$	: DMA Acknowledge
DREQ0, DREQ1	: DMA Request
$\overline{\text{HLD\text{AK}}}$	: Hold Acknowledge
$\overline{\text{HLDRQ}}$	: Hold Request
$\overline{\text{INTP00-INTP03}}, \overline{\text{INTP10-INTP13}}$	: Interrupt Request
$\overline{\text{IORD}}$	: I/O Read
$\overline{\text{IOWR}}$	: I/O Write
$\overline{\text{LCAS}}$	: Lower Column Address Strobe
$\overline{\text{LMWR}}$	: Lower Memory Write
$\overline{\text{MRD}}$	: Memory Read
$\overline{\text{NMI}}$	: Non-maskable Interrupt Request
P00-P09	: Port
$\overline{\text{RAS}}$	: Row Address Strobe
$\overline{\text{READY}}$	: Ready
$\overline{\text{REFRQ}}$	: Refresh Request
$\overline{\text{RESET}}$	: Reset
$\overline{\text{RXD}}$	: Receive Data
$\overline{\text{SCLK}}$	: Serial Clock
SI	: Serial Input
SO	: Serial Output
$\overline{\text{TC}}$	: Terminal Count
TCLR	: Timer Clear
TI	: Timer Input
TO00, TO01	: Timer Output
$\overline{\text{TXD}}$	: Transmit Data
$\overline{\text{UBE}}$	: Upper Byte Enable
$\overline{\text{UCAS}}$	: Upper Column Address Strobe
$\overline{\text{UMWR}}$	: Upper Memory Write
$\overline{\text{WDTOUT}}$	: Watchdog Timer Output
$\overline{\text{WE}}$	: Write Enable
X1, X2	: Crystal Oscillator

## 1.4 FUNCTIONAL BLOCK CONFIGURATION

### ★ 1.4.1 Internal Block Diagram



### 1.4.2 Internal Units

#### (1) Bus interface unit (BIU)

Controls the pins of the address bus, data bus, and control bus. A bus cycle activated by the CPU or DMAC is controlled via the WCU, DRAMC, and ROMC.

#### (2) Wait control unit (WCU)

Manages the four blocks corresponding to four chip select signals ( $\overline{CS0}$ – $\overline{CS3}$ ).

This block generates chip select signals, performs wait control, and selects a bus cycle type.

#### (3) DRAM controller (DRAMC)

Generates the  $\overline{RAS}$ ,  $\overline{UCAS}$ , and  $\overline{LCAS}$  signals (2CAS control) and controls access to DRAM.

This block supports DRAM high-speed page mode. Access to DRAM can be of either of two types, each having a different cycle, normal access (off-page) or high-speed page access (on-page).

#### (4) ROM controller (ROMC)

Supports access to ROM supporting a page access function.

Performs address comparison relative to the previous bus cycle and performs wait control for normal access (off-page)/page access (on-page). It supports page widths of 8-64 bytes.

#### (5) Interrupt controller

Handles maskable interrupt requests ( $\overline{INTP00}$ – $\overline{INTP03}$ ,  $\overline{INTP10}$ – $\overline{INTP13}$ ) from both the built-in and external peripheral hardware. Priorities can be specified for these interrupt requests, in units of four groups. It can apply multiple handling control to the interrupt sources.

#### (6) DMA controller (DMAC)

Transfers data between memory and I/O, as instructed by the CPU.

There are two address modes, fly-by (1-cycle) transfer and 2-cycle transfer. There are three bus modes, single transfer, single-step transfer, and block transfer.

#### (7) Serial interfaces (UART/CSI)

As serial interfaces, the V821 features an asynchronous serial interface (UART) and a synchronous serial interface (CSI), one channel being assigned to each.

The UART transfers data via pins TXD and RXD.

The CSI transfers data via pins SO, SI, and  $\overline{SCLK}$ .

Either the baud rate generator or the system clock can be selected as the serial clock source.

#### (8) Real-time pulse unit (RPU)

This block incorporates a 16-bit timer/event counter and a 16-bit interval timer. It can calculate pulse intervals and frequencies and output programmable pulses.

#### (9) Watchdog timer (WDT)

This block incorporates an 8-bit watchdog timer to detect a program hanging up or system errors. If the watchdog timer overflows, the WDTOUT pin becomes active.

**(10) Clock generator (CG)**

Supplies clock pulses at a frequency five times greater than that of the oscillator connected to pins X1 and X2 (when the built-in PLL is being used) or at half the frequency (when the built-in PLL is not being used) of the operating clock pulses for the CPU. Also, instead of connecting an oscillator, external clock pulses can be input.

**(11) Bus arbitration unit (BAU)**

Arbitrates any contention over bus mastership between the bus masters (CPU, DRAMC, DMAC, external bus master). Bus mastership can be switched in each bus cycle and also in the idle state.

**(12) Port**

Port 0 provides a total of ten input/output port pins. The pins can be used as either port or control pins.

[MEMO]



## CHAPTER 2 PIN FUNCTIONS

The pins of the V821 are classed either as port pins or non-port pins, according to their function.

### 2.1 PIN FUNCTIONS

#### 2.1.1 Port Pins

Pin name	Input/output	Function	Dual-function pin
P00	Input/output	Port 0 10-bit input/output port Can be set for input/output bit.	TCLR
P01			DREQ0
P02			$\overline{\text{DACK0}}$
P03			DREQ1
P04			$\overline{\text{DACK1}}$
P05			SI
P06			SO
P07			$\overline{\text{SCLK}}$
P08			TXD/ $\overline{\text{UBE}}$
P09			RXD/ $\overline{\text{TC}}$

## 2.1.2 Non-Port Pins

(1/2)

Pin name	Input/output	Function	Dual-function pin
A0-A23	Tristate output	Address bus signal	—
D0-D15	Tristate input/output	Bidirectional data bus signal	—
$\overline{\text{READY}}$	Input	Bus cycle termination permit signal	—
$\overline{\text{HLDRQ}}$	Input	Bus mastership request signal	—
$\overline{\text{HLDK}}$	Output	Bus mastership permit signal	—
BLOCK	Output	Bus mastership prohibit signal	WDTOUT
$\overline{\text{MRD}}$	Tristate output	Read strobe signal to memory	—
$\overline{\text{LMWR}}$	Tristate output	Write strobe signal to lower data in memory	$\overline{\text{WE}}$
$\overline{\text{UMWR}}$	Tristate output	Write strobe signal to upper data in memory	—
$\overline{\text{IORD}}$	Tristate output	Read strobe signal to I/O data	—
$\overline{\text{IOWR}}$	Tristate output	Write strobe signal to I/O data	—
$\overline{\text{UBE}}$	Tristate output	Data bus upper data enable signal	TXD/P08
$\overline{\text{RESET}}$	Input	System reset input	—
X1, X2	Input	Crystal connection/external clock input	—
CLKOUT	Output	System clock output	—
$\overline{\text{CS0}}$	Tristate output	Chip select signal	$\overline{\text{REFRQ}}$
$\overline{\text{CS1}}$			—
$\overline{\text{CS2}}$			—
$\overline{\text{CS3}}$			—
$\overline{\text{INTP00}}$	Input	Interrupt request input	TO00
$\overline{\text{INTP01}}$			—
$\overline{\text{INTP02}}$			TO01
$\overline{\text{INTP03}}$			—
$\overline{\text{INTP10}}$			—
$\overline{\text{INTP11}}$			—
$\overline{\text{INTP12}}$			—
$\overline{\text{INTP13}}$			TI

(2/2)

Pin name	Input/output	Function	Dual-function pin
$\overline{\text{NMI}}$	Input	Nonmaskable interrupt request input	–
$\overline{\text{REFRQ}}$	Tristate output	Refresh request signal to DRAM	$\overline{\text{CS0}}$
$\overline{\text{RAS}}$	Tristate output	Row address strobe signal to DRAM	–
$\overline{\text{LCAS}}$	Tristate output	Column address strobe signal to lower data in DRAM	–
$\overline{\text{UCAS}}$	Tristate output	Column address strobe signal to upper data in DRAM	–
$\overline{\text{WE}}$	Tristate output	Write strobe signal to DRAM	$\overline{\text{LMWR}}$
DREQ0	Input	DMA request signal (channel 0)	P01
DREQ1	Input	DMA request signal (channel 1)	P03
$\overline{\text{DACK0}}$	Output	DMA permit signal (channel 0)	P02
$\overline{\text{DACK1}}$	Output	DMA permit signal (channel 1)	P04
$\overline{\text{TC}}$	Output	DMA end signal	RXD/P09
TO00	Output	RPU pulse output	$\overline{\text{INTP00}}$
TO01			$\overline{\text{INTP02}}$
TCLR	Input	External clear or start signal input to timer 0	P00
TI	Input	External count clock input to timer 0	$\overline{\text{INTP13}}$
TXD	Output	UART serial data output	$\overline{\text{UBE/P08}}$
RXD	Input	UART serial data input	$\overline{\text{TC/P09}}$
$\overline{\text{SCLK}}$	Input/output	CSI serial clock input/output	P07
SO	Output	CSI serial data output	P06
SI	Input	CSI serial data input	P05
WDTOUT	Output	WDT overflow signal	BLOCK
IC	–	Internal connection (must be connected to GND via a resistor)	–
V <sub>DD</sub>	–	Supplies positive power.	–
GND	–	Ground potential	–

## 2.2 PIN STATES

The table below indicates the state of each pin in each operating state.

Pin \ Operating state	Reset	Bus hold	HALT mode	IDLE mode	STOP mode
CLKOUT	<b>Note 1</b>	<b>Note 1</b>	<b>Note 1</b>	<b>Note 1</b>	1
$\overline{\text{CS0/REFRQ}}$	1	Hi-Z	1	1	1
$\overline{\text{CS1}}$	1	Hi-Z	1	1	1
$\overline{\text{CS2}}$	1	Hi-Z	1	1	1
$\overline{\text{CS3}}$	1	Hi-Z	1	1	1
A0-A23	x	Hi-Z	<b>Note 2</b>	<b>Note 2</b>	<b>Note 2</b>
D0-D15	Hi-Z	Hi-Z	Hi-Z	Hi-Z	Hi-Z
$\overline{\text{MRD}}$	1	Hi-Z	1	1	1
$\overline{\text{LMWR/WE}}$	1	Hi-Z	1	1	1
$\overline{\text{UMWR}}$	1	Hi-Z	1	1	1
$\overline{\text{IORD}}$	1	Hi-Z	1	1	1
$\overline{\text{IOWR}}$	1	Hi-Z	1	1	1
$\overline{\text{HLDAK}}$	1	0	x	x	1
$\overline{\text{RAS}}$	1	Hi-Z	1	0	0
$\overline{\text{LCAS}}$	1	Hi-Z	1	0	0
$\overline{\text{UCAS}}$	1	Hi-Z	1	0	0
BLOCK/WDTOUT	0	0	x	x	0

**Notes 1.** Clock output

**2.** PC of the HALT instruction

**Remark** Hi-Z : High impedance

0 : Low-level output

1 : High-level output

x : Not defined

## 2.3 PIN FUNCTIONS

This section describes the pins according to their functions.

### 2.3.1 Address Bus

#### (1) A0-A23 (Address Bus): tristate output

Output an address signal when the V821 is accessing main external storage or an input/output device. An address space of up to  $2^{24}$  bytes can be accessed. The states of the pins change in sync with the rising edge of a clock pulse of the T1 state in a bus cycle.

### 2.3.2 Data Bus

#### (1) D0-D15 (Data Bus): tristate input/output

Pins for inputting and outputting write and read data when the V821 is accessing main external storage or an input/output device. During a write bus cycle, the user can specify that write data driving start at the falling edge of a clock pulse in the T1 state or at the rising edge of a clock pulse in the T2 state. The data is held until the bus hold state and the rising edge of a clock pulse in the next bus cycle. In a read bus cycle, data is sampled at the rise of the clock pulse following the last T2 state in the cycle.

### 2.3.3 Bus Control Signals

#### (1) $\overline{\text{READY}}$ (Ready): input

Receives the signal used to prolong a bus cycle to align it with memory or I/O access time. For DRAM off-page access and during a CBR refresh cycle, this signal is sampled at the rising edge of the second clock pulse in the T2 state. In another cycle in which the  $\overline{\text{READY}}$  pin is valid, the signal is sampled at the falling edge of a clock pulse in the T2 state.

The setup/hold time for  $\overline{\text{READY}}$  input must always be observed. Otherwise, normal operation cannot be guaranteed.

#### (2) $\overline{\text{HLDRQ}}$ (Hold Request): input

Input pin used to issue a request to the CPU to relinquish bus mastership.

The signal is sampled at the falling edge of a clock pulse in the T2, T1, and TH states. If this pin and the  $\overline{\text{READY}}$  pin are both active at the falling edge of a clock pulse in the T2 state, the CPU changes to the T1 state. Then, if this signal is active at the falling edge of a clock pulse in the T1 state, the CPU changes to the TH state, sets the address, data, and control buses to high impedance, makes the  $\overline{\text{HLDK}}$  signal active, and relinquishes bus mastership.

During a bus clock cycle, the CPU does not change to the TH state.

**(3)  $\overline{\text{HLDAK}}$  (Hold Acknowledge): output**

Outputs the acknowledge signal for  $\overline{\text{HLDRQ}}$  input.

The state of this pin changes at the falling edge of a clock pulse. After relinquishing up bus mastership, the CPU sets this signal to the active state. If  $\overline{\text{HLDRQ}}$  input subsequently becomes inactive, the CPU sets this signal to the inactive state and regains bus mastership.

If, however, a request is issued for a higher priority cycle during bus hold, the  $\overline{\text{HLDAK}}$  signal becomes inactive.

**(4) BLOCK (Bus Lock): output**

Outputs the signal prohibiting use of a bus to a bus user other than the V821 itself.

The signal becomes active at the start of the first bus cycle (rising edge of a clock pulse in the T1 state) and inactive at the rising edge of the clock pulse following the end (last T2 state) of the last bus cycle.

**(5)  $\overline{\text{MRD}}$  (Memory Read): tristate output**

Outputs a strobe signal indicating that the current bus cycle is an external memory read cycle.

The signal always becomes inactive at the rising edge of a clock pulse in the T1 state during a bus cycle.

If the bus cycle is for memory read, it becomes active at the falling edge of a clock pulse in the T1 state.

For the duration of a page-ROM cycle, however, the signal remains active. Also, it remains active for the duration of a refresh cycle.

**(6)  $\overline{\text{LMWR}}$  (Lower Memory Write): tristate output**

Outputs the strobe signal for writing data to external memory.

The lower bytes on the data bus are valid. If the bus cycle is for lower memory write, the signal becomes active at the falling edge of a clock pulse in the T1 state, and inactive at the falling edge of a clock pulse in the last T2 state of the bus cycle.

**(7)  $\overline{\text{UMWR}}$  (Upper Memory Write): tristate output**

Outputs the strobe signal for writing data to external memory.

The upper bytes on the data bus are valid. If the bus cycle is for upper memory write, the signal becomes active at the falling edge of a clock pulse in the T1 state, and inactive at the falling edge of a clock pulse in the last T2 state of the bus cycle.

**(8)  $\overline{\text{IORD}}$  (I/O Read): tristate output**

Outputs the strobe signal indicating that the current bus cycle is a read cycle for external I/O.

This pin always becomes inactive at the rising edge of a clock pulse in the T1 state of a bus cycle. If the bus cycle is for I/O read, it becomes active at the falling edge of a clock pulse in the T1 state.

**(9)  $\overline{\text{IOWR}}$  (I/O Write): tristate output**

Outputs the strobe signal for writing data to external I/O.

If the bus cycle is for I/O write, the signal becomes active at the falling edge of a clock pulse in the T1 state, and inactive at the falling edge of a clock pulse in the last T2 state of the bus cycle.

**(10)  $\overline{\text{UBE}}$  (Upper Byte Enable): tristate output**

Outputs signal indicating access to the upper bytes on the data bus. The state of the pin changes in sync with the rising edge of a clock pulse in the T1 state.

Control over the lower bytes on the data bus is performed using the A0 signal.

Access	$\overline{\text{UBE}}$	A0
Word	0	0
Halfword		
Even byte	1	0
Odd byte	0	1

### 2.3.4 System Control Signals

**(1)  $\overline{\text{RESET}}$  (Reset): input**

This pin is used to initialize the V821.

To initialize the V821, the signal must remain active for at least 30 clock pulses.

When the  $\overline{\text{RESET}}$  signal is accepted, the V821 initializes each signal and internal register and executes instructions starting from address FFFFFFF0H.

**(2) X1, X2 (Crystal Oscillator): input**

Used to connect a crystal oscillator when the internal clock generator is to be used. When using external clock, input its pulses to pin X1.

**(3) CLKOUT (System Clock Out): output**

Outputs internally generated system clock pulses.

**(4)  $\overline{\text{CS0}}$ - $\overline{\text{CS3}}$  (Chip Select): tristate output**

Output chip select signals to the address spaces. The address blocks to which the chip select signals are output are predetermined. When the bus cycle for accessing an address block is activated, the corresponding signal becomes active at the rising edge of a clock pulse in the T1 state during the bus cycle.

However,  $\overline{\text{CS3}}$  remains active for the duration of a page-ROM cycle.

### 2.3.5 Interrupt Control Signals

#### (1) $\overline{\text{INTP00}}\text{--}\overline{\text{INTP03}}$ (Interrupt Request): input

Receive an asynchronous interrupt request signal that has been issued to the interrupt control unit (ICU). Either an edge trigger or level trigger (low level) can be selected.

If the capture/compare register (CC0) of timer 0 (TM0) in the real-time pulse unit (RPU) is specified as a capture register, the value of TM0 is latched using the valid edge (either edge) of this signal as a capture trigger.

#### (2) $\overline{\text{INTP10}}\text{--}\overline{\text{INTP13}}$ (Interrupt Request): input

Receive an asynchronous interrupt request signal that has been issued to the interrupt control unit (ICU). Either an edge trigger or level trigger (low level) can be selected.

#### (3) $\overline{\text{NMI}}$ (Non-maskable Interrupt Request): input

Receives a nonmaskable interrupt signal that has been issued to the CRU.

This signal indicates an interrupt level to the CPU. It is sampled at the falling edge of a clock pulse. This signal must be held at the active level until the CPU starts interrupt handling and posts notification, by software, that the interrupt has been accepted.

### 2.3.6 DRAM Control Signals

#### (1) $\overline{\text{REFRQ}}$ (Refresh Request): tristate output

Outputs a refresh request signal to DRAM.

This pin is used for  $\overline{\text{RAS}}$  control during a refresh cycle when DRAM is to be expanded by decoding the addresses in an external circuit.

The signal becomes active at the rising edge of a clock pulse in the second T2 state of a refresh cycle. Then, if the cycle is a self-refresh cycle, the signal is held active at the falling edge of a clock pulse. In the case of a normal refresh cycle, the signal becomes inactive at the falling edge of a clock pulse in the fourth T2 state.

#### (2) $\overline{\text{RAS}}$ (Row Address Strobe): tristate output

Outputs a row address strobe signal to DRAM.

For DRAM off-Page access, the signal becomes inactive at the falling edge of a clock pulse in the T1 state of a bus cycle, and active at the rising edge of a clock pulse in the second T2 state.

The timing differs for a refresh cycle. (See **Section 7.1.7.**)



**(3)  $\overline{\text{LCAS}}$  (Lower Column Address Strobe): tristate output**

Outputs a column address strobe signal to DRAM.

The lower bytes on the data bus are valid.

For read, if the cycle is an off-page cycle, the signal becomes active at the rising edge of a clock pulse in the third T2 state, and inactive at the rising edge of a clock pulse in the T1 state of the next bus cycle.

If the cycle is an on-page cycle, the signal becomes active at the falling edge of a clock pulse in the T1 state and inactive at the rising edge of a clock pulse in the T1 state of the next bus cycle.

For write, the signal becomes active at the rising edge of a clock pulse in the last T2 state of a bus cycle, and inactive at the rising edge of a clock pulse in the T1 state of the next bus cycle.

The timing differs for a refresh cycle. (See **Section 7.1.7.**)

**(4)  $\overline{\text{UCAS}}$  (Upper Column Address Strobe): tristate output**

Outputs a column address strobe signal to DRAM.

The upper bytes on the data bus are valid.

For read, if the cycle is an off-page cycle, the signal becomes active at the rising edge of a clock pulse in the third T2 state, and inactive at the rising edge of a clock pulse in the T1 state of the next bus cycle.

If the cycle is an on-page cycle, the signal becomes active at the falling edge of a clock pulse in the T1 state, and inactive at the rising edge of a clock pulse in the T1 state of the next bus cycle.

For write, the signal becomes active at the rising edge of a clock pulse in the last T2 state of a bus cycle, and inactive at the rising edge of a clock pulse in the T1 state of the next bus cycle.

The timing differs for a refresh cycle. (See **Section 7.1.7.**)

**(5)  $\overline{\text{WE}}$  (Write Enable): tristate output**

Outputs the signal indicating that the current bus cycle is a DRAM write cycle.

If the bus cycle is for DRAM write, the signal becomes active at the falling edge of a clock pulse in the T1 state, and inactive at the falling edge of a clock pulse in the last T2 state of the bus cycle.

**2.3.7 DMA Control Signals****(1)  $\overline{\text{DREQ0}}$ ,  $\overline{\text{DREQ1}}$  (DMA Request): input**

Receive a DMA service request signal. The pins correspond to DMA channels 0 and 1, respectively, and are independent of each other. Priorities are fixed:  $\text{DREQ1} < \text{DREQ0}$ .

This signal is sampled at the falling edge of the CLKOUT signal. It must be held at the active level until the DMA request is accepted.

**(2)  $\overline{\text{DACK0}}$ ,  $\overline{\text{DACK1}}$  (DMA Acknowledge): output**

Output a signal indicating that a DMA service request can be accepted. The pins correspond to DMA channels 0 and 1, respectively, and are independent of each other.

This signal becomes active at the rising edge of a clock pulse in the T1R or T1F state of a DMA cycle and is held at the active level during DMA transfer.

**(3)  $\overline{TC}$  (Terminal Count): output**

Outputs a signal indicating that DMA transfer using the DMA controller has ended.

This signal becomes active at the rising edge of a clock pulse.

The OR of  $\overline{TC}$  for channel 0 and  $\overline{TC}$  for channel 1 is output. The  $\overline{TC}$  for channel 0 and for channel 1 must, therefore, be generated by ANDing it with each of the  $\overline{DACK0}$  and  $\overline{DACK1}$  signals in an external circuit. (See **Chapter 8**.)

**2.3.8 Real-Time Pulse Control Signals****(1) TO00, TO01 (Timer Output): output**

Output the compare match signal of the capture/compare register (CC0) of timer 0 (TM0) in the real-time pulse unit (RPU). This signal is set upon the detection of a match between the CC00 and CC02 registers. It is reset upon the detection of a match between the CC01 and CC03 registers. The output can be reversed by RPU mode setting.

**(2) TCLR (Timer Clear): input**

Receives a clear signal for TM0 of RPU.

**(3) TI (Timer Input): input**

Receives the external clock signal used by TM0 of RPU. Whether TM0 of RPU is to use this external clock signal or the clock pulses obtained by dividing the internal system clock pulses must be specified at initialization.

**2.3.9 Serial Control Signals****(1) TXD (Transmit Data): output**

Output pin for UART serial transmission data. This signal changes in sync with an internal serial clock pulse. It is held at the high level while transmission is not being performed.

**(2) RXD (Receive Data): input**

Input pin for UART serial receive data.

**(3)  $\overline{SCLK}$  (Serial Clock): input/output**

Input/output pin for CSI serial clock. Can be set to input/output through register setting.

**(4) SO (Serial output): output**

Output pin for CSI serial transmission data. The state of the pin changes in sync with the fall of the  $\overline{SCLK}$  signal. It is set to high impedance while transmission is not being performed.

**(5) SI (Serial Input): input**

Input pin for CSI serial receive data. This signal is sampled at the rising edge of the  $\overline{SCLK}$  signal.

### **2.3.10 Watchdog Timer Control Signal**

#### **(1) WDTOUT (Watchdog Timer Output): output**

Outputs a signal indicating that the watchdog timer (WDT) has overflowed. This signal becomes active if WDT overflows, again becoming inactive once 32 clock pulses have been issued.

### **2.3.11 Port Control Signal**

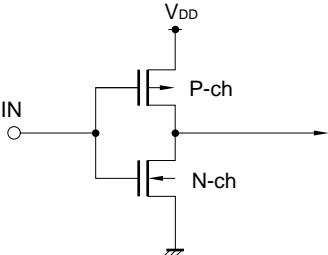
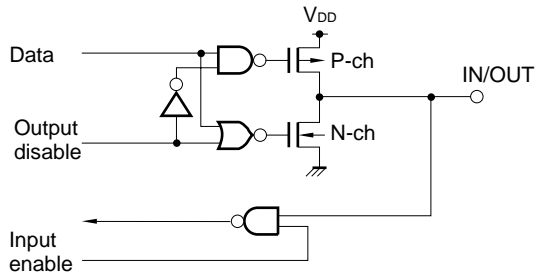
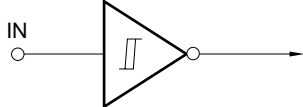
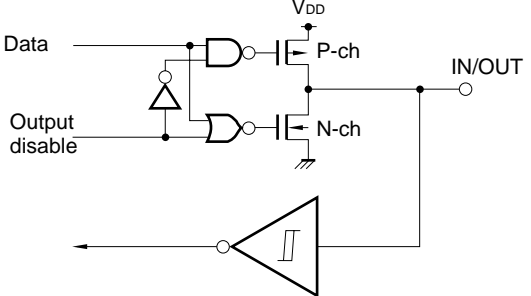
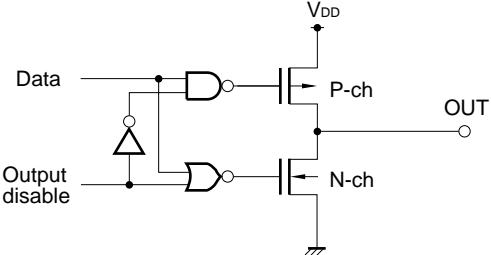
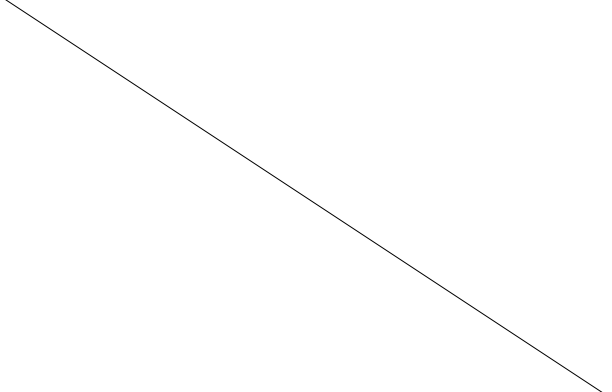
#### **(1) P00-P09 (Port): tristate input/output**

Port signal pins. Can be set to input/output through register setting.

## 2.4 PIN I/O CIRCUITS AND PROCESSING OF UNUSED PINS

Pin	I/O circuit type	Recommended connection
P00/TCLR	5	Input status: Individually connect these pins to the V <sub>DD</sub> pin or GND via a resistor. Output status: Leave these pins open.
P01/DREQ0		
P02/DACK0		
P03/DREQ1		
P04/DACK1		
P05/SI		
P06/SO		
P07/SCLK		
P08/TXD/UBE		
P09/RXD/TC		
D0-D15	5	Leave these pins open.
A0-A7, A16-A18		
A8-A15, A19-A23	4	Leave these pins open.
READY	1	Connect this pin to GND via a resistor.
HLDRQ		Connect this pin to the V <sub>DD</sub> pin via a resistor.
HLDACK	4	Leave these pins open.
BLOCK/WDTOUT		
MRD		
LMWR/WE		
UMWR		
IORD		
IOWR		
CLKOUT		
CS0/REFRQ		
CS1-CS3		
INTP00/TO00	8	Connect this pin to the V <sub>DD</sub> pin via a resistor.
INTP01	2	Connect this pin to the V <sub>DD</sub> pin via a resistor.
INTP02/TO01	8	Connect this pin to the V <sub>DD</sub> pin via a resistor.
INTP03	2	Connect these pins to the V <sub>DD</sub> pin via a resistor.
INTP10-INTP12		
INTP13/TI		
NMI		
RESET		
RAS	4	Leave these pins open.
LCAS		
UCAS		
X2	-	Connect this pin to GND via a resistor.
IC	-	

## 2.5 PIN I/O CIRCUITS

<p>Type 1</p> 	<p>Type 5</p> 
<p>Type 2</p>  <p>Schmitt trigger input with hysteresis characteristics</p>	<p>Type 8</p> 
<p>Type 4</p>  <p>Push-pull output which can output high impedance (Both the positive and negative channels are off.)</p>	

[MEMO]

## CHAPTER 3 CPU FUNCTIONS

The CPU has functions equivalent to those of the V810 microprocessor, designed for built-in control. It offers bit string instructions, floating-point instructions, and quick real-time response.

### 3.1 FEATURES

The features of the CPU are:

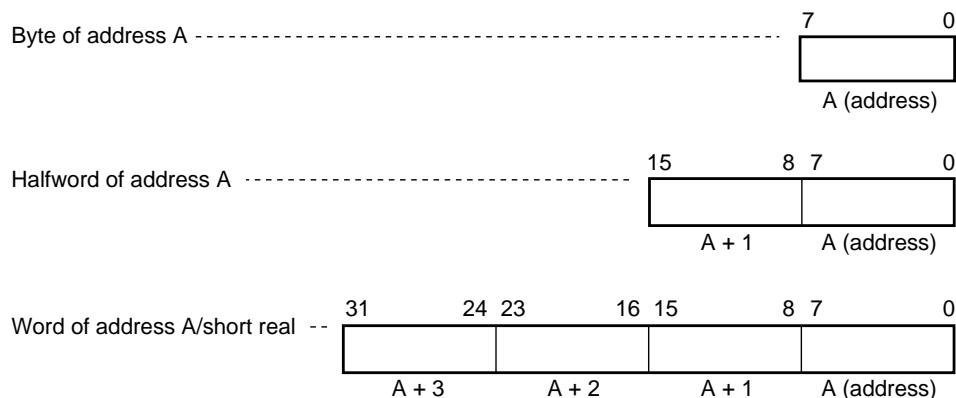
- High-performance 32-bit RISC microprocessor
  - Built-in 1-Kbyte cache memory
  - 1-clock-pulse pitch pipeline structure
  - 16-bit data bus
  - 32-bit general-purpose register x 32
  - 4-Gbyte linear address space
- Instruction groups suitable for a wide range of applications
  - Floating-point instructions (conforming to the IEEE754 data format)
  - Bit string instructions
- High-speed interrupt response
- Debug support functions

### 3.2 ADDRESS SPACE

The V821 supports internal memory and I/O spaces of 4 Gbytes each. The V821 outputs 24-bit addresses to memory and I/O, such that the addresses range from 0 to  $2^{24}-1$ .

In byte data, bit 0 is defined as the LSB (Least Significant Bit) and bit 7 as the MSB (Most Significant Bit). In multiple-byte data, bit 0 of the byte data in the lower address is defined as the LSB and bit 7 of the byte data in the upper address as the MSB, unless noted otherwise.

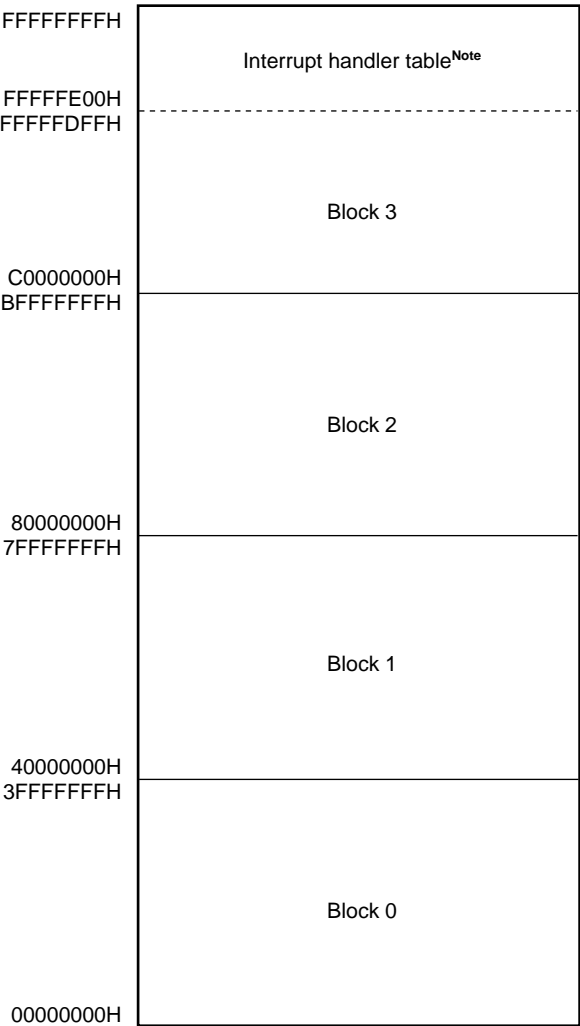
In the case of the V821, 2-byte data is referred to as halfword data, and 4-byte data as word data. In this manual, in representations of multiple-byte memory and I/O data, the right address corresponds to the lower address and the left address to the upper address, as shown below.



3.2.1 Memory Map

Figure 3-1 shows the memory map of the V821.  
The internal 4-Gbyte memory space is divided into blocks of 1 Gbyte each.  
Each block has a linear address space of 16 Mbytes. (The lower 24 bits of a 32-bit address are output.)

Figure 3-1. Memory Map



**Note** See Table 4-1 for details.



### 3.2.2 I/O Map

Figure 3-2 shows the I/O map of the V821.

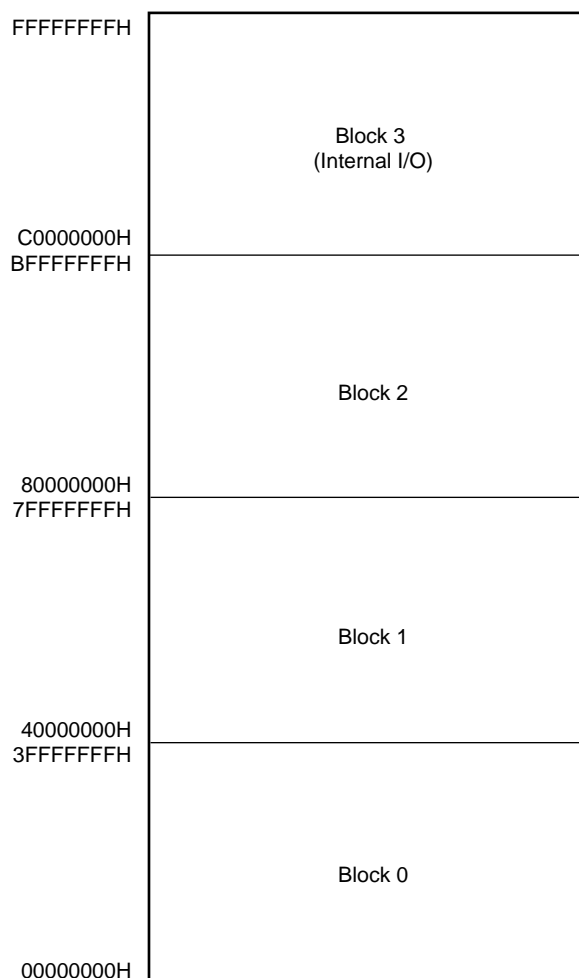
The internal 4-Gbyte memory space is divided into blocks of 1 Gbyte each.

Each block has a linear address space of 16 Mbytes. (The lower 24 bits of a 32-bit address are output.)

The V821 reserves I/O addresses C0000000H-FFFFFFFFH (I/O block 3) as an internal I/O space. Each unit is mapped to this internal I/O space.

See **Section 3.4** for details of the configuration of the internal I/O space.

**Figure 3-2. I/O Map**



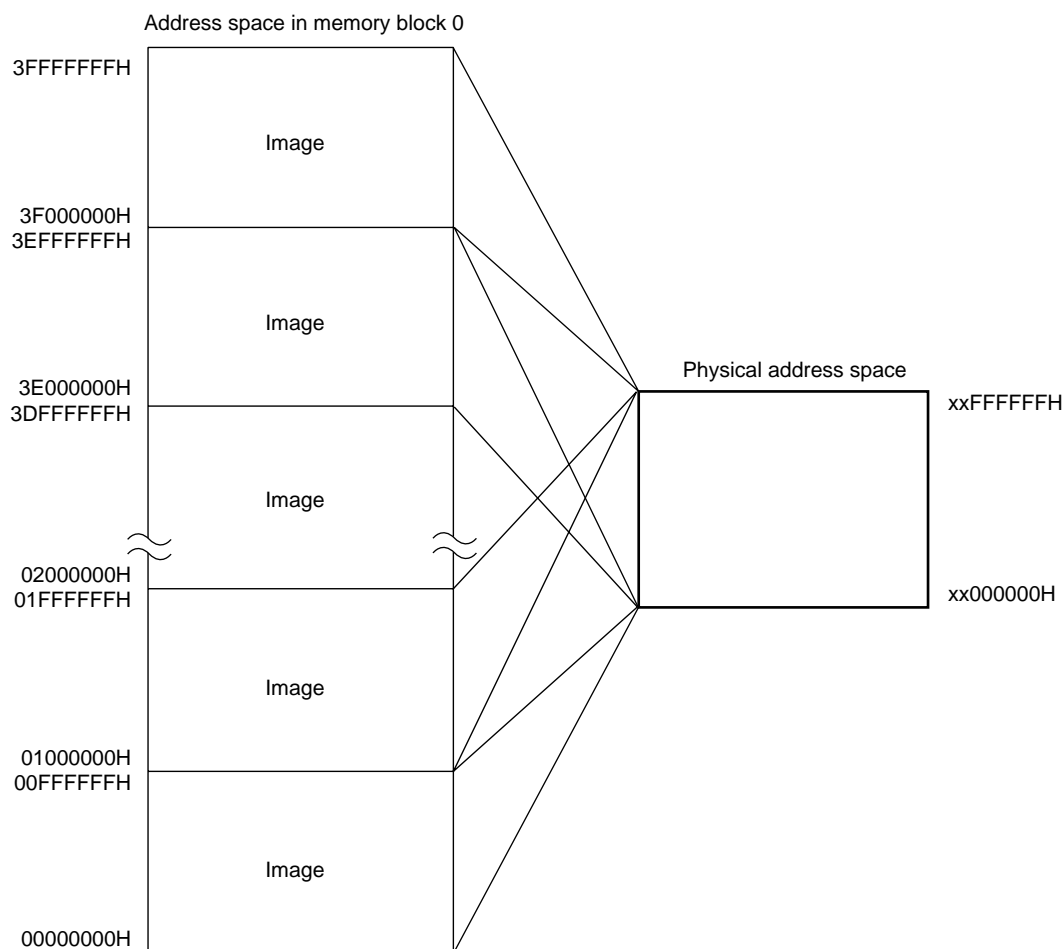
### 3.2.3 Images

The 4-Gbyte memory and I/O spaces are each divided into four blocks of 1 Gbyte each. Each block is selected with the  $\overline{CS}$  signal. (See **Chapter 6**.)

Each block has a linear address space of 16 Mbytes. In one block, a 16-Mbyte physical address space is regarded as being 64 images. Thus, irrespective of the values of bits 29-24 of an address, the same 16-Mbyte physical address space is accessed. Figure 3-3 illustrates images in an address space.

The 32-bit memory address in memory block 0 is used as a 24-bit physical address, the upper 8 bits being ignored. Thus, physical address xx000000H can be regarded not only as memory address 00000000H but also as 01000000H, 02000000H, ..., 3E000000H, and 3F000000H.

Figure 3-3. Images in an Address Space



### 3.2.4 Differences between the Memory and I/O Maps

The memory and I/O maps differ in the control procedure that is applied when access is performed.

If an interrupt occurs during read access (LD instruction) to the memory map, the interrupt takes precedence and the data read by executing the LD instruction may be nullified. The LD instruction may, therefore, need to be reexecuted upon the completion of interrupt handling.

If an interrupt occurs during read access (IN instruction) to the I/O map, the read access takes precedence. The IN instruction will not, therefore, be executed after the completion of interrupt handling.

For the above reason, any devices that are cleared by read operations (such as the read buffer in the serial controller) and those that are updated depending on the read count must be placed in the I/O map, not the memory map.

## 3.3 CPU REGISTER SET

The registers of the V821 belong to one of two sets, the general-purpose program register set and the dedicated system register set. All registers are 32 bits in wide.

### Program register set

31	0
r0	Zero Register
r1	Reserved for Address Generation
r2	Handler Stack Pointer (hp)
r3	Stack Pointer (sp)
r4	Global Pointer (gp)
r5	Text Pointer (tp)
r6	
r7	
r8	
r9	
r10	
r11	
r12	
r13	
r14	
r15	
r16	
r17	
r18	
r19	
r20	
r21	
r22	
r23	
r24	
r25	
r26	String Destination Bit Offset
r27	String Source Bit Offset
r28	String Length
r29	String Destination
r30	String Source
r31	Link Pointer (lp)
31	0
PC	Program Counter

### System register set

31	0
EIPC	Exception/Interrupt PC
EIPSW	Exception/Interrupt PSW
31	0
FEPC	Fatal Error PC
FEPSW	Fatal Error PSW
31	0
ECR	Exception Cause Register
31	0
PSW	Program Status Word
31	0
PIR	Processor ID Register
31	0
TKCW	Task Control Word
31	0
CHCW	Cache Control Word
31	0
ADTRE	Address Trap Register

### 3.3.1 Program Register Set

The program register set includes general-purpose registers and a program counter.

#### (1) General-purpose registers

The V821 has 32 general-purpose registers, r0-r31. These registers can be used for data or address variables.

Registers r0 and r26-r30 are used implicitly with instructions. Caution is therefore necessary when using these registers. Registers r1-r5 and r31 are used implicitly by the assembler and the C compiler. Before using these registers, therefore, the contents of the registers must be saved to prevent their being destroyed. After using the registers, their contents must be restored.

**Table 3-1. Program Registers**

Name	Use	Explanation
r0	Zero register	Always stores zeros.
r1	Assembler-reserved register	Used as a working register to create 32-bit immediate.
r2	Handler stack pointer	Used as a stack pointer for the handler.
r3	Stack pointer	Used to create a stack frame at a function call.
r4	Global pointer	Used to access a global variable in a data area.
r5	Text pointer	Points to the top of a text area
r6-r25	-	Register for an address/data variable
r26	String destination start bit offset	Used to execute a bit string instruction.
r27	String source start bit offset	
r28	String length register	
r29	String destination start address register	
r30	String start address register	
r31	Link pointer	Stores a return point address according to the execution of a JAL instruction.

#### (2) Program counter

Stores the address of an instruction while a program is running. Bit 0 of the program counter (PC) is fixed to 0, thus preventing a branch to an odd address. It is initialized to FFFFFFF0H at reset.

### 3.3.2 System Register Set

System registers are used to control the state of the CPU and store interrupt information.

**Table 3-2. System Register Numbers**

No.	Register name	Use	Explanation
0	EIPC	Registers for saving the current status upon the occurrence of an exception or interrupt	Retain the contents of PC and PSW if an exception or interrupt occurs. Note, however, that there is only one pair of these registers. When multiple interrupts are allowed, therefore, the contents of the registers must be saved by the program.
1	EIPSW		
2	FEPC	Registers for saving the current status upon the occurrence of an NMI or double exception	Retain the contents of PC and PSW if an NMI or double exception occurs.
3	FEPSW		
4	ECR	Exception source register	Stores the source of an exception, maskable interrupt, or NMI. The upper 16 bits of this register are called "FECC" and set to the exception code of an NMI/double exception. The lower 16 bits are called "EICC" and set to the exception code of an exception/interrupt.
5	PSW	Program status word	The program status word is a set of flags indicating the state of the program (result of executing an instruction) and the state of the CPU.
6	PIR	Processor ID register	Used to identify a CPU type number.
7	TKCW	Task control word	Used to control a floating-point operation.
8-23	Reserved		
24	CHCW	Cache control word	Used to control the built-in instruction cache.
25	ADTRE	Address trap register	Stores the address used to detect an address match with PC, and to generate an address trap.
26-31	Reserved		

Read and write operations made to these system registers can be performed using the system register load/store instructions (LDSR and STSR) with the system register numbers specified.

### 3.4 BUILT-IN PERIPHERAL I/O REGISTERS

The built-in peripheral I/O registers are allocated to the 256-byte area between C0000000H and C00000FFH in the 1-Gbyte space between C0000000H and FFFFFFFFH. Starting from address C0000100H, 256-byte images are created every 256 bytes.

The least significant bit of an address is not decoded. Thus, when byte access is attempted to a register at an odd address (2n+1), a register at an even address (2n) is actually performed.

When 16-bit access is attempted to an 8-bit I/O register, the upper eight bits are ignored for write, and become undefined for read.

Table 3-3 lists the built-in peripheral I/O registers.

**Table 3-3. Built-in Peripheral I/O Registers (1/2)**

Address	Function register name	Abbreviation	Manipulatable bits		Initial value
			8-bit	16-bit	
C0000010	Port mode control register 0	PMC0		o	0000H
C0000012	Port mode register 0	PM0		o	03FFH
C0000014	Port register 0	P0		o	Not defined
C0000020	Bus cycle type control register	BCTC	o		01H
C0000022	Programmable wait control register 0	PWC0	o		77H
C0000024	Programmable wait control register 1	PWC1	o		77H
C0000026	Programmable wait control register 2	PWC2	o		77H
C0000028	DRAM configuration register	DRC	o		81H
C000002A	Refresh control register	RFC	o		80H
C000002C	Page-ROM configuration register	PRC	o		80H
C0000040	DMA source address register 0H	DSA0H		o	Not defined
C0000042	DMA source address register 0L	DSA0L		o	Not defined
C0000044	DMA destination address register 0H	DDA0H		o	Not defined
C0000046	DMA destination address register 0L	DDA0L		o	Not defined
C0000048	DMA source address register 1H	DSA1H		o	Not defined
C000004A	DMA source address register 1L	DSA1L		o	Not defined
C000004C	DMA destination address register 1H	DDA1H		o	Not defined
C000004E	DMA destination address register 1L	DDA1L		o	Not defined
C0000050	DMA byte count register 0	DBC0		o	Not defined
C0000052	DMA byte count register 1	DBC1		o	Not defined
C0000054	DMA channel control register 0	DCHC0		o	0000H
C0000056	DMA channel control register 1	DCHC1		o	0000H

**Table 3-3. Built-in Peripheral I/O Registers (2/2)**

Address	Function register name	Abbreviation	Manipulatable bits		Initial value
			8-bit	16-bit	
C0000060	Timer unit mode register 0	TUM0		o	0A00H
C0000062	Timer control register 0	TMC0	o		00H
C0000064	Timer control register 1	TMC1	o		00H
C0000066	Timer output control register 0	TOC0	o		03H
C0000068	Timer overflow status register	TOVS	o		00H
C0000070	Timer register 0	TM0		o	0000H
C0000072	Capture/compare register 00	CC00		o	Not defined
C0000074	Capture/compare register 01	CC01		o	Not defined
C0000076	Capture/compare register 02	CC02		o	Not defined
C0000078	Capture/compare register 03	CC03		o	Not defined
C000007C	Timer register 1	TM1		o	0000H
C000007E	Compare register 1	CM1		o	Not defined
C0000080	Asynchronous serial interface mode register	ASIM	o		00H
C0000082	Asynchronous serial interface status register	ASIS	o		00H
C0000084	Reception buffer	RXB		o	Not defined
C0000086	Reception buffer L	RXBL	o		Not defined
C0000088	Transmission shift register	TXS		o	Not defined
C000008A	Transmission shift register L	TXSL	o		Not defined
C0000090	Synchronous serial interface mode register	CSIM	o		00H
C0000092	Serial I/O shift register	SIO	o		Not defined
C00000A0	Baud rate generator register	BRG	o		Not defined
C00000A2	Baud rate generator prescale mode register	BPRM	o		00H
C00000B0	Interrupt group priority register	IGP	o		E4H
C00000B2	Interrupt clear register	ICR		o	0000H
C00000B4	Interrupt request register	IRR		o	0000H
C00000B6	Interrupt request mask register	IMR		o	FFFFH
C00000B8	ICU mode register	IMOD		o	AAAAH
C00000C0	WDT mode register	WDTM	o		00H
C00000D0	Standby control register	STBC	o		00H
C00000E0	Clock control register	CGC	o		03H

[MEMO]



## CHAPTER 4 INTERRUPT/EXCEPTION HANDLING FUNCTIONS

The V821 features an interrupt controller (ICU) that is dedicated to interrupt handling. The V821 thus supports a powerful interrupt handling function capable of handling interrupt requests issued by up to 16 sources.

As referred to in this manual, an interrupt is an event which occurs independently of program execution while an exception is an event that depends on program execution. In general, an exception assumes a higher priority than an interrupt.

The V821 can handle interrupt requests issued by both built-in peripheral hardware and external devices. Exception handling can be triggered by executing an instruction (TRAP instruction) as well as by the occurrence of an exception (such as an address trap or invalid instruction code).

### 4.1 FEATURES

- Interrupts
  - Nonmaskable interrupt : 1 source
  - Maskable interrupt : 15 sources
  - Programmable priority control with four groups
  - Multiple interrupt handling control according to priority
  - Mask specification for each maskable interrupt request
  - Valid edge specification for external interrupt requests
  - The noise eliminator introducing an analog delay (60 to 300 ns) is incorporated into the nonmaskable interrupt ( $\overline{\text{NMI}}$ ) pin.
- Exceptions
  - Software exception : 32 sources
  - Exception trap : 10 sources

Table 4-1 lists the interrupt and exception sources.

Table 4-1. Interrupts (1/2)

Type	Category	Group	Priority in group	Interrupt/exception source			Exception code	Handler address	Return PC <sup>Note 1</sup>
				Name	Source	Unit			
Reset	Interrupt	-	-	RESET	Reset input	-	FFF0H	FFFFFFF0H	Undefined
Non-maskable	Interrupt	-	-	NMI	NMI input	-	FFD0H	FFFFFFD0H	Next PC <sup>Note 2</sup>
Software exception	Exception	-	-	TRAP1nH	trap instruction	-	FFBnH	FFFFFFB0H	Next PC
		-	-	TRAP0nH	trap exception	-	FFAnH	FFFFFFA0H	Next PC
Exception trap	Exception	-	-	DP-EX	Double exception	-	<b>Note 3</b>	FFFFFFD0H	Current PC
		-	-	AD-TR	Address trap	-	FFC0H	FFFFFFC0H	
		-	-	I-OPC	Invalid instruction code	-	FF90H	FFFFFF90H	
		-	-	DIV0	Division by zero	-	FF80H	FFFFFF80H	
		-	-	FIZ	Invalid floating-point operation	-	FF70H	FFFFFF60H	
		-	-	FZD	Floating-point division by zero	-	FF68H	FFFFFF60H	
		-	-	FOV	Floating-point overflow	-	FF64H	FFFFFF60H	
		-	-	FUD	Floating-point underflow <sup>Note 4</sup>	-	FF62H	FFFFFF60H	
		-	-	FPR	Floating-point degraded precision <sup>Note 4</sup>	-	FF61H	FFFFFF60H	
		-	-	FRO	Floating-point reserved operand	-	FF60H	FFFFFF60H	

**Remark** n = 0H to FH

- Notes**
1. PC value saved in EIPC or FEPC at the beginning of interrupt/exception handling
  2. Return PC = current PC if an interrupt occurred during the execution of an instruction which was stopped by an interrupt (DIV/DIVU, floating-point, and bit string instructions).
  3. The exception code for the exception which occurred first is written into in the 16 low-order bits of ECR, while and that for the second exception is written into the 16 high-order bits.
  4. The V821 is not subject to floating-point underflow or degraded precision exceptions.

**Table 4-1. Interrupts (2/2)**

Type	Category	Group	Priority in group	Interrupt/exception source			Exception code	Handler address	Return PC <sup>Note 1</sup>
				Name	Source	Unit			
Maskable	Interrupt	GR3	3	RESERVED	Reserved	-	FEF0H	FFFFFFEF0H	Next PC <sup>Note 2</sup>
			2	INTOV0	Timer 0 overflow	RPU	FEE0H	FFFFFFEE0H	
			1	INTSER	UART reception error	UART	FED0H	FFFFFFED0H	
			0	INTP13	$\overline{\text{INTP13}}$ pin input	External	FEC0H	FFFFFFEC0H	
		GR2	3	INTSR	UART reception end	UART	FEB0H	FFFFFFEB0H	
			2	INTST	UART transmission end	UART	FEA0H	FFFFFFEA0H	
			1	INTCSI	CSI transmission/reception end	CSI	FE90H	FFFFFFE90H	
			0	INTP12	$\overline{\text{INTP12}}$ pin input	External	FE80H	FFFFFFE80H	
		GR1	3	INTDMA	DMA transfer end	DMAC	FE70H	FFFFFFE70H	
			2	INTP00/ INTCC00	$\overline{\text{INTP00}}$ pin input/CC00 match	External/ RPU	FE60H	FFFFFFE60H	
			1	INTP01/ INTCC01	$\overline{\text{INTP01}}$ pin input/CC01 match	External/ RPU	FE50H	FFFFFFE50H	
			0	INTP11	$\overline{\text{INTP11}}$ pin input	External	FE40H	FFFFFFE40H	
		GR0	3	INTCM1	CM1 match	RPU	FE30H	FFFFFFE30H	
			2	INTP02/ INTCC02	$\overline{\text{INTP02}}$ pin input/CC02 match	External/ RPU	FE20H	FFFFFFE20H	
			1	INTP03/ INTCC03	$\overline{\text{INTP03}}$ pin input/CC03 match	External/ RPU	FE10H	FFFFFFE10H	
			0	INTP10	$\overline{\text{INTP10}}$ pin input	External	FE00H	FFFFFFE00H	

**Notes 1.** PC value saved in EIPC or FEPC at the beginning of interrupt/exception handling

**2.** Return PC = current PC if an interrupt occurred during the execution of an instruction which was stopped by an interrupt (DIV/DIVU, floating-point, and bit string instructions).

**Caution** The exception code and handler address for a maskable interrupt assume the values existing when the default priority is specified. Table 4-2 lists the values in the order of their priorities.

## 4.2 NONMASKABLE INTERRUPTS

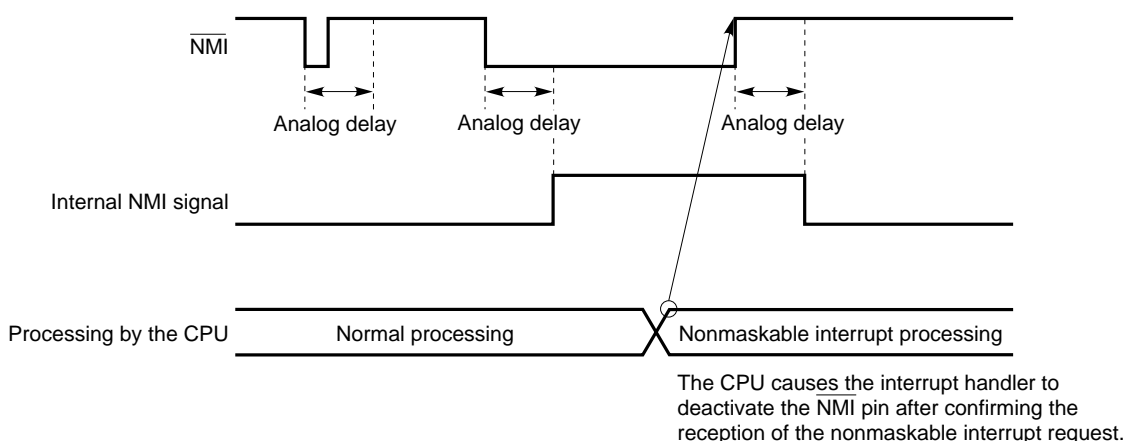
A nonmaskable interrupt request is accepted unconditionally, even in the interrupt disabled (DI) state ( $\text{PSW.ID} = 1$ ). Interrupt priority control is not applied to nonmaskable interrupt requests, which always take the highest priority.

Nonmaskable interrupt requests are input to the  $\overline{\text{NMI}}$  pin. Because  $\overline{\text{NMI}}$  is a level-input pin, the input must be maintained until it can be confirmed that control has branched to the handler and that the request has been accepted.

During the execution of the service program for a nonmaskable interrupt ( $\text{PSW.NP} = 1$ ), nonmaskable interrupt requests are not accepted and instead enter pending status. Pending nonmaskable interrupt requests are accepted only once the execution of the current service program has been completed (after the execution of the RETI instruction) or by setting  $\text{PSW.NP}$  to 0 by using the LDSR instruction.

The noise eliminator introducing an analog delay (60 to 300 ns) is used for input of a nonmaskable interrupt request. Using this noise eliminator prevents the nonmaskable interrupt request from being received mistakenly when noise having a fine pulse width as shown in Figure 4-1 enters.

**Figure 4-1. Noise Elimination of the  $\overline{\text{NMI}}$  Pin Introducing an Analog Delay**

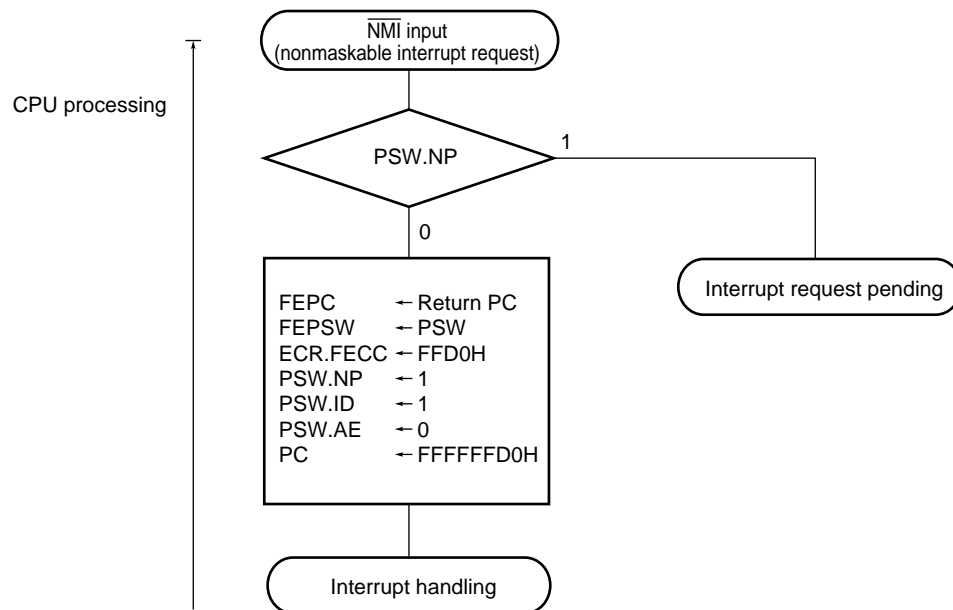


### 4.2.1 Operation

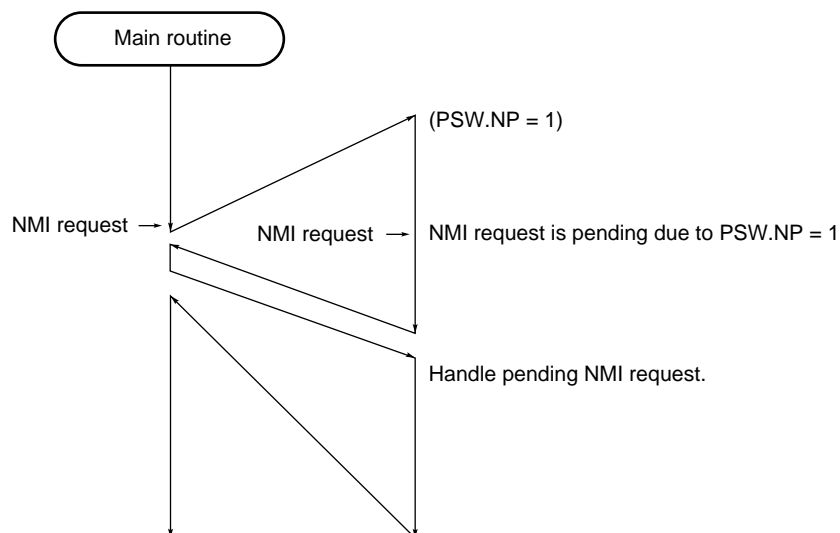
Once a nonmaskable interrupt request has been input to the  $\overline{\text{NMI}}$  pin, the CPU performs the following processing, then passes control to the handler routine.

- (1) Saves the return PC into FEPC.
- (2) Saves the current PSW into FEPSW.
- (3) Writes exception code FFD0H into the 16 high-order bits of ECR (FECC).
- (4) Sets the NP and ID bits of PSW and clears the AE bit.
- (5) Sets the nonmaskable interrupt handler address (FFFFFFD0H) in the PC, thus passing control to the handler routine.

Figure 4-2 illustrates the nonmaskable interrupt handling procedure.

**Figure 4-2. Nonmaskable Interrupt Handling Procedure****Figure 4-3. Accepting a Nonmaskable Interrupt Request**

When a new NMI request is input during the execution of the NMI service program



### 4.3 MASKABLE INTERRUPTS

An interrupt request for which acceptance can be masked (disabled), by setting the corresponding bit of the interrupt request mask register, is called a maskable interrupt request. The V821 supports 15 maskable interrupt sources.

When two or more maskable interrupt requests are issued at the same time, they are handled in the order of their default priorities. In addition to the default priority, a group interrupt priority can be specified for each of the four groups, by using the interrupt group priority register (IGP) (the priority of each interrupt within a group is fixed).

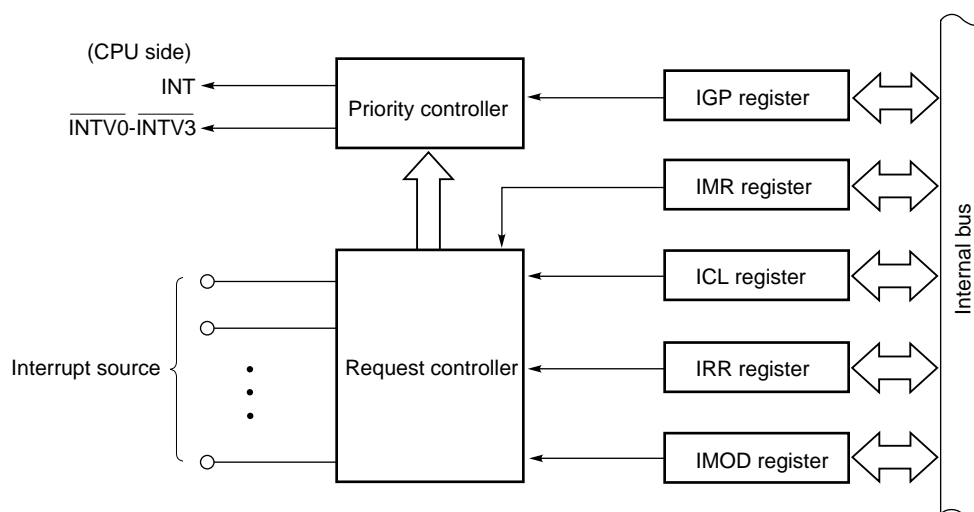
A maskable interrupt is masked according to the logical sum of the NP, EP, and ID bits of PSW. Interrupt level  $n$  for a maskable interrupt is specified by the  $\overline{\text{INTV0}}$  to  $\overline{\text{INTV3}}$  signals and sent to the CPU. The maskable interrupt is not accepted if  $n$  is lower than the interrupt enable level specified in PSW (I0 to I3). An interrupt of the highest interrupt level ( $n = 15$ ), cannot, therefore, be disabled by specifying the interrupt enable level.

Once an interrupt request has been accepted, the system enters the interrupt disabled state ( $\text{PSW.ID} = 1$ ), thus disabling the acceptance of subsequent maskable interrupt requests. Then, the accepted interrupt level ( $n$ ) plus one ( $n + 1$ ) is set in the I0 to I3 bits of PSW.

When enabling multiple interrupts, save the contents of EIPC and EIPSW into memory or registers, then specify the interrupt enabled state ( $\text{PSW.ID} = 0$ ,  $\text{EP} = 0$ ). After the completion of interrupt handling, specify the interrupt disabled state ( $\text{PWS.ID} = 1$ ) before executing the RETI instruction, which restores EIPC and EIPSW.

#### 4.3.1 Block Diagram

Figure 4-4. Maskable Interrupts

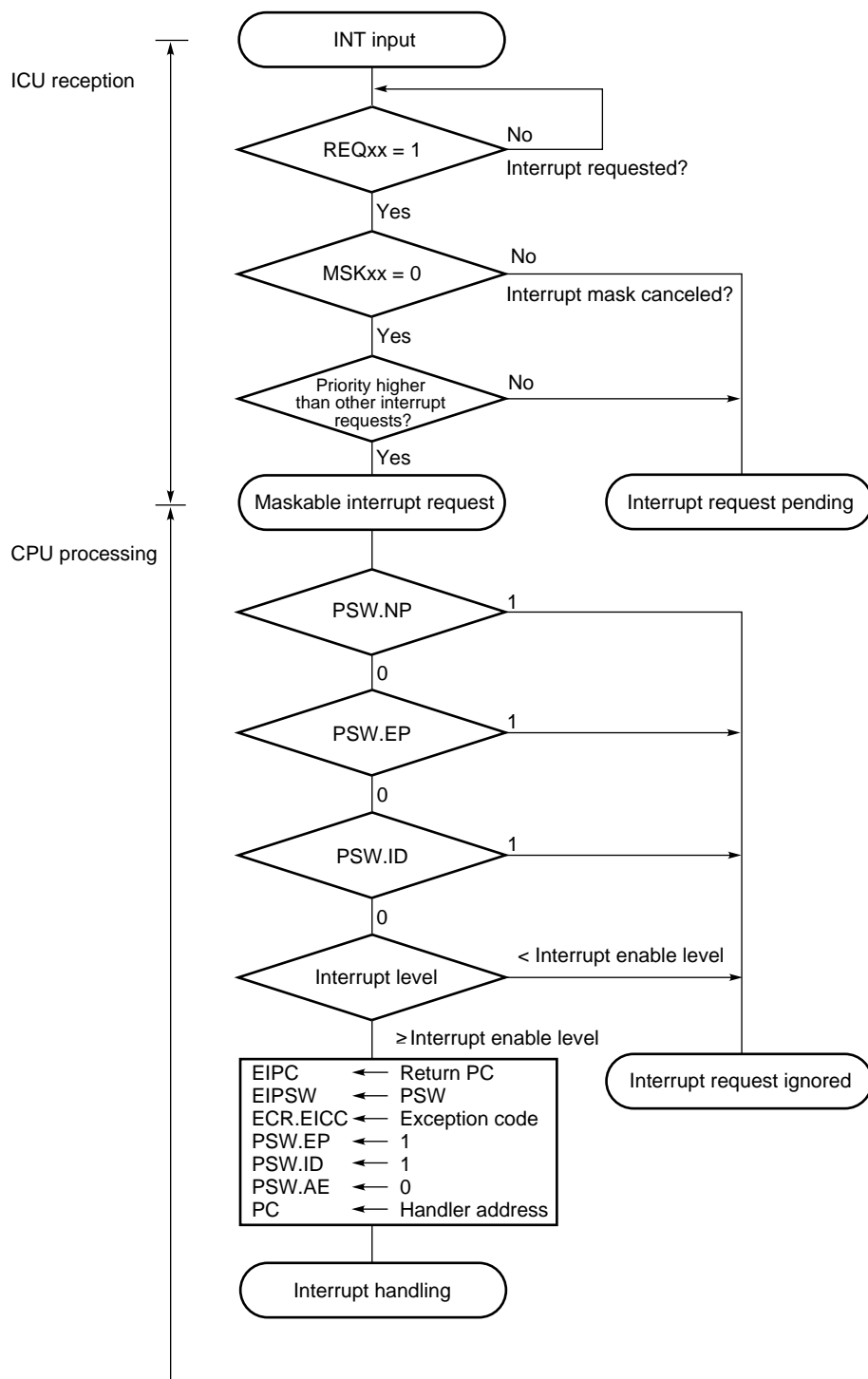


#### 4.3.2 Operation

Once a maskable interrupt has been issued by using an INT input, the CPU performs the following processing, then passes control to the handler routine.

- (1) Saves the return PC into EIPC.
- (2) Saves the current PSW into EIPSW.
- (3) Writes the exception code into the 16 low-order bits of ECR (EICC).
- (4) Sets the EP and ID bits of PSW and clears the AE bit.
- (5) Sets the accepted interrupt level ( $n$ ) plus one ( $n + 1$ ) in the I field (I0 to I3 bits) of PSW. When the accepted interrupt level is of the highest level ( $n = 15$ ), however, sets 15.
- (6) Sets the handler address for the interrupt in the PC, thus passing control to the handler routine.

Figure 4-5 shows the maskable interrupt handling procedure.

**Figure 4-5. Maskable Interrupt Handling Procedure**



### 4.3.3 Maskable Interrupt Priority

While handling one interrupt, the V821 can accept a second interrupt (multiple interrupt handling). Multiple interrupt handling is controlled based on the interrupt priorities.

Priority control is classified into either that based on the default priorities (default priority control) or that based on the settings of the interrupt group priority register (programmable priority control). When default priority control is applied, the V821 handles two or more simultaneously requested interrupts according to the default priority of each interrupt request group (a group consists of four interrupt requests) (see **Table 4-1**). Programmable priority control is applied by specifying a priority (0, 1, 2, or 3) for each interrupt request group, using the interrupt group priority control register. The priority of each interrupt request within a group is, however, fixed.

Once an interrupt request has been accepted, the ID and EP flags of PSW are automatically set to 1. Therefore, when performing multiple interrupt handling, enable interrupts (by specifying PSW.ID = 0 and PSW.EP = 0) in the interrupt handling program.

Maskable interrupt or exception service program

```

...
...
• Save EIPC into memory or a register.
• Save EIPSW into memory or a register.
• Enable interrupt acceptance (PSW.ID = 0, PSW.EP = 0).
...
...
...
...
• Disable interrupt acceptance (PSW.ID = 1, PSW.EP = 1).
• Restore EIPSW.
• Restore EIPC.
• Execute the RETI instruction.

```

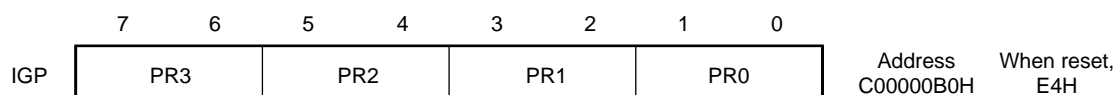
<- Accept an interrupt, such as an INTP input.

## 4.4 CONTROL REGISTERS

### 4.4.1 Interrupt Group Priority Register (IGP)

This register is used to specify the priority of each interrupt request group. **Table 4-2** lists the correspondence between each priority and the exception code and handler address of each interrupt in the group (when the IGP register is set to 4EH).

This register can be read/written in 8-bit units.



Bit position	Bit name	Description															
7-0	PR3-PR0	<p>Group Priority Specifies the priority for interrupt group n (GRn). 0 is the lowest priority, while 3 is the highest.</p> <table border="1" style="margin: 10px auto; border-collapse: collapse;"> <thead> <tr> <th colspan="2">PRn</th><th>Priority</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>0</td></tr> <tr> <td>0</td><td>1</td><td>1</td></tr> <tr> <td>1</td><td>0</td><td>2</td></tr> <tr> <td>1</td><td>1</td><td>3</td></tr> </tbody> </table> <p>n = 0-3</p>	PRn		Priority	0	0	0	0	1	1	1	0	2	1	1	3
PRn		Priority															
0	0	0															
0	1	1															
1	0	2															
1	1	3															

**Caution** Do not specify the same priority for multiple groups. Otherwise, interrupt handling will be unpredictable.

**Table 4-2. Correspondence between Priorities, Exception Codes, Handler Addresses, and Interrupt Levels (When the IGP Register Is Set to 4EH)**

IGP register setting		Group	Priority of the group	Priority within the group (fixed)	Name of the interrupt/exception source	Exception code	Handler address	Interrupt level
Bit	Value							
PR1	11	GR1	3	3	INTDMA	FEF0H	FFFFFFEF0H	15 (Highest)
				2	INTP00/INTCC00	FEE0H	FFFFFFE0H	14
				1	INTP01/INTCC01	FED0H	FFFFFFED0H	13
				0	INTP11	FEC0H	FFFFFFEC0H	12
PR0	10	GR0	2	3	INTCM1	FEB0H	FFFFFFEB0H	11
				2	INTP02/INTCC02	FEA0H	FFFFFFEA0H	10
				1	INTP03/INTCC03	FE90H	FFFFFFE90H	9
				0	INTP10	FE80H	FFFFFFE80H	8
PR3	01	GR3	1	3	RESERVED	FE70H	FFFFFFE70H	7
				2	INTOV0	FE60H	FFFFFFE60H	6
				1	INTSER	FE50H	FFFFFFE50H	5
				0	INTP13	FE40H	FFFFFFE40H	4
PR2	00	GR2	0	3	INTSR	FE30H	FFFFFFE30H	3
				2	INTST	FE20H	FFFFFFE20H	2
				1	INTCSI	FE10H	FFFFFFE10H	1
				0	INTP12	FE00H	FFFFFFE00H	0 (Lowest)

#### 4.4.2 Interrupt Clear Register (ICR)

This register is used to clear interrupt requests. **Table 4-3** lists the correspondence between each bit and interrupt request. This register can be written in 16-bit units.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
ICR	0	CLR14	CLR13	CLR12	CLR11	CLR10	CLR9	CLR8	CLR7	CLR6	CLR5	CLR4	CLR3	CLR2	CLR1	CLR0	Address When reset, C00000B2H 0000H

Bit position	Bit name	Description
14-0	CLR14-CLR0	<p>Clear interrupt request</p> <p>Setting a bit of this register to 1 clears the corresponding interrupt request (the REQn bit of the IRR register). This register, however, cannot be used to clear interrupt requests in level trigger mode.</p> <p>This is a write-only register. Attempting to read this register always results in 0 being read, regardless of its actual value.</p> <p>0: No operation</p> <p>1: Clears the REQn bit of the IRR register.</p>

#### 4.4.3 Interrupt Request Register (IRR)

This register holds interrupt requests. **Table 4-3** indicates the correspondence between each bit and the interrupt requests. This register can be read in 16-bit units.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
IRR	0	REQ14	REQ13	REQ12	REQ11	REQ10	REQ9	REQ8	REQ7	REQ6	REQ5	REQ4	REQ3	REQ2	REQ1	REQ0	Address When reset, C00000B4H 0000H

Bit position	Bit name	Description
14-0	REQ14-REQ0	<p>Interrupt Request</p> <p>Once an interrupt request has been issued, the corresponding bit of this register is set, regardless of the mask register setting. For edge-detected interrupts, manipulating the ICR register resets the relevant bits.</p> <p>0: The corresponding interrupt request has not been issued.</p> <p>1: The corresponding interrupt request has been issued.</p>

**4.4.4 Interrupt Request Mask Register (IMR)**

This register is used to mask interrupt requests. **Table 4-3** indicates the correspondence between each bit and the interrupt requests. This register can be read/written in 16-bit units.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
IMR	0	MSK14	MSK13	MSK12	MSK11	MSK10	MSK9	MSK8	MSK7	MSK6	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0	Address When reset, C00000B6H FFFFH

Bit position	Bit name	Description
14-0	MSK14-MSK0	Mask interrupt request Setting a bit of this register to 1 masks the corresponding interrupt request. An interrupt request can be masked only before it is issued. 0: Does not mask the corresponding interrupt request. 1: Masks the corresponding interrupt request.

**Table 4-3. Correspondence between Interrupt Control Register Bits and Interrupt Request Signals**

ICR register bit	IRR register bit	IMR register bit	Interrupt request signal
CLR14	REQ14	MSK14	INTOV0
CLR13	REQ13	MSK13	INTSER
CLR12	REQ12	MSK12	INTP13
CLR11	REQ11	MSK11	INTSR
CLR10	REQ10	MSK10	INTST
CLR9	REQ9	MSK9	INTCSI
CLR8	REQ8	MSK8	INTP12
CLR7	REQ7	MSK7	INTDMA
CLR6	REQ6	MSK6	INTP00/INTCC00
CLR5	REQ5	MSK5	INTP01/INTCC01
CLR4	REQ4	MSK4	INTP11
CLR3	REQ3	MSK3	INTCM1
CLR2	REQ2	MSK2	INTP02/INTCC02
CLR1	REQ1	MSK1	INTP03/INTCC03
CLR0	REQ0	MSK0	INTP10

#### 4.4.5 ICU Mode Register (IMOD)

This register is used to specify the trigger mode for external interrupt requests INTP00 to INTP03 and INTP10 to INTP13, each of which is input to the pin having the same name. There are two trigger modes: level trigger and edge trigger.

##### (1) Level trigger

Interrupts are sampled at each clock. An interrupt request signal must remain at the active level until the request is recognized by the interrupt handler of the CPU. If an interrupt request is canceled before it can be recognized, an error may occur, such as an undefined branch destination vector.

##### (2) Edge trigger

Interrupts are sampled at the rising edge of each clock. When an interrupt request signal changes at a clock rising edge, as specified with the IMOD register, the request is accepted. If identical interrupt requests are input consecutively, only one is recognized.

Internal interrupts are also edge-detected. To use external interrupt requests, clear the corresponding bits of the interrupt request register (IRR) in the interrupt handling routine.

This register can be read/written in 16-bit units.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
IMOD	ITM13	ITM12	ITM11	ITM10	ITM03	ITM02	ITM01	ITM00									
	Address C00000B8H																When reset, AAAAH

Bit position	Bit name	Description															
15-0	ITM13-ITM10 ITM03-ITM00	<p>Interrupt Trigger Mode Specifies the trigger mode for the INTPn pin. The trigger mode can be changed only before input of the relevant interrupt request.</p> <table border="1"> <tr> <th colspan="2">ITMn</th><th>Trigger mode</th></tr> <tr> <td>0</td><td>0</td><td>Level trigger</td></tr> <tr> <td>0</td><td>1</td><td>Falling edge trigger</td></tr> <tr> <td>1</td><td>0</td><td>Rising edge trigger</td></tr> <tr> <td>1</td><td>1</td><td>Both edge trigger</td></tr> </table> <p>Do not change the settings of ITM03 to ITM00 during timer 0 operation.</p>	ITMn		Trigger mode	0	0	Level trigger	0	1	Falling edge trigger	1	0	Rising edge trigger	1	1	Both edge trigger
ITMn		Trigger mode															
0	0	Level trigger															
0	1	Falling edge trigger															
1	0	Rising edge trigger															
1	1	Both edge trigger															

#### 4.5 EXCEPTION HANDLING (SOFTWARE EXCEPTION AND EXCEPTION TRAP)

A software exception occurs when the CPU executes the TRAP instruction. The format of the TRAP instruction is as follows ("vector" is 0 to 1FH):

TRAP vector

In addition, the following exception traps may occur:

- Double exception : Occurs upon the occurrence of an exception during the handling of another exception (PSW.EP = 1).
- Address trap : Occurs once the PC has assumed the address specified with the address trap register.
- Invalid instruction code : Occurs when the instruction to be executed next is found to be invalid.
- Division by zero : Occurs upon the occurrence of a division by zero error.
- Invalid floating-point operation : Occurs if an invalid floating-point operation is performed.
- Floating-point division by zero : Occurs upon the occurrence of a division by zero error during a floating-point operation.
- Floating-point overflow : Occurs upon the occurrence of an overflow during a floating-point operation.
- Floating-point underflow : Occurs upon the occurrence of an underflow during a floating-point operation.
- Floating-point degraded precision : Occurs upon the occurrence of degraded precision during a floating-point operation.
- Floating-point reserved operand : Occurs upon the detection of a reserved operand (infinity or non-numeric) during a floating-point operation.

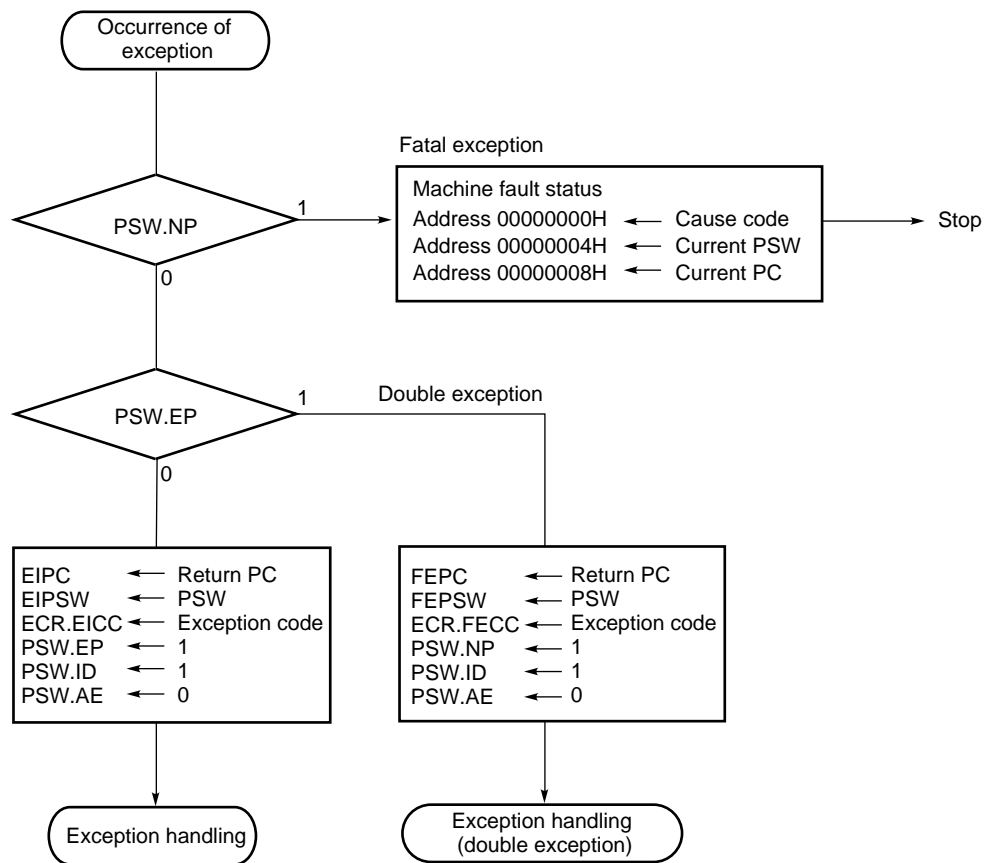
The V821 is not subject to floating-point underflow and degraded precision exceptions.

#### 4.5.1 Operation

Upon the occurrence of an exception, the CPU performs the following processing, then passes control to the handler routine.

- (1) If NP of PSW has been set, proceeds to step (8).
- (2) If EP of PSW has been set, proceeds to step (9).
- (3) Saves the return PC into EIPC.
- (4) Saves the current PSW into EIPSW.
- (5) Writes the exception code into the 16 low-order bits of ECR (EICC).
- (6) Sets the EP and ID bits of PSW and clears the AE bit.
- (7) Sets the handler address for the exception in the PC, thus passing control to the handler routine.
- (8) For a fatal exception
  - (a) Starts a machine fault cycle, then sequentially outputs the following data to the data bus: cause code for the fatal exception (logical sum of FFFF0000H and the exception code) to address 00000000H, the current PSW to address 00000004H, and the current PC to address 00000008H.
  - (b) Suspends operation until the system is reset.
- (9) For a double exception
  - (a) Saves the return PC into FEPC.
  - (b) Saves the current PSW into FEPSW.
  - (c) Writes the exception code for the exception which caused the double exception, into the 16 high-order bits of ECR (FECC).
  - (d) Sets the NP and ID bits of PSW and clears the AE bit.
  - (e) Sets the double exception handler address (FFFFFFD0H) in the PC, thus passing control to the handler routine.

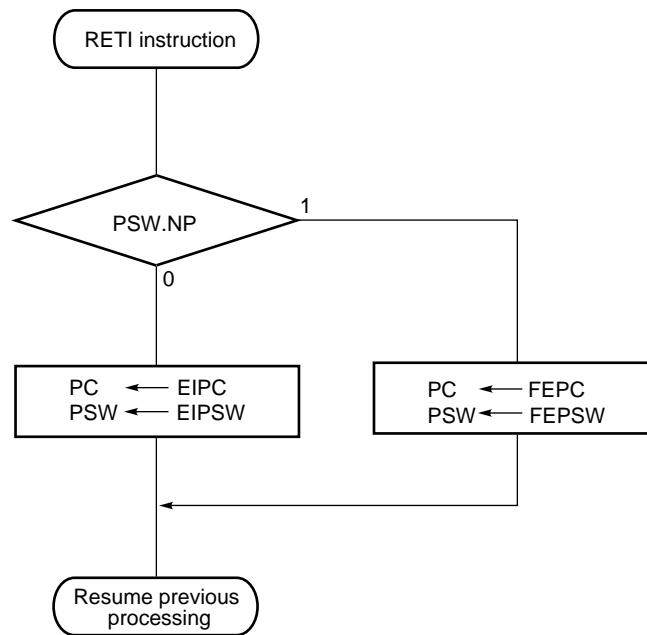




#### 4.5.2 Return from an Exception or Interrupt

The RETI instruction is used to perform return from an exception or interrupt, except in the case of a fatal exception.

- (1) Fetches the return PC and PSW from FEPC and FEPSW (when NP of PSW is set to 1) or EIPC and EIPSW (when NP is set to 0).
- (2) Writes the fetched values into the PC and PSW, thus passing control to the address indicated by the PC.



## 4.6 PRIORITY CONTROL

### 4.6.1 Priorities of Interrupts and Exceptions

The following table lists the priorities of interrupts and exceptions. If two or more interrupts and/or exceptions occur at the same time, they are handled according to their priorities.

**Table 4-4. Priorities of Interrupts and Exceptions**

	RESET	NMI	INT	AD-TR	TRAP	I-OPC	DIV0	FLOAT
RESET		*	*	*	*	*	*	*
NMI	x		←	←	←	←	←	←
INT	x	↑		←	←	←	←	←
AD-TR	x	↑	↑		←	←	←	←
TRAP	x	↑	↑	↑		—	—	—
I-OPC	x	↑	↑	↑	—		—	—
DIV0	x	↑	↑	↑	—	—		—
FLOAT	x	↑	↑	↑	—	—	—	

RESET : Reset

NMI : Nonmaskable interrupt

INT : Maskable interrupt

AD-TR : Address trap

TRAP : Trap instruction

I-OPC : Invalid instruction code

DIV0 : Division by zero

FLOAT : Floating-point exception (invalid operation, division by zero, overflow, and reserved operand)

\* : The item on the left causes the item at the top to be ignored.

x : The item at the top causes the item on the left to be ignored.

— : The item on the left and that at the top cannot occur at the same time.

← : The item on the left has a higher priority than the item at the top.

↑ : The item at the top has a higher priority than the item on the left.

#### 4.6.2 Priorities of Floating-Point Exceptions

The following table lists the priorities of floating-point exceptions:

**Table 4-5. Priorities of Floating-Point Exceptions**

	FRO	FIV	FZD	FOV	FUD	FPR
FRO		*	*	*	—	—
FIV	x		*	*	—	—
FZD	x	x		*	—	—
FOV	x	x	x		—	—
FUD	—	—	—	—		—
FPR	—	—	—	—	—	

FRO : Floating-point reserved operand

FIV : Invalid floating-point operation

FZD : Floating-point division by zero

FOV : Floating-point overflow

FUD : Floating-point underflow

FPR : Floating-point degraded precision

\* : The item on the left causes the item at the top to be ignored.

x : The item at the top causes the item on the left to be ignored.

— : The item on the left and that at the top cannot occur at the same time.

**Remark** The V821 is not subject to FUD and FPR.

## CHAPTER 5 BUS CONTROL FUNCTION

The V821 interfaces with external memory and external I/O devices via the external bus.

### 5.1 CPU BUS STATES

The V821 bus cycles can assume the following eight states:

#### (1) TI and TIS states

When no access request has been issued, or when the TH or THS state (hold state) is released, the bus enters the TI or TIS state. At the falling edge of the clock signal in the TI and TIS states, the  $\overline{\text{HLDRQ}}$  signal is sampled.

#### (2) T1 and T1S states

The bus enters the T1 or T1S state at the start of a bus cycle. In these states, an address is output to the bus at the rising edge of the clock pulse, and valid data is output to the data bus at the falling edge of the clock pulse. After entering the T1 or T1S state, the bus invariably enters the T2 or T2S state.

#### (3) T2 and T2S states

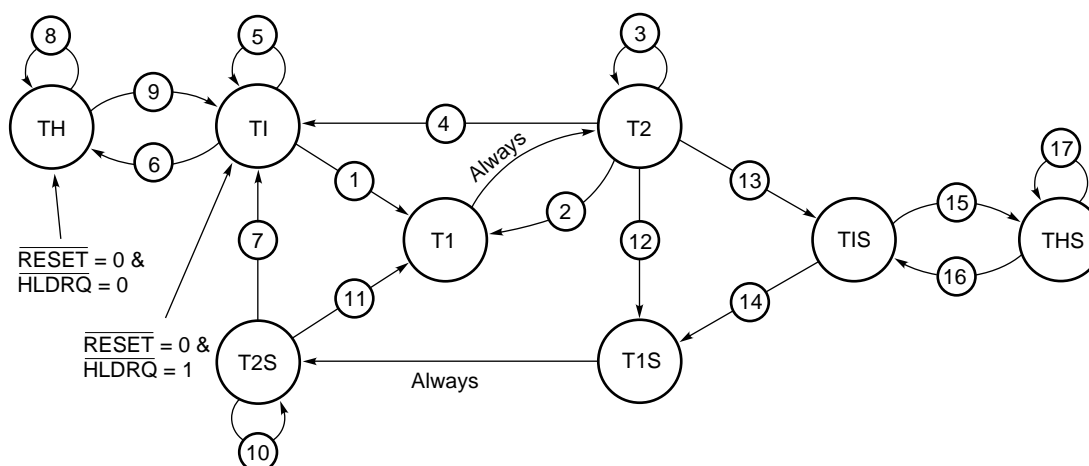
The T2 and T2S states correspond to the final state of a bus cycle or a wait state. At the falling edge of the clock signal in the T2 and T2S states, the  $\overline{\text{HLDRQ}}$  signal is sampled. At the rising edge of the clock pulse that occurs immediately after the last T2 and T2S states, read data is sampled.

#### (4) TH and THS states

The bus enters the TH or THS state when the hold state is caused by an  $\overline{\text{HLDRQ}}$  input. In these states, the  $\overline{\text{HLDRQ}}$  signal is sampled at the falling edge of the clock signal. If sampling indicates that the  $\overline{\text{HLDRQ}}$  signal has become inactive, the bus then enters the TI or TIS state.

**Remark** The TIS, T1S, T2S, and THS states are assumed by additional bus cycles which occur only when word accesses are performed. The TIS, T1S, T2S, and THS states correspond to the TI, T1, T2, and TH states of normal bus cycles, respectively.

Figure 5-1. CPU Bus Cycle State Transition



**Remark** When the  $\overline{\text{RESET}}$  input becomes active, the bus enters the T1 or TH state according to the  $\overline{\text{HLDRQ}}$  signal status.

#### Conditions for bus cycle state transition

- <1>  $\overline{\text{HLDRQ}} = 1$  or hold disable state) and access source present
- <2>  $\overline{\text{READY}} = 0$  and (non-word access) and access source present and ( $\overline{\text{HLDRQ}} = 1$  or hold disable state)
- <3>  $\overline{\text{READY}} = 1$
- <4>  $\overline{\text{READY}} = 0$  and (non-word access) and (access source not present or ( $\overline{\text{HLDRQ}} = 0$  and hold enable state))
- <5> ( $\overline{\text{HLDRQ}} = 1$  or hold disable state) and access source not present
- <6>  $\overline{\text{HLDRQ}} = 0$  and hold enable state
- <7>  $\overline{\text{READY}} = 0$  and (access source not present or ( $\overline{\text{HLDRQ}} = 0$  and hold enable state))
- <8>  $\overline{\text{HLDRQ}} = 0$
- <9>  $\overline{\text{HLDRQ}} = 1$
- <10>  $\overline{\text{READY}} = 1$
- <11>  $\overline{\text{READY}} = 0$  and access source present and ( $\overline{\text{HLDRQ}} = 1$  or hold disable state)
- <12>  $\overline{\text{READY}} = 0$  and word access and ( $\overline{\text{HLDRQ}} = 1$  or hold disable state)
- <13>  $\overline{\text{READY}} = 0$  and word access and  $\overline{\text{HLDRQ}} = 0$  and hold enable state
- <14>  $\overline{\text{HLDRQ}} = 1$
- <15>  $\overline{\text{HLDRQ}} = 0$
- <16>  $\overline{\text{HLDRQ}} = 1$
- <17>  $\overline{\text{HLDRQ}} = 0$

**Remark** After the BLOCK signal becomes active, the hold disable state continues until the bus enters the last bus cycle of the BLOCK cycle (the last T2 state or, when an additional bus cycle exists, the last T2S state). (During the last bus cycle, the hold enable state exists.)

## 5.2 DMAC BUS STATE

The DMAC bus cycles can assume the following 10 states:

### (1) T1 state

The T1 state is the idle state, during which no access request is issued.

The DREQ signal is sampled at the falling edge of the clock pulse.

### (2) T0 state

DMA transfer ready state. (A DMA transfer request has been issued, causing the bus to be acquired for the first DMA transfer.)

### (3) T1R state

The bus enters the T1R state at the beginning of a read operation in two-cycle transfer mode.

After entering the T1R state, the bus invariably enters the T2R state. Address driving starts.

### (4) T2R state

The T2R state corresponds to the last state of a read operation in two-cycle transfer mode, or to a wait state.

In the last T2R state, read data is sampled. After entering the last T2R state, the bus invariably enters the T1W state.

### (5) T1W state

The bus enters the T1W state at the beginning of a write operation in two-cycle transfer mode.

After entering the T1W state, the bus invariably enters the T2W state. Address driving starts.

### (6) T2W state

The T2W state corresponds to the last state of a write operation in two-cycle transfer mode, or to a wait state.

In the last T2W state, the write strobe signal is made inactive.

### (7) T1F state

The bus enters the T1F state at the beginning of a fly-by transfer.

After entering the T1F state, the bus invariably enters the T2F state. Address driving starts.

### (8) T2F state

The T2F state corresponds to the last state of a fly-by transfer or a wait state.

In the last T2F state, the write strobe signal is made inactive.

### (9) T3 state

The bus enters the T3 state when a DMA transfer has been completed, and the bus has been released.

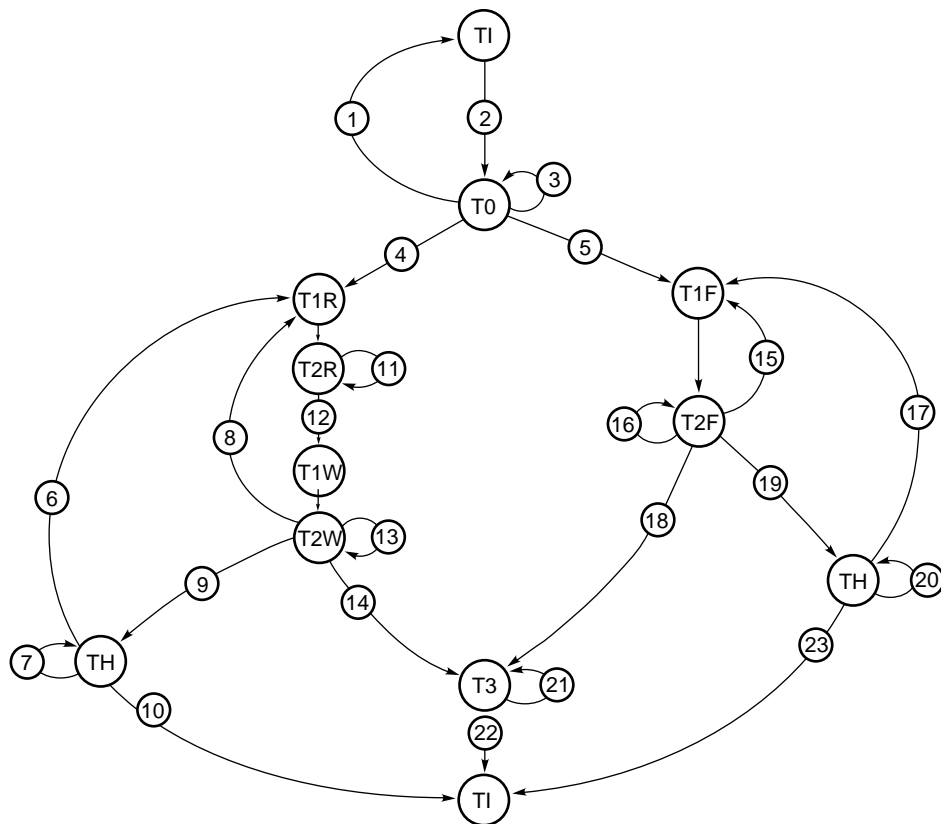
After entering the T3 state, the bus invariably enters the T1 state.

**(10) TH state**

From the TH state, the bus enters the DMA hold state (when a bus request is made by a bus master having a higher priority during DMA block transfer).

Each time the processing for a DMA service is completed, the bus is released (the bus enters bus release mode).

**Figure 5-2. DMAC Bus Cycle State Transition**





## Conditions governing bus cycle state transition

- <1>  $\overline{\text{NMI}} = 0$
- <2> (External terminal  $\text{DREQ} = 1$  or software  $\text{DREQ} = 1$  or internal I/O  $\text{DREQ} = 1$ ), transfer not terminated, DMA enable and  $\overline{\text{NMI}} = 1$
- <3>  $\overline{\text{DHLD\text{AK}}} = 1$
- <4>  $\overline{\text{DHLD\text{AK}}} = 0$  and two-cycle transfer  $\overline{\text{NMI}} = 1$
- <5>  $\overline{\text{DHLD\text{AK}}} = 1$  and fly-by transfer  $\overline{\text{NMI}} = 1$
- <6>  $\overline{\text{NMI}} = 1$  and  $\overline{\text{DHLD\text{AK}}} = 0$ , transfer not terminated and block transfer
- <7>  $\overline{\text{DHLD\text{AK}}} = 1$
- <8>  $\overline{\text{NMI}} = 1$  and  $\overline{\text{DREADY}} = 0$  and  $\overline{\text{DHLD\text{AK}}} = 0$ , transfer not terminated, and block transfer
- <9>  $\overline{\text{NMI}} = 1$  and  $\overline{\text{DREADY}} = 0$  and  $\overline{\text{DHLD\text{AK}}} = 1$ , transfer not terminated and block transfer
- <10>  $\overline{\text{NMI}} = 0$
- <11>  $\overline{\text{DREADY}} = 1$
- <12>  $\overline{\text{DREADY}} = 0$
- <13>  $\overline{\text{DREADY}} = 1$
- <14>  $\overline{\text{DREADY}} = 0$  and ( $\overline{\text{NMI}} = 0$  or transfer terminated or non-block transfer)
- <15>  $\overline{\text{NMI}} = 1$  and  $\overline{\text{DREADY}} = 0$  and  $\overline{\text{DHLD\text{AK}}} = 0$ , transfer not terminated and block transfer
- <16>  $\overline{\text{DREADY}} = 1$
- <17>  $\overline{\text{NMI}} = 1$  and  $\overline{\text{DHLD\text{AK}}} = 0$ , transfer not terminated and block transfer
- <18>  $\overline{\text{DREADY}} = 0$  and ( $\overline{\text{NMI}} = 0$  or transfer terminated or non-block transfer)
- <19>  $\overline{\text{NMI}} = 1$  and  $\overline{\text{DREADY}} = 0$ ,  $\overline{\text{DHLD\text{AK}}} = 1$ , transfer not terminated and block transfer
- <20>  $\overline{\text{DHLD\text{AK}}} = 1$
- <21>  $\overline{\text{DHLD\text{AK}}} = 0$
- <22>  $\overline{\text{DHLD\text{AK}}} = 1$
- <23>  $\overline{\text{NMI}} = 0$

**Remark**  $\overline{\text{DHLD\text{AK}}}$  : Internal signal from BAU that grants bus mastership to DMAC

$\overline{\text{DREADY}}$  : Internal signal from WCU to indicate the DMAC ready state, and external  $\overline{\text{READY}}$  signal

### 5.3 BUS PRIORITY

The bus arbitration unit (BAU) arbitrates bus mastership among the bus masters (those that can acquire the bus). If a bus master having a certain priority requests the use of the bus while another bus master having a lower priority is using the bus, the BAU immediately transfers bus mastership from the low-priority bus master to the high-priority bus master.

Table 5-1 lists the types of the bus masters and their priorities.

**Table 5-1. Bus Priority**

Bus master	Bus cycle	Priority
CPU	Bus lock	1 (highest)
	Fetch	5 (lowest)
	Read/write	
DRAMC	Refresh	2
External bus master	Bus cycles driven by external devices	3
DMAC	DMA	4

Although the DRAMC takes priority over external bus masters, it cannot acquire the bus forcibly while it is being used by an external bus master. In response to a refresh request received from the DRAMC, the BAU makes the  $\overline{\text{HLDAK}}$  pin inactive to indicate the presence of the refresh request. When the  $\overline{\text{HLDRQ}}$  pin is made inactive while the  $\overline{\text{HLDAK}}$  is inactive, a refresh cycle is initiated.

Bus mastership is switched during each bus cycle or idle state. In the halt and IDLE modes upon reset, the  $\overline{\text{HLDRQ}}$  signal is accepted. During a halt acknowledge cycle in which halt or IDLE mode is entered, or during the first six clock cycles of a self-refresh cycle in which IDLE mode is entered, however, the  $\overline{\text{HLDRQ}}$  signal is not accepted.

At power-on reset, the  $\overline{\text{HLDRQ}}$  signal must be made inactive.

## 5.4 DATA FLOW WHEN USING AN ADDITIONAL BUS CYCLE

A bus cycle is added only when word access is performed.

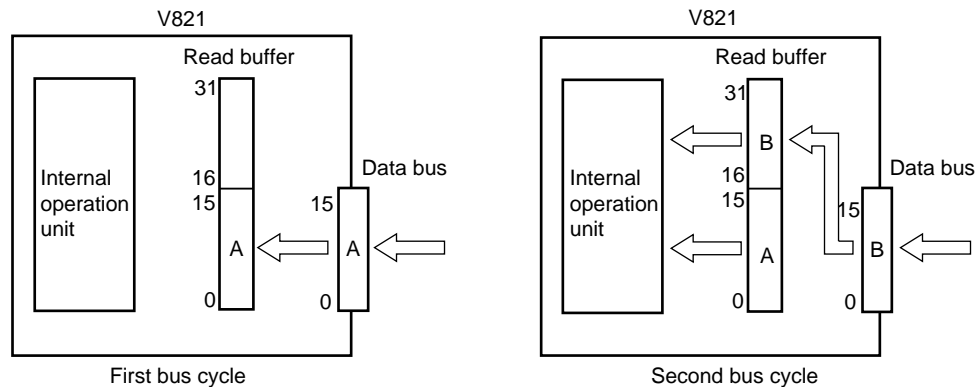
During read cycles, the low-order 16 bits of word data are sampled during the first bus cycle. The high-order 16 bits are sampled during the second bus cycle.

During write cycles, the low-order 16 bits of word data are output at the falling edge of the clock signal in the T1 state of the first bus cycle. The remaining high-order 16 bits are output at the falling edge of the clock signal in the T1S state of the second bus cycle. Then, the output data is held until the clock signal in the T1 state of the next bus cycle goes low.

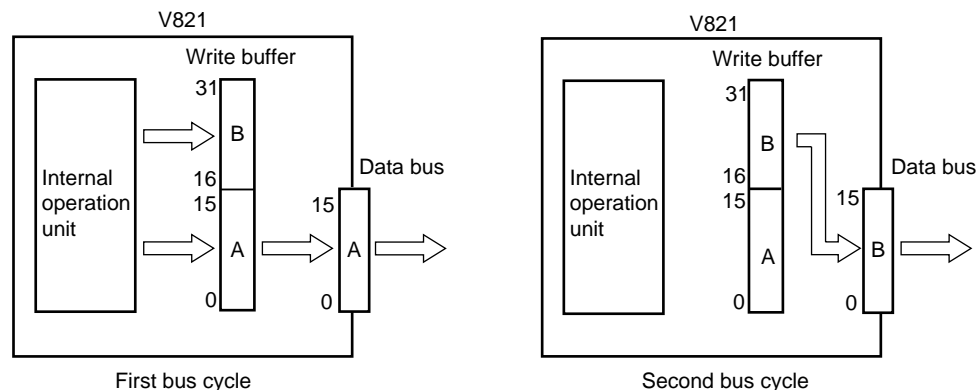
Figure 5-3 illustrates read and write operations. In the figure, A is assumed to be the low-order 16 bits of the word data, while B is assumed to be the high-order 16 bits of the word data.

**Figure 5-3. Data Flow When an Additional Bus Cycle Is Used**

### (a) Read cycle



### (b) Write cycle



## 5.5 RELATIONSHIP BETWEEN EXTERNAL ACCESSES AND THE DATA BUS

This section explains data I/O and the byte enable signal when external access is performed.

### 5.5.1 Relationship between External Access and Byte Enable Signal

Table 5-2 shows the relationship between external accesses and the byte enable signal ( $\overline{UBE}$ ).

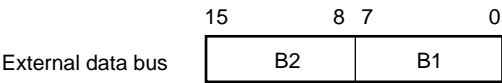
**Table 5-2. Relationship between Addresses, Data Length,  $\overline{UBE}$ , and Address Pins**

Data length	Operand address		$\overline{UBE}$	Address pin		Bus cycle sequence
	Bit 1	Bit 0		A1	A0	
Byte	0	0	1	0	0	1
	0	1	0	0	1	1
	1	0	1	1	0	1
	1	1	0	1	1	1
Halfword	0	0	0	0	0	1
	1	0	0	1	0	1
Word	0	0	0	0	0	1
			0	1	0	2

### 5.5.2 Operand Read

Figure 5-4 shows the relationship between external access and data input during an operand read operation.

In the figure, read buffer "n;Bm" indicates that the mth byte on the external data bus is read during the nth bus cycle. Bm in the external data bus is illustrated below:



**Figure 5-4. Read Cycle**

#### (a) Data length: Byte

Operand address		Internal register			
Bit 1	Bit 0	31	24	23	16 15 8 7 0
0	0				1 ; B1
0	1				1 ; B2
1	0				1 ; B1
1	1				1 ; B2

#### (b) Data length: Halfword

Operand address		Internal register			
Bit 1	Bit 0	31	24	23	16 15 8 7 0
0	0				1 ; B2 1 ; B1
1	0				1 ; B2 1 ; B1

#### (c) Data length: Word

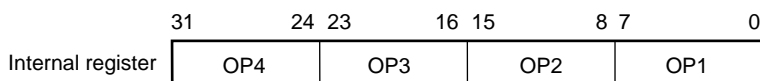
Operand address		Internal register			
Bit 1	Bit 0	31	24	23	16 15 8 7 0
0	0	2 ; B2	2 ; B1	1 ; B2	1 ; B1

### 5.5.3 Operand Write

Figure 5-5 shows the relationship between external access and data output during an operand write operation.

In the figure, OPm (m = 1 to 4) indicates the byte position in the internal register.

The position of OPm in the internal register is shown below.



**Figure 5-5. Write Cycle**

#### (a) Data length: Byte

Operand address		External data bus			Bus cycle sequence
Bit 1	Bit 0	15	8 7	0	
0	0	Note 1	OP1		1
0	1	OP1	Note 2		1
1	0	Note 1	OP1		1
1	1	OP1	Note 2		1

**Notes** 1. The undefined data is output from bits D8 to D15.

2. The undefined data is output from bits D0 to D7.

#### (b) Data length: Halfword

Operand address		External data bus			Bus cycle sequence
Bit 1	Bit 0	15	8 7	0	
0	0	OP2	OP1		1
1	0	OP2	OP1		1

#### (c) Data length: Word

Operand address		External data bus			Bus cycle sequence
Bit 1	Bit 0	15	8 7	0	
0	0	OP2	OP1		1
0	0	OP4	OP3		2

#### 5.5.4 Notes on Bit Strings

Bit strings are processed word by word (in blocks of 4 bytes) to speed up operation. If the end of the data does not correspond to a word boundary when a write operation is performed, that is, if bits 1 and 0 of the last byte address of the locations that hold the bit string in the destination are other than "11," a maximum of three extra bytes are written as shown in Figure 5-6. In this case, as the extra data, the original value is written as is. In this way, the data is not destroyed.

During read operations, up to three extra words subsequent to the last word of the data may be read during read cycles at both the source and destination. This extra data is discarded.

**Figure 5-6. Write Cycles for Bit Strings**

Operand address		External data bus			Bus cycle sequence
Bit 1	Bit 0	15	8 7	0	
0	0	OP2 <sup>Note</sup>	OP1		1
0	0	OP4 <sup>Note</sup>	OP3 <sup>Note</sup>		2
0	1	OP2	OP1		1
0	1	OP4 <sup>Note</sup>	OP3 <sup>Note</sup>		2
1	0	OP2	OP1		1
1	0	OP4 <sup>Note</sup>	OP3		2

**Note** Extra bytes written unnecessarily

## 5.6 EXTERNAL I/O ACCESS

### 5.6.1 External I/O Read Cycle

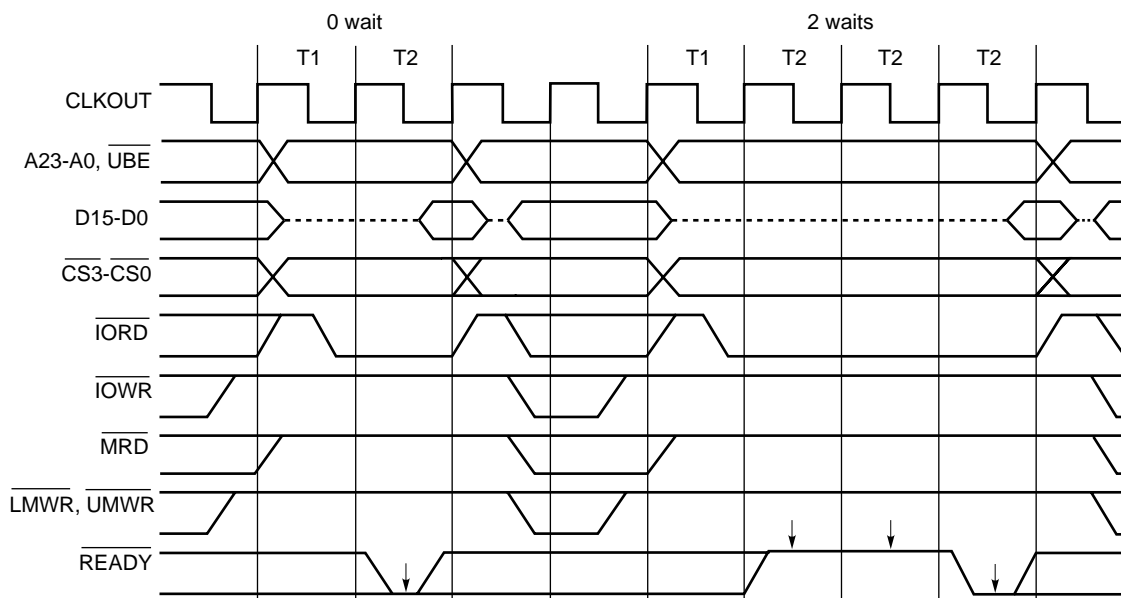
The timing for each state is explained below:

T1 state : The output of an address and  $\overline{\text{CSn}}$  signal starts. At the falling edge of the clock signal, the  $\overline{\text{IOR}}$  signal becomes active.

T2 state : If the  $\overline{\text{READY}}$  signal is active at the falling edge of the clock signal, the bus cycle terminates. The  $\overline{\text{IOR}}$  signal is made inactive, and at the same time, data is sampled.

The wait count set by the WSn bit of the PWCn register is ORed with the wait count controlled by  $\overline{\text{READY}}$ . Therefore, as many wait states as the larger of these wait counts are inserted. (See **Chapter 6**.)

**Figure 5-7. External I/O Read Cycle**



**Remarks 1.** An arrow indicates a sampling timing when the programmable wait is set to 0.

**2.** A broken line indicates a high impedance.



### 5.6.2 External I/O Write Cycle

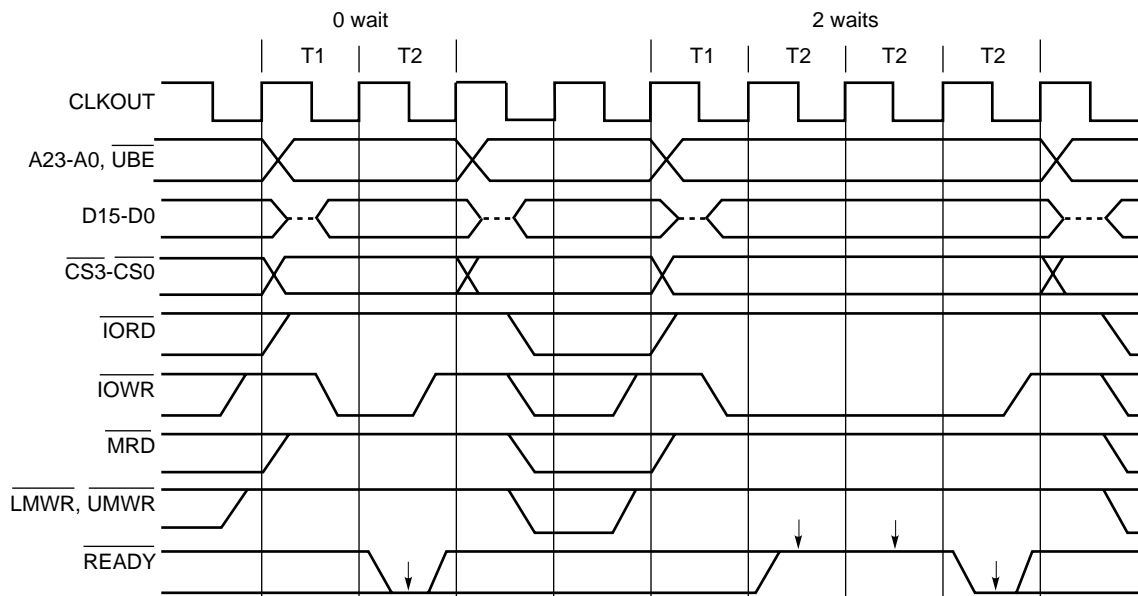
The timing for each state is explained below:

T1 state : The output of an address and the  $\overline{\text{CSn}}$  signal starts. At the falling edge of the clock signal, the  $\overline{\text{IOWR}}$  signal becomes active.

T2 state : If the  $\overline{\text{READY}}$  signal is active at the falling edge of the clock signal, the bus cycle terminates, and the  $\overline{\text{IOWR}}$  signal is made inactive. The write data is maintained until the clock signal goes high in the T1 or T1 state of the next bus cycle.

The wait count set by the WSn bit of the PWCn register is ORed with the wait count controlled by  $\overline{\text{READY}}$ . Therefore, as many wait states as the larger of these wait counts are inserted. (See **Chapter 6**.)

**Figure 5-8. External I/O Write Cycle**



- Remarks 1.** An arrow indicates a sampling timing when the programmable wait is set to 0.  
**2.** A broken line indicates a high impedance.

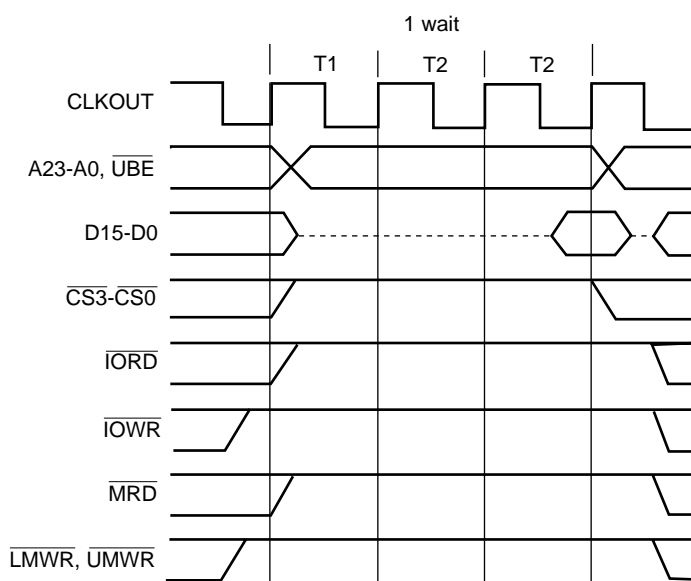
## 5.7 INTERNAL I/O ACCESS

Internal I/O read cycles and internal I/O write cycles are similar to external I/O read cycles and external I/O write cycles, except in the following points:

- Neither the  $\overline{\text{IORD}}$ ,  $\overline{\text{IOWR}}$ , nor  $\overline{\text{CSn}}$  signals become active.
- The number of wait states that can be inserted is always 1 or 2. (Wait control using the  $\overline{\text{READY}}$  pin and  $\text{PWCn}$  register is impossible.)

### 5.7.1 Internal I/O Read Cycle

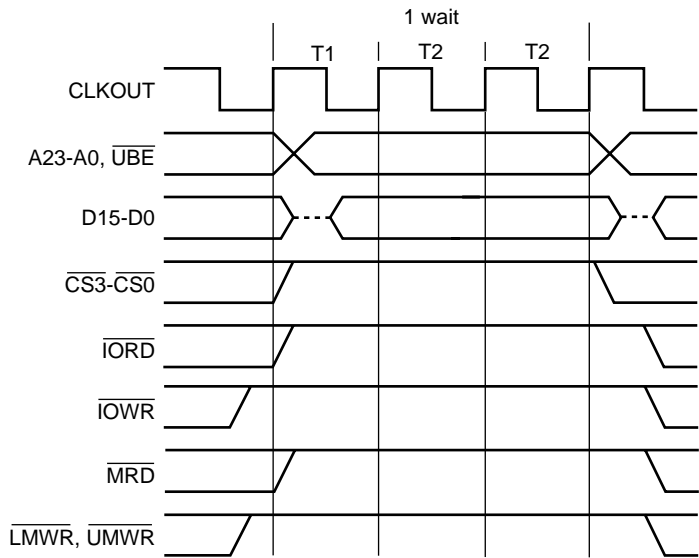
**Figure 5-9. Internal I/O Read Cycle**



**Remark** A broken line indicates a high impedance.

### 5.7.2 Internal I/O Write Cycle

**Figure 5-10. Internal I/O Write Cycle**



**Remark** A broken line indicates a high impedance.

## 5.8 SRAM (ROM) ACCESS

### 5.8.1 SRAM (ROM) Read Cycle

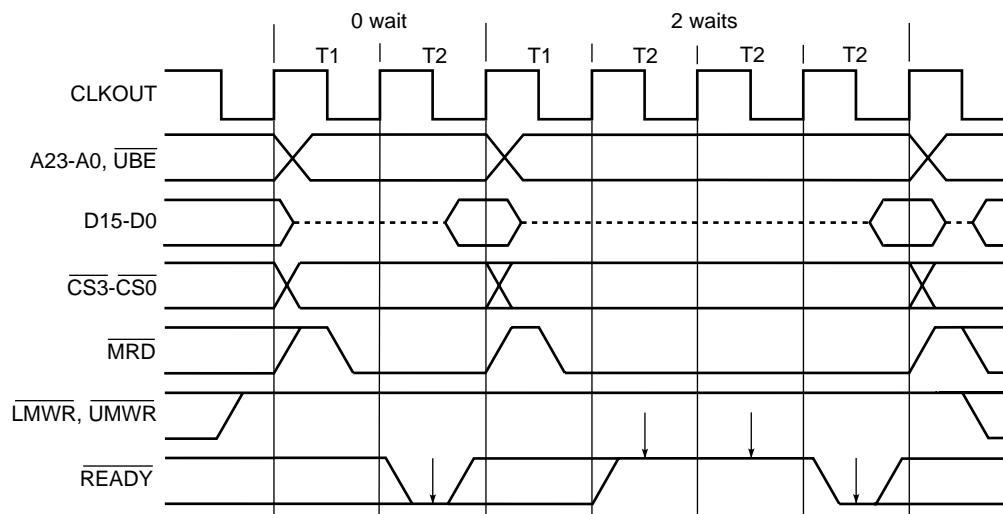
The timing in each state is explained below:

T1 state : The output of an address and the  $\overline{\text{CSn}}$  signal starts. At the falling edge of the clock signal, the  $\overline{\text{MRD}}$  signal becomes active.

T2 state : If the  $\overline{\text{READY}}$  signal is active at the falling edge of the clock signal, the bus cycle terminates. The  $\overline{\text{MRD}}$  signal is made inactive and, at the same time, data is sampled.

The wait count set with the WSn bit of the PWCn register is ORed with the wait count controlled by  $\overline{\text{READY}}$ . Therefore, as many wait states as the larger of these wait counts are inserted. (See **Chapter 6**.)

**Figure 5-11. SRAM (ROM) Read Cycle**



**Remarks 1.** An arrow indicates a sampling timing when the programmable wait is set to 0.

**2.** A broken line indicates a high impedance.

### 5.8.2 SRAM Write Cycle

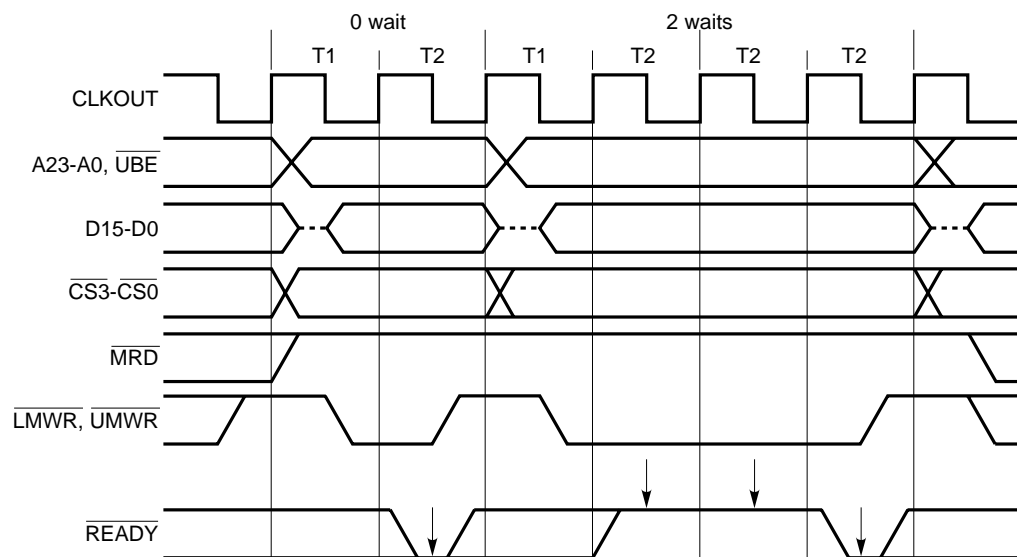
The timing in each state is explained below:

T1 state : The output of an address and the  $\overline{\text{CSn}}$  signal starts. At the falling edge of the clock signal, the  $\overline{\text{LMWR}}$  and  $\overline{\text{UMWR}}$  signals become active.

T2 state : If the  $\overline{\text{READY}}$  signal is active at the falling edge of the clock signal, the bus cycle terminates, and the  $\overline{\text{LMWR}}$  and  $\overline{\text{UMWR}}$  signals are made inactive. The write data is maintained until the clock signal goes high in the T1 or TI state of the next bus cycle.

The wait count set by the WSn bit of the PWCn register is ORed with the wait count controlled by  $\overline{\text{READY}}$ . Therefore, as many wait states as the larger of these wait counts are inserted. (See **Chapter 6**.)

**Figure 5-12. SRAM Write Cycle**



- Remarks 1.** An arrow indicates a sampling timing when the programmable wait is set to 0.  
**2.** A broken line indicates a high impedance.

## 5.9 PAGE-ROM ACCESS

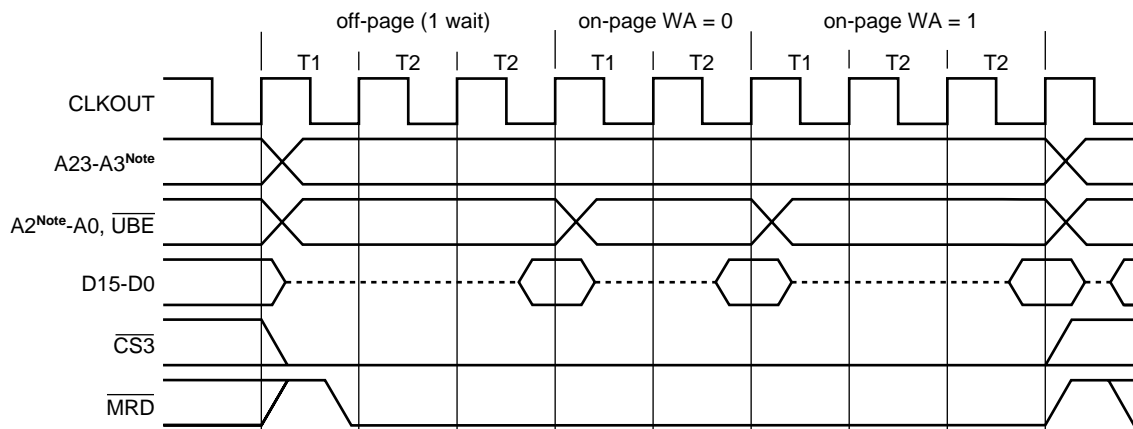
Page-ROM cycles are selected by setting 1 in the CT3 bit of the BCTC register, a WCU control register.

### 5.9.1 Page-ROM Read Cycle

The page-ROM read cycle is similar to the SRAM read cycle, except in the following points:

- The wait count for each on-page cycle is specified with the WA bit of the PRC register (0 or 1). The wait count for the off-page cycles is specified with the WS3 bit of the PWC1 register (0 to 7).
- Wait control using the  $\overline{\text{READY}}$  pin is impossible.
- The  $\overline{\text{MRD}}$  signal does not become inactive for the on-page cycles.

**Figure 5-13. Page-ROM Read Cycle**



**Note** When MA5 to MA3 of the PRC register are 000 (See **Section 7.2.3.**)

**Remark** A broken line indicates a high impedance.

## 5.10 DRAM ACCESS

DRAM cycles are selected by setting 01 or 11 in the CT0 bits of the BCTC register, a WCU control register.

<1> When CT0 bits = 01 : The  $\overline{CS0}$  pin functions as the  $\overline{REFRQ}$  output.

<2> When CT0 bits = 11 : The  $\overline{CS0}$  pin functions as the  $\overline{CS0}$  output. ( $\overline{CS0}$  becomes active when I/O block 0 is accessed.)

During the DRAM cycle, the  $\overline{CS0}$  signal becomes active in the case of <2> above.

To generate a cycle for the DRAM high-speed page mode, set the PAE bit of the DRAM configuration register to 1. (See **Section 7.1.4.**)

### 5.10.1 DRAM Read Cycle

#### (1) Off-page

The output of a column address starts in the T1 state. At the falling edge of the clock signal, the  $\overline{RAS}$  signal becomes inactive (RAS precharge), and the  $\overline{MRD}$  signal becomes active.

During the first clock period of the T2 state that follows T1, RAS precharge is performed. At the falling edge of the clock cycle, a row address is output.

In a second T2 state, the  $\overline{RAS}$  signal is made active and, at the falling edge of the clock signal, a column address is output.

In a third T2 state, the  $\overline{LCAS}$  and  $\overline{UCAS}$  signals are made active. If the CWS bit of the DRC register is 1, a wait state is inserted into the next cycle.

At the rising edge of the clock signal in the T1 state of the next cycle, the  $\overline{LCAS}$ ,  $\overline{UCAS}$ , and  $\overline{MRD}$  signals are made inactive, data is sampled, and the cycle is completed.

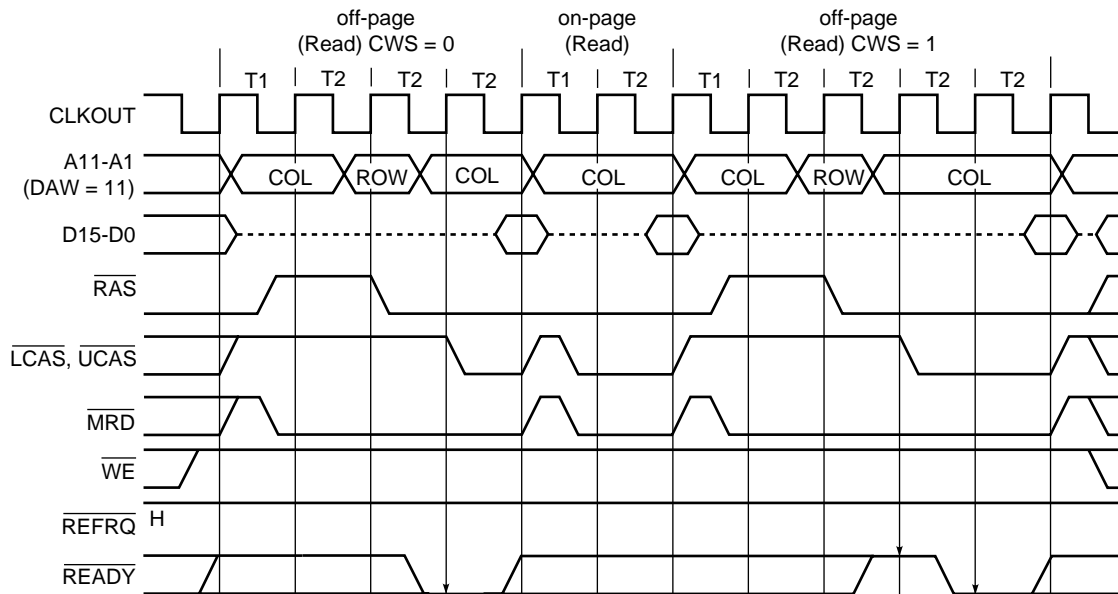
Wait states can be inserted by inputting the external  $\overline{READY}$  signal.

#### (2) On-page

In the T1 state, output of a column address starts. At the falling edge of the clock signal, the  $\overline{MRD}$ ,  $\overline{LCAS}$ , and  $\overline{UCAS}$  signals are made active. The  $\overline{RAS}$  signal is held active.

At the rising edge of the clock signal in the T1 state of the next cycle, the  $\overline{LCAS}$ ,  $\overline{UCAS}$ , and  $\overline{MRD}$  signals are made inactive, data is sampled, and the cycle is completed.

Figure 5-14. DRAM Read Cycle



- Remarks 1.** ROW : Row address  
COL : Column address
- 2.** A broken line indicates a high impedance.
- 3.** An arrow indicates a sampling timing.



### 5.10.2 DRAM Write Cycle

#### (1) Off-page

In the T1 state, output of a column address starts. At the falling edge of the clock signal, the  $\overline{\text{RAS}}$  signal becomes inactive (RAS precharge), and the  $\overline{\text{WE}}$  signal becomes active.

During the first clock period in the T2 state following T1, RAS precharge is performed. At the falling edge of the clock cycle, a row address is output.

During the second T2 state, the  $\overline{\text{RAS}}$  signal is made active, and a column address is output at the falling edge of the clock signal. If the CWS bit of the DRC register is 1, a wait state is inserted into the next cycle.

In a third T2 state, the  $\overline{\text{LCAS}}$  and  $\overline{\text{UCAS}}$  signals are made active. At the falling edge of the clock signal, the  $\overline{\text{WE}}$  signal is made inactive.

At the rising edge of the clock signal in the T1 state of the next cycle, the  $\overline{\text{LCAS}}$  and  $\overline{\text{UCAS}}$  signals are made inactive, and the cycle is completed.

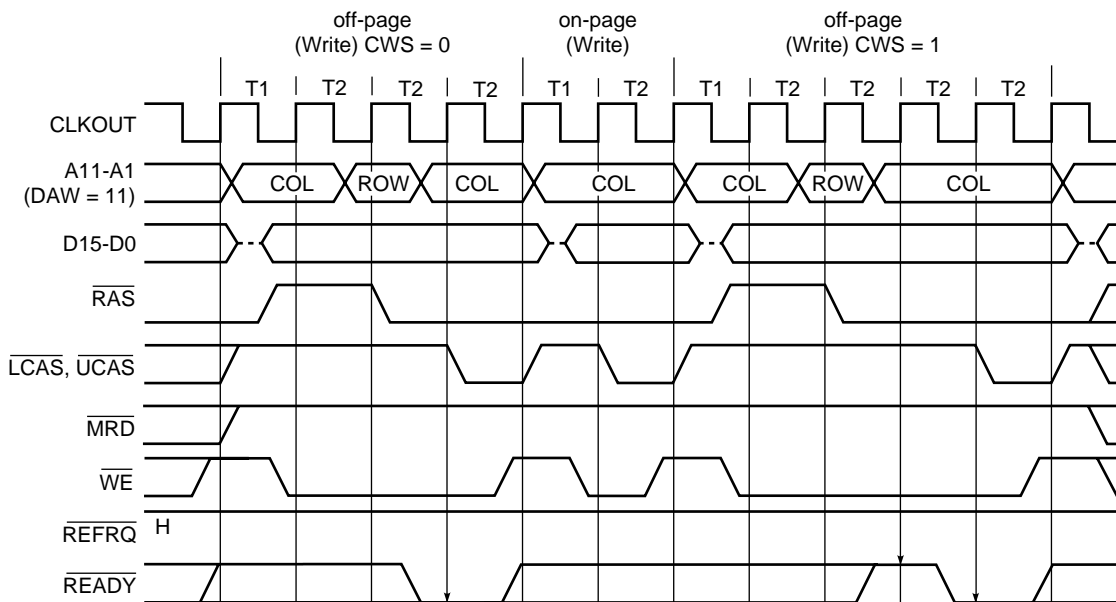
Wait states can be inserted by inputting the external  $\overline{\text{READY}}$  signal.

#### (2) On-page

In the T1 state, output of a column address starts. At the falling edge of the clock signal, the  $\overline{\text{WE}}$  signal is made active. The  $\overline{\text{RAS}}$  signal is held active.

In the T2 state, the  $\overline{\text{LCAS}}$  and  $\overline{\text{UCAS}}$  signals are made active. If the ADC bit of the DRAM configuration register is set to 1, one wait state is inserted unless the previous cycle was a write cycle. In this case, during the next T2 state, the  $\overline{\text{LCAS}}$  and  $\overline{\text{UCAS}}$  signals are made active and, at the falling edge of the clock signal, the  $\overline{\text{WE}}$  signal is made inactive.

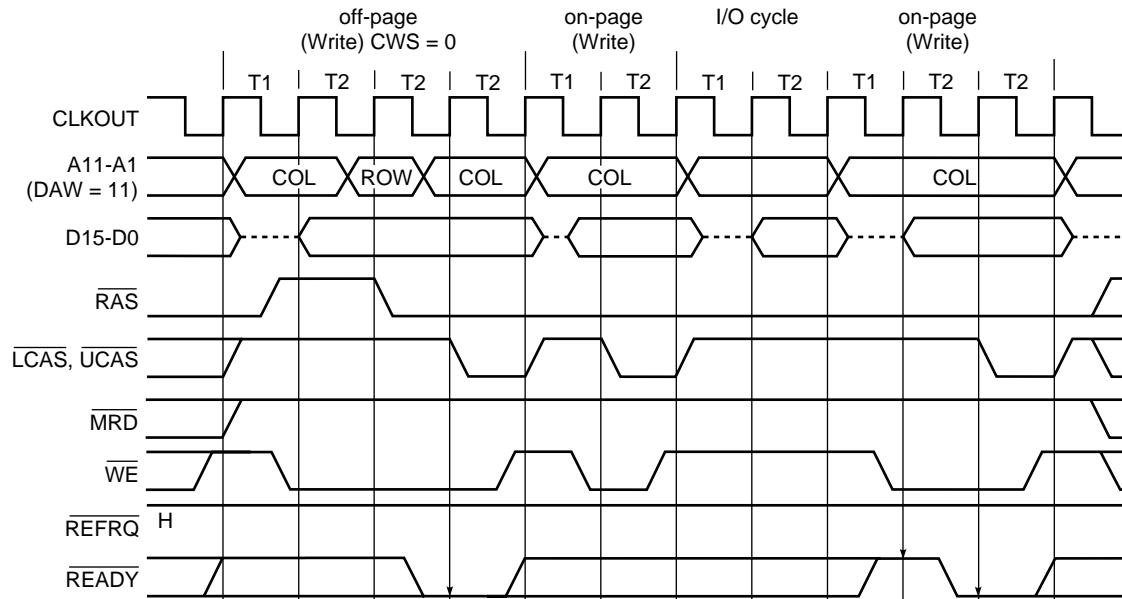
At the rising edge of the clock signal in the T1 state of the next cycle, the  $\overline{\text{LCAS}}$  and  $\overline{\text{UCAS}}$  signals are made inactive, and the cycle is completed.

**Figure 5-15. DRAM Write Cycle****(1) When ADC = 0****Remarks 1.** ROW : Row address

COL : Column address

**2.** A broken line indicates a high impedance.**3.** An arrow indicates a sampling timing.

## (2) When ADC = 1

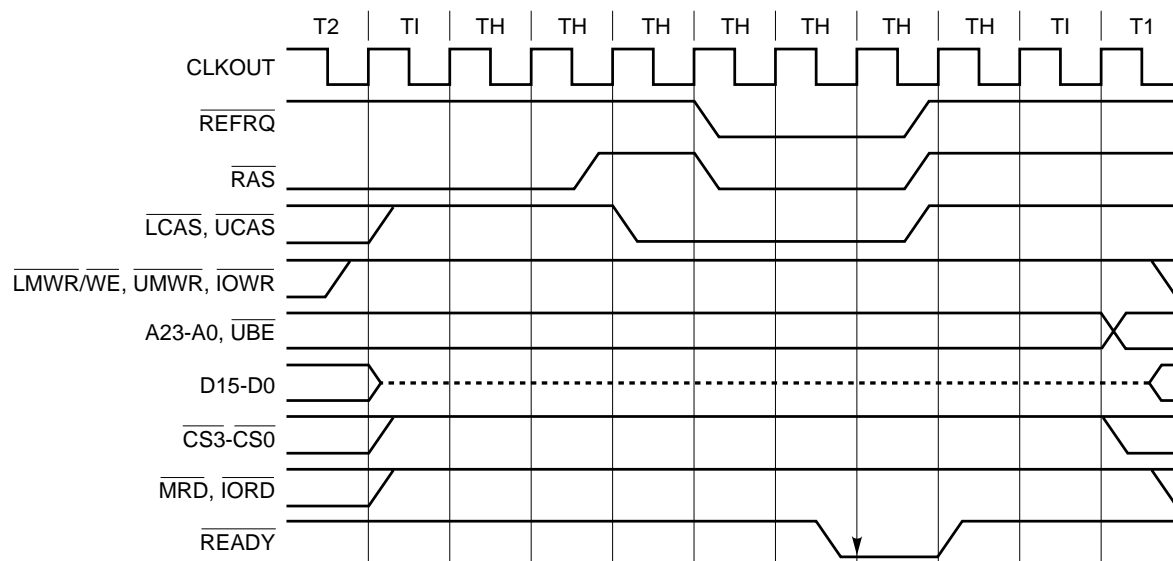


- Remarks 1.** ROW : Row address  
COL : Column address
- A broken line indicates a high impedance.
  - An arrow indicates a sampling timing.

### 5.10.3 CBR Refresh Cycle

A bus cycle for a refresh cycle requires at least seven clock periods. Including the idle cycles entered before and after the refresh cycle, at least nine clock periods are required. During the CBR refresh cycle, the external  $\overline{\text{READY}}$  signal can be used for control. At the rising edge of the sixth clock cycle of the CBR refresh cycle, the external  $\overline{\text{READY}}$  pin is sampled. If it is found to be inactive, the CBR refresh cycle is not terminated until the external  $\overline{\text{READY}}$  pin becomes active at the rising clock edge.

Figure 5-16. CBR Refresh Cycle



**Remarks 1.** A broken line indicates a high impedance.

**2.** An arrow indicates a sampling timing.

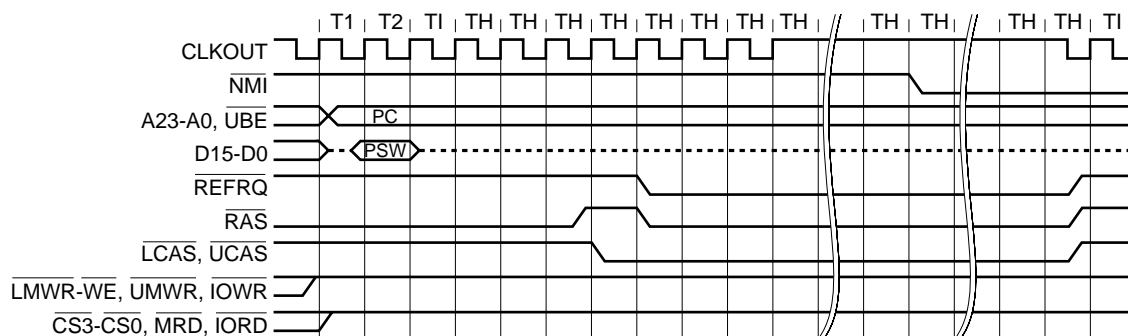
#### 5.10.4 Self-Refresh Cycle

The self-refresh cycle is initiated by setting the SMD bit of the standby control register (STBC) to specify idle or STOP mode, and executing the HALT instruction. Before DRAM can perform a self-refresh operation, the specified  $\overline{\text{RAS}}$  pulse width of DRAM (CBR self-refresh) ( $100\ \mu\text{s}$  or more) must be satisfied. When the  $\overline{\text{NMI}}$  signal is generated during the CBR self-refresh cycle, the  $\overline{\text{REFRQ}}$ ,  $\overline{\text{RAS}}$ ,  $\overline{\text{LCAS}}$ , and  $\overline{\text{UCAS}}$  signals are made inactive, and CBR self-refresh is canceled.

During the halt acknowledge cycle and the first six clock cycles of the IDLE mode self-refresh cycle, all external hold requests are ignored. External hold requests can be made only from the seventh clock cycle of the self-refresh cycle.

In the halt acknowledge cycle and self-refresh cycle in STOP mode, all external hold requests are ignored.

**Figure 5-17. CBR Self-Refresh Cycle**



**Remark** A broken line indicates a high impedance.

### 5.10.5 Bus Cycle during Fly-by Transfer

This cycle occurs only when fly-by transfer is specified for the DMA.

In the fly-by cycle, I/O access and memory access are performed at the same time. Basically, the I/O and memory accesses are made in the same way as in the I/O and memory access cycles, respectively.

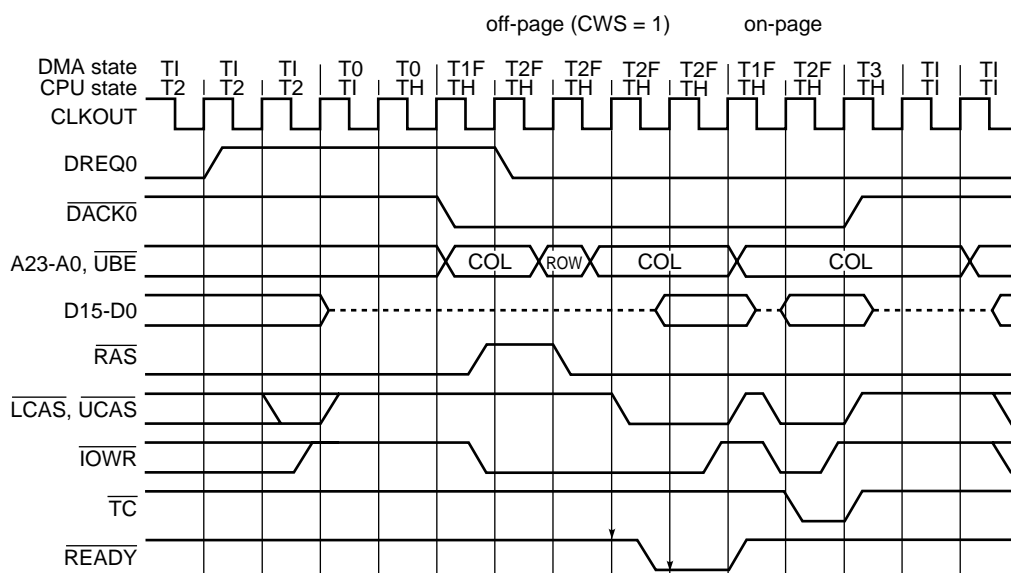
The wait count is set with the DWSn bit of programmable wait control register 2 (PWC2).

During DRAM off-page access and the SRAM cycle, wait states can be inserted by inputting the external  $\overline{\text{READY}}$  signal.

#### (1) Fly-by read cycle

##### (a) Block transfer

Figure 5-18. Fly-by Read Cycle during DMA Block Transfer (DRAM to External I/O Device)



**Remarks 1.** ROW : Row address

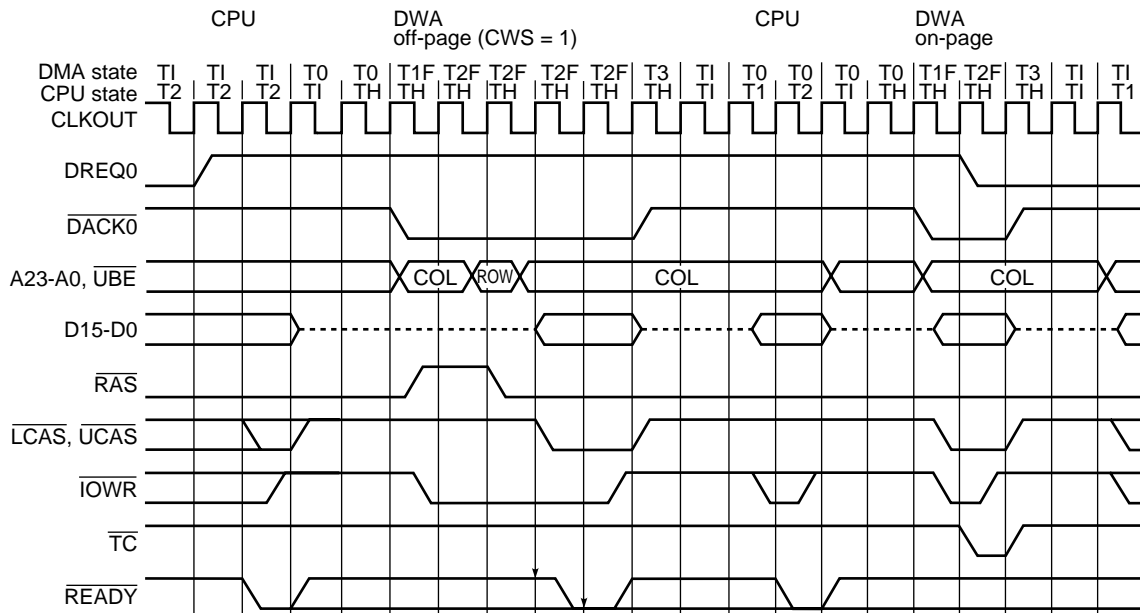
COL : Column address

**2.** A broken line indicates a high impedance.

**3.** An arrow indicates a sampling timing.

**(b) Single transfer and single-step transfer**

**Figure 5-19. Fly-by Read Cycle during DMA Single Transfer and Single-Step Transfer (DRAM to External I/O Device)**

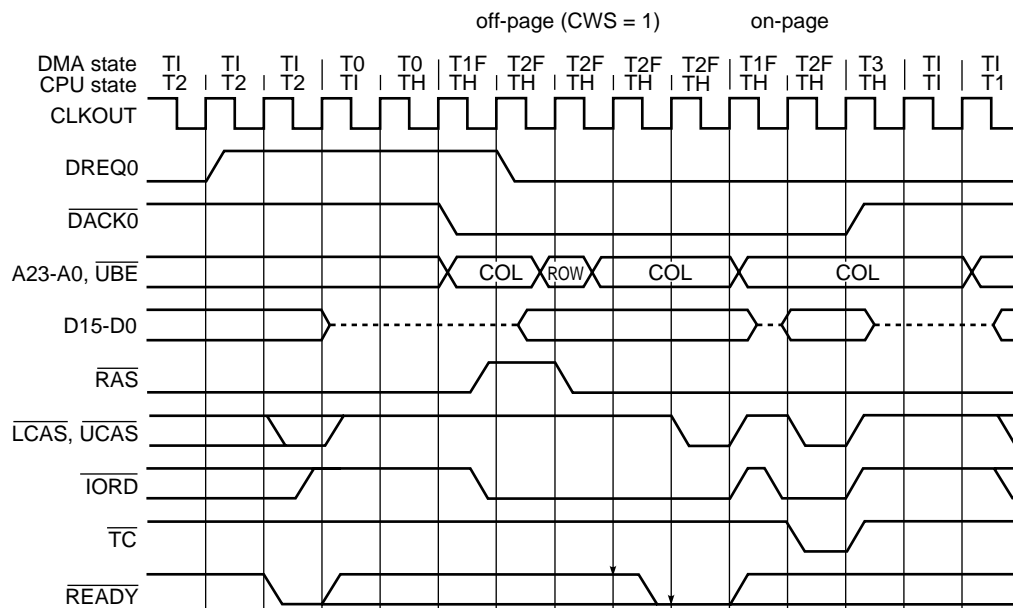


- Remarks 1.** ROW : Row address  
COL : Column address
- 2.** A broken line indicates a high impedance.
- 3.** An arrow indicates a sampling timing.

## (2) Fly-by write cycle

## (a) Block transfer

Figure 5-20. Fly-by Write Cycle during DMA Block Transfer (External I/O Device to DRAM)



**Remarks 1.** ROW : Row address

COL : Column address

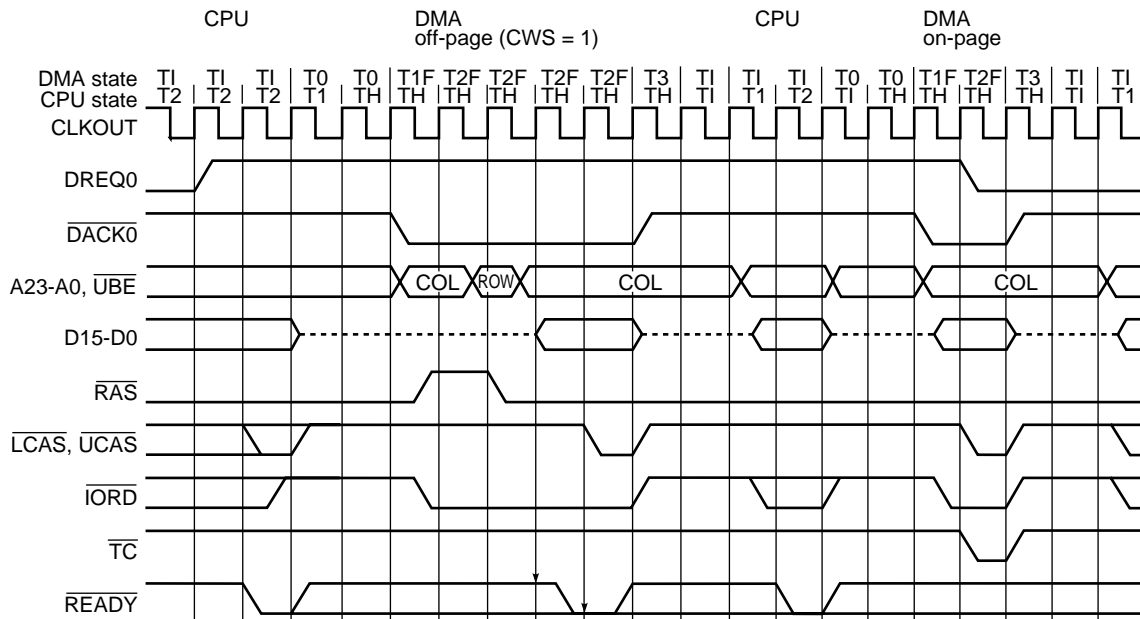
**2.** A broken line indicates a high impedance.

**3.** An arrow indicates a sampling timing.



**(b) Single transfer and single-step transfer**

**Figure 5-21. Fly-by Write Cycle during DMA Single Transfer and Single-Step Transfer  
(External I/O Device to DRAM)**



- Remarks 1.** ROW : Row address  
COL : Column address
- 2.** A broken line indicates a high impedance.
- 3.** An arrow indicates a sampling timing.

## 5.11 EXCEPTION HANDLING CYCLES

### 5.11.1 Machine Fault Cycle

When a machine fault occurs, the machine fault cycle is initiated. During the I/O write cycle, the source code of a fatal exception (generated by ORing 0000H with the exception code), and the PSW and PC contents at that time are output onto the data bus, one after the other.

Table 5-3 indicates the correspondence between the data bus and address bus.

**Table 5-3. Correspondence between the Address Bus and Data Bus in the Machine Fault Cycle**

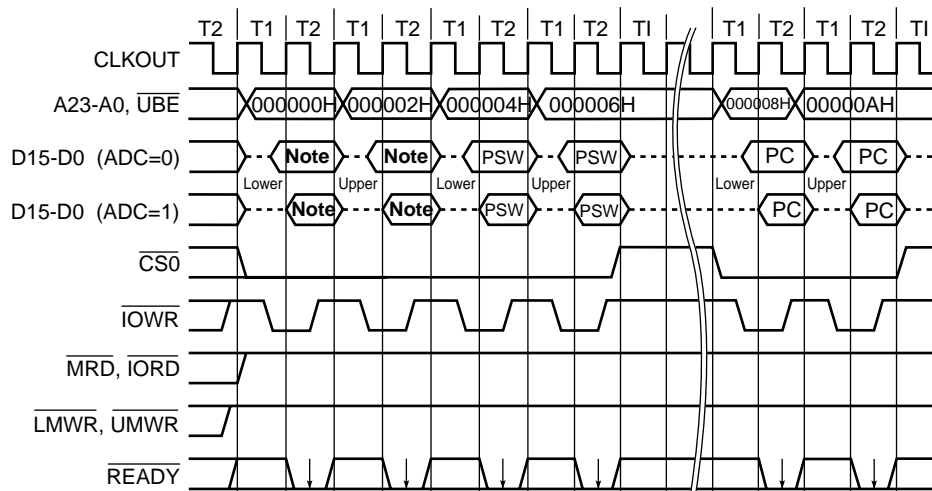
Sequence	Address bus (A0 - A23)	Data bus (D0 - D15)
1	000000H	Source code of fatal exception (low-order)
2	000002H	Source code of fatal exception (high-order) (The value is always FFFFH.)
3	000004H	PSW at that time (low-order)
4	000006H	PSW at that time (high-order)
5	000008H	PC at that time (low-order)
6	00000AH	PC at that time (high-order)

Do not use addresses 000000H to 00000BH of address block 0 in the I/O space. The machine fault cycle replaces any data in these locations with arbitrary data.

During this write cycle, all wait and bus hold requests are valid. Requests using  $\overline{\text{HLDRQ}}$  and  $\overline{\text{READY}}$  are also valid during the machine fault cycle and the subsequent TI state.

The machine fault state can be released only by inputting a reset.

**Figure 5-22. Machine Fault Cycle**



**Note** Source code of a fatal exception

**Remarks 1.** A broken line indicates a high impedance.

**2.** An arrow indicates a sampling timing.

### 5.11.2 Halt Acknowledge Cycle

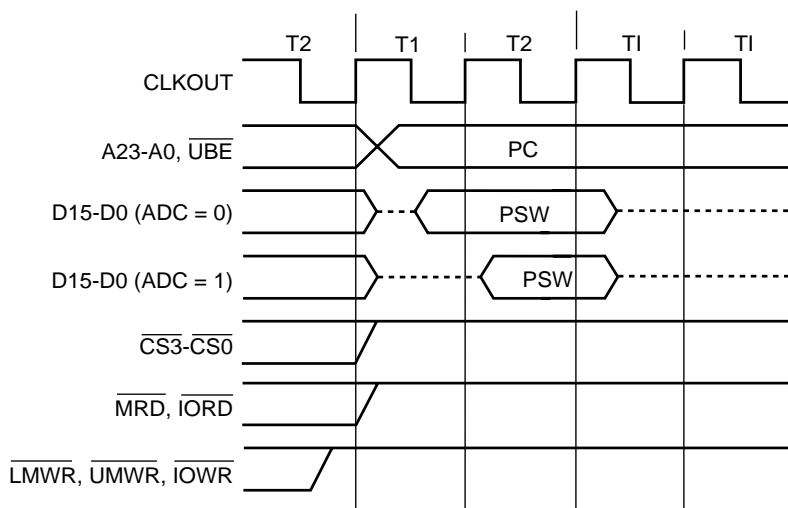
In the halt acknowledge cycle, the low-order 24 bits of the PC existing when the HALT instruction is executed are output onto the address bus, while the low-order 16 bits of the PSW are output onto the data bus.

Wait states cannot be inserted by inputting the external  $\overline{\text{READY}}$  signal. The wait count is always 0.

During the cycle, hold requests are ignored.

When idle or STOP mode is specified with the SMD bit of the standby control register (STBC), a self-refresh cycle is initiated immediately after the hold acknowledge cycle. (See **Section 5.10.4.**)

**Figure 5-23. Halt Acknowledge Cycle**



**Remark** A broken line indicates a high impedance.

## 5.12 CONTROL SIGNAL TIMING

This section explains the timings of the bus hold and bus lock operations.

### 5.12.1 Bus Hold

When another processor requests bus mastership from the V821, the bus enters the bus hold state, and the bus is relinquished to the processor. In this case, the bus is placed in the floating state.

Figure 5-24 shows the timing at which the bus enters the bus hold state. In the different states, the following operations are performed:

**T2 state :** At the falling edge of the clock signal, the  $\overline{\text{READY}}$  and  $\overline{\text{HLDRQ}}$  signals are sampled. If both signals are active, the bus enters the TI state.

**TI state :** At the falling edge of the clock signal, the  $\overline{\text{HLDRQ}}$  signal is sampled. If the signal is active, the bus enters the TH state.

**TH state :** The bus is placed in the floating state, and the  $\overline{\text{HLDK}}$  signal is made active. If the  $\overline{\text{HLDRQ}}$  signal remains active, the TH state continues. If the signal becomes inactive, the bus enters the TI state, and bus mastership is returned to the CPU.

Bus hold requests are accepted in ordinary mode, HALT mode, and IDLE mode. In addition, bus hold requests can be accepted during a reset. Under the following conditions, however, a bus hold request is not accepted:

- When the CAXI instruction is executed (The bus is locked.)
- In STOP mode (internal clock stopped state)
- During the halt acknowledge cycle in which halt or IDLE mode is entered, or during the first six cycles of the self-refresh cycle in IDLE mode

At power-on reset, the  $\overline{\text{HLDRQ}}$  signal must be made inactive.

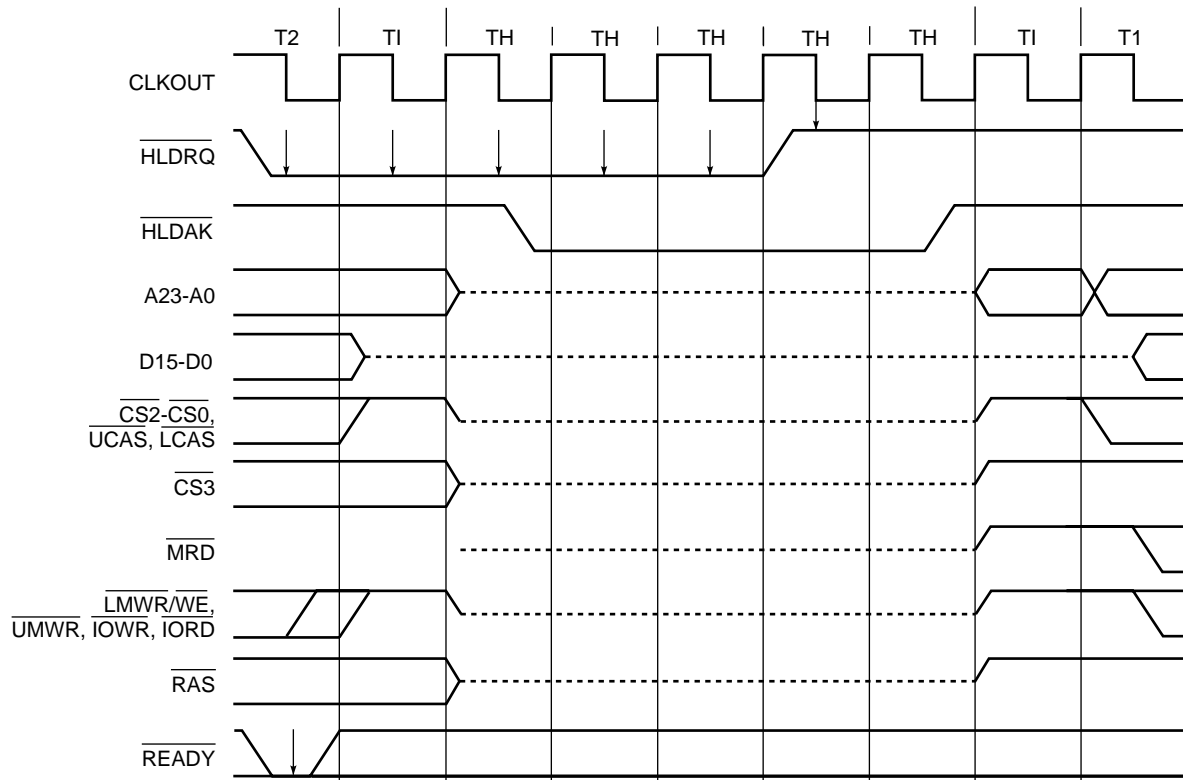
When the bus enters the bus hold state in IDLE mode, the  $\overline{\text{REFRQ}}$ ,  $\overline{\text{RAS}}$ ,  $\overline{\text{UCAS}}$ ,  $\overline{\text{LCAS}}$ ,  $\overline{\text{MRD}}$ , and  $\overline{\text{WE}}$  signals originally in the self-refresh cycle state, enter the floating state. Later, when the bus hold request is released, these signals become inactive, then, a self-refresh cycle is initiated.

If the bus cycle immediately before the bus hold state was a page-ROM cycle, the  $\overline{\text{CS3}}$  and  $\overline{\text{MRD}}$  pins, originally at the active level, are placed in the floating state. When the bus hold state is released, these pins become inactive.

Similarly, if the bus cycle immediately before the bus hold state was a DRAM cycle and an on-page access was enabled, the  $\overline{\text{RAS}}$  pin, originally at the active level, is placed in the floating state. When the bus hold state is released, the  $\overline{\text{RAS}}$  pin becomes inactive.

**Caution** When the bus hold cycle is to be repeated, assure at least three clocks at the inactive period for the  $\overline{\text{HLDRQ}}$  signal during the bus hold cycle.

**Figure 5-24. Bus Hold Cycle**



- Remarks 1.** A broken line indicates a high impedance.  
**2.** An arrow indicates a sampling timing.

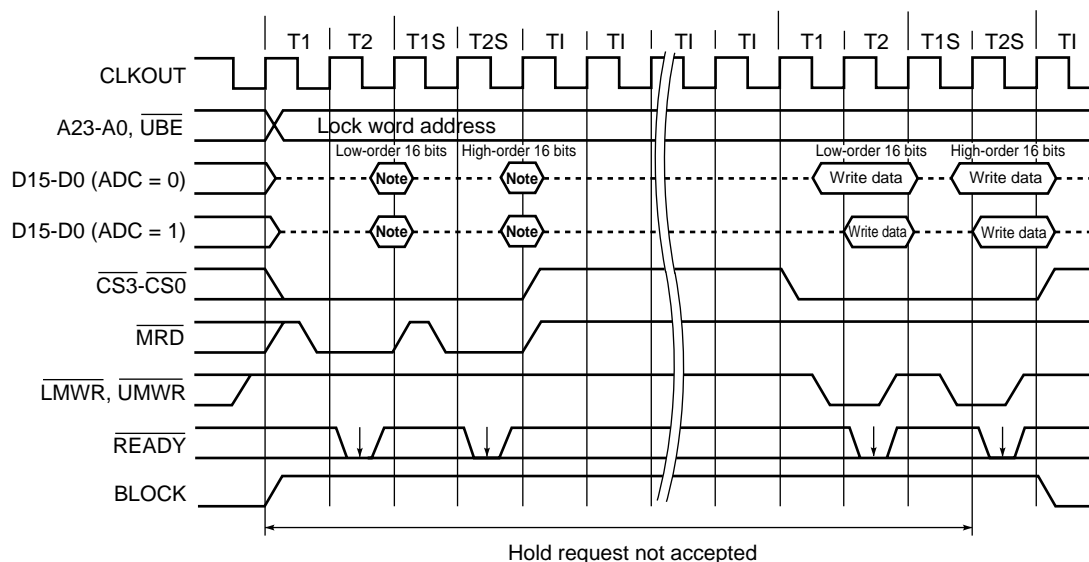
### 5.12.2 Bus Lock

To prevent processors other than the V821 from using the bus, the BLOCK signal can be made active to lock the bus.

A bus lock occurs when a CAXI instruction is executed. For example, during a lock word access made using the CAXI instruction, the BLOCK signal becomes active in sync with the start of a read bus cycle, and the signal becomes inactive in sync with the end of the last write cycle.

The BLOCK pin is also used as the WDTOUT pin. To output a BLOCK signal to the pin, the BWOS bit of the WDT mode register must be set. (See **Section 11.4.**)

**Figure 5-25. Bus Lock Cycle When the CAXI Instruction Is Executed**



**Note** Read data

**Remarks 1.** A broken line indicates a high impedance.

**2.** An arrow indicates a sampling timing.

## CHAPTER 6 WAIT CONTROL FUNCTIONS

The wait control unit (WCU) manages the four blocks corresponding to the four chip select signals, generates the chip select signals, performs wait control, and selects the bus cycle types.

### 6.1 FEATURES

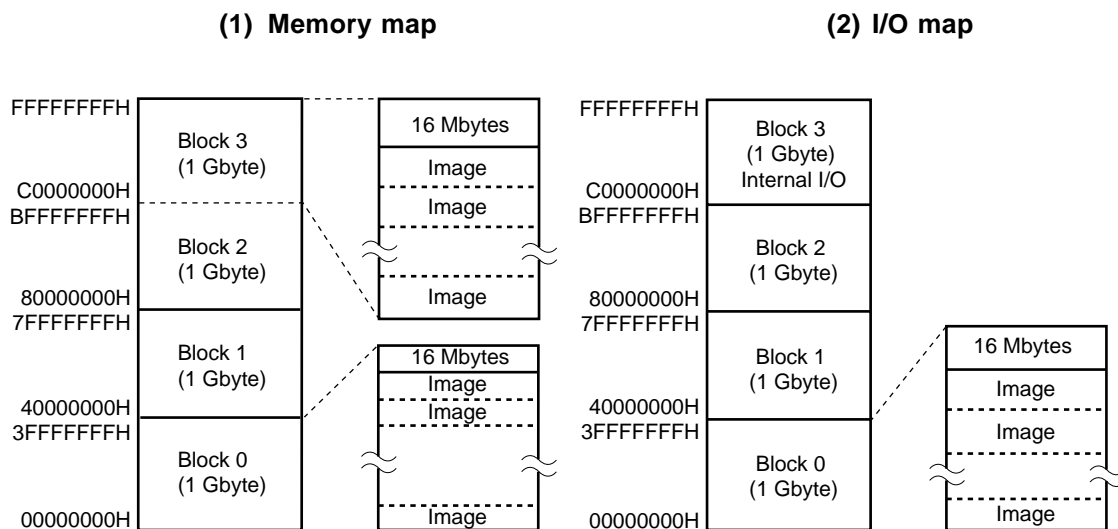
- Able to control up to four blocks in the memory and I/O spaces
- Linear address space of each block: 16 Mbytes
- Wait control
  - Automatic insertion of 0-7 waits per block
  - Insertion of waits using the  $\overline{\text{READY}}$  pin
- Bus cycle selection function
  - Page-ROM cycle selectable (address block 3)
  - DRAM cycle selectable (address block 0)

### 6.2 ADDRESS SPACES AND BLOCKS

The internal 4-Gbyte memory and I/O spaces are each divided into blocks of 1 Gbyte each. Built-in peripheral I/O registers are allocated to I/O block 3.

Each block has a linear address space of 16 Mbytes. For a 32-bit address, only the lower 24 bits are output.)

**Figure 6-1. Memory and I/O Maps**



### 6.3 CHIP SELECT CREATION AND BUS CYCLE SELECTION FUNCTIONS

The chip select signals ( $\overline{CS0}$ – $\overline{CS3}$ ) are output to block 0-3, as follows:

Block name	Access type	Chip select signal
Block 0	Memory or I/O access	$\overline{CS0}$ signal
Block 1	Memory or I/O access	$\overline{CS1}$ signal
Block 2	Memory or I/O access	$\overline{CS2}$ signal
Block 3	Memory access	$\overline{CS3}$ signal

When memory access is selected for block 0, the DRAM cycle can be specified. In this case, the  $\overline{CS0}$ / $\overline{REFRQ}$  pin is specified for either  $\overline{REFRQ}$  or  $\overline{CS0}$  output.  $\overline{CS0}$  output becomes active at both memory access and I/O access.

For memory access to block 3, the page-ROM cycle can be specified.

### 6.4 PROGRAMMABLE WAIT FUNCTION

WCU allows the user to specify the number of waits to be inserted, using the  $WS_n$  bits of the programmable wait control register (PWC $_n$ ) corresponding to the signal  $\overline{CS}_n$ . The programmable wait control register allows the specification of 0 to 7 waits. Additional waits can be inserted by using the  $\overline{READY}$  pin.

#### 6.4.1 Wait Control Using the $\overline{READY}$ Pin

Waits of eight or more clock pulses can be inserted using the  $\overline{READY}$  pin. In the cycles listed below, all of which have fixed waits, the  $\overline{READY}$  pin cannot be used to insert waits, however.

- Page-ROM cycle
- DRAM cycle (on-page)
- Internal I/O cycle
- CBR self refresh cycle
- Halt acknowledge cycle

The sampling of the  $\overline{READY}$  pin is performed as follows, according to the cycle:

- DRAM cycle (off-page): At the rising edge of the third clock pulse in the T2 cycle
- CBR refresh cycle: At the rising edge of the fourth clock pulse in the T2 cycle
- Other cycles in which the  $\overline{READY}$  pin is effective: At the falling edge of a clock pulse in the T2 cycle

The number of waits specified with the PWC $_n$  register and that specified with  $\overline{READY}$  input are ORed, that is, that having the greater number of waits is inserted.

**Caution** The CLKOUT signal must be synchronized so that the setup/hold time of  $\overline{READY}$  input is ensured. Otherwise, the operation cannot be guaranteed.



### 6.4.2 Wait Control during DMA Transfer

#### (1) During two-cycle DMA transfer

- Read cycle: Dependent on the wait setting made for an address block specified with the SBN bit of the DMA source address register (DSA).
- Write cycle: Dependent on the wait setting made for an address block specified with the DBN bit of the DMA destination address register (DDA).

#### (2) During fly-by DMA transfer

The 0-7 waits specified with the DWS0 and DWS1 bits of the PWC2 register can be inserted. An external  $\overline{\text{READY}}$  signal is effective at off-page access during the DRAM cycle and during the SRAM cycle of each block.

### 6.4.3 Bus Cycles during Which the Wait Function Is Effective

V821 supports the following wait insertion sources:

- Programmable wait (set with the PWC0 and PWC1 registers)
- Programmable wait (set with PWC2): During DMA fly-by transfer
- Wait with the  $\overline{\text{READY}}$  pin
- Wait at off/on-page time in the DRAM cycle with DRAMC (fixed)
- Wait at off/on-page time in the page-ROM cycle with ROMC (fixed)

Table 6-1 lists the relationship between each bus cycle and the wait function. Table 6-2 lists the bus cycles for which programmable waits can be set, together with the corresponding control registers.

Table 6-1. Bus Cycles during Which the Wait Function Is Effective

Bus cycle		Programmable wait	Wait with the $\overline{\text{READY}}$ pin
SRAM (ROM) cycle (Blocks 0-3)		0-7 waits	o
DRAM cycle (Block 0)	off-page	2 or 3 waits	o
	on-page	0 or 1 wait	x
Page-ROM cycle (Block 3)	off-page	0-7 waits	x
	on-page	0 or 1 wait	x
External I/O cycle (Blocks 0-2)		0-7 waits	o
Internal I/O cycle (Block 3)		1 or 2 waits	x
CBR refresh cycle		Fixed (3 waits)	o
CBR self-refresh cycle		-	x
Fly-by DMA transfer			
SRAM (ROM) cycle (Blocks 0-3)		0-7 waits	o
DRAM cycle (Block 0)	off-page	2-7 waits	o
	on-page	0-7 waits	x
Page-ROM cycle (Block 3)	off-page	0-7 waits	x
	on-page	0-7 waits	x
Halt acknowledge cycle		Fixed (0 wait)	x
Machine fault cycle (I/O block 0 write)		0-7 wait	o

**Remark** o: Effective

x: Not effective

**Table 6-2. Cycles during Which Programmable Waits Can Be Set, and Corresponding Control Registers**

Bus cycle		Control register
SRAM (ROM) cycle		Programmable wait control registers 0 and 1 (PWC0 and PWC1)
DRAM cycle	off-page	DRAM configuration register (DRC)
	on-page	
Page-ROM cycle	off-page	Programmable wait control register 1 (PWC1)
	on-page	Page-ROM configuration register (PRC)
External I/O cycle		Programmable wait control registers 0 and 1 (PWC0 and PWC1)
Fly-by DMA transfer		
SRAM (ROM) cycle		Programmable wait control register 2 (PWC2)
DRAM cycle	off-page	Programmable wait control register 2 (PWC2) and DRAM configuration register (DRC) <b>Note</b>
	on-page	
Page-ROM cycle	off-page	Programmable wait control register 2 (PWC2)
	on-page	
Machine fault cycle		Programmable wait control register 0 (PWC0)

**Note** See Table 6-3.

## 6.5 CONTROL REGISTERS

WCU includes a bus cycle type control register (BCTC) and programmable wait control registers 0-2 (PWC0-PWC2), used for wait control.

### 6.5.1 Bus Cycle Type Control Register (BCTC)

This register is used to set the bus cycle types for address blocks 0-3.

The register supports read/write in units of eight bits.

	7	6	5	4	3	2	1	0		
BCTC	0	CT3	0	CT2	0	CT1	CT0		Address C0000020H	When reset, 01H

Bit position	Bit name	Description															
6	CT3	<p>Cycle Type3 When memory block 3 is accessed, the <math>\overline{CS3}</math> signal is output. Specifies the cycle to be activated at that time. 0: SRAM (ROM) cycle 1: Page-ROM cycle</p>															
4	CT2	<p>Cycle Type2 Specifies whether the <math>\overline{CS2}</math> signal should be output at memory or I/O access, together with the cycle to be activated. 0: SRAM (ROM) cycle 1: I/O cycle</p>															
2	CT1	<p>Cycle Type1 Specifies whether the <math>\overline{CS1}</math> signal should be output at memory or I/O access, together with the cycle to be activated. 0: SRAM (ROM) cycle 1: I/O cycle</p>															
1, 0	CT0	<p>Cycle Type0 Specifies whether the <math>\overline{CS0}</math> signal should be output at memory or I/O access, together with the cycle to be activated. For the DRAM cycle, the <math>\overline{CS0}</math>/REFRQ pin can be specified for <math>\overline{CS0}</math> output for an I/O block, or for REFRQ output for DRAM. When the pin is specified for <math>\overline{CS0}</math> output (CT0 = 11), the <math>\overline{CS0}</math> signal becomes active at both I/O access and memory access.</p> <table border="1"> <thead> <tr> <th colspan="2">CT0</th><th>Cycle to be activated</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>SRAM (ROM) cycle</td></tr> <tr> <td>0</td><td>1</td><td>DRAM cycle (<math>\overline{REFRQ}</math> output)</td></tr> <tr> <td>1</td><td>0</td><td>I/O cycle</td></tr> <tr> <td>1</td><td>1</td><td>I/O cycle, DRAM cycle (<math>\overline{CS0}</math> output)</td></tr> </tbody> </table>	CT0		Cycle to be activated	0	0	SRAM (ROM) cycle	0	1	DRAM cycle ( $\overline{REFRQ}$ output)	1	0	I/O cycle	1	1	I/O cycle, DRAM cycle ( $\overline{CS0}$ output)
CT0		Cycle to be activated															
0	0	SRAM (ROM) cycle															
0	1	DRAM cycle ( $\overline{REFRQ}$ output)															
1	0	I/O cycle															
1	1	I/O cycle, DRAM cycle ( $\overline{CS0}$ output)															

## 6.5.2 Programmable Wait Control Register 0 (PWC0)

This register is used to set the number of waits for access to address blocks 0 and 1.

The register supports read/write in units of eight bits.

	7	6	5	4	3	2	1	0	
PWC0	0	WS1			0	WS0			Address C0000022H
									When reset, 77H

Bit position	Bit name	Description																																				
6-4	WS1	<p>Wait States1</p> <p>Specifies the number of waits. When address block 1 corresponding to the <math>\overline{CS1}</math> signal is accessed, the number of wait states specified with WS1 is automatically inserted.</p> <table><tr><th colspan="3">WS1</th><th>Number of waits</th></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td><td>2</td></tr><tr><td>0</td><td>1</td><td>1</td><td>3</td></tr><tr><td>1</td><td>0</td><td>0</td><td>4</td></tr><tr><td>1</td><td>0</td><td>1</td><td>5</td></tr><tr><td>1</td><td>1</td><td>0</td><td>6</td></tr><tr><td>1</td><td>1</td><td>1</td><td>7</td></tr></table>	WS1			Number of waits	0	0	0	0	0	0	1	1	0	1	0	2	0	1	1	3	1	0	0	4	1	0	1	5	1	1	0	6	1	1	1	7
WS1			Number of waits																																			
0	0	0	0																																			
0	0	1	1																																			
0	1	0	2																																			
0	1	1	3																																			
1	0	0	4																																			
1	0	1	5																																			
1	1	0	6																																			
1	1	1	7																																			
2-0	WS0	<p>Wait States0</p> <p>Specifies the number of waits. When address block 0 is accessed, the number of wait states specified with WS0 is automatically inserted. If the DRAM cycle is specified with the CT0 bit of the BCTC register, WS0 is ignored and DRAMC inserts a fixed wait state(s).</p> <p>WS0 specifies the number of waits to be inserted for an I/O block.</p> <table><tr><th colspan="3">WS0</th><th>Number of waits</th></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td><td>2</td></tr><tr><td>0</td><td>1</td><td>1</td><td>3</td></tr><tr><td>1</td><td>0</td><td>0</td><td>4</td></tr><tr><td>1</td><td>0</td><td>1</td><td>5</td></tr><tr><td>1</td><td>1</td><td>0</td><td>6</td></tr><tr><td>1</td><td>1</td><td>1</td><td>7</td></tr></table>	WS0			Number of waits	0	0	0	0	0	0	1	1	0	1	0	2	0	1	1	3	1	0	0	4	1	0	1	5	1	1	0	6	1	1	1	7
WS0			Number of waits																																			
0	0	0	0																																			
0	0	1	1																																			
0	1	0	2																																			
0	1	1	3																																			
1	0	0	4																																			
1	0	1	5																																			
1	1	0	6																																			
1	1	1	7																																			

**6.5.3 Programmable Wait Control Register 1 (PWC1)**

This register is used to set the number of waits for access to address blocks 2 and 3.

The register supports read/write in units of eight bits.

	7	6	5	4	3	2	1	0	
PWC1	0	WS3			0	WS2			Address C0000024H      When reset, 77H

Bit position	Bit name	Description																																				
6-4	WS3	<p>Wait States3</p> <p>Specifies the number of waits. When address block 3 is accessed, the number of wait states specified with WS3 is automatically inserted. If the page-ROM cycle is specified with the CT3 bit of the BCTC register, the number of waits at off-page time is assumed.</p> <table><tr><th colspan="3">WS3</th><th>Number of waits</th></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td><td>2</td></tr><tr><td>0</td><td>1</td><td>1</td><td>3</td></tr><tr><td>1</td><td>0</td><td>0</td><td>4</td></tr><tr><td>1</td><td>0</td><td>1</td><td>5</td></tr><tr><td>1</td><td>1</td><td>0</td><td>6</td></tr><tr><td>1</td><td>1</td><td>1</td><td>7</td></tr></table>	WS3			Number of waits	0	0	0	0	0	0	1	1	0	1	0	2	0	1	1	3	1	0	0	4	1	0	1	5	1	1	0	6	1	1	1	7
WS3			Number of waits																																			
0	0	0	0																																			
0	0	1	1																																			
0	1	0	2																																			
0	1	1	3																																			
1	0	0	4																																			
1	0	1	5																																			
1	1	0	6																																			
1	1	1	7																																			
2-0	WS2	<p>Wait States2</p> <p>Specifies the number of waits. When memory or I/O block 2 is accessed, the number of wait states specified with WS2 is automatically inserted.</p> <table><tr><th colspan="3">WS2</th><th>Number of waits</th></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td><td>2</td></tr><tr><td>0</td><td>1</td><td>1</td><td>3</td></tr><tr><td>1</td><td>0</td><td>0</td><td>4</td></tr><tr><td>1</td><td>0</td><td>1</td><td>5</td></tr><tr><td>1</td><td>1</td><td>0</td><td>6</td></tr><tr><td>1</td><td>1</td><td>1</td><td>7</td></tr></table>	WS2			Number of waits	0	0	0	0	0	0	1	1	0	1	0	2	0	1	1	3	1	0	0	4	1	0	1	5	1	1	0	6	1	1	1	7
WS2			Number of waits																																			
0	0	0	0																																			
0	0	1	1																																			
0	1	0	2																																			
0	1	1	3																																			
1	0	0	4																																			
1	0	1	5																																			
1	1	0	6																																			
1	1	1	7																																			

#### 6.5.4 Programmable Wait Control Register 2 (PWC2)

This register is used to set the number of waits for fly-by DMA transfer.

The register is used to set the number of waits to be inserted during fly-by DMA transfer, during which memory access (DRAM, page-ROM, ROM, and SRAM) and I/O access are performed concurrently.

This register supports read/write in units of eight bits.

	7	6	5	4	3	2	1	0	
PWC2	0	DWS1			0	DWS0			Address C0000026H
									When reset, 77H

Bit position	Bit name	Description																																				
6-4	DWS1	<div>DMA Wait States1 Specifies the number of waits to be inserted during fly-by transfer with DMA channel 1.</div> <table><tr><th colspan="3">DWS1</th><th>Number of waits</th></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td><td>2</td></tr><tr><td>0</td><td>1</td><td>1</td><td>3</td></tr><tr><td>1</td><td>0</td><td>0</td><td>4</td></tr><tr><td>1</td><td>0</td><td>1</td><td>5</td></tr><tr><td>1</td><td>1</td><td>0</td><td>6</td></tr><tr><td>1</td><td>1</td><td>1</td><td>7</td></tr></table>	DWS1			Number of waits	0	0	0	0	0	0	1	1	0	1	0	2	0	1	1	3	1	0	0	4	1	0	1	5	1	1	0	6	1	1	1	7
DWS1			Number of waits																																			
0	0	0	0																																			
0	0	1	1																																			
0	1	0	2																																			
0	1	1	3																																			
1	0	0	4																																			
1	0	1	5																																			
1	1	0	6																																			
1	1	1	7																																			
2-0	DWS0	<div>DMA Wait States0 Specifies the number of waits to be inserted during fly-by transfer with DMA channel 0.</div> <table><tr><th colspan="3">DWS0</th><th>Number of waits</th></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>0</td><td>1</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td><td>2</td></tr><tr><td>0</td><td>1</td><td>1</td><td>3</td></tr><tr><td>1</td><td>0</td><td>0</td><td>4</td></tr><tr><td>1</td><td>0</td><td>1</td><td>5</td></tr><tr><td>1</td><td>1</td><td>0</td><td>6</td></tr><tr><td>1</td><td>1</td><td>1</td><td>7</td></tr></table>	DWS0			Number of waits	0	0	0	0	0	0	1	1	0	1	0	2	0	1	1	3	1	0	0	4	1	0	1	5	1	1	0	6	1	1	1	7
DWS0			Number of waits																																			
0	0	0	0																																			
0	0	1	1																																			
0	1	0	2																																			
0	1	1	3																																			
1	0	0	4																																			
1	0	1	5																																			
1	1	0	6																																			
1	1	1	7																																			

At off-page time during the DRAM cycle, and when the  $\overline{\text{CS0}}$  signal is specified for the DRAM cycle (CT0 in the BCTC register = 01 or 11), the number of waits for fly-by DMA transfer to and from memory block 0 will be influenced by the CWS bit of the DRAM configuration register (DRC), as follows:

**Table 6-3. Number of Waits to Be Inserted during Fly-by DMA Transfer  
(at off-page Time during DRAM Cycle)**

Number of waits specified with the CWS bit	2								3							
Number of waits specified with the DWS bit	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
Number of waits	2	2	2	3	4	5	6	7	3	3	3	3	4	5	6	7



## CHAPTER 7 MEMORY ACCESS CONTROL FUNCTIONS

### 7.1 DRAM CONTROLLER (DRAMC)

The DRAM controller (DRAMC) generates the  $\overline{\text{REFRQ}}$ ,  $\overline{\text{RAS}}$ ,  $\overline{\text{LCAS}}$ , and  $\overline{\text{UCAS}}$  signals, and controls access to DRAM. Access to DRAM is achieved by multiplexing the DRAM row and column addresses and outputting them from the address pins.

The microprocessor assumes the connected DRAM to be of x 4 bits or more, and that it supports high-speed page mode. There are two types of DRAM access cycles, on-page (2 or 3 clock pulses) and off-page (4 or 5 pulses).

Refresh uses the  $\overline{\text{CAS}}$  before  $\overline{\text{RAS}}$  method, allowing the user to set any refresh period. In idle and STOP modes, CBR self-refresh is performed.

#### 7.1.1 Features

- Generates the  $\overline{\text{REFRQ}}$ ,  $\overline{\text{RAS}}$ ,  $\overline{\text{LCAS}}$ , and  $\overline{\text{UCAS}}$  signals.
- Supports DRAM high-speed page mode.
- Address multiplexing function: 8, 9, 10, and 11 bits
- CBR refresh and CBR self-refresh functions

#### 7.1.2 Connecting DRAM

The V821 can be directly connected to DRAM of x 4 bits or more, and which supports a high-speed page mode function (512 Kbytes to 8 Mbytes).

The DRAM cycle is activated upon being selected with CT0 of the bus cycle type control register (BCTC) and address block 0 is accessed.

Table 7-1 lists the DRAM control pins of the V821.

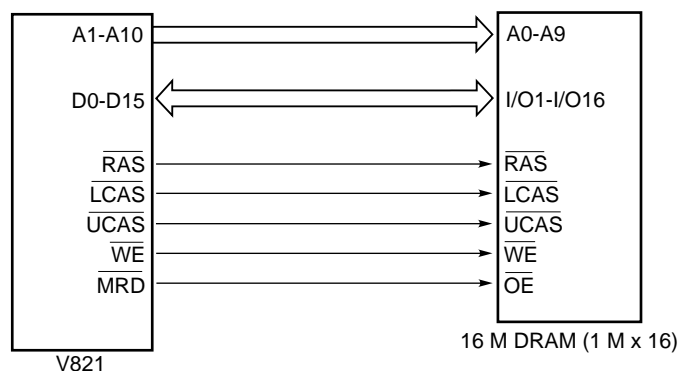
**Table 7-1. DRAM Control Pins**

V821 DRAM control pins	DRAM pins
A0-A23	Address pins
D0-D15	Data pins
$\overline{\text{RAS}}$ , $\overline{\text{REFRQ}}$	$\overline{\text{RAS}}$
$\overline{\text{LCAS}}$	$\overline{\text{CAS}}$
$\overline{\text{UCAS}}$	
$\overline{\text{WE}}$	$\overline{\text{WE}}$
$\overline{\text{MRD}}$	$\overline{\text{OE}}$

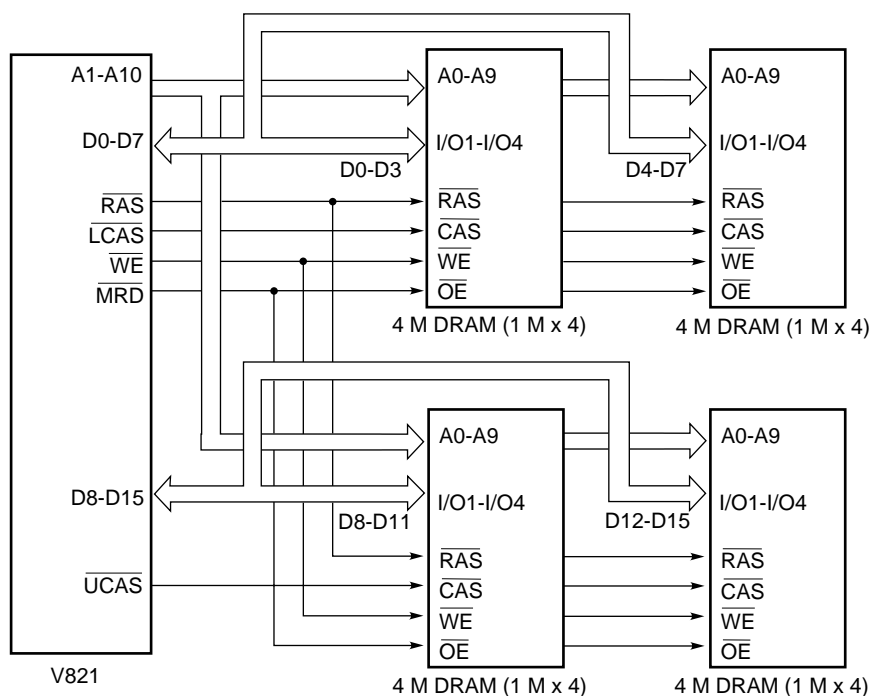
Byte control during DRAM access is performed using only the  $\overline{\text{LCAS}}$  and  $\overline{\text{UCAS}}$  signals. Thus, throughout DRAM access, the  $\overline{\text{UMWR}}$  signals are inactive. To prevent the DRAM from entering test mode, the  $\overline{\text{WE}}$  signal is always inactive during the refresh cycle. **Chapter 2** explains each pin in detail.

Figures 7-1 and 7-2 show examples of connecting DRAM.

**Figure 7-1. Connection of 16-Mbyte (1-Mbyte x 16) DRAM**



**Figure 7-2. Connection of 4-Mbyte (1-Mbyte x 4) DRAM**

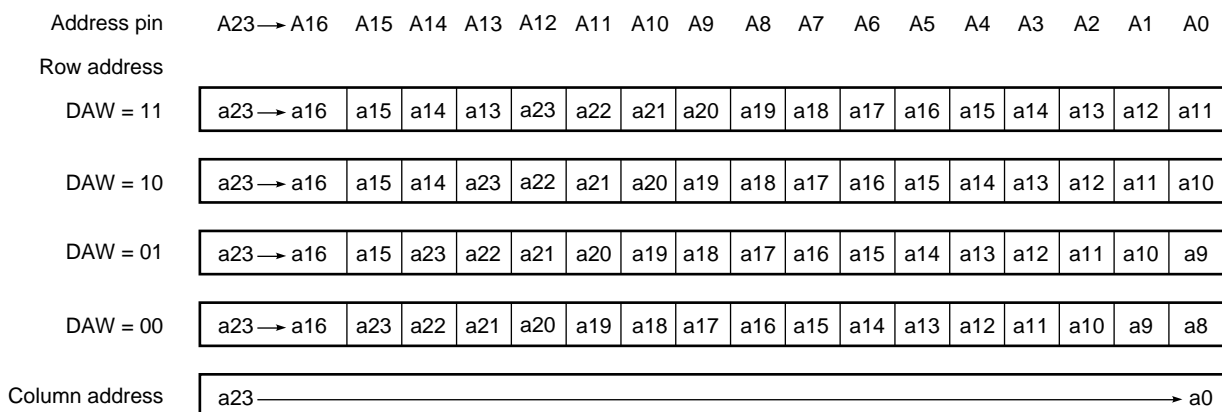


### 7.1.3 Address Multiplexing Function

In the DRAM cycle, row and column addresses are multiplexed according to the value of the DAW bits of the DRAM configuration register (DRC), then output, as shown in Figure 7-3. In Figure 7-3, a0-a23 are the addresses output from the CPU, while A0-A23 are the address pins of the V821. For example, if DAW = 11, row address a12-a22 and column address a1-a11 are output from address pins (A1-A11).

Table 7-2 lists the relationship between the connectable DRAMs and address multiplexing widths. Depending on the connected DRAM, the DRAM space can be between 128 Kbytes and 8 Mbytes.

**Figure 7-3. Output of Row and Column Addresses**



**Table 7-2. Examples of DRAM and Address Multiplexing Width**

Address multiplexing width	DRAM capacity (in bits) and configuration				DRAM space (in bytes)
	256 K	1 M	4 M	16 M	
8 bits	64 K x 4	-	-	-	128 K
9 bits	-	256 K x 4	256 K x 16	-	512 K
	-	-	512 K x 8	-	1 M
10 bits	-	-	1 M x 4	1 M x 16	2 M
	-	-	-	2 M x 8	4 M
11 bits	-	-	-	4 M x 4	8 M

**7.1.4 DRAM Configuration Register (DRC)**

This register is used to set the address multiplexing width, number of waits, and the data output timing used during DRAM access.

The register supports read/write in units of eight bits.

	7	6	5	4	3	2	1	0	
DRC	ADC	PAE	0	0	0	CWS	DAW		Address C0000028H           When reset, 81H

Bit position	Bit name	Description															
7	ADC	Avoid Data Conflict Sets the output timing of the write data. Effective during SRAM and I/O cycles other than the DRAM cycle. 0: The output of data starts at the falling edge of a clock pulse in T1. 1: The output of data starts at the rising edge of a clock pulse in T2. During the DRAM on-page write cycle, one wait state is inserted when ADC is 1, unless the previous cycle was a write cycle.															
6	PAE	Page-mode Access Enable Controls the activation of the on-page access cycle conforming to the DRAM high-speed page mode. 0: Activation disabled 1: Activation enabled															
2	CWS	Compulsory Wait State Sets the number of waits to be inserted in the DRAM off-page cycle. 0: 2 waits 1: 3 waits															
1, 0	DAW	DRAM Address Width Sets the address multiplexing width used when a row address is output during the DRAM cycle. <table border="1"> <thead> <tr> <th colspan="2">DAW</th><th>Multiplexing width</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>8 bits</td></tr> <tr> <td>0</td><td>1</td><td>9 bits</td></tr> <tr> <td>1</td><td>0</td><td>10 bits</td></tr> <tr> <td>1</td><td>1</td><td>11 bits</td></tr> </tbody> </table>	DAW		Multiplexing width	0	0	8 bits	0	1	9 bits	1	0	10 bits	1	1	11 bits
DAW		Multiplexing width															
0	0	8 bits															
0	1	9 bits															
1	0	10 bits															
1	1	11 bits															

### 7.1.5 DRAM Read/Write Cycles

The DRAM cycle is activated when CT0 of the BCTC register is 01 or 11 address block 0 is accessed. The width of the row and column addresses to be output to DRAM is selected by setting the DAW bits of the DRC register.

The DRAM write cycle is an early write cycle.

There are two types of DRAM read/write cycles, on-page cycle and off-page cycle, according to the number of clock pulses in the cycle and the timing of the DRAM control signals ( $\overline{\text{RAS}}$ ,  $\overline{\text{LCAS}}$ , and  $\overline{\text{UCAS}}$ ). In the on-page write cycle, a wait state is inserted when ADC of the DRC register is 1, unless the previous cycle was a write cycle.

**Table 7-3. Differences between on-page and off-page Cycles**

Cycle	Conditions	Number of clock pulses
on-page cycle	<ul style="list-style-type: none"> <li><math>\overline{\text{RAS}} = 0</math> and the same row address as that in the previous DRAM read or write cycle</li> </ul>	2 (0 wait) <b>Note</b>
off-page cycle	<ul style="list-style-type: none"> <li><math>\overline{\text{RAS}} = 1</math></li> <li><math>\overline{\text{RAS}} = 0</math> and a row address that differs from that used in the previous DRAM read or write cycle</li> </ul>	4 + n (2 + n waits)

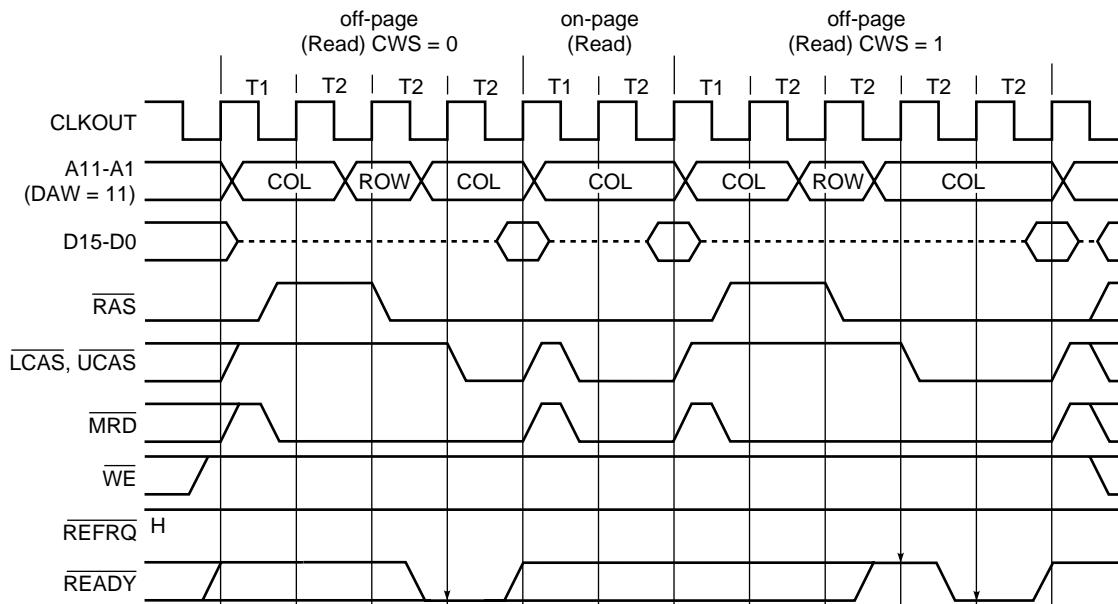
**Note** During the write cycle, when ADC of the DRC register is 1 and the previous bus cycle was a read cycle for other than internal I/O: 3 (1 wait) clock pulses

**Remark** n: Values of the CWS bits of the DRC register

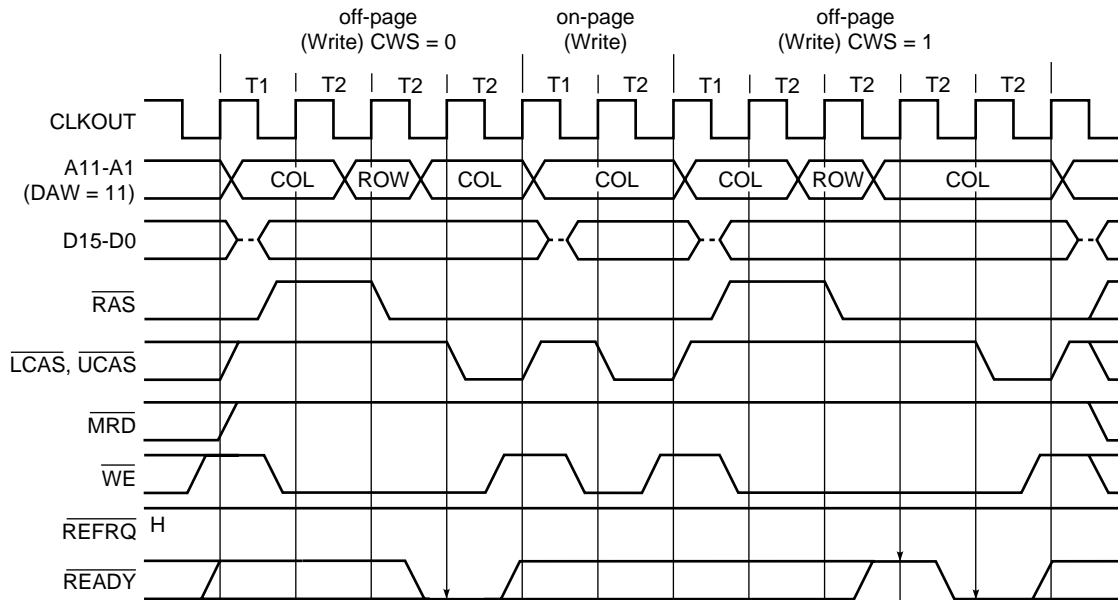
Once active, the  $\overline{\text{RAS}}$  signal remains active until an off-page cycle or refresh cycle occurs; when an off-page cycle occurs, the signal becomes active again after the completion of RAS precharge; when a refresh cycle occurs, it becomes inactive after performing refresh.

In an off-page cycle, wait control can be applied by using the  $\overline{\text{READY}}$  pin.

★ The DRAM cycle for address block 0 is always in the off-page cycle after the I/O cycle.

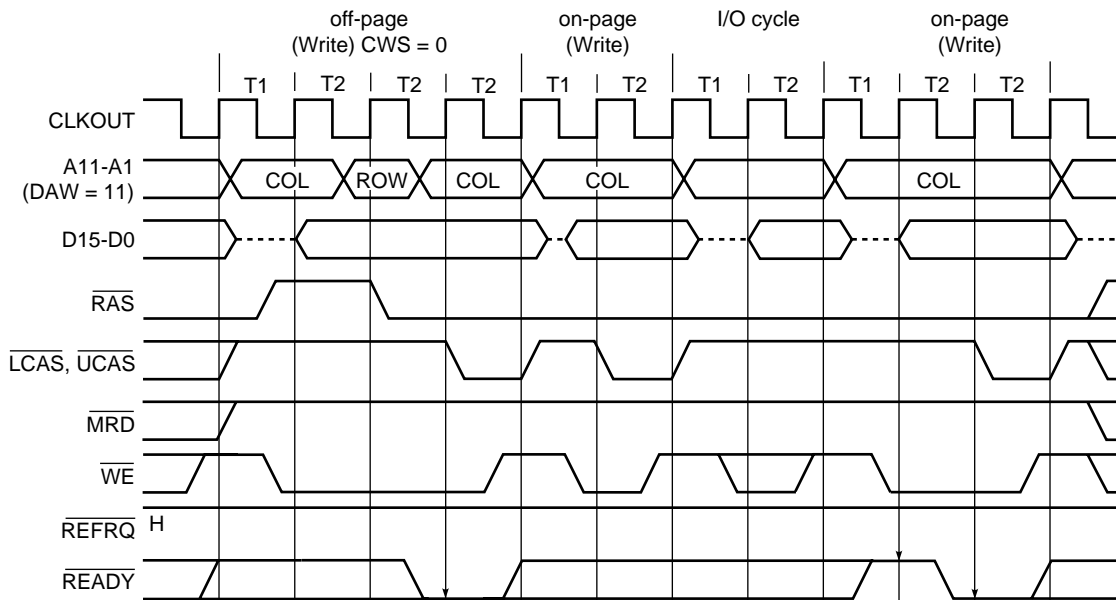
**Figure 7-4. DRAM Read Cycle**

- Remarks 1.** ROW : Row address  
COL : Column address
- 2.** A broken line indicates a high impedance.
- 3.** An arrow indicates a sampling timing.

**Figure 7-5. DRAM Write Cycle****(1) When ADC = 0****Remarks 1.** ROW : Row address

COL : Column address

**2.** A broken line indicates a high impedance.**3.** An arrow indicates a sampling timing.

**(2) When ADC = 1**

- Remarks 1.** ROW : Row address  
COL : Column address
- 2.** A broken line indicates a high impedance.
- 3.** An arrow indicates a sampling timing.



### 7.1.6 Wait Control for DRAM Cycle

Programmable wait control for the DRAM cycle is possible during fly-by DMA transfer only. The number of waits to be inserted is set with programmable wait control register 2 (PWC2). (See **Section 6.5.4**.)

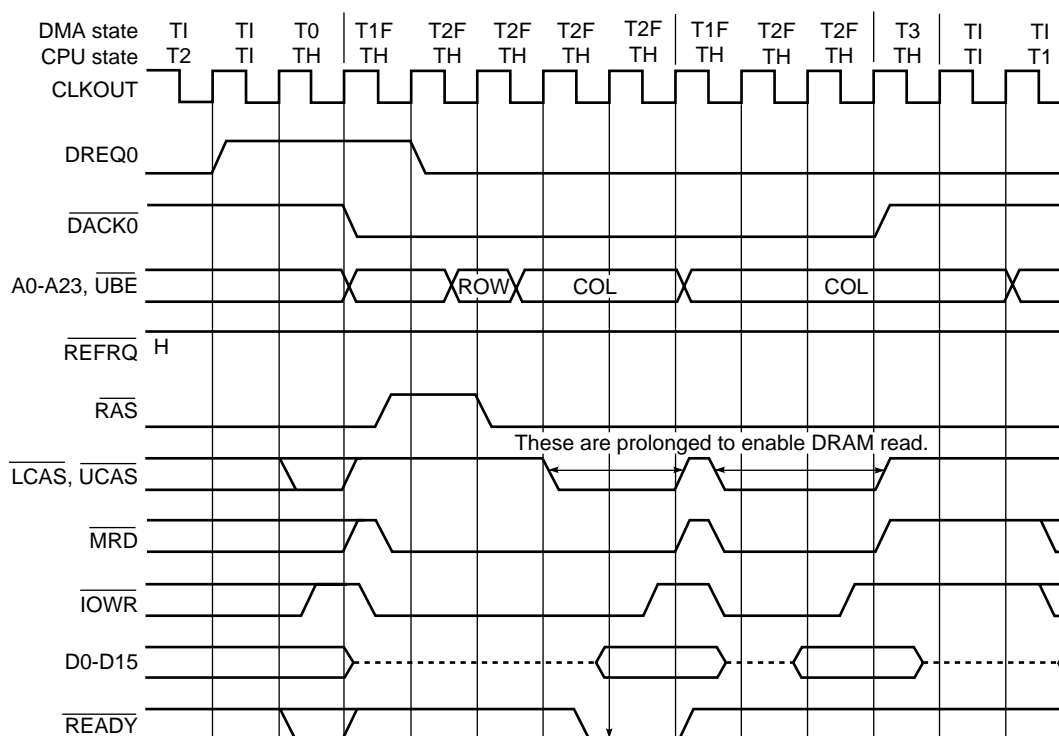
The timing of wait insertion differs during read and write. During DRAM read, the access time for the  $\overline{\text{LCAS}}$  and  $\overline{\text{UCAS}}$  signals is extended; during DRAM write, the timing at which the  $\overline{\text{LCAS}}$  and  $\overline{\text{UCAS}}$  signal become active is delayed.

Wait control using the external  $\overline{\text{READY}}$  pin is effective during the off-page cycle for normal DRAM access, DRAM access with fly-by DMA, and during the CBR refresh cycle.

The  $\overline{\text{READY}}$  signal is sampled from the T2 cycle at the third clock pulse, at the rising edge of the clock pulse, half a clock pulse earlier than during a normal SRAM cycle.

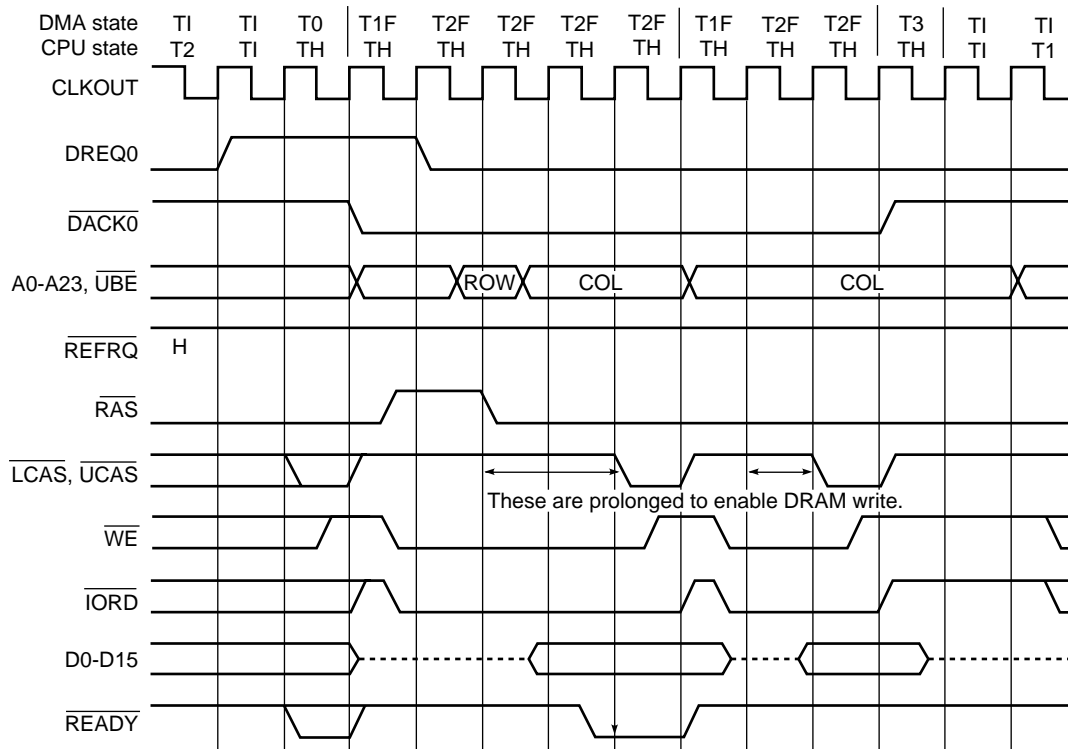
Figures 7-6 and 7-7 show the timings of the DRAM read and write cycles when waits are inserted.

**Figure 7-6. Insertion of Wait State (1 Wait) during DRAM Read for Fly-by DMA Transfer (DRAM to External I/O)**



- Remarks 1.** ROW : Row address  
COL : Column address
- 2.** A broken line indicates a high impedance.
- 3.** An arrow indicates a sampling timing.

**Figure 7-7. Insertion of Wait State (1 Wait) during DRAM Write for Fly-by DMA Transfer (External I/O to DRAM)**



**Remarks 1.** ROW : Row address

COL : Column address

**2.** A broken line indicates a high impedance.

**3.** An arrow indicates a sampling timing.

### 7.1.7 Refresh Function

DRAMC can automatically generate the distributed CBR refresh cycle needed to refresh external DRAM. Whether refresh should be enabled or disabled, and the refresh interval, are specified using the refresh control register (RFC).

While another bus master is occupying a bus, DRAMC cannot forcibly acquire the bus. In this case, in response to a refresh request issued from DRAMC, BAU makes the  $\overline{\text{HLDAK}}$  pin inactive to post notification of the occurrence of a refresh request. In this state, by making the  $\overline{\text{HLDRQ}}$  pin inactive, the refresh cycle is activated. **Section 5.3** explains the priorities of bus cycles other than refresh.

#### (1) Refresh control register (RFC)

This register is used to specify whether refresh should be enabled or disabled, as well as the refresh interval.

The refresh interval is determined using the following formula:

$$\text{Refresh interval (in } \mu\text{s)} = \text{Refresh count clock pulses (T}_{\text{RCY}}) \times \text{Interval factor}$$

The refresh count clock pulses and interval factor are specified with the RCC and RI bits of the RFC register, respectively.

The register supports read/write in units of eight bits.

	7	6	5	4	3	2	1	0	
RFC	REN	RCC	RI						Address C000002AH
									When reset, 80H

Bit position	Bit name	Description																																				
7	REN	Refresh Enable Specifies whether to enable or disable CBR refresh. 0: Refresh disabled 1: Refresh enabled																																				
6	RCC	Refresh Count Clock Specifies refresh count clock pulses ( $T_{RCY}$ ). 0: $T_{RCY} = 32/\phi$ 1: $T_{RCY} = 128/\phi$																																				
5-0	RI	Refresh Interval Sets the interval factor for the interval timer used for creating refresh timing. <table><tr><th colspan="5">RI</th><th>Interval factor</th></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>2</td></tr><tr><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>3</td></tr><tr><td></td><td></td><td>:</td><td></td><td></td><td>:</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>64</td></tr></table>	RI					Interval factor	0	0	0	0	0	1	0	0	0	0	1	2	0	0	0	1	0	3			:			:	1	1	1	1	1	64
RI					Interval factor																																	
0	0	0	0	0	1																																	
0	0	0	0	1	2																																	
0	0	0	1	0	3																																	
		:			:																																	
1	1	1	1	1	64																																	

**(2) DRAM refresh intervals and examples of setting the RFC register**

Table 7-4 lists the refresh cycles and the refresh intervals for a DRAM of between 256 Kbits to 16 Mbits.

Table 7-5 lists examples of setting the interval factor (set with the RI bit of the RFC register) needed to determine the refresh interval.

**Table 7-4. DRAM Refresh Intervals**

Capacity	Refresh cycles (cycles/ms)	Refresh interval (μs)
256 K	256/4	15.6
1 M	512/8	15.6
	512/64	125
4 M	512/128	250
	1 K/16	15.6
	1 K/128	125
16 M	1 K/256	250
	2 K/256	125
	4 K/64	15.6
	4 K/256	62.5

**Table 7-5. Examples of Setting the Interval Factor**

- When  $T_{RCY} = 32/\phi$

Standard refresh interval (μs)	Value of the interval factor			
	12.5 MHz	16 MHz	20 MHz	25 MHz
15.6	5 (12.8)	7 (14.0)	9 (14.4)	11 (14.1)
62.5	24 (61.4)	31 (62.0)	38 (60.8)	48 (61.4)
125	48 (122.9)	62 (124)	-	-
250	-	-	-	-

- When  $T_{RCY} = 128/\phi$

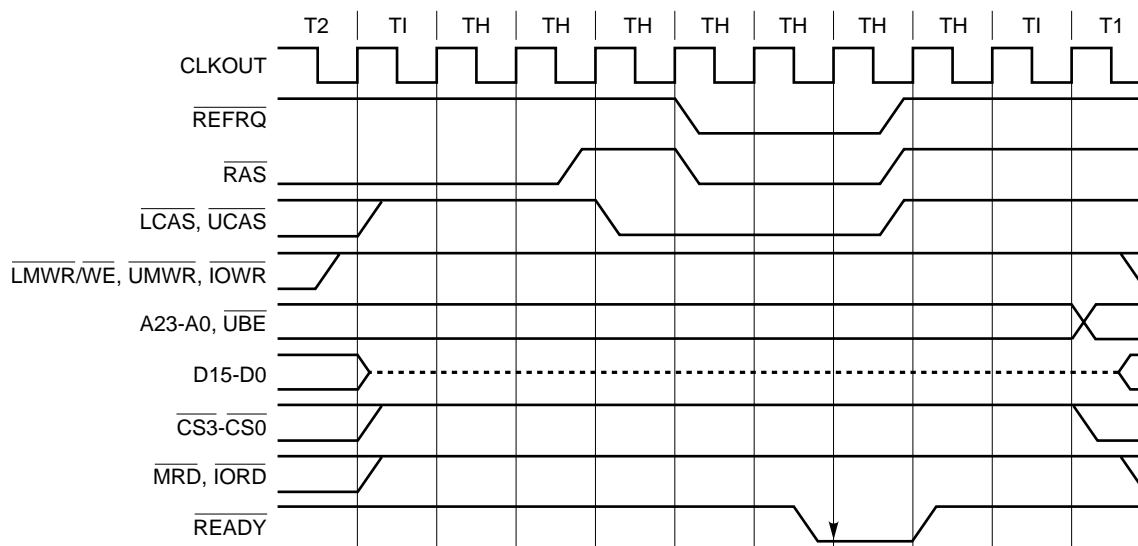
Standard refresh interval ( $\mu\text{s}$ )	Value of the interval factor			
	12.5 MHz	16 MHz	20 MHz	25 MHz
15.6	-	-	-	-
62.5	-	-	-	-
125	-	-	19 (121.6)	24 (122.9)
250	24 (245.8)	31 (248.0)	38 (243.2)	48 (245.8)

**Remark** The values in parentheses are calculated refresh interval values ( $\mu\text{s}$ ).  
 -: Not applicable.

### (3) CBR refresh cycle

The refresh cycle is performed by holding the bus, as shown in Figure 7-8. One bus cycle requires at least seven clock pulses, or at least nine clock pulses when including the preceding and succeeding idle cycles.

**Figure 7-8. CBR Refresh Cycle**



- Remarks**
1. A broken line indicates a high impedance.
  2. An arrow indicates a sampling timing.

### 7.1.8 Self-Refresh Function

DRAMC generates the CBR self-refresh cycle in idle and STOP modes. The self-refresh cycle is activated by setting the SMD bit of the standby control register (STBC) to idle or STOP mode and executing the HALT instruction.

To enable DRAM to perform self-refresh, the standard  $\overline{\text{RAS}}$  pulse width for DRAM (100  $\mu\text{s}$  or greater) must be ensured.

Self-refresh is canceled using the  $\overline{\text{RESET}}$  or  $\overline{\text{NMI}}$  pin. The procedure for cancellation by  $\overline{\text{RESET}}$  input is the same as that for normal reset. The other procedures for canceling self-refresh are explained below.

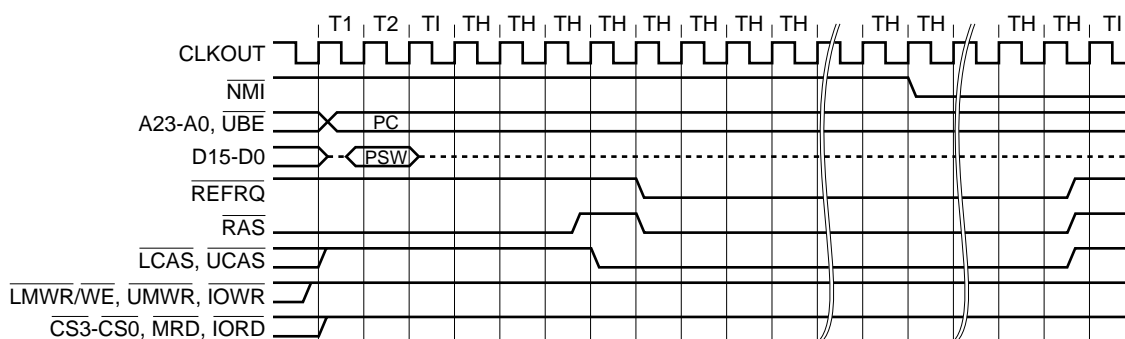
### (1) Cancellation with $\overline{\text{NMI}}$ input

As soon as an  $\overline{\text{NMI}}$  interrupt occurs during the CBR self-refresh cycle in IDLE mode, DRAMC makes the  $\overline{\text{REFRQ}}$ ,  $\overline{\text{RAS}}$ ,  $\overline{\text{LCAS}}$ , and  $\overline{\text{UCAS}}$  signals inactive to cancel CBR self-refresh.

When an  $\overline{\text{NMI}}$  interrupt occurs during the CBR self-refresh cycle in STOP mode, DRAMC secures the time required for oscillation to settle, then makes the above signals inactive, thus canceling CBR self-refresh.

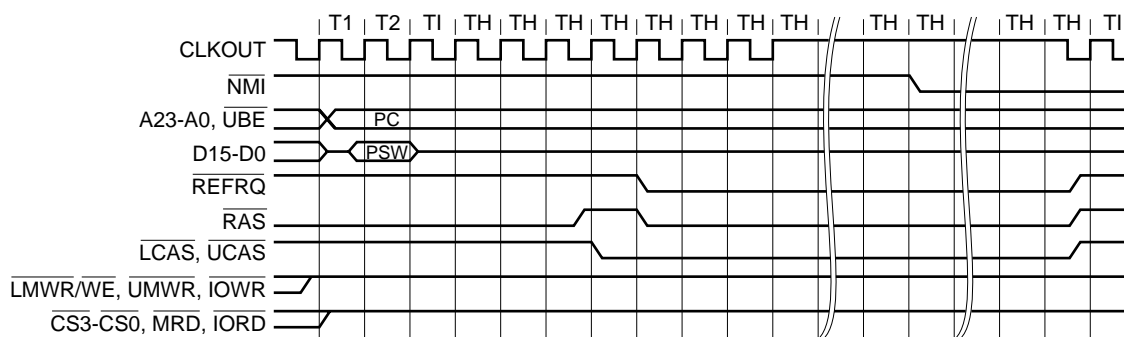
**Figure 7-9. Cancellation of CBR Self Refresh with  $\overline{\text{NMI}}$  Input**

**(a) In IDLE mode**



**Remarks 1.** A broken line indicates a high impedance.

2. The noise eliminator introducing an analog delay is used for input of  $\overline{\text{NMI}}$ . Therefore, the time from when  $\overline{\text{NMI}}$  is input until  $\overline{\text{REFRQ}}$ ,  $\overline{\text{RAS}}$ ,  $\overline{\text{LCAS}}$ , and  $\overline{\text{UCAS}}$  become inactive depends on the use condition.

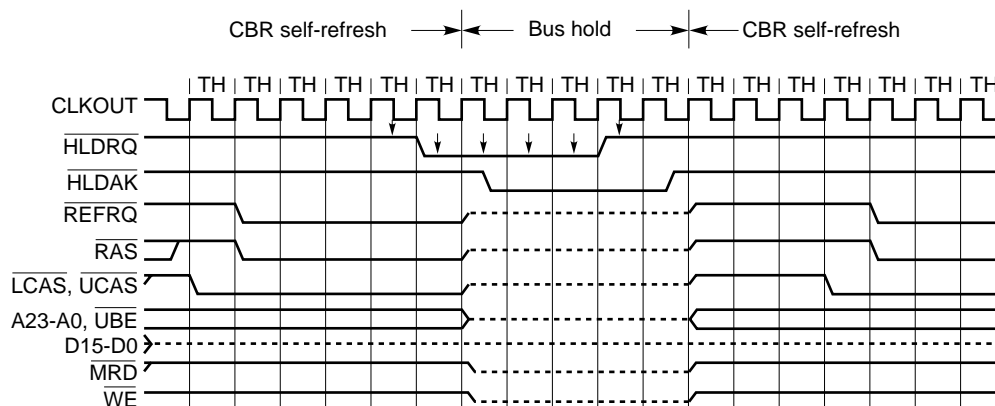
**(b) In STOP mode**

**Remark** A broken line indicates a high impedance.

**(2) Cancellation with  $\overline{\text{HLDRQ}}$  input (in IDLE mode)**

When a hold request is issued during the CBR self-refresh cycle in IDLE mode, the DRAM control pins are set to high impedance and enter the bus hold state. When the hold request is canceled, the CBR self refresh cycle is reactivated.

**Figure 7-10. Cancellation of CBR Self-Refresh with  $\overline{\text{HLDRQ}}$  Input (in IDLE mode)**



**Remarks 1.** A broken line indicates a high impedance.

**2.** An arrow indicates a sampling timing.

## 7.2 ROM CONTROLLER (ROMC)

The ROM controller supports access to ROM having a page access function (page-ROM).

The ROM controller performs address comparison with the previous bus cycle and performs wait control for normal access (off-page)/page access (on-page). It supports page widths of 8-64 bytes.

The page-ROM cycle is supported with address block 3.

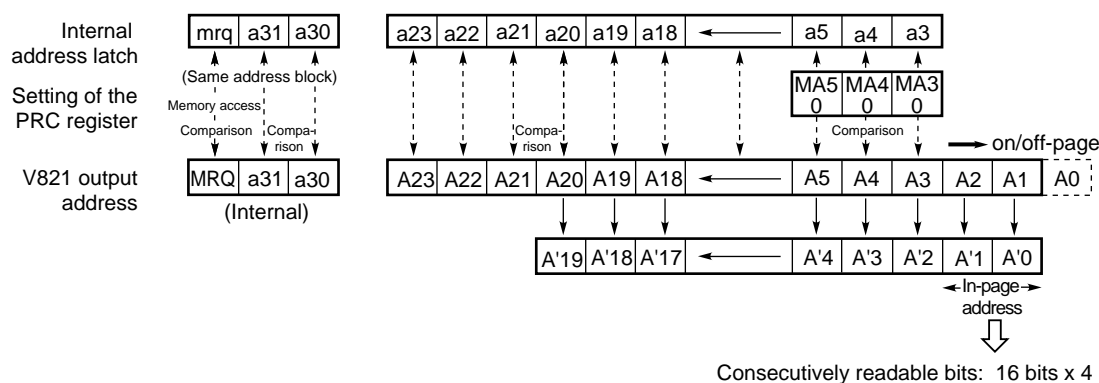
### 7.2.1 on-page/off-page Decision

Whether the page-ROM cycle is on-page or off-page is determined by latching the address during the previous cycle and comparing it with the address during the current cycle.

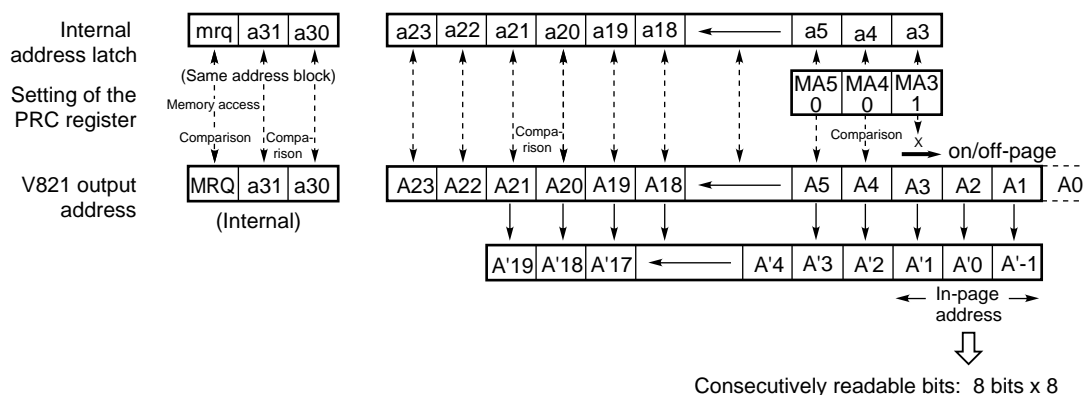
The address(es) (A3-A5) to be masked (not compared) is set using the page-ROM configuration register (PRC), according to the configuration of the connected page-ROM and the number of consecutively readable bits.

**Figure 7-11. on-page/off-page Decision When ROM Having a Page Access Function Is Connected**

#### (1) For 16-Mbit ROM (1-Mbit x 16)



#### (2) For 16-Mbit ROM (2-Mbit x 8)



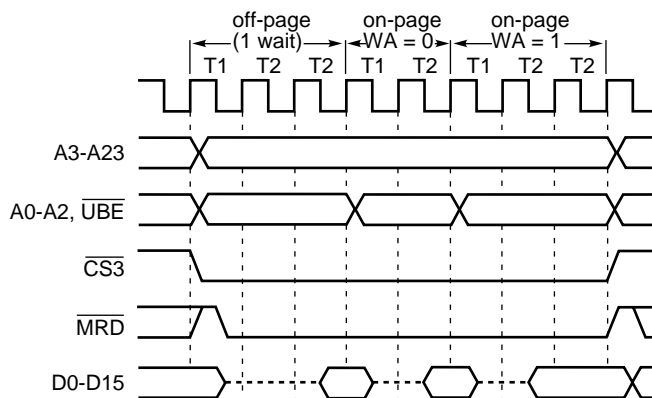


### 7.2.2 Page-ROM Access

Access to page-ROM is made with 0 or 1 wait during the on-page cycle, and with 0-7 waits during the off-page cycle. The number of waits in the on-page cycle is specified with the WA bit of the page-ROM configuration register (PRC).

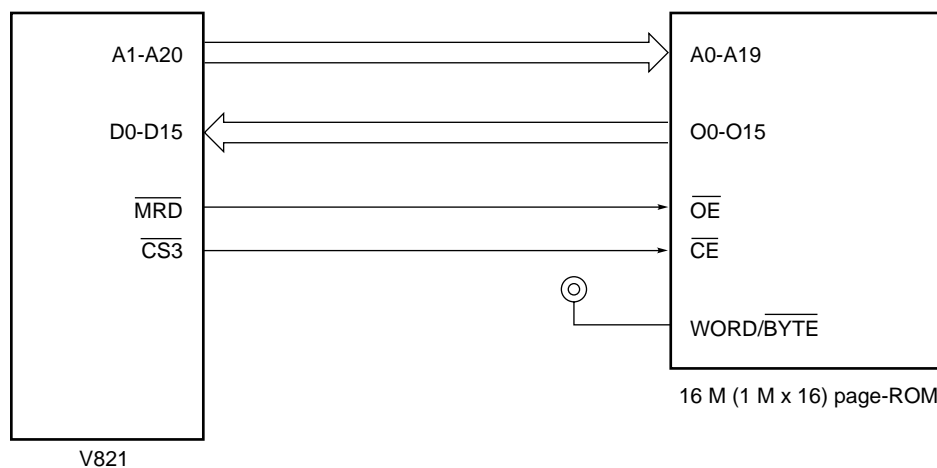
Figure 7-12 shows the timing of the page-ROM cycle. Figures 7-13 and 1-14 show examples of connecting page-ROM to the V821.

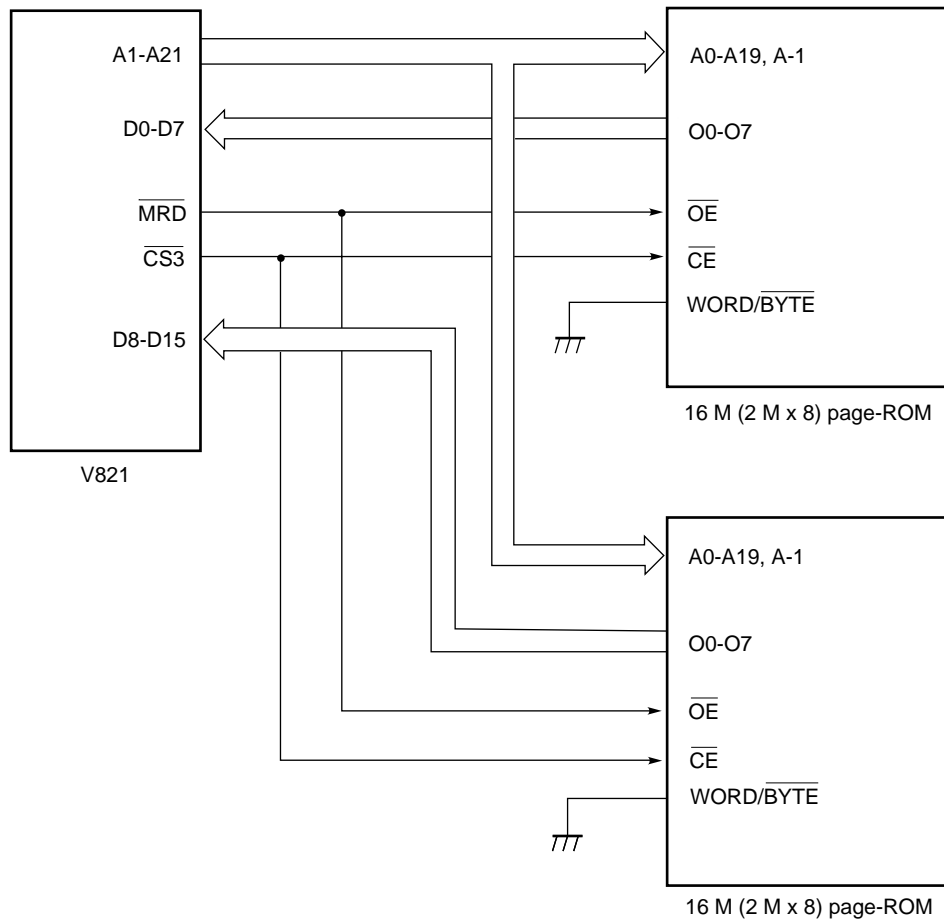
**Figure 7-12. Timing of the Page-ROM Cycle (16-Mbit (1-Mbit x 16) Page-ROM)**



**Remark** A broken line indicates a high impedance.

**Figure 7-13. Connection of 16-Mbit (1-Mbit x 16) Page-ROM**



**Figure 7-14. Connection of 16-Mbit (2-Mbit x 8) Page-ROM**

### 7.2.3 Page-ROM Configuration Register (PRC)

This register is used to set the address(es) (A3-A5) to be masked (not compared) according to the configuration of the connected page-ROM and the number of consecutively readable bits. It is also used to set a wait according to the system clock.

The register supports read/write in units of eight bits.

	7	6	5	4	3	2	1	0		
PRC	WA	0	0	0	0	MA5	MA4	MA3	Address C000002CH	When reset, 80H

Bit position	Bit name	Description																				
7	WA	<p>Wait Adjustment</p> <p>Sets a wait according to the system clock.</p> <p>The wait set with this bit is inserted in the on-page cycle only. In the off-page cycle, the wait(s) set with WS3 of the PWC1 register are inserted.</p> <p>0: 0 wait</p> <p>1: 1 wait</p>																				
2-0	MA5-MA3	<p>Mask Address</p> <p>Masks an address(es) (A5-A3) according to the values of MA5-MA3 (1 for mask). A masked address is not compared as part of on/off-page decision.</p> <p>Set the bits according to the number of consecutively readable bits.</p> <table><tr><th>MA5</th><th>MA4</th><th>MA3</th><th>Number of consecutively readable bits</th></tr><tr><td>0</td><td>0</td><td>0</td><td>4 words x 16 bits (8 words x 8 bits)</td></tr><tr><td>0</td><td>0</td><td>1</td><td>8 words x 16 bits (16 words x 8 bits)</td></tr><tr><td>0</td><td>1</td><td>1</td><td>16 words x 16 bits (32 words x 8 bits)</td></tr><tr><td>1</td><td>1</td><td>1</td><td>32 words x 16 bits (64 words x 8 bits)</td></tr></table>	MA5	MA4	MA3	Number of consecutively readable bits	0	0	0	4 words x 16 bits (8 words x 8 bits)	0	0	1	8 words x 16 bits (16 words x 8 bits)	0	1	1	16 words x 16 bits (32 words x 8 bits)	1	1	1	32 words x 16 bits (64 words x 8 bits)
MA5	MA4	MA3	Number of consecutively readable bits																			
0	0	0	4 words x 16 bits (8 words x 8 bits)																			
0	0	1	8 words x 16 bits (16 words x 8 bits)																			
0	1	1	16 words x 16 bits (32 words x 8 bits)																			
1	1	1	32 words x 16 bits (64 words x 8 bits)																			

[MEMO]

## CHAPTER 8 DMA FUNCTIONS (DMA CONTROLLER)

The V821 includes a DMA (Direct Memory Access) controller that executes and controls DMA transfer.

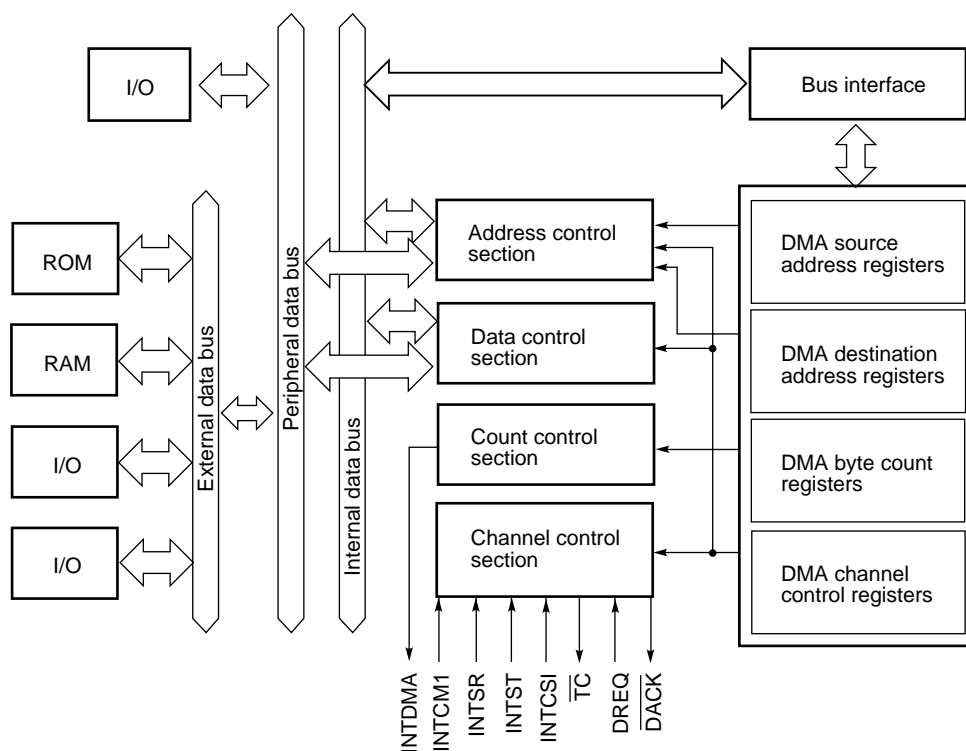
The DMAC (DMA controller) transfers data between memory and I/O, or within memory, based on DMA requests issued by the built-in peripheral hardware (serial interfaces and timer), external DREQ pins, or software triggers.

### 8.1 FEATURES

- Two independent DMA channels
- Transfer units: 8/16 bits
- Maximum transfer count: 65,536 ( $2^{16}$ )
- Two types of transfer
  - Fly-by (one-cycle) transfer
  - Two-cycle transfer
- Three transfer modes
  - Single transfer mode
  - Single-step transfer mode
  - Block transfer mode
- Transfer requests
  - External DREQ pin (x 2)
  - Requests from built-in peripheral hardware (serial interfaces and timer)
  - Requests from software
- Transfer objects
  - Memory to I/O and vice versa
  - Memory to memory and vice versa
- Programmable wait function
- DMA transfer end output signal ( $\overline{TC}$ )

## 8.2 CONFIGURATION

Figure 8-1. Block Diagram of DMAC



## 8.3 DMA CONTROL REGISTERS

### 8.3.1 DMA Source Address Registers 0 and 1 (DSA0 and DSA1)

These registers are used to set the DMA source addresses (24 bits each) and the DMA source address blocks for DMA channels 0 and 1. They are divided into two 16-bit registers, DSAnH and DSAnL, respectively.

During DMA transfer, the registers store the next DMA source addresses.

When fly-by transfer is specified with the TTYP bits of the DMA channel control registers (DCHC0 and DCHC1), the memory addresses are set with the DSAn registers. The settings made with DMA destination address registers 0 and 1 (DDA0 and DDA1) are ignored.

**(1) DMA source address registers 0H and 1H (DSA0H and DSA1H)**

These registers support read/write in 16-bit units.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
DSA0H	0	0	0	0	0	SBT	SBN	SA23	SA22	SA21	SA20	SA19	SA18	SA17	SA16		Address	When reset,
DSA1H																	C0000040H	Not defined
																	C0000048H	Not defined

Bit position	Bit name	Description															
10	SBT	Source Block Type Specifies whether the source address block will be a memory or I/O block. Must be set to 0 for fly-by transfer. 0: memory 1: I/O															
9, 8	SBN	Source Block Number Specifies the source address block number. <table border="1"> <tr> <th colspan="2">SBN</th><th>Address block</th></tr> <tr> <td>0</td><td>0</td><td>0</td></tr> <tr> <td>0</td><td>1</td><td>1</td></tr> <tr> <td>1</td><td>0</td><td>2</td></tr> <tr> <td>1</td><td>1</td><td>3</td></tr> </table>	SBN		Address block	0	0	0	0	1	1	1	0	2	1	1	3
SBN		Address block															
0	0	0															
0	1	1															
1	0	2															
1	1	3															
7-0	SA23-SA16	Source Address Sets the DMA source address (A23-A16). During DMA transfer, it stores the next DMA source address. During fly-by transfer, it stores a memory address.															

**(2) DMA source address registers 0L and 1L (DSA0L and DSA1L)**

The registers support read/write in 16-bit units.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
DSA0L	SA15	SA14	SA13	SA12	SA11	SA10	SA9	SA8	SA7	SA6	SA5	SA4	SA3	SA2	SA1	SA0	Address	When reset,
DSA1L																	C0000042H	Not defined
																	C000004AH	Not defined

Bit position	Bit name	Description
15-0	SA15-SA0	Source Address Sets the DMA source address (A15-A0). During DMA transfer, it stores the next DMA source address.

**8.3.2 DMA Destination Address Registers 0 and 1 (DDA0 and DDA1)**

These registers are used to set the DMA destination addresses (24 bits each) and the DMA destination address blocks for DMA channels 0 and 1. They are divided into two 16-bit registers, DDAnH and DDAnL, respectively.

During DMA transfer, these registers store the next DMA destination addresses.

When fly-by transfer is specified with the TTYP bits of the DMA channel control registers (DCHC0 and DCHC1), the settings made with registers DDA0 and DDA1 are ignored.

**(1) DMA destination address registers 0H and 1H (DDA0H and DDA1H)**

The registers allow read/write in 16-bit units.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
DDA0H	0	0	0	0	0	DBT	DBN	DA23	DA22	DA21	DA20	DA19	DA18	DA17	DA16		Address	When reset,
DDA1H																	C0000044H	Not defined
																	C000004CH	Not defined

Bit position	Bit name	Description															
10	DBT	Destination Block Type Specifies whether the destination address block will be a memory or I/O block. 0: memory 1: I/O															
9, 8	DBN	Destination Block Number Specifies the destination address block number. <table border="1"> <tr> <th colspan="2">DBN</th><th>Address block</th></tr> <tr> <td>0</td><td>0</td><td>0</td></tr> <tr> <td>0</td><td>1</td><td>1</td></tr> <tr> <td>1</td><td>0</td><td>2</td></tr> <tr> <td>1</td><td>1</td><td>3</td></tr> </table>	DBN		Address block	0	0	0	0	1	1	1	0	2	1	1	3
DBN		Address block															
0	0	0															
0	1	1															
1	0	2															
1	1	3															
7-0	DA23-DA16	Destination Address Sets the DMA destination address (A23-A16). During DMA transfer, it stores the next DMA source address. During fly-by transfer, it is ignored.															

**(2) DMA destination address registers 0L and 1L (DDA0L and DDA1L)**

These registers support read/write in 16-bit units.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
DDA0L	DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0	Address	When reset,
DDA1L																	C0000046H	Not defined
																	C000004EH	Not defined

Bit position	Bit name	Description
15-0	DA15-DA0	Destination Address Sets the DMA destination address (A15-A0). During DMA transfer, it stores the next DMA destination address. During fly-by transfer, it is ignored.



**8.3.3 DMA Byte Count Registers 0 and 1 (DBC0 and DBC1)**

These 16-bit registers are used to set the byte transfer counts for DMA channels 0 and 1.

They store the remaining transfer counts during DMA transfer.

These registers are decremented by 1 for byte transfer and by two for 16-bit transfer. Transfer ends when a borrow occurs. Thus, "transfer count –1" should be set for byte transfer and "(transfer count –1) x 2" for 16-bit transfer.

The registers support read/write in 16-bit units.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
DBC0	BC																Address	When reset,
DBC1																	C0000050H	Not defined
																	C0000052H	Not defined

Bit position	Bit name	Description										
15-0	BC	<div>Byte Count Sets the byte transfer count. During DMA transfer, it stores the remaining byte transfer count.</div> <table><tr><th>BC</th><th>State</th></tr><tr><td>00H</td><td>Byte transfer count 1 or the remaining byte transfer count</td></tr><tr><td>01H</td><td>Byte transfer count 2 or the remaining byte transfer count</td></tr><tr><td>⋮</td><td>⋮</td></tr><tr><td>FFFFH</td><td>Byte transfer count 65,536 (2<sup>16</sup>) or the remaining byte transfer count</td></tr></table>	BC	State	00H	Byte transfer count 1 or the remaining byte transfer count	01H	Byte transfer count 2 or the remaining byte transfer count	⋮	⋮	FFFFH	Byte transfer count 65,536 (2 <sup>16</sup> ) or the remaining byte transfer count
BC	State											
00H	Byte transfer count 1 or the remaining byte transfer count											
01H	Byte transfer count 2 or the remaining byte transfer count											
⋮	⋮											
FFFFH	Byte transfer count 65,536 (2 <sup>16</sup> ) or the remaining byte transfer count											

### 8.3.4 DMA Channel Control Registers 0 and 1 (DCHC0 and DCHC1)

These 16-bit registers are used to control the DMA transfer modes for DMA channels 0 and 1.

Note that 16-bit transfer with an odd address is not possible. (The A0 pin must be set to 0).

These registers support read/write in 16-bit units. (Bit 15 can be read only.)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
DCHC0	TC	0	0	0	SAD	DAD	TTYP	TDIR	TM	DS	EN						Address	When reset,
DCHC1																	C0000054H	0000H
																	C0000056H	0000H

Bit position	Bit name	Description															
15	TC	<p>Terminal Count</p> <p>This status bit indicates whether DMA transfer through DMA channel n has ended.</p> <p>This bit can only be read. It is set when DMA transfer ends with a terminal count and reset when it is read.</p> <p>0: DMA transfer has not ended.</p> <p>1: DMA transfer has ended.</p>															
11, 10	SAD	<p>Source Address count Direction</p> <p>Sets the count direction of the source address for DMA channel n.</p> <table border="1"> <thead> <tr> <th colspan="2">SAD</th><th>Count direction</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>Increment</td></tr> <tr> <td>0</td><td>1</td><td>Decrement</td></tr> <tr> <td>1</td><td>0</td><td>Fixed</td></tr> <tr> <td>1</td><td>1</td><td>Setting prohibited</td></tr> </tbody> </table>	SAD		Count direction	0	0	Increment	0	1	Decrement	1	0	Fixed	1	1	Setting prohibited
SAD		Count direction															
0	0	Increment															
0	1	Decrement															
1	0	Fixed															
1	1	Setting prohibited															
9, 8	DAD	<p>Destination Address count Direction</p> <p>Sets the count direction of the destination address for DMA channel n.</p> <table border="1"> <thead> <tr> <th colspan="2">DAD</th><th>Count direction</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>Increment</td></tr> <tr> <td>0</td><td>1</td><td>Decrement</td></tr> <tr> <td>1</td><td>0</td><td>Fixed</td></tr> <tr> <td>1</td><td>1</td><td>Setting prohibited</td></tr> </tbody> </table>	DAD		Count direction	0	0	Increment	0	1	Decrement	1	0	Fixed	1	1	Setting prohibited
DAD		Count direction															
0	0	Increment															
0	1	Decrement															
1	0	Fixed															
1	1	Setting prohibited															

Bit position	Bit name	Description																																				
7-5	TTYP	<div>Transfer Type Sets the cause of the activation of DMA transfer and the transfer direction.</div> <table><tr><th colspan="3">TTYP</th><th>Activation cause and transfer direction</th></tr><tr><td>0</td><td>0</td><td>0</td><td>Activated by the DREQn signal: two-cycle transfer</td></tr><tr><td>0</td><td>0</td><td>1</td><td>Activated by the DREQn signal: fly-by transfer</td></tr><tr><td>0</td><td>1</td><td>0</td><td>Activated by software: two-cycle transfer</td></tr><tr><td>0</td><td>1</td><td>1</td><td>Setting prohibited</td></tr><tr><td>1</td><td>0</td><td>0</td><td>Activated by the INTST signal: two-cycle transfer</td></tr><tr><td>1</td><td>0</td><td>1</td><td>Activated by the INTSR signal: two-cycle transfer</td></tr><tr><td>1</td><td>1</td><td>0</td><td>Activated by the INTCSI signal: two-cycle transfer</td></tr><tr><td>1</td><td>1</td><td>1</td><td>Activated by the INTCM1 signal: two-cycle transfer</td></tr></table>	TTYP			Activation cause and transfer direction	0	0	0	Activated by the DREQn signal: two-cycle transfer	0	0	1	Activated by the DREQn signal: fly-by transfer	0	1	0	Activated by software: two-cycle transfer	0	1	1	Setting prohibited	1	0	0	Activated by the INTST signal: two-cycle transfer	1	0	1	Activated by the INTSR signal: two-cycle transfer	1	1	0	Activated by the INTCSI signal: two-cycle transfer	1	1	1	Activated by the INTCM1 signal: two-cycle transfer
TTYP			Activation cause and transfer direction																																			
0	0	0	Activated by the DREQn signal: two-cycle transfer																																			
0	0	1	Activated by the DREQn signal: fly-by transfer																																			
0	1	0	Activated by software: two-cycle transfer																																			
0	1	1	Setting prohibited																																			
1	0	0	Activated by the INTST signal: two-cycle transfer																																			
1	0	1	Activated by the INTSR signal: two-cycle transfer																																			
1	1	0	Activated by the INTCSI signal: two-cycle transfer																																			
1	1	1	Activated by the INTCM1 signal: two-cycle transfer																																			
4	TDIR	<div>Transfer Direction Sets the transfer direction during transfer between I/O and memory. The setting is valid during fly-by transfer only; ignored during two-cycle transfer. 0: Memory to I/O (read) 1: I/O to memory (write)</div>																																				
3, 2	TM	<div>Transfer Mode Sets the transfer mode during DMA transfer.</div> <table><tr><th colspan="2">TM</th><th>Transfer mode</th></tr><tr><td>0</td><td>0</td><td>Single mode</td></tr><tr><td>0</td><td>1</td><td>Single-step mode</td></tr><tr><td>1</td><td>0</td><td>Block mode</td></tr><tr><td>1</td><td>1</td><td>Setting prohibited</td></tr></table>	TM		Transfer mode	0	0	Single mode	0	1	Single-step mode	1	0	Block mode	1	1	Setting prohibited																					
TM		Transfer mode																																				
0	0	Single mode																																				
0	1	Single-step mode																																				
1	0	Block mode																																				
1	1	Setting prohibited																																				
1	DS	<div>Data Size Sets the transfer data size for DMA transfer. 0: 8 bits 1: 16 bits</div>																																				
0	EN	<div>Enable Specifies whether DMA transfer through DMA channel n is to be enabled or disabled. It is reset when DMA transfer ends with a terminal count. It is also reset when transfer is forcibly ended by means of NMI input. 0: DMA transfer disabled 1: DMA transfer enabled</div>																																				

## 8.4 TRANSFER MODES

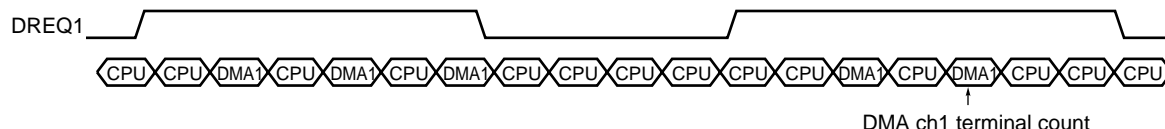
### 8.4.1 Single Transfer Mode

In single transfer mode, DMAC releases the bus at each byte/word transfer. If there is subsequently a DMA transfer request, transfer is performed again. This operation continues until a terminal count occurs.

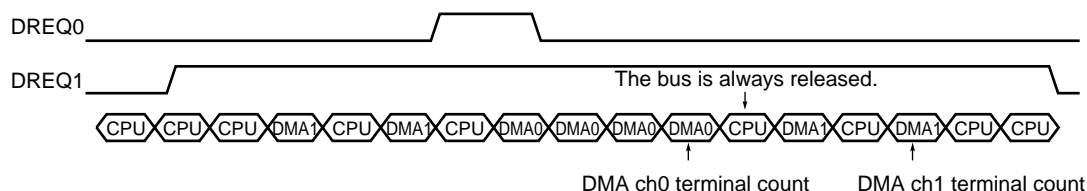
While DMAC is releasing the bus, if another higher priority DMA transfer request is issued, the higher priority DMA request always takes precedence.

Figures 8-2 and 8-3 show examples of single transfer. Figure 8-3 shows an example of single transfer in which a higher priority DMA request is issued. DMA channel 0 is in block transfer mode and channel 1 is in single transfer mode.

**Figure 8-2. Single Transfer Example 1**



**Figure 8-3. Single Transfer Example 2**



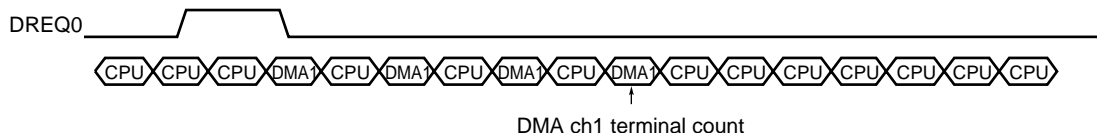
### 8.4.2 Single-Step Transfer Mode

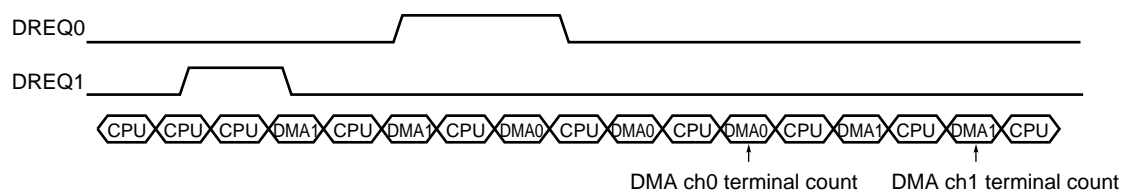
In single-step transfer mode, DMAC releases the bus at each byte/word transfer. Once a request signal (DREQ) is received, this operation continues until a terminal count occurs.

While DMAC is releasing the bus, if another higher priority DMA transfer request is issued, the higher priority DMA request always takes precedence.

Figures 8-4 and 8-5 show examples of single-step transfer.

**Figure 8-4. Single-Step Transfer Example 1**

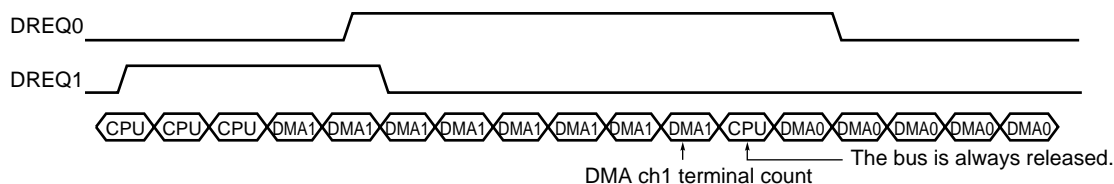


**Figure 8-5. Single-Step Transfer Example 2****8.4.3 Block Transfer Mode**

In block transfer mode, once transfer starts, the transfer continues without the bus being released, until a terminal count occurs. No other DMA requests are accepted during block transfer.

After the block transfer ends and DMAC releases the bus, another DMA transfer can be accepted.

Figure 8-6 shows an example of block transfer. In this block transfer example, a higher priority DMA request is issued. DMA channels 0 and 1 are in block transfer mode.

**Figure 8-6. Block Transfer Example**

## 8.5 DMA TRANSFER TYPES AND TRANSFER OBJECTS

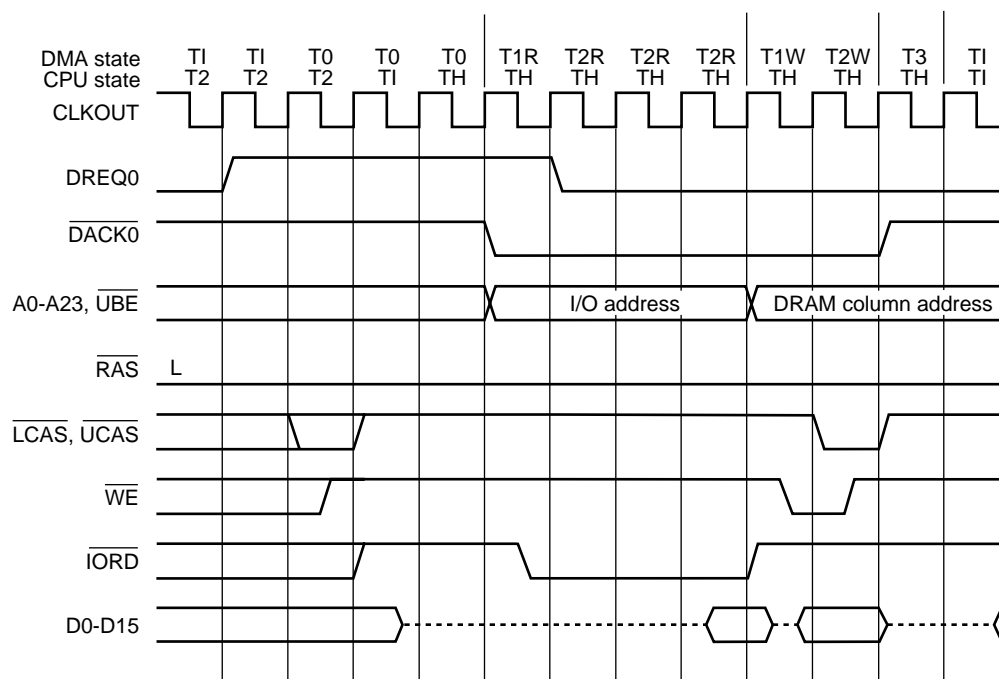
### 8.5.1 Two-Cycle Transfer

In two-cycle transfer, data transfer is performed in two cycles, source to DMAC then DMAC to destination.

In the first cycle, the source address is output to perform reading from the source to DMAC. In the second cycle, the destination address is output to perform writing from DMAC to the destination.

Figure 8-7 shows an example of two-cycle transfer.

**Figure 8-7. Timing of Two-Cycle, Single-Step DMA Transfer (External I/O to DRAM (on-page))**



**Remark** A broken line indicates a high impedance.

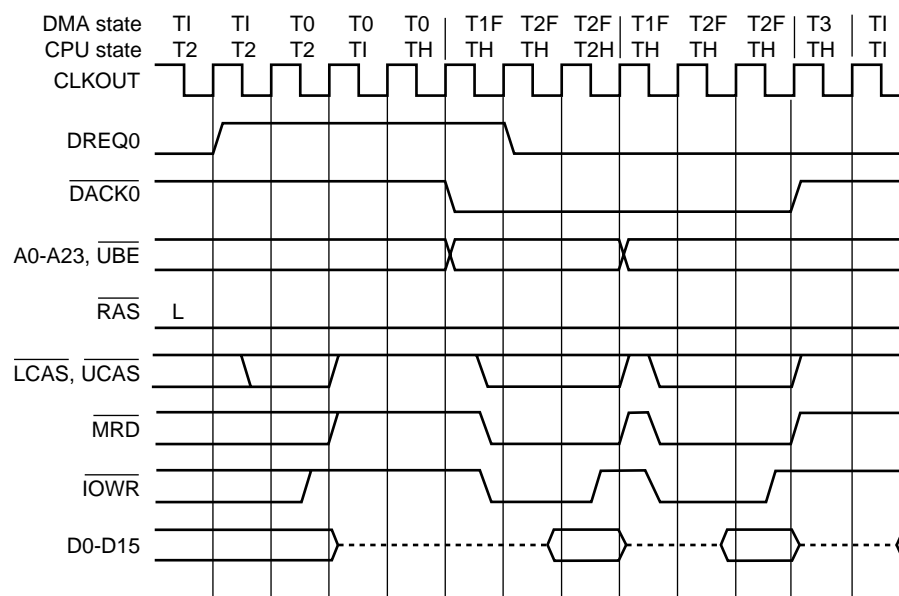
### 8.5.2 Fly-by Transfer

In fly-by transfer, data transfer between memory and I/O is performed in one cycle. To achieve single-cycle transfer, the memory address is always output irrespective of whether it is that of the source or the destination, and the read/write signals for the memory and I/O are made active at the same time.

I/O is selected with the  $\overline{\text{DACK}}$  signal.

Figure 8-8 shows an example of fly-by DMA transfer in which one wait is inserted.

**Figure 8-8. Timing of Fly-by DMA Transfer (DRAM (on-page) to External I/O)**



**Remark** A broken line indicates a high impedance.

### 8.5.3 Transfer Objects

Table 8-1 lists the relationships between transfer types and transfer objects.

**Table 8-1. Transfer Types vs. Transfer Objects**

DMA transfer objects	Transfer type
External I/O selected with the $\overline{\text{DACK}}$ signal to memory and vice versa	Fly-by
I/O (external or internal) to memory and vice versa	Two-cycle
Memory to memory and vice versa	Two-cycle
I/O (external or internal) to I/O (external or internal) and vice versa	Two-cycle

## 8.6 DMA CHANNEL PRIORITIES

The DMA channel priorities are fixed, as follows:

DMA channel 0 > DMA channel 1

These priorities are valid in the TI state only. In block transfer mode, the channel used for transfer is never switched.

In single-step transfer mode, if a higher priority DMA transfer request is issued while the bus is being released (in the TI state), higher priority DMA transfer request is accepted.

## 8.7 DMA TRANSFER REQUESTS

There are three types of DMA transfer requests, those received from the external DREQ pins, those issued by software, and those received from built-in peripheral hardware. These are set with the DMA channel control registers (DCHC).

### (1) Requests received from the DREQ pins

A request received from a DREQ pin is sampled at the falling edge of every clock pulse. The request must not be released until the corresponding  $\overline{\text{DACK}}$  signal becomes active.

If a DREQ pin becomes active while DMAC is in the TI state, DMAC enters the T0 state to start DMA transfer.

### (2) Requests issued by software

### (3) Requests received from built-in peripheral hardware

There are four types of transfer request signals (interrupt request signals) that can be received from built-in peripheral hardware:

- Reception completion interrupt (INTSR) from UART
- Transmission completion interrupt (INTST) from UART
- Transmission/reception completion interrupt (INTCSI) from CSI
- Match interrupt of compare register 1 (CM1) (INTCM1) from RPU

## 8.8 DMA TRANSFER END INTERRUPTS

When DMA transfer ends and the TC bit of the corresponding DCHC register is set to "1," a DMA transfer end interrupt is issued to the interrupt controller. The interrupt is the OR of the interrupt information for channel 0 and channel 1. (See **Table 4-1.**)



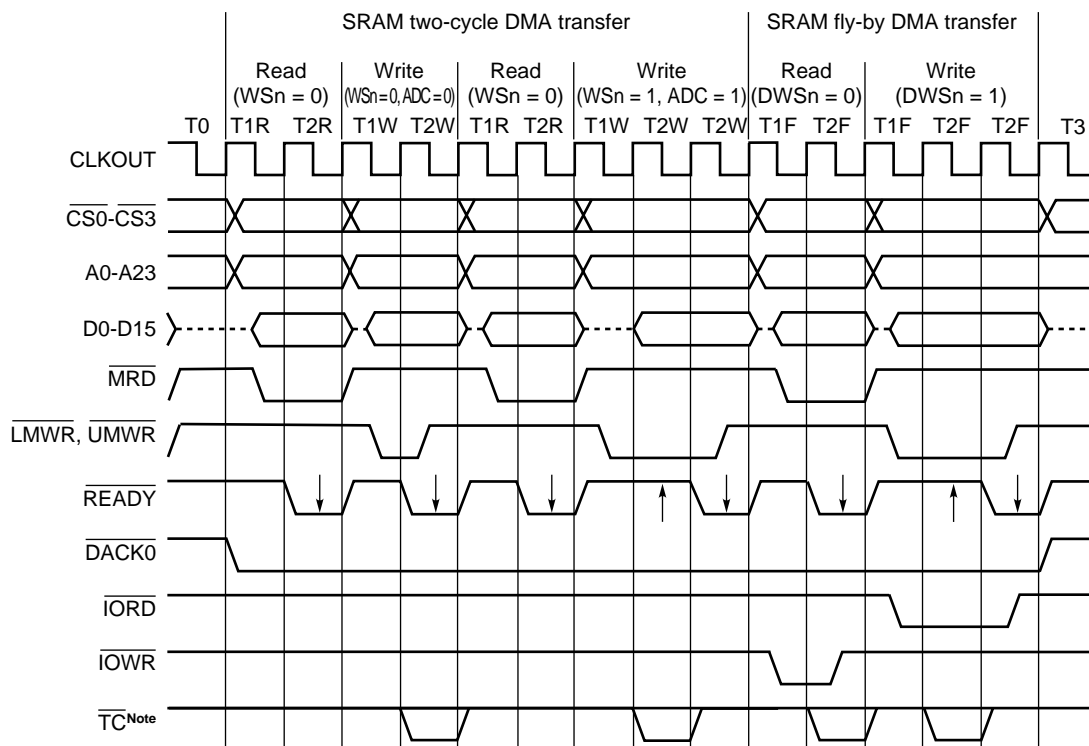
## 8.9 DMA TRANSFER END OUTPUT

In the T2F or T2W state of the cycle in which DMA transfer ends,  $\overline{TC}$  output becomes active for one clock pulse. Figure 8-9 shows the timing of  $\overline{TC}$  output during SRAM two-cycle DMA transfer and SRAM fly-by DMA transfer.

$\overline{TC}$  output is the OR of the DMA transfer end outputs for channels 0 and 1. By ANDing the  $\overline{TC}$  output with the  $\overline{DACK0}$  and  $\overline{DACK1}$  signal in an external circuit, the DMA transfer end outputs for channels 0 and 1 can be obtained.

★

**Figure 8-9. Timing of DMA Transfer End Output**



**Note** Active after completion of DMA transfer.

**Remarks 1.** A broken line indicates a high impedance.

**2.** An arrow indicates a sampling timing.

## 8.10 FORCIBLE INTERRUPTION

During DMA transfer, DMA transfer can be interrupted forcibly with  $\overline{\text{NMI}}$  input. With this input, DMAC sets the EN bits of the DCHC registers for both channels to 0 to disable DMA transfer.

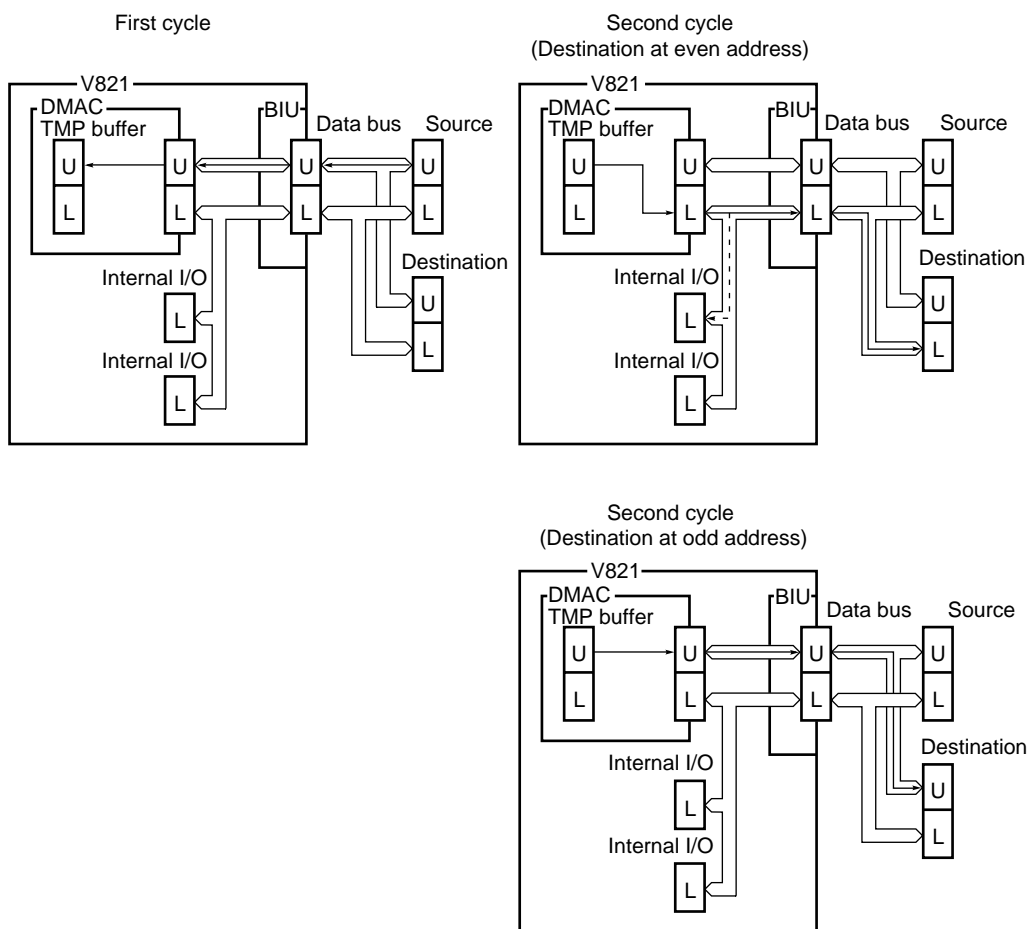
In this state, the DMA transfer can be resumed from the point at which it was interrupted by setting the EN bits to 1.

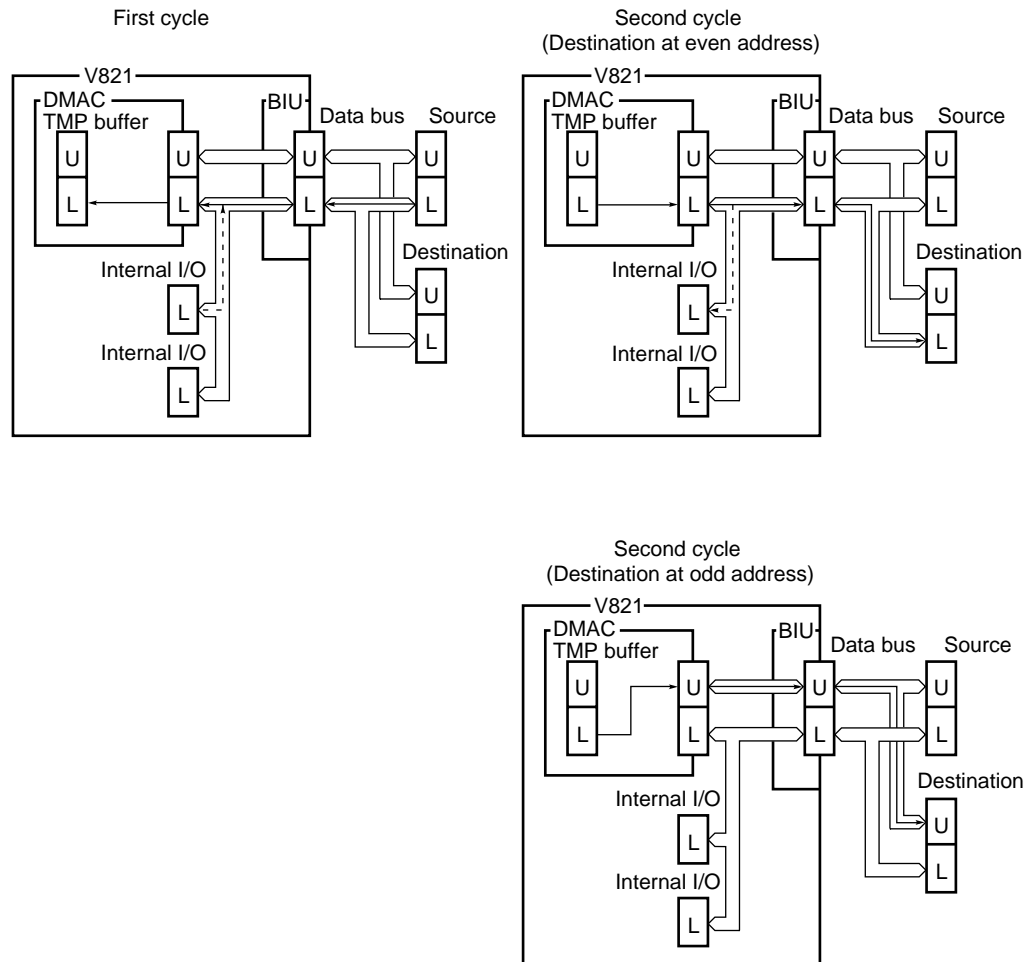
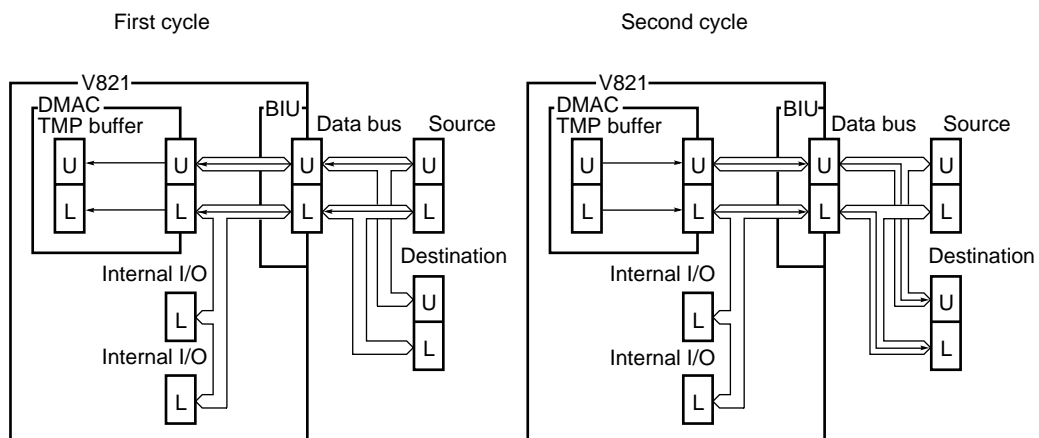
## 8.11 DATA FLOW DURING DMA TRANSFER

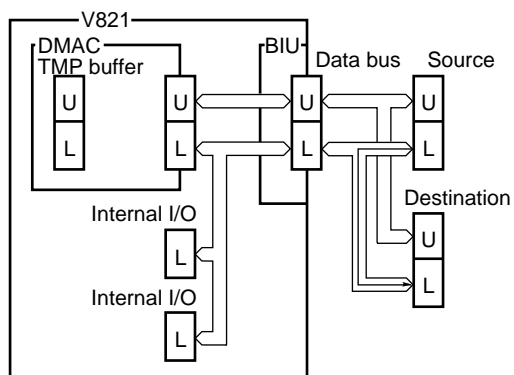
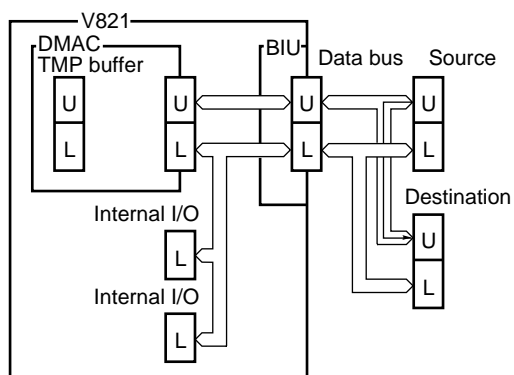
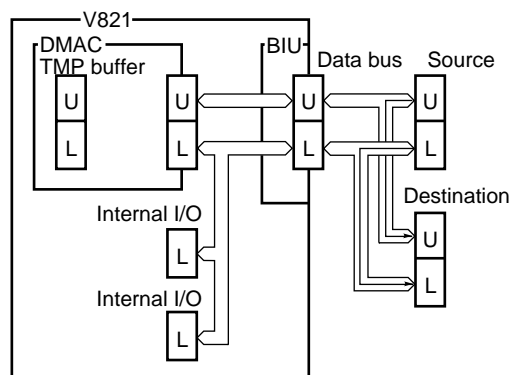
Figures 8-10 and 8-11 illustrate two-cycle byte transfer and 16-bit transfer. Figure 8-12 illustrates fly-by transfer. In these figures, L represents the lower eight bits of 16-bit data and U the upper eight bits.

**Figure 8-10. Data Flow during DMA Transfer (Two-Cycle Byte Transfer)**

### (1) When the source is at an odd address



**(2) When the source is at an even address****Figure 8-11. Data Flow during DMA Transfer (Two-Cycle, 16-Bit Transfer)**

**Figure 8-12. Data Flow during DMA Transfer (Fly-by Transfer)****(1) Byte transfer with an even memory address****(2) Byte transfer with an odd memory address****(3) 16-bit transfer**

## CHAPTER 9 SERIAL INTERFACE FUNCTION

### 9.1 FEATURES

The V821 provides two transmission and reception channels as part of its serial interface function.

The two interface modes listed below are supported, one channel being provided for each mode. The two modes operate independently of each other.

- (1) Asynchronous serial interface (UART)
- (2) Synchronous serial interface (CSI)

In UART mode, one-byte serial data is transmitted or received after a start bit, and full-duplex communication is enabled.

In CSI mode, data is transferred using three signal lines (three-wire serial I/O): the serial clock ( $\overline{\text{SCLK}}$ ), serial input (SI), and serial output (SO).

### 9.2 ASYNCHRONOUS SERIAL INTERFACE (UART)

#### 9.2.1 Features

- Transfer rate    110 bps to 38,400 bps (when BRG is used with  $\phi = 25$  MHz)  
                      781 Kbps maximum    (when  $\phi/2$  is used with  $\phi = 25$  MHz)
- Full-duplex communication
- Two-pin configuration    TXD: Transmission data output pin  
                                     RXD: Reception data input pin
- Reception error detection function
  - Parity error
  - Framing error
  - Overrun error
- Interrupt source (3 types)
  - Reception error interrupt (INTSER)
  - Reception completion interrupt (INTSR)
  - Transmission completion interrupt (INTST)
- The character length for transmission and reception data is specified upon ASIM reception.
- Character length:    7 or 8 bits  
                              9 bits (when an extended bit is used)
- Parity function:    Odd parity, even parity, zero parity, without parity
- Transmission stop bit: 1 or 2 bits
- On-chip baud rate generator

### 9.2.2 Configuration

The asynchronous serial interface is controlled by the asynchronous serial interface mode register (ASIM) and asynchronous serial interface status register (ASIS). Reception data is held in the reception buffer (RXB), and transmission data is written into a transmission shift register (TXS).

The asynchronous serial interface is configured as shown in Figure 9-1.

#### (1) Asynchronous serial interface mode register (ASIM)

The ASIM register is an 8-bit register for specifying asynchronous serial interface operation.

#### (2) Asynchronous serial interface status register (ASIS)

The ASIS register consists of flags for indicating the states of reception errors as well as a transmission status flag. If a reception error occurs, the corresponding reception error flag is set to 1; the flag is reset to 0 when data is read from the reception buffer (RXB/RXBL) or the next data is received. (If the next data contains an error, the corresponding error flag is set.)

The transmission status flag is set to 1 when data transmission starts; the flag is reset to 0 when data transmission ends.

#### (3) Reception control parity check

Reception is controlled according to the setting of the ASIM register. During reception, error checks such as a parity error check are made. If an error is detected, the corresponding value is set in the ASIS register.

#### (4) Reception shift register

The reception shift register converts serial data, applied to the RXD pin, to parallel data. When one-byte data is received, the received data is transferred to the reception buffer.

The reception shift register cannot be manipulated directly from the CPU.

#### (5) Reception buffer (RXB, RXBL)

RXB is a 9-bit buffer register which is used to hold received data; when a 7-bit or 8-bit character is received, 0 is held in the high-order bit position(s).

Specify RXB for 16-bit access to this register. Specify RXBL to access the low-order 8 bits.

In the reception enabled state, reception data is transferred from the reception shift register to the reception buffer upon the completion of shift-in processing for one frame.

When data is transferred to the reception buffer, a reception completion interrupt request (INTSR) is generated.

#### (6) Transmission shift register (TXS, TXSL)

TXS is a 9-bit shift register for transmission processing. When data is written into this register, transmission is started.

Upon the completion of the transmission of one frame including TXS data, a transmission completion interrupt request (INTST) is generated.

Specify TXS for 16-bit access to this register. Specify TXSL to access the low-order 8 bits.

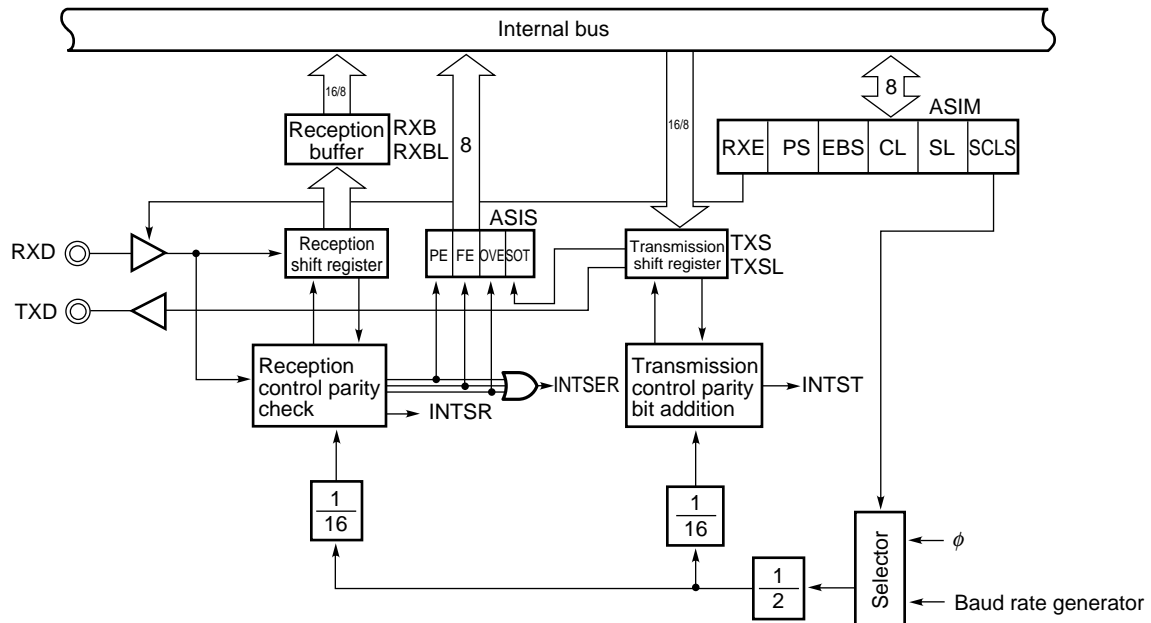
### (7) Transmission control parity bit addition

Transmission is controlled by adding a start bit, parity bit, and stop bit to the data written into the TXS register, according to the setting of the ASIM register.

### (8) Selector

The selector is used to select a serial clock source.

**Figure 9-1. Asynchronous Serial Interface**



### 9.2.3 UART Control Registers

#### (1) Asynchronous serial interface mode register (ASIM)

The asynchronous serial interface mode register specifies the UART transfer mode.

This register allows 8-bit read and write operations.

	7	6	5	4	3	2	1	0		
ASIM	RXE	PS	EBS	0	CL	SL	SCLS		Address C0000080H	When reset, 00H

Bit position	Bit name	Description
7	RXE	<p><b>Receive Enable</b></p> <p>This bit specifies the reception enabled or reception disabled state.</p> <p>0: Reception disabled state</p> <p>1: Reception enabled state</p> <p>In the reception disabled state, the reception shift register does not detect a start bit. Shift-in processing and transfer to the reception buffer are not performed, but the contents of the reception buffer are preserved.</p> <p>In the reception enabled state, reception shift operation is started upon the detection of a start bit. Upon the completion of reception of one frame, the contents of the reception shift register are transferred to the reception buffer. A reception completion interrupt (INTSR) is generated in sync with transfer to the reception buffer.</p>



Bit position	Bit name	Description															
6, 5	PS	<p>Parity Select These bits specify the parity bit.</p> <table border="1"> <thead> <tr> <th colspan="2">PS</th><th>Operation</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>Without parity. Extended bit operation is enabled.</td></tr> <tr> <td>0</td><td>1</td><td>Zero parity Transmission side -&gt; Data is transmitted with the parity bit set to 0. Reception side -&gt; No parity error is generated when data is received.</td></tr> <tr> <td>1</td><td>0</td><td>Odd parity</td></tr> <tr> <td>1</td><td>1</td><td>Even parity</td></tr> </tbody> </table> <ul style="list-style-type: none"> <li>• Even parity When the transmission data contains an odd number of bits set to 1, the parity bit is set to 1. The parity bit is set to 0 when the transmission data contains an even number of bits set to 1. Thus, control is applied so that the number of bits set to 1 in transmission data that includes a parity bit is even. When data is received, the number of bits set to 1 in the received data that includes a parity bit is counted; a parity error is generated if the number is found to be odd.</li> <li>• Odd parity In contrast to even parity, control is exercised so that the number of bits set to 1 in transmission data that includes a parity bit is odd. When data is received, a parity error is generated when the number of bits set to 1 in received data that includes a parity bit is found to be even.</li> <li>• Zero parity When data is transmitted, the parity bit is set to 0, regardless of the transmission data. When data is received, no parity bit check is performed. That is, no parity error is generated, regardless of whether the parity bit is set to 0 or 1.</li> <li>• Without parity No parity bit is added to the transmission data. When data is received, the use of no parity bit is assumed. No parity error is generated because no parity bit is used. With the EBS bit, extended bit operation can be specified.</li> </ul>	PS		Operation	0	0	Without parity. Extended bit operation is enabled.	0	1	Zero parity Transmission side -> Data is transmitted with the parity bit set to 0. Reception side -> No parity error is generated when data is received.	1	0	Odd parity	1	1	Even parity
PS		Operation															
0	0	Without parity. Extended bit operation is enabled.															
0	1	Zero parity Transmission side -> Data is transmitted with the parity bit set to 0. Reception side -> No parity error is generated when data is received.															
1	0	Odd parity															
1	1	Even parity															

Bit position	Bit name	Description								
4	EBS	<p>Extended Bit Select</p> <p>This bit specifies extended bit operation for transmission and reception data when no parity bit is used (PS = 00).</p> <p>0: Disables extended bit operation.</p> <p>1: Enables extended bit operation.</p> <p>When extended bit operation is specified, a higher data bit is added to the 8 bit transmission and reception data to enable 9-bit data communication. Extended bit operation is enabled only when the PS bits are set for operation without parity. When zero parity, odd parity, or even parity operation is specified, the EBS bit is ignored; no extended bit is added.</p>								
2	CL	<p>Character Length</p> <p>This bit specifies the character length of one frame.</p> <p>0: 7 bits</p> <p>1: 8 bits</p>								
1	SL	<p>Stop Bit Length</p> <p>This bit specifies the number of stop bits.</p> <p>0: 1 bit</p> <p>1: 2 bits</p>								
0	SCLS	<p>Serial Clock Source</p> <p>This bit specifies a serial clock source.</p> <p>0: A serial clock source is specified using the BRG and BPRM registers.</p> <p>1: <math>\phi/2</math></p> <ul style="list-style-type: none"><li>When SCLS = 1</li></ul> <p>As the serial clock source, <math>\phi/2</math> (serial clock) is selected. In asynchronous mode, a sampling rate of x 16 is used, so that the baud rate can be expressed by the following:</p> $\text{Baud rate} = \frac{\phi/2}{16} \text{ bps}$ <p>The baud rates determined from the above expression when using typical clocks are:</p> <table><tr><td><math>\phi</math></td><td>25 MHz</td><td>20 MHz</td><td>16 MHz</td></tr><tr><td>Baud rate</td><td>781 K</td><td>625 K</td><td>500 K</td></tr></table> <ul style="list-style-type: none"><li>When SCLS = 0</li></ul> <p>As the serial clock source, the output from the baud rate generator is selected. For details of the baud rate generator, see <b>Section 9.4</b>.</p>	$\phi$	25 MHz	20 MHz	16 MHz	Baud rate	781 K	625 K	500 K
$\phi$	25 MHz	20 MHz	16 MHz							
Baud rate	781 K	625 K	500 K							

**Caution** If the value of this register is changed while the UART is used for transmission or reception, the operation of the UART will be unpredictable.

## (2) Asynchronous serial interface status register (ASIS)

The asynchronous serial interface status register consists of three error flags (three bits), used to indicate any error status that may be present when reception using the UART has ended, and a transmission status flag.

Each of the status flags used to indicate a reception error indicates the latest error state. That is, if the same error occurs more than once before the reception data is read, the state of only the last error is preserved.

If a reception error occurs, read the reception buffer RXB or RXBL after reading the ASIS register, then clear the error flag.

This register supports only 8-bit read operations.

	7	6	5	4	3	2	1	0		
ASIS	SOT	0	0	0	0	PE	FE	OVE	Address C0000082H	When reset, 00H

Bit position	Bit name	Description
7	SOT	<b>Status Of Transmission</b> This bit is a status flag for indicating the transmission status. Set to 1 : Transmission start timing (write to the TXS register) Cleared to 0 : Transmission end timing (INTST generation) This flag is used to determine whether it is possible to write to the transmission shift register before serial data transfer is started.
2	PE	<b>Parity Error</b> This bit is a status flag for indicating a parity error. Set to 1 : When there is a mismatch between the transmission and reception parities Cleared to 0 : Processing to read data from the reception buffer
1	FE	<b>Framing Error</b> This bit is a status flag for indicating a framing error. Set to 1 : When no stop bit is detected Cleared to 0 : Processing to read data from the reception buffer
0	OVE	<b>Overflow Error</b> This bit is a status flag for indicating an overflow error. Set to 1 : When the UART completes the next receive processing before reception data is read from the reception buffer Cleared to 0 : Processing to read data from the reception buffer Note that, to receive each frame, the contents of the reception shift register are transferred to the reception buffer. So, if an overflow error occurs, the reception buffer is overwritten by the next reception data, thus causing the previous reception data to be discarded.

**(3) Reception buffer (RXB, RXBL)**

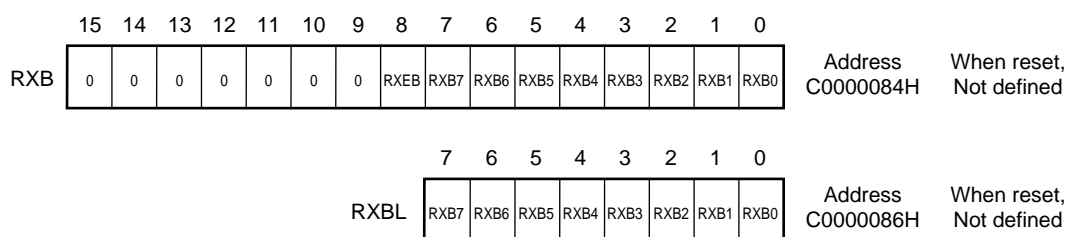
RXB is a 9-bit buffer register used to hold received data; when a 7- or 8-bit character is received, 0 is held at the higher bit position(s).

Specify RXB for 16-bit access to this register. Specify RXBL to access the low-order 8 bits.

In the reception enabled state, reception data is transferred from the reception shift register to the reception buffer upon the completion of shift-in processing for one frame. When data is transferred to the reception buffer, a reception completion interrupt request (INTSR) is generated.

In the reception disabled state, reception data is not transferred to the reception buffer, even upon the completion of shift-in processing for one frame; the contents of the reception buffer are preserved, and no reception completion interrupt request is generated.

RXB supports only 16-bit read access, while RXBL supports only 8-bit read access.



Bit position	Bit name	Description
8	RXEB	Receive Extended Buffer This bit is an extended bit for 9-bit character reception. Zero is read from this bit when a 7-or 8-bit character is received.
7-0	RXBn (n = 7-0)	Receive Buffer These bits are used to store received data. Zero is read from RXB7 when a 7-bit character is received.

#### (4) Transmission shift register (TXS, TXSL)

TXS is a 9-bit shift register used for transmission. When data is written to this register, transmission is started.

When the transmission of one frame including TXS data is completed, a transmission completion interrupt request (INTST) is generated.

Specify TXS for 16-bit access to this register. Specify TXSL to access the low-order 8 bits.

TXS supports only 16-bit write access. TXSL supports only 8-bit write access.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
TXS	0	0	0	0	0	0	0	TXED	TXS7	TXS6	TXS5	TXS4	TXS3	TXS2	TXS1	TXS0	Address C0000088H	When reset, Not defined

	7	6	5	4	3	2	1	0		
TXSL	TXS7	TXS6	TXS5	TXS4	TXS3	TXS2	TXS1	TXS0	Address C000008AH	When reset, Not defined

Bit position	Bit name	Description
8	TXED	Transmit Extended Data This bit is an extended bit for 9-bit character transmission.
7-0	TXSn (n = 7-0)	Transmit Shifter Transmission data is written to these bit positions.

**Caution** The UART of the V821 has no transmission buffer, so that an interrupt request is generated not at the time of transmission completion (completion of transfer to a buffer) but upon the completion of the transmission of one-frame data.

### 9.2.4 Interrupt Requests

The UART generates three types of interrupt requests:

- Reception error interrupt
- Reception completion interrupt
- Transmission completion interrupt

The default priorities for these types of interrupt requests are such that the reception error interrupt has the highest priority, the reception completion interrupt has the next-highest priority, and the transmission completion interrupt has the lowest priority.

**Table 9-1. Generated Interrupts and Default Priorities**

Interrupt	Priority
Reception error	1
Reception completion	2
Transmission completion	3

#### (1) Reception error interrupt (INTSER)

In the reception enabled state, a reception error interrupt is generated according to the logical OR of the three types of reception errors described in **Section 9.2.3 (2)**.

In the reception disabled state, no reception error interrupt is generated.

#### (2) Reception completion interrupt (INTSR)

In the reception enabled state, a reception completion interrupt is generated when data is shifted into the reception shift register, then is transferred to the reception buffer.

A reception completion interrupt is also generated when a reception error occurs. In this case, however, the reception error interrupt is assigned higher priority.

In the reception disabled state, a reception completion interrupt is not generated.

#### (3) Transmission completion interrupt (INTST)

The UART of the V821 does not have a transmission buffer. So, a transmission completion interrupt is generated when one-frame transmission data including a 7-, 8-, or 9-bit character is shifted out from the transmission shift register.

A transmission completion interrupt is output when transmission of the last bit of the transmission data is started.

## 9.2.5 Operation

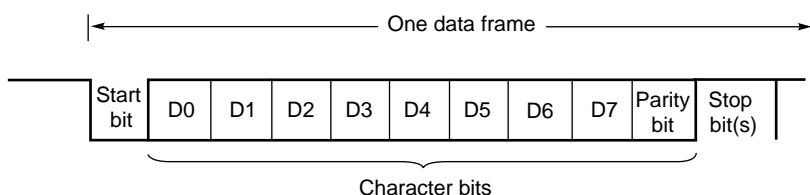
### (1) Data format

Full-duplex serial data transmission and reception are performed.

Figure 9-2 shows the format of the transmission and reception data. As shown in the figure, one data frame consists of a start bit, character bits, a parity bit, and one or two stop bits.

The asynchronous serial interface mode register (ASIM) is used to specify the number of character bits, select a parity option, and specify a stop bit length within a data frame.

**Figure 9-2. Format of Transmission and Reception Data Transferred via Asynchronous Serial Interface**



- Start bit : 1 bit
- Character bit : 7 bits/8 bits/9 bits (when an extended bit is used)
- Parity bit : Even parity/odd parity/zero parity/without parity
- Stop bit : 1 bit/2 bits

### (2) Transmission

Transmission is started when data is written into the transmission shift register (TXS or TXSL). When a transmission completion interrupt (INTST) is generated, the next data is written into the TXS or TXSL register.

#### (a) Transmission enabled state

The UART of the V821 is always placed in the transmission enabled state. The V821 does not have an input pin for a transmission enable signal. To check whether the remote receiver is in the reception enabled state, therefore, a general I/O port should be used.

#### (b) Starting transmission

Transmission is started when data is written into the transmission shift register (TXS/TXSL). Transmission data is transferred, starting with a start bit and followed by the LSB of the character data. A start bit, parity bit, and stop bit(s) are added automatically.

**(c) Transmission interrupt request**

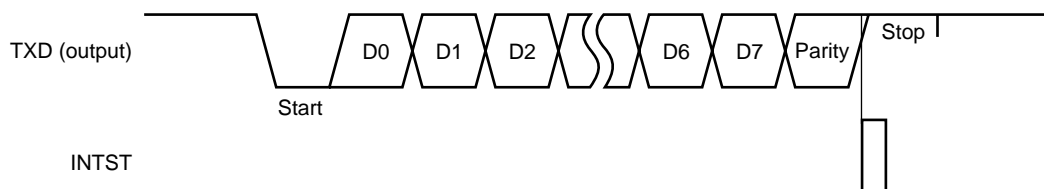
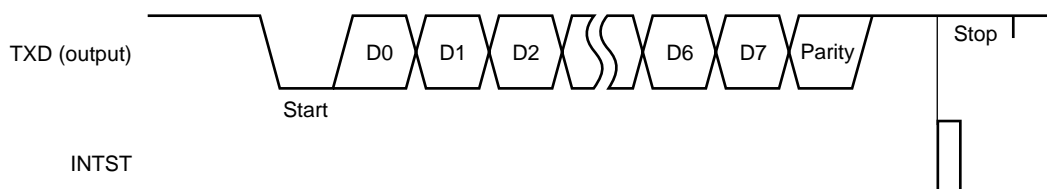
Upon the completion of the transmission of one-frame data, a transmission completion interrupt request (INTST) is generated.

If the next transmission data is not written into TXS or TXSL, transmission stops.

If the next transmission data is not written into TXS or TXSL immediately after the completion of one transmission operation, the communication rate is reduced.

- Cautions**
1. A transmission completion interrupt (INTST) is usually generated once the transmission shift register (TXS or TXSL) becomes empty. However, no transmission completion interrupt (INTST) is generated even when the transmission shift register (TXS or TXSL) is forcibly emptied by inputting a **RESET** signal.
  2. During transmission before INTST generation, write data, even if written into the TXS or TXSL register, is ignored.

**Figure 9-3. Timing of Transmission Completion Interrupt via Asynchronous Serial Interface**

**(a) Number of stop bits: 1****(b) Number of stop bits: 2****(3) Reception**

When reception is enabled, sampling of the RXD pin starts. Then, when a start bit is detected, data reception is started. Each time one-frame data reception ends, a reception completion interrupt (INTSR) is generated. Usually, reception data is transferred from the reception buffer (RXB/RXBL) to memory by means of this interrupt handling.



### (a) Reception enabled state

Reception can be enabled by setting the RXE bit of the ASIM register to 1.

RXE = 1: Reception enabled state

RXE = 0: Reception disabled state

When reception is disabled, the reception hardware enters the initial state.

In the reception disabled state, neither a reception completion interrupt nor a reception error interrupt is generated, the data in the reception buffer being preserved.

### (b) Starting reception

Reception is started upon the detection of a start bit.

The RXD pin is sampled using the serial clock specified with the ASIM register. Eight serial clock cycles after a falling edge is detected on the RXD pin, the RXD pin is sampled again. If a low level is detected at this time, the presence of a start bit is assumed, and receive processing is started. Then, the RXD pin is sampled every 16 clock cycles.

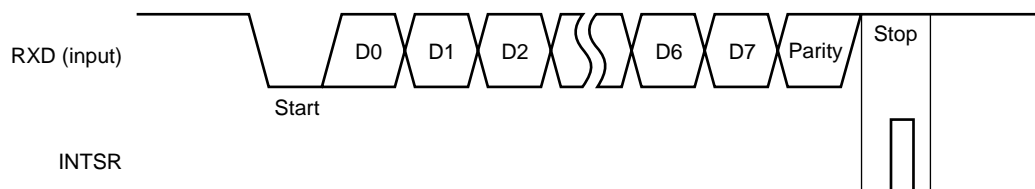
If a high level is detected eight clock cycles after a falling edge is detected on the RXD pin, the falling edge is not regarded as being a start bit. The serial clock counter, used to generate a sampling timing signal, is initialized and stops operating, then waits for the next falling edge to be applied.

### (c) Reception completion interrupt request

If one-frame data reception is completed when RXE = 1, the reception data held in the shift register is transferred to RXB, after which a reception completion interrupt request (INTSR) is generated. Even if an error occurs, reception data including an error is transferred to the reception buffer (RXB/RXBL), and both a reception completion interrupt (INTSR) and a reception error interrupt (INTSER) are generated.

When the RXE bit is reset to 0 during reception, receive processing stops immediately. At this time, the contents of the reception buffer (RXB/RXBL) and the asynchronous serial interface status register (ASIS) remain as is, and neither a reception completion interrupt (INTSR) nor a reception error interrupt (INTSER) is generated.

**Figure 9-4. Timing of Reception Completion Interrupt via Asynchronous Serial Interface**



**(d) Reception error flags**

The three error flags for a parity error, framing error, and overrun error are affected by receive processing.

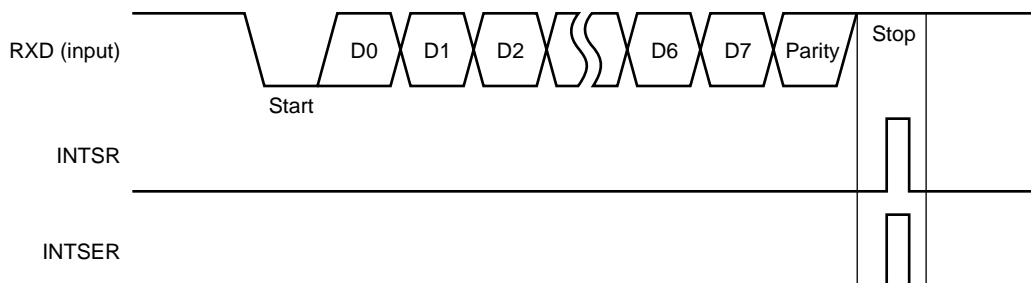
A reception error interrupt is generated according to the logical OR of these three error flags.

By reading the ASIS register in reception error interrupt (INTSER) handling, the error which occurred during reception can be determined.

The contents of the ASIS register are reset to 0 when the reception buffer (RXB/RXBL) is read or the next data is received. (If the next reception data contains an error, the corresponding error flag is set.)

Reception error	Cause
Parity error	Parity specified for transmission does not match the parity of the reception data.
Framing error	No stop bit is detected
Overrun error	Reception of the next data is completed before data is read from the reception buffer.

**Figure 9-5. Timing of Reception Error**



## 9.3 SYNCHRONOUS SERIAL INTERFACE (CSI)

### 9.3.1 Features

- High-speed transfer 6.25 Mbps maximum (when  $\phi/2$  is used with  $\phi = 25$  MHz)
- Half-duplex communication
- Character length: 8 bits
- Switchable between the MSB and LSB to lead data transfer
- Allows selection between external serial clock input and internal serial clock output
- Three-wire method
  - SO : Serial data output
  - SI : Serial data input
  - $\overline{\text{SCLK}}$  : Serial clock I/O pin
- One interrupt source
  - Interrupt request signal (INTCSI)

The synchronous serial interface is controlled using the synchronous serial interface mode register (CSIM). Transmission/reception data can be read from or written to the SIO register.

#### (1) Synchronous serial interface mode register (CSIM)

The CSIM register is an 8-bit register for specifying synchronous serial interface operation.

#### (2) Shift register (SIO)

The shift register (SIO) is an 8-bit register used to convert serial data to parallel data or vice versa. SIO is used for both transmission and reception.

Data is shifted in (for reception) or shifted out (for transmission), starting from the MSB or LSB.

Transmission and reception are actually controlled by SIO read/write operation.

#### (3) Serial clock selector

The serial clock selector is used to select the serial clock to be used.

#### (4) Serial clock control circuit

The serial clock control circuit controls the supply of a serial clock to the shift register. When an internal clock is used, the serial clock control circuit controls the clock signal output on the  $\overline{\text{SCLK}}$  pin.

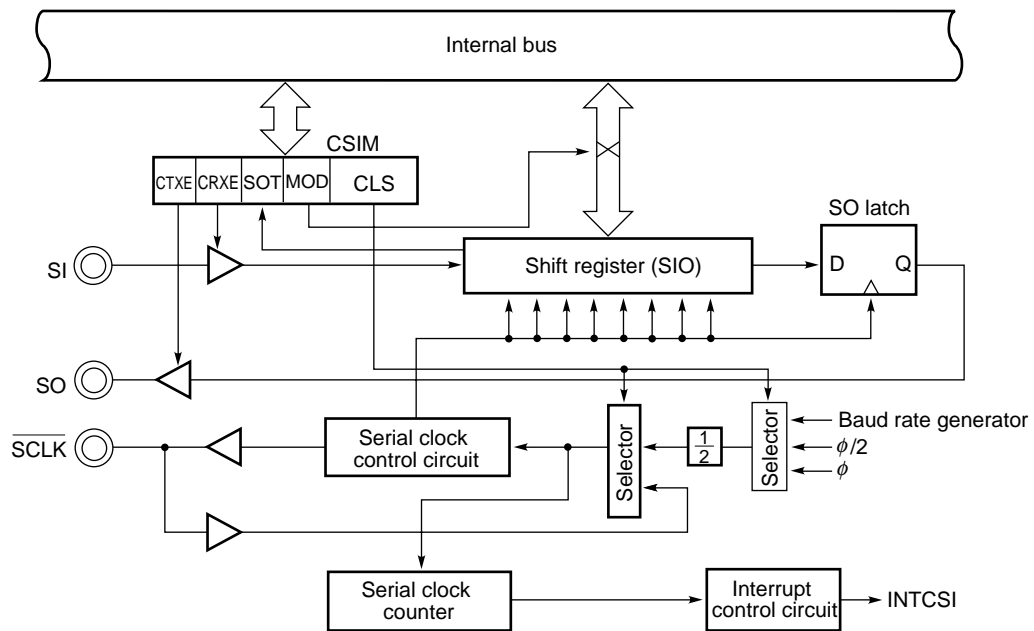
#### (5) Serial clock counter

To check whether 8-bit data transfer is performed, the serial clock counter counts the number of serial clock pulses output or applied in reception or transmission.

#### (6) Interrupt signal generation control circuit

The interrupt signal generation control circuit applies control to determine whether to generate an interrupt request when 8 serial clock pulses are counted with the serial clock counter.

### 9.3.2 Configuration



### 9.3.3 CSI Control Registers

#### (1) Synchronous serial interface mode register (CSIM)

The synchronous serial interface mode register specifies a basic CSI operating mode.

This register supports 8-bit read and write operations. However, bit 5 can be only read.

	7	6	5	4	3	2	1	0	
CSIM	CTXE	CRXE	SOT	0	0	MOD	CLS		
									Address C0000090H
									When reset, 00H

Bit position	Bit name	Description																							
7	CTXE	CSI Transmit Enable This bit specifies the transmission enabled state or transmission disabled state. 0: Transmission disabled state 1: Transmission enabled state When CTXE = 0, the output buffers of the SO and SI pins become high-impedance.																							
6	CRXE	CSI Receive Enable This bit specifies the reception enabled state or reception disabled state. 0: Reception disabled state 1: Reception enabled state If a serial clock is applied while transmission is disabled (CTXE = 0) and reception is disabled, 0s are written into the shift register.																							
5	SOT	Status Of Transmission This bit indicates that transfer is in progress. Set to 1 : Transmission start timing (write to the SIO register) Cleared to 0 : Transmission end timing (INTCSI generation) This bit is used to check whether data can be written into the serial I/O shift register (SIO) when serial data transfer is to be started in the transmission enabled state (CTXE = 1).																							
2	MOD	Mode This bit specifies the bit from which data transfer is to start. 0: MSB 1: LSB																							
1,0	CLS	Clock Source These bits specify a serial clock. <table><tr><th colspan="2">CLS</th><th colspan="2">Serial clock specification</th><th>SCLK pin</th></tr><tr><td>0</td><td>0</td><td colspan="2">External clock</td><td>Input</td></tr><tr><td>0</td><td>1</td><td rowspan="3">Internal clock</td><td>Specified by the BPRM register<sup>Note 1</sup></td><td>Output</td></tr><tr><td>1</td><td>0</td><td><math>\phi/4</math><sup>Note 2</sup></td><td>Output</td></tr><tr><td>1</td><td>1</td><td><math>\phi/2</math><sup>Note 2</sup></td><td>Output</td></tr></table> <p><b>Notes 1.</b> For details of the setting of the BPRM register, see <b>Section 9.4</b>. <b>2.</b> <math>\phi/4</math> and <math>\phi/2</math> are frequency-divided signals (<math>\phi</math> = system clock).</p>	CLS		Serial clock specification		SCLK pin	0	0	External clock		Input	0	1	Internal clock	Specified by the BPRM register <sup>Note 1</sup>	Output	1	0	$\phi/4$ <sup>Note 2</sup>	Output	1	1	$\phi/2$ <sup>Note 2</sup>	Output
CLS		Serial clock specification		SCLK pin																					
0	0	External clock		Input																					
0	1	Internal clock	Specified by the BPRM register <sup>Note 1</sup>	Output																					
1	0		$\phi/4$ <sup>Note 2</sup>	Output																					
1	1		$\phi/2$ <sup>Note 2</sup>	Output																					

**(2) Serial I/O shift register (SIO)**

The serial I/O shift register converts 8-bit serial data to 8-bit parallel data or vice versa. Transmission and reception are actually controlled by an SIO register read/write operation.

When CTXE = 1 or CRXE = 1, a shift operation is performed.

This register supports 8-bit read and write operations.

	7	6	5	4	3	2	1	0	
SIO	SIO7	SIO6	SIO5	SIO4	SIO3	SIO2	SIO1	SIO0	
									Address C0000092H
									When reset, Not defined

Bit position	Bit name	Description
7-0	SIO <sub>n</sub> (n = 7-0)	Serial I/O Data is shifted in (for reception) or shifted out (for transmission), starting from the MSB or LSB.

**9.3.4 Basic Operation****(1) Transfer format**

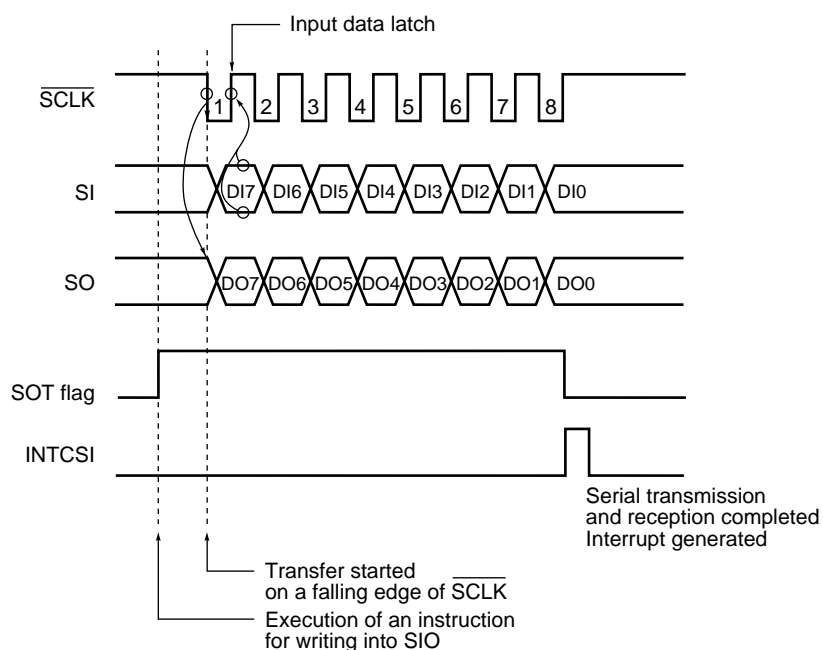
For the CSI of the V821, three wires are used: one clock line, and two data lines.

Serial transfer is started by executing an instruction for writing transfer data into the SIO register.

During transmission, data is output on the SO pin on a falling edge of  $\overline{\text{SCLK}}$ .

During reception, input data applied to the SI pin is latched on a rising edge of  $\overline{\text{SCLK}}$ .  $\overline{\text{SCLK}}$  stops when the serial clock counter overflows (at the eighth rising edge), and  $\overline{\text{SCLK}}$  remains high until the next data is transmitted, or reception is started. At this time, an interrupt request signal (INTCSI) is generated.

**Caution** Serial transfer is not performed even if the value of CTXE is changed from 0 to 1 after transmission data is written into the shift register.



## (2) Enabling transmission and reception

The CSI of the V821 has only one 8-bit shift register and has no buffer, so that transmission and reception are performed at the same time.

### (a) Prerequisite transmission and reception conditions

When CTXE = 1, transmission is enabled.

When CRXE = 1, reception is enabled.

When CTXE = CRXE = 1, both transmission and reception are enabled.

#### (i) Disabling SIO output with CTXE

When CTXE = 0, the serial output becomes high-impedance.

When CTXE = 1, the data held in the shift register is output.

#### (ii) Disabling SIO input with CRXE

When CRXE = 0, 0s are written into the shift register.

When CRXE = 1, serial input data is applied to the shift register.

#### (iii) Transmission data check

By setting CTXE = CRXE = 1, transmission data can be locally received to check where there is a bus contention.

### (b) Starting transmission and reception

Transmission/reception is started by reading from or writing to the shift register (SIO). The start of transmission/reception is controlled by setting the transmission enable bit (CTXE) and reception enable bit (CRXE) as indicated below.

CTXE	CRXE	Start condition
0	0	Not started
0	1	Reading from the shift register
1	0	Writing to the shift register
1	1	Writing to the shift register
1	0 -> 1	Rewriting the CRXE bit

Data transfer does not take place even if the shift register is read from or written to when CTXE = 0, after which CTXE is set to 1.

If the setting of CRXE is changed from 0 to 1 when CTXE = 0, a serial clock is generated to start reception.

### 9.3.5 Transmission in Three-Wire Serial I/O Mode

Transmission starts when data is written into the SIO register after transmission is enabled with the synchronous serial interface mode register (CSIM).

#### (1) Starting transmission

Transmission is started by writing transmission data into the shift register (SIO) after setting the CTXE bit of the synchronous serial interface mode register (CSIM) to 1 and setting the CRXE bit to 0.

Note that when the CTXE bit is reset to 0, the SO pin is placed in the high-impedance state.

#### (2) Data transmission synchronized with the serial clock

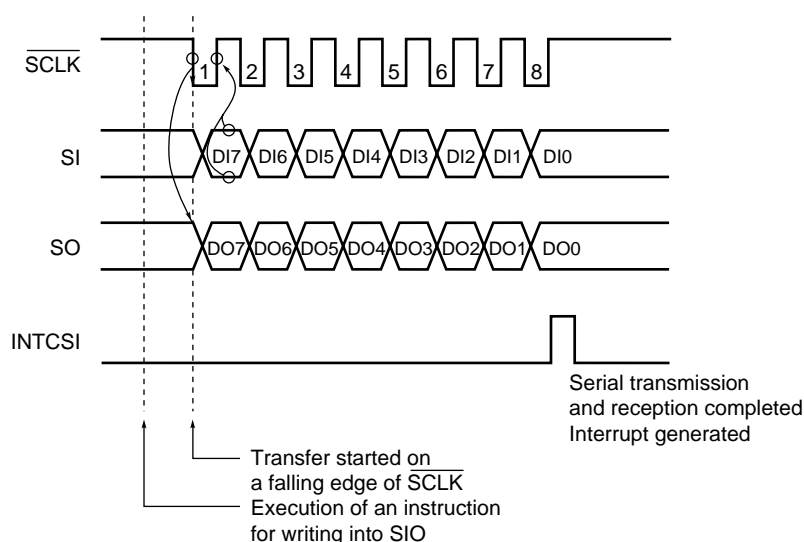
##### (a) When an internal clock is selected as the serial clock

When transmission is started, a serial clock is output on the  $\overline{\text{SCLK}}$  pin, and data is sequentially shifted out, from SIO to the SO pin, on a falling edge of the serial clock.

##### (b) When an external clock is selected as the serial clock

When transmission is started, data is sequentially shifted out from SIO to the SO pin on a falling edge of a serial clock applied to the  $\overline{\text{SCLK}}$  pin after the start of transmission. If a serial clock is applied to the  $\overline{\text{SCLK}}$  pin before transmission is started, a shift operation is not performed, and the output level of the SO pin does not change.

**Figure 9-6. Timing of Three-Wire Serial I/O Mode (Transmission)**





### 9.3.6 Reception in Three-Wire Serial I/O Mode

Reception starts when the reception disabled state is changed to the reception enabled state with the synchronous serial interface mode register (CSIM), or when the SIO register is read when reception is enabled.

#### (1) Starting reception

Reception is enabled in either of two ways:

- <1> Changing the setting of the CRXE bit of the CSIM register from 0 (to disable reception) to 1 (to enable reception)
- <2> Reading reception data from the shift register (SIO) when the CRXE bit of the CSIM register is set to 1 (to enable reception)

When the CRXE bit of the CSIM register is set to 1, reception is not started even if 1 is again written into the CRXE bit. Note that when CRXE = 0, 0s are written into the shift register.

#### (2) Data reception synchronized with the serial clock

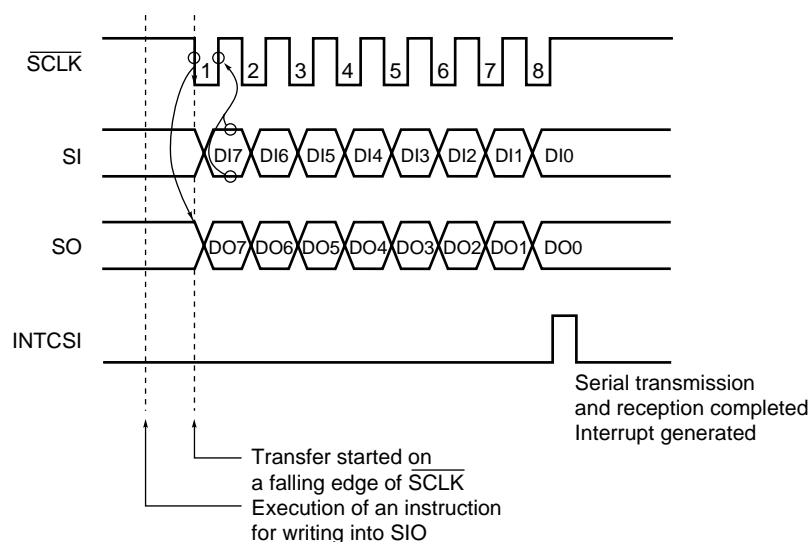
##### (a) When an internal clock is selected as the serial clock

When reception is started, a serial clock is output on the SCLK pin, and data applied to the SI pin is sequentially shifted into SIO on a rising edge of the serial clock.

##### (b) When an external clock is selected as the serial clock

When reception is started, data applied to the SI pin is sequentially shifted into SIO on a rising edge of a serial clock applied to the SCLK pin after the start of reception. If a serial clock is applied to the SCLK pin before reception is started, a shift operation is not performed.

**Figure 9-7. Timing of Three-Wire Serial I/O Mode (Reception)**



### 9.3.7 Transmission and Reception in Three-Wire Serial I/O Mode

Transmission and reception can be performed at the same time by enabling transmission and reception with the synchronous serial interface mode register (CSIM).

#### (1) Starting transmission and reception

When both the CTXE bit and CRXE bit of the synchronous serial interface mode register (CSIM) are set to 1, transmission and reception can be performed at the same time.

Transmission and reception can be started in the following:

- Writing transmission data into the shift register (SIO) when both the CTXE bit and CRXE bit of the CSIM register are set to 1 (to enable transmission and reception).

#### (2) Data transmission synchronized with the serial clock

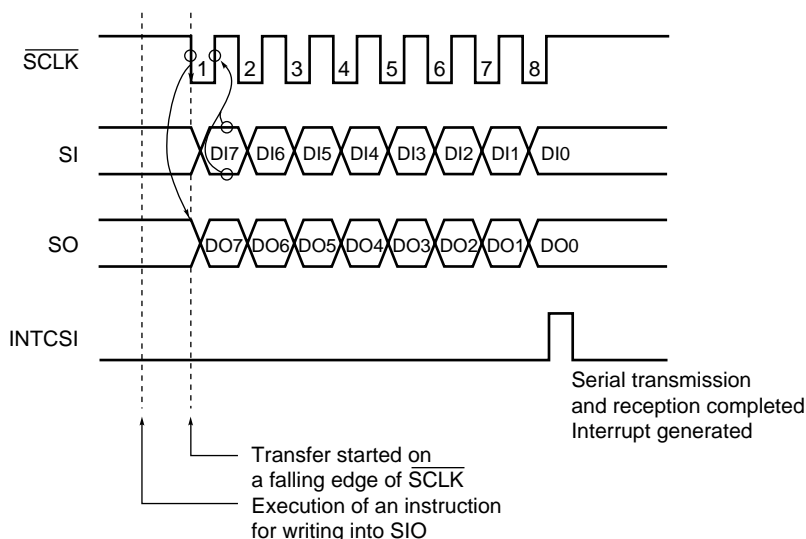
##### (a) When an internal clock is selected as the serial clock

When transmission and reception are started, a serial clock is output on the  $\overline{\text{SCLK}}$  pin, and data is sequentially shifted out from SIO to the SO pin on a falling edge of the serial clock. At the same time, data applied to the SI pin is sequentially shifted into SIO on a rising edge of the serial clock.

##### (b) When an external clock is selected as the serial clock

When transmission and reception are started, data is sequentially shifted out from SIO to the SO pin on a falling edge of a serial clock applied to the  $\overline{\text{SCLK}}$  pin, after the start of transmission and reception. At the same time, data applied to the SI pin is sequentially shifted into SIO on a rising edge of the serial clock. If a serial clock is applied to the  $\overline{\text{SCLK}}$  pin when transmission and reception are not started, a shift operation is not performed, and the output level on the SO pin does not change.

**Figure 9-8. Timing of Three-Wire Serial I/O Mode (Transmission and Reception)**



**Caution** When transmission and reception are started for the first time, the setting of the CRXE bit is always changed from 0 to 1. This change causes transmission and reception to start immediately, such that arbitrary data may be output. Accordingly, while transmission and reception are disabled (by resetting both the CTXE bit and CRXE bit to 0), write initial transmission data into the SIO register. Then, enable transmission and reception.

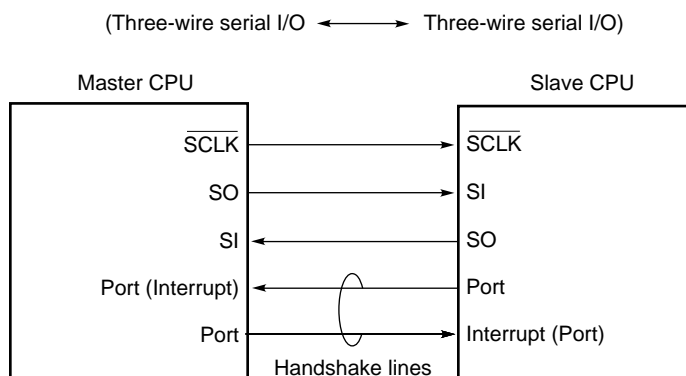
### 9.3.8 Example System Configuration

This section introduces an example system configuration, in which 8-bit data is transferred using three signal lines: the serial clock ( $\overline{\text{SCLK}}$ ), serial input (SI), and serial output (SO). This configuration is useful for making a connection to a display controller or peripheral I/O device having a built-in conventional synchronous serial interface.

To enable connection to multiple devices, handshaking lines are also required.

The user can select either the MSB or LSB as the bit from which data transmission will begin, thus enabling communications with a wide variety of devices.

**Figure 9-9. Example System Configuration Using CSI**



## 9.4 BAUD RATE GENERATOR (BRG)

### 9.4.1 Configuration and Function

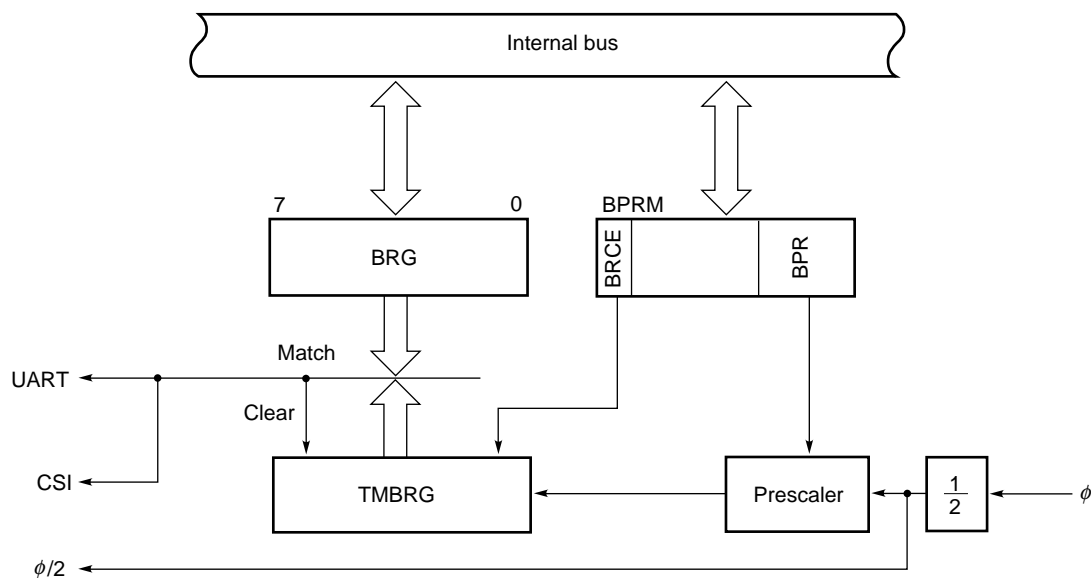
With the serial interface, a serial clock chosen from the baud rate generator output and clocks generated using the system clock ( $\phi$ ) can be used as a baud rate.

A serial clock source can be specified by using the SCLS bit of the ASIM register when the UART is used, or by using the CLS bit of the CSIM register when the CSI is used.

When baud rate generator output is specified, the baud rate generator is selected as the clock source.

The same serial clock is used for both transmission and reception on a channel, so that the same baud rate applies to transmission and reception.

**Figure 9-10. Block Diagram**



The dedicated baud rate generator, BRG, consists of a dedicated 8-bit timer (TMBRG), compare register (BRG), and prescaler for generating a shift clock for transmission and reception.

### (1) Input clock

The system clock ( $\phi$ ) is applied to BRG.

### (2) Setting for BRG

#### (a) UART

If the dedicated baud rate generator is specified when the UART is used, the actual baud rate is given by the following formula because a sampling rate of x 16 is used:

$$\text{Baud rate} = \frac{\phi}{2 \times m \times 2^n \times 16 \times 2} \text{ [bps]}$$

$\phi$  = System clock frequency [Hz]

m = BRG setting ( $1 \leq m \leq 256$  **Note**)

n = BRG prescaler setting (n = 0, 1, 2, 3, 4)

**Note** By writing 0s to the BRG register, m = 256 can be set.

#### (b) CSI

If use of the dedicated baud rate generator is specified when the CSI is being used, the actual baud rate is given by the following formula:

$$\text{Baud rate} = \frac{\phi}{2 \times m \times 2^n \times 2} \text{ [bps]}$$

$\phi$  = System clock frequency [Hz]

m = BRG setting ( $1 \leq m \leq 256$  **Note**)

n = BRG prescaler setting (n = 0, 1, 2, 3, 4)

**Note** By writing 0s to the BRG register, m = 256 can be set.

Table 9-2 lists examples of baud rate generator settings using typical clocks.

Table 9-2. BRG Setting Data

Baud rate [bps]		$\phi = 25$ MHz			$\phi = 20$ MHz			$\phi = 16$ MHz		
UART	CSI	BPR	BRG	Error	BPR	BRG	Error	BPR	BRG	Error
110	1,760	4	222	0.02%	4	178	0.25%	4	142	0.03%
150	2,400	4	163	0.15%	4	130	0.16%	3	208	0.16%
300	4,800	3	163	0.15%	3	130	0.16%	2	208	0.16%
600	9,600	2	163	0.15%	2	130	0.16%	1	208	0.16%
1,200	19,200	1	163	0.15%	1	130	0.16%	0	208	0.16%
2,400	38,400	0	163	0.15%	0	130	0.16%	0	104	0.16%
4,800	76,800	0	81	0.47%	0	65	0.16%	0	52	0.16%
9,600	153,600	0	41	0.76%	0	33	1.36%	0	26	0.16%
10,400	166,400	0	38	1.16%	0	30	0.16%	0	24	0.16%
19,200	307,200	0	20	1.73%	0	16	1.73%	0	13	0.16%
38,400	614,400	0	10	1.73%	0	8	1.73%	0	7	6.99% <sup>Note</sup>
76,800	1,228,800	0	5	1.73%	0	4	1.73%	-	-	-
153,600	2,457,600	0	2	27.2% <sup>Note</sup>	0	2	1.73%	-	-	-

**Note** Unusable because of an excessive error

### (3) Baud rate error

A baud rate error is expressed as follows:

$$\text{Error [\%]} = \left( \frac{\text{Actual baud rate (baud rate including an error)}}{\text{Desired baud rate (normal baud rate)}} - 1 \right) \times 100$$

**Example:**  $(9,520/9,600 - 1) \times 100 = -0.833$  [%]  
 $(5,000/4,800 - 1) \times 100 = +4.167$  [%]

### (4) Allowable error ranges for baud rate generator

The allowable error ranges depend on the number of bits in one frame.

When there are 16 bits in one frame, a baud rate error of  $\pm 5\%$  and a sampling timing of  $\pm 4.5\%$  are the basic allowable ranges.

Practically, however, a baud rate error of only  $\pm 2.3\%$  can be allowed, assuming that both transmission and reception involve a degree of error.

### 9.4.2 Baud Rate Generator Register (BRG)

The baud rate generator register is an 8-bit compare register, used to set a timer count value for the dedicated baud rate generator.

This register supports 8-bit read and write operations.

	7	6	5	4	3	2	1	0		
BRG	BRG7	BRG6	BRG5	BRG4	BRG3	BRG2	BRG1	BRG0	Address C00000A0H	When reset, Not defined

**Caution** BRG register write clears the internal timer (TMBRG). So, never rewrite the BRG register by software during transmission or reception.

### 9.4.3 Baud Rate Generator Prescaler Mode Register (BPRM)

The baud rate generator prescaler mode register controls the timer count operation, and selects a count clock.

This register supports 8-bit read and write operations.

	7	6	5	4	3	2	1	0		
BPRM	BRCE	0	0	0	0	BPR			Address C00000A2H	When reset, 00H

Bit position	Bit name	Description																								
7	BRCE	Baud Rate Generator Count Enable This bit controls the BRG count operation. 0: Stops the count operation, with BRG cleared. 1: Enables count operation.																								
2-0	BPR	Baud Rate Generator Prescaler These bits specify the count clock to be applied to TMBRG. <table border="1"><thead><tr><th colspan="3">BPR</th><th>Count clock</th></tr></thead><tbody><tr><td>0</td><td>0</td><td>0</td><td><math>\phi/2</math> (n = 0)</td></tr><tr><td>0</td><td>0</td><td>1</td><td><math>\phi/4</math> (n = 1)</td></tr><tr><td>0</td><td>1</td><td>0</td><td><math>\phi/8</math> (n = 2)</td></tr><tr><td>0</td><td>1</td><td>1</td><td><math>\phi/16</math> (n = 3)</td></tr><tr><td>1</td><td>x</td><td>x</td><td><math>\phi/32</math> (n = 4)</td></tr></tbody></table> n: Prescaler setting $\phi$ : System clock	BPR			Count clock	0	0	0	$\phi/2$ (n = 0)	0	0	1	$\phi/4$ (n = 1)	0	1	0	$\phi/8$ (n = 2)	0	1	1	$\phi/16$ (n = 3)	1	x	x	$\phi/32$ (n = 4)
BPR			Count clock																							
0	0	0	$\phi/2$ (n = 0)																							
0	0	1	$\phi/4$ (n = 1)																							
0	1	0	$\phi/8$ (n = 2)																							
0	1	1	$\phi/16$ (n = 3)																							
1	x	x	$\phi/32$ (n = 4)																							

**Caution** Never change the count clock during transmission or reception.

[MEMO]



## CHAPTER 10 TIMER/COUNTER FUNCTIONS (REAL-TIME PULSE UNIT)

The real-time pulse unit (RPU) measures pulse intervals and frequencies, and outputs programmable pulses. It is capable of 16-bit measurement. It can also generate various types of pulses, such as interval pulse and one-shot pulse.

### 10.1 FEATURES

- Timer 0 (TM0)
  - 16-bit timer/event counter
  - Two count clock sources (system clock frequency division selected or external pulse input)
  - Four capture/compare registers
  - Count clear pin (TCLR)
  - Five interrupt sources
  - Two external pulse outputs
- Timer 1 (TM1)
  - 16-bit interval timer
  - Count clock generated by dividing the system clock frequency
  - Compare register
  - Interrupt source

## 10.2 BASIC CONFIGURATION

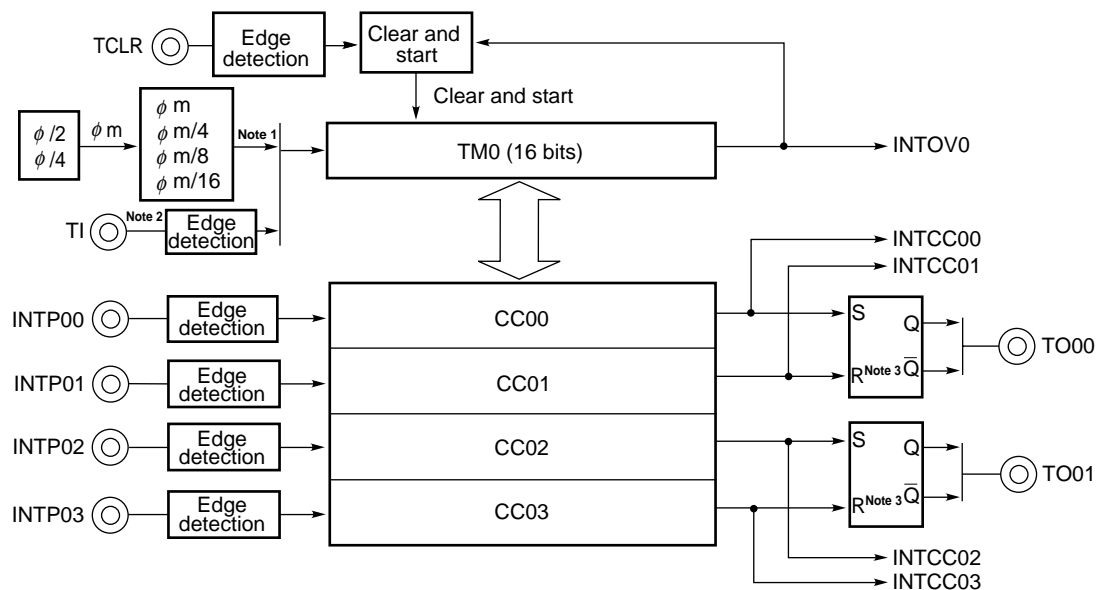
The basic configuration of the real-time pulse unit is listed below.

**Table 10-1. Configuration of the RPU**

Timer	Count clock	Register	Read/write	Generated interrupt signal	Capture trigger	Timer output SR	Other functions
Timer 0	$\phi/2$	TM0	Read	INTOV0	-	-	External clear
	$\phi/4$	CC00	Read/write	INTCC00	INTP00	TO00 (S)	-
	$\phi/8$	CC01	Read/write	INTCC01	INTP01	TO00 (R)	-
	$\phi/16$	CC02	Read/write	INTCC02	INTP02	TO01 (S)	-
	$\phi/32$ $\phi/64$ TI pin input	CC03	Read/write	INTCC03	INTP03	TO01 (R)	-
Timer 1	$\phi/32$	TM1	Read	-	-	-	-
	$\phi/64$ $\phi/128$ $\phi/256$	CM1	Read/write	INTCM1	-	-	-

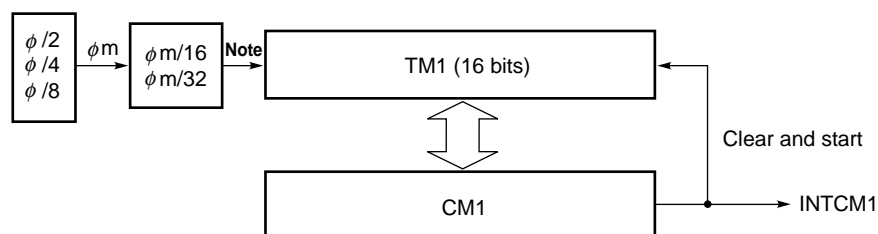
**Remark**  $\phi$  : System clock

SR : Set/reset

**(1) Timer 0 (16-bit timer/event counter)**

- Notes**
1. Internal count clock
  2. External count clock
  3. A reset takes precedence.

**Remark**  $\phi$ : System clock

**(2) Timer 1 (16-bit interval timer)**

**Note** Internal count clock

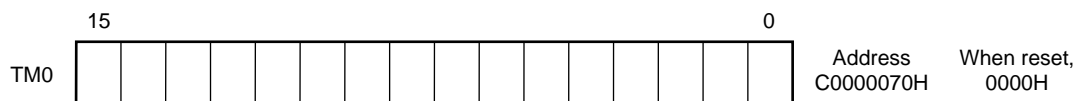
**Remark**  $\phi$ : System clock

### 10.2.1 Timer 0

#### (1) Timer 0 (TM0)

TM0 functions as a 16-bit free-running timer or external signal event counter. It can be used mainly for period measurement, frequency measurement, and pulse output.

TM0 can be read from only in 16-bit units.



TM0 is used to count the internal or external count clock pulses. The timer is started and stopped using bit CEO of timer control register 0 (TMC0).

TMC0 is used to select whether internal or external count clock is to be selected.

#### (a) When the external count clock is selected

TM0 functions as an event counter. An effective edge is specified using timer unit mode register 0 (TUM0), and TM0 is incremented according to the TI pin input.

#### (b) When the internal count clock is selected

TM0 functions as a free-running timer. It is used to count the internal clock ( $\phi/2$  to  $\phi/64$ ) pulses specified in register TMC0.

When the timer overflows, an overflow interrupt can be generated. Setting register TUM0 properly can stop the timer after an overflow occurs.

External input TCLR can clear and start the timer. Because the prescaler is cleared at the same time, the time between the input of TCLR and the first time the timer reaches its full count is fixed according to the frequency division ratio of the prescaler. TUM0 specifies the behavior.

Inputting RESET clears (0) all bits of TM0.

**(2) Capture/compare registers 00 to 03 (CC00 to CC03)**

Each capture/compare register consists of 16 bits and is connected to TM0. The capture/compare registers are used as either capture or compare registers according to the setting of timer unit mode register 0 (TMU0). They can be read from and written to in 16-bit units.

CC00	15 [16-bit register] 0	Address C0000072H	When reset, Not defined
CC01	15 [16-bit register] 0	Address C0000074H	When reset, Not defined
CC02	15 [16-bit register] 0	Address C0000076H	When reset, Not defined
CC03	15 [16-bit register] 0	Address C0000078H	When reset, Not defined

**(a) When a capture/compare register is specified as a capture register**

If a capture/compare register is specified as a capture register, it detects the effective edge of the corresponding external interrupt (INTP00 to INTP03) as a capture trigger. Timer 0 latches the count in synchronization with the capture trigger (capturing). Capturing occurs asynchronously with the count clock. The latched value is held in the capture register until capturing occurs again.

If capturing (latching) and writing by an instruction contend for the capture register, instruction execution takes precedence, and capturing is ignored.

The effective edges (rising edge, falling edge or both rising and falling edges) of an external interrupt can be selected using the ICU mode register (IMOD). (See **Section 4.4.5.**)

If a capture/compare register is specified as a capture register, an interrupt occurs on the effective edge of INTP00 to INTP03. In this case, INTCCn (compare register match signal) cannot be used to cause an interrupt.

**(b) When a capture/compare register is specified as a compare register**

If a capture/compare register is specified as a compare register, the value in the register is compared with the content of the timer each time the timer is incremented. When they match, an interrupt is generated.

The compare register provides a set/reset output function. It sets or resets the corresponding timer output in synchronization with the occurrence of the a match signal.

The interrupt source that becomes effective varies with the function of the selected register.

If a capture/compare register is specified as a compare register, register TUM0 can be used to select either INTCCn (match signal) or INTPn (effective edge detection) as an interrupt signal. If INTPn is selected, an external interrupt becomes acceptable in concurrence with timer output specification.

### 10.2.2 Timer 1

#### (1) Timer 1 (TM1)

TM1 is a 16-bit timer. It can be used mainly as an interval timer for software.

TM1 can be read from in 16-bit units. It is a read-only register.



TM1 is started and stopped using bit CE1 of timer control register 1 (TMC1).

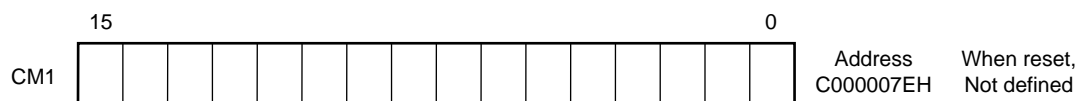
For the count clock, the ratio of frequency division by the prescaler can be selected from  $\phi/32$ ,  $\phi/64$ ,  $\phi/128$ , and  $\phi/256$  according to the setting of register TMC1.

Inputting RESET clears (0) all bits of TM1.

**Caution** Once a comparison match occurs, the timer is cleared on the next count clock pulse. So, if the frequency division ratio is large, the timer value may not be zero even when it is read immediately after the occurrence of a comparison match interrupt.

#### (2) Compare register 1 (CM1)

CM1 is a 16-bit register and is connected to TM1. It can be read from and written to in 16-bit units.



The content of TM1 is compared with that of CM1 each time TM1 is incremented. When a match is detected, it causes an interrupt (INTCM1). At the same time, TM1 is cleared.

**10.3 CONTROL REGISTERS****(1) Timer unit mode register 0 (TUM0)**

TUM0 is a register used to control the behavior of timer 0. It specifies the operation mode of the capture/compare registers.

TUM0 can be read from and written to in 16-bit units.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
TUM0	0	0	OST	ECLR0	TES0		CES0	CMS03	CMS02	CMS01	CMS00	IMS03	IMS02	IMS01	IMS00		Address C0000060H When reset, 0A00H

Bit position	Bit name	Description															
13	OST	<p>Overflow Stop Specifies the action to be taken after the timer overflows. This flag is effective only for TM0.</p> <p>0: The timer keeps to increment after it overflows. 1: The timer stops and holds 0000H when it overflows. CE0 of TMC0 remains to be 1. The timer is caused to restart incrementing by: Setting CE0 to 1, when ECLR0 = 0, or Inputting a trigger to the timer clear pin (TCLR) , when ECLR0 = 1</p>															
12	ECLR0	<p>External Input Timer Clear Enables the timer to be cleared according to an external clear input (TCLR) to TM0.</p> <p>0: TM0 is not cleared by an external input. 1: TM0 is cleared by an external input. After cleared, it restarts incrementing.</p>															
11, 10	TES0	<p>TI Edge Select Specifies the effective edge of an external clock input (TI).</p> <table border="1"> <thead> <tr> <th colspan="2">TES0</th><th>Effective edge</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>RFU (reserved)</td></tr> <tr> <td>0</td><td>1</td><td>Falling edge</td></tr> <tr> <td>1</td><td>0</td><td>Rising edge</td></tr> <tr> <td>1</td><td>1</td><td>Both rising and falling edges</td></tr> </tbody> </table>	TES0		Effective edge	0	0	RFU (reserved)	0	1	Falling edge	1	0	Rising edge	1	1	Both rising and falling edges
TES0		Effective edge															
0	0	RFU (reserved)															
0	1	Falling edge															
1	0	Rising edge															
1	1	Both rising and falling edges															

Bit position	Bit name	Description															
9, 8	CES0	<p>TCLR Edge Select Specifies the effective edge of an external clear input (TCLR).</p> <table border="1"> <thead> <tr> <th colspan="2">CES0</th><th>Effective edge</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td>RFU (reserved)</td></tr> <tr> <td>0</td><td>1</td><td>Falling edge</td></tr> <tr> <td>1</td><td>0</td><td>Rising edge</td></tr> <tr> <td>1</td><td>1</td><td>Both rising and falling edges</td></tr> </tbody> </table>	CES0		Effective edge	0	0	RFU (reserved)	0	1	Falling edge	1	0	Rising edge	1	1	Both rising and falling edges
CES0		Effective edge															
0	0	RFU (reserved)															
0	1	Falling edge															
1	0	Rising edge															
1	1	Both rising and falling edges															
7-4	CMS00-CMS03	<p>Capture/Compare Mode Select Selects the operation mode of the capture/compare registers (CC00 to CC03). 0: The capture/compare registers work as capture registers. Capturing can occur only when CE0 of TMC0 is 1. 1: The capture/compare registers work as compare registers.</p>															
3-0	IMS00-IMS03	<p>Interrupt Mode Select Selects INTPn or INTCCn as an interrupt source (n = 00 to 03). 0: Compare register match signal INTCCn is used as an interrupt signal. 1: An external input signal INTPn is used as an interrupt signal.</p>															



**(2) Timer control register 0 (TMC0)**

TMC0 controls the behavior of TM0.

TMC0 can be read from and written to in 8-bit units.

	7	6	5	4	3	2	1	0		
TMC0	CE0	0	0	ETI	PRSO	0	0	PRM0	Address C0000062H	When reset, 00H

Bit position	Bit name	Description															
7	CE0	<p>Count Enable Controls the behavior of the timer.</p> <p>0: The timer stops at 0000H. It does not work.</p> <p>1: The timer works, except when ECLR0 of TUM0 is 1. When ECLR0 is 1, the timer waits for a TCLR input. When a TCLR input occurs, the timer starts counting.</p> <p>If ECLR0 = 0, the timer starts the instant 1 is written to CE0. If CE0 is set to 1 with ECLR0 = 1, the timer will not start even when ECLR0 is reset to 0.</p>															
4	ETI	<p>External TI Input Switches between the external and internal count clocks.</p> <p>0: Specifies a <math>\phi</math>-based clock (internal).</p> <p>1: Specifies TI (external).</p>															
3, 2	PRSO	<p>Prescaler Clock Select Selects an internal count clock (<math>\phi_m</math> is an intermediate clock).</p> <table border="1"> <thead> <tr> <th colspan="2">PRSO</th><th>Count clock</th></tr> </thead> <tbody> <tr> <td>0</td><td>0</td><td><math>\phi_m</math></td></tr> <tr> <td>0</td><td>1</td><td><math>\phi_m/4</math></td></tr> <tr> <td>1</td><td>0</td><td><math>\phi_m/8</math></td></tr> <tr> <td>1</td><td>1</td><td><math>\phi_m/16</math></td></tr> </tbody> </table>	PRSO		Count clock	0	0	$\phi_m$	0	1	$\phi_m/4$	1	0	$\phi_m/8$	1	1	$\phi_m/16$
PRSO		Count clock															
0	0	$\phi_m$															
0	1	$\phi_m/4$															
1	0	$\phi_m/8$															
1	1	$\phi_m/16$															
0	PRM0	<p>Prescaler Clock Mode Selects an intermediate count clock <math>\phi_m</math> (where <math>\phi</math> is the system clock).</p> <p>0: <math>\phi/2</math></p> <p>1: <math>\phi/4</math></p>															

**Caution** Do not change the count clock when the timer is operating.

**(3) Timer control register 1 (TMC1)**

TMC1 controls the behavior of TM1.

TMC1 can be read from and written to in 8-bit units.

	7	6	5	4	3	2	1	0	
TMC1	CE1	0	0	0	0	PRS1	PRM1		Address C0000064H When reset, 00H

Bit position	Bit name	Description															
7	CE1	Count Enable Controls the behavior of the timer. 0: The timer stops at 0000H. It does not work. 1: The timer works.															
2	PRS1	Prescaler Clock Select Selects an internal count clock ( $\phi_m$ is an intermediate clock). 0: $\phi_m/16$ 1: $\phi_m/32$															
1, 0	PRM1	Prescaler Clock Mode Selects an intermediate count clock $\phi_m$ (where $\phi$ is the system clock). <table border="1"> <tr> <th colspan="2">PRM1</th><th><math>\phi_m</math></th></tr> <tr> <td>0</td><td>0</td><td><math>\phi/2</math></td></tr> <tr> <td>0</td><td>1</td><td><math>\phi/4</math></td></tr> <tr> <td>1</td><td>0</td><td><math>\phi/8</math></td></tr> <tr> <td>1</td><td>1</td><td>RFU (reserved)</td></tr> </table>	PRM1		$\phi_m$	0	0	$\phi/2$	0	1	$\phi/4$	1	0	$\phi/8$	1	1	RFU (reserved)
PRM1		$\phi_m$															
0	0	$\phi/2$															
0	1	$\phi/4$															
1	0	$\phi/8$															
1	1	RFU (reserved)															

**Caution** Do not switch the count clock when the timer is operating.

**(4) Timer output control register 0 (TOC0)**

TOC0 controls the timer output from the pins TO00 and TO01.

TOC0 can be read from and written to in 8-bit units.

	7	6	5	4	3	2	1	0		
TOC0	ENTO01	ALV01	ENTO00	ALV00	0	0	TOPC01	TOPC00	Address C0000066H	When reset, 03H

Bit position	Bit name	Description
7, 5	ENTO01, ENTO00	Enable TOxx pin Disables or enables the corresponding timer outputs (TO00 and TO01). 0: The timer output is disabled. The reversed (inactive) level of the ALV bit is output from the corresponding TO00 and TO01 pins. If a match signal occurs with the corresponding compare register, the level at TO00 or TO01 does not change. 1: The timer output is enabled. When a match signal occurs with the corresponding compare register, the timer output changes. The reverse (inactive) level of the ALV bit is output, until the first match signal occurs after the timer output is enabled.
6, 4	ALV01, ALV00	Active Level TOxx pin Specifies the active level of the timer output. 0: Active low level 1: Active high level
1, 0	TOPC01, TOPC00	Timer Output Control Specifies the pin function of the TO01/ $\overline{\text{INTP02}}$ and TO00/ $\overline{\text{INTP00}}$ dual-function pins. 0: $\overline{\text{TO01}}$ and $\overline{\text{TO00}}$ outputs 1: $\overline{\text{INTP02}}$ and $\overline{\text{INTP00}}$ inputs

**Remark** For the TO00 and TO01 output flip-flops, a reset takes precedence.

**Caution** The TO00 or TO01 output does not change on an external interrupt signal ( $\overline{\text{INTP0n}}$ ). To use TO00 and TO01, it is necessary to specify the capture/compare registers as the compare registers ( $\text{CMS0n} = 1$ ).

**(5) ICU mode register (IMOD)**

This is a control register for specifying the effective edge of an external interrupt signal. (See **Chapter 4** for details.)

**(6) Timer overflow status register (TOVS)**

TOVS holds flags to indicate an overflow from TM0 or TM1.

TOVS can be read from and written to in 8-bit units.

TOVS can be tested and reset by software to poll about the occurrence of an overflow.

	7	6	5	4	3	2	1	0	
TOVS	0	0	0	0	0	0	OVF1	OVF0	Address C0000068H           When reset, 00H

Bit position	Bit name	Description
1, 0	OVFn	<p>Overflow Flag Overflow flag for TMn (n = 0 or 1) 0: TMn has not overflowed. 1: TMn has overflowed.</p> <p><b>Caution</b> TM0 causes an interrupt request signal INTOV0 to be sent to the interrupt controller in synchronization with an overflow. However, an interrupt operation is completely independent of TOVS. Similarly to other overflow flags, the TM0 overflow flag can be operated on by software. This does not affect the interrupt request flag in the interrupt controller corresponding to INTOV0.</p> <p>No transfer occurs to the TOVS register during access from the CPU. If an overflow occurs while the TOVS register is being read from, the flag is not changed. Instead, it is changed at the next read timing.</p>

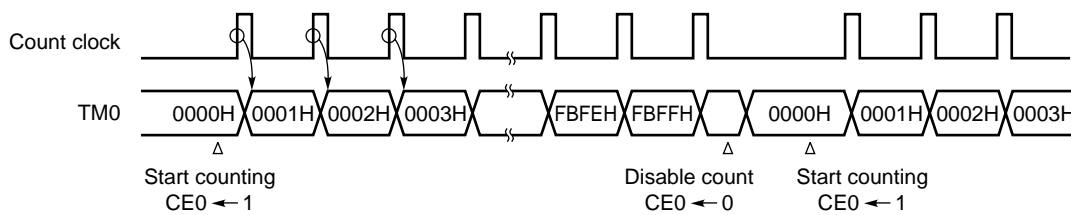
## 10.4 OPERATION OF TIMER 0

### 10.4.1 Counting

Timer 0 functions as a 16-bit free-running timer or external signal event counter. Timer control register 0 (TMC0) is used to specify the behavior of timer 0.

If timer 0 is working as a free-running timer, when the content of a register (CC00 to CC03) matches the count in TM0, an interrupt signal can be generated, and timer output TOxx can be set and reset. In addition, the count in TM0 can be captured in a register (CC00 to CC03) in synchronization with the effective edge detected at the external interrupt request input pin (external trigger). The captured value is retained until the next capture trigger occurs.

**Figure 10-1. Basic Operation of Timer 0**



### 10.4.2 Count Clock Selection

There are two types of count clocks for timer 0, that is internal and external. The ETI bit of register TMC0 specifies which count clock is to be used.

**Caution** Do not switch the count clock when the timer is operating.

#### (1) Internal count clock (ETI bit = 0)

An internal count clock is selected from  $\phi/2$ ,  $\phi/4$ ,  $\phi/8$ ,  $\phi/16$ ,  $\phi/32$ , and  $\phi/64$  according to the setting of bits PRS0 and PRM0 of register TMC0.

PRS0		PRM0	Count clock
0	0	0	$\phi/2$
0	0	1	$\phi/4$
0	1	0	$\phi/8$
0	1	1	$\phi/16$
1	0	0	$\phi/16$
1	0	1	$\phi/32$
1	1	0	$\phi/32$
1	1	1	$\phi/64$

**(2) External count clock (ETI bit = 1)**

TM0 counts signals input at the TI pin. It can function also as an event counter.

The effective edge of a signal at TI is specified using bit TES0 of register TUM0.

TES0		Effective edge
0	0	RFU (reserved)
0	1	Falling edge
1	0	Rising edge
1	1	Both rising and falling edges

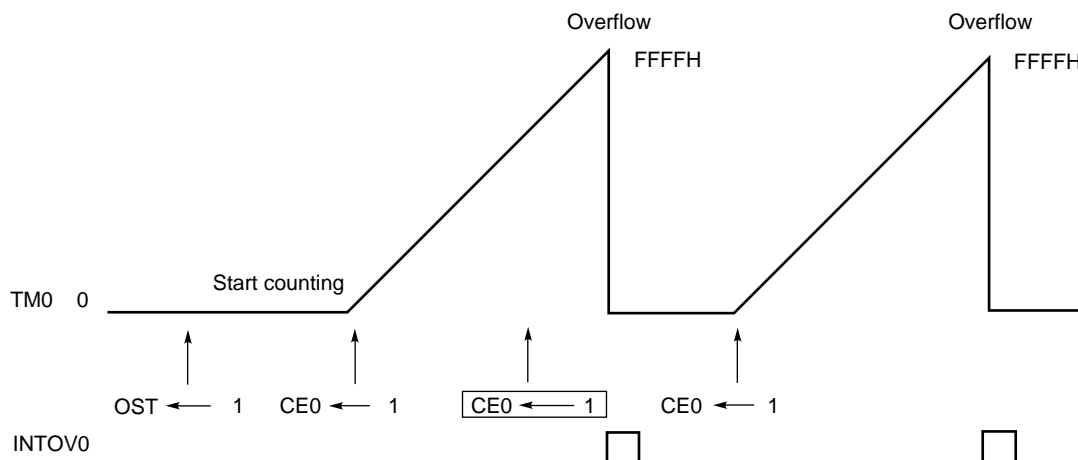
**10.4.3 Overflow**

If TM0 overflows when it is counting external or internal count clock pulses, bit OVFO of the TOVS register is set to generate an overflow interrupt (INTOV0).

Setting the OST bit of TUM0 to 1 can cause the timer to stop after an overflow occurs. If the timer stops due to an overflow, it will not restart until CE0 is set to 1 by software.

Setting CE0 to 1 when the timer is counting does not affect the operation of the timer.

**Figure 10-2. Operation after an Overflow (When ECLR0 = 0 and OST = 1)**



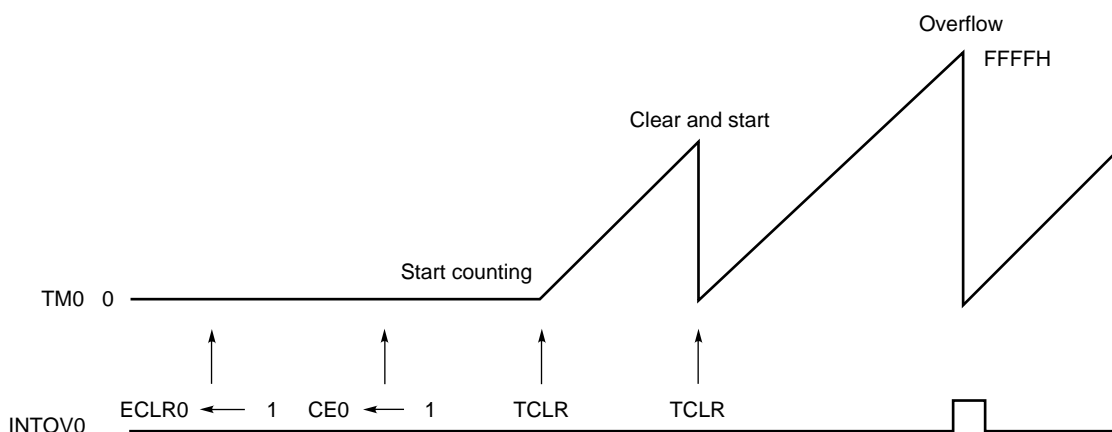
**10.4.4 Clearing/Starting the Timer by the TCLR Input**

Timer 0 usually starts counting when bit CE0 of register TMC0 is set to 1. However, external input TCLR can be used to clear TM0 and cause it to start counting.

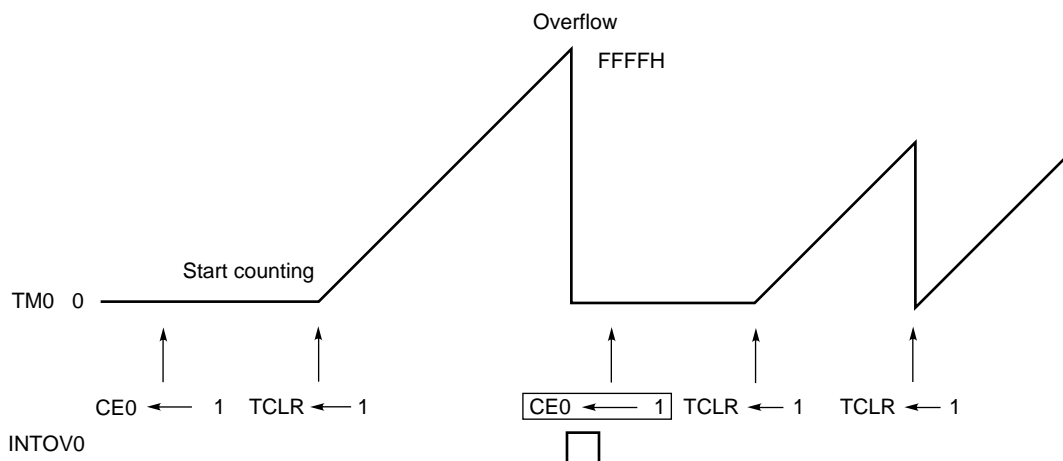
When ECLR0 = 1 and OST = 0, setting CE0 to 1, then inputting an effective edge to TCLR causes the timer to start counting. When TM0 is operating, inputting an effective edge to TCLR clears the timer and causes it to restart counting. (See **Figure 10-3**.)

When ECLR0 = 0 and OST = 1, setting CE0 to 1, then inputting an effective edge to TCLR causes TM0 to start counting. When TM0 overflows, it stops counting and is kept from restarting until an effective edge is input to TCLR. If an effective edge is detected at TCLR when TM0 is operating, TM0 is cleared, then starts and continues counting. (See **Figure 10-4**.)

**Figure 10-3. Clearing/Starting TM0 by a TCLR Input (When ECLR0 = 1 and OST = 0)**



**Figure 10-4. Clearing/Starting TM0 by a TCLR Input and the Related Overflow (When ECLR0 = 0 and OST = 1)**



### 10.4.5 Capturing

The capture register can capture the count value from TM0 in synchronization with an external trigger, independently of the count clock of TM0. This operation is called capturing. An effective edge detected at an external interrupt request input pin INTP<sub>n</sub> (n = 00 to 03) is used as an external trigger (called capture trigger) for capturing. The count value is sent from TM0 to a capture register in synchronization with this capture trigger signal. The capture register retains its content until another capture trigger occurs.

In addition, the INTP<sub>n</sub> input signal causes an interrupt signal INTCC<sub>n</sub>.

**Table 10-2. Capture Trigger Signal for the 16-Bit Capture Register (TM0)**

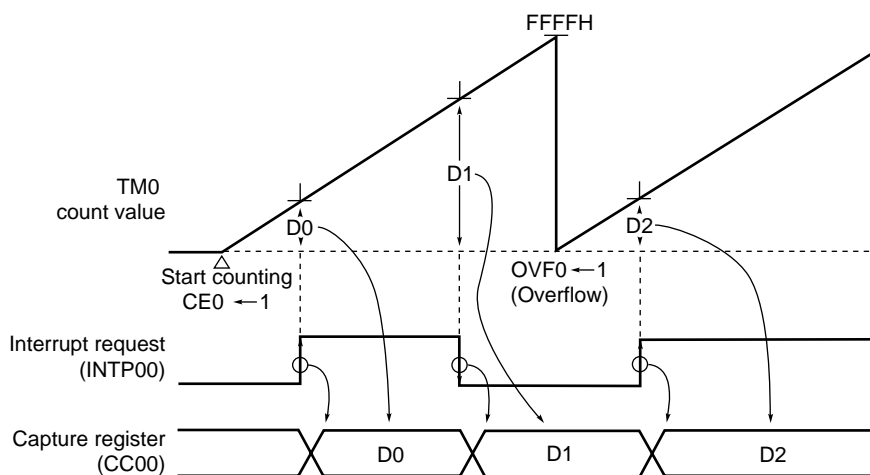
Capture register	Capture trigger signal
CC00	INTP00
CC01	INTP01
CC02	INTP02
CC03	INTP03

**Remark** CC00 to CC03 are capture/compare registers. Timer unit mode register 0 (TUM0) is used to specify whether a capture/compare register is to be used as a capture register or compare register.

The effective edge of the capture trigger is specified using the ICU mode register (IMOD).

If both rising and falling edges are specified as capture triggers, it is possible to measure the width of an external input pulse. If either a rising or a falling edge is specified as a capture trigger, it is possible to measure the period of an input pulse.

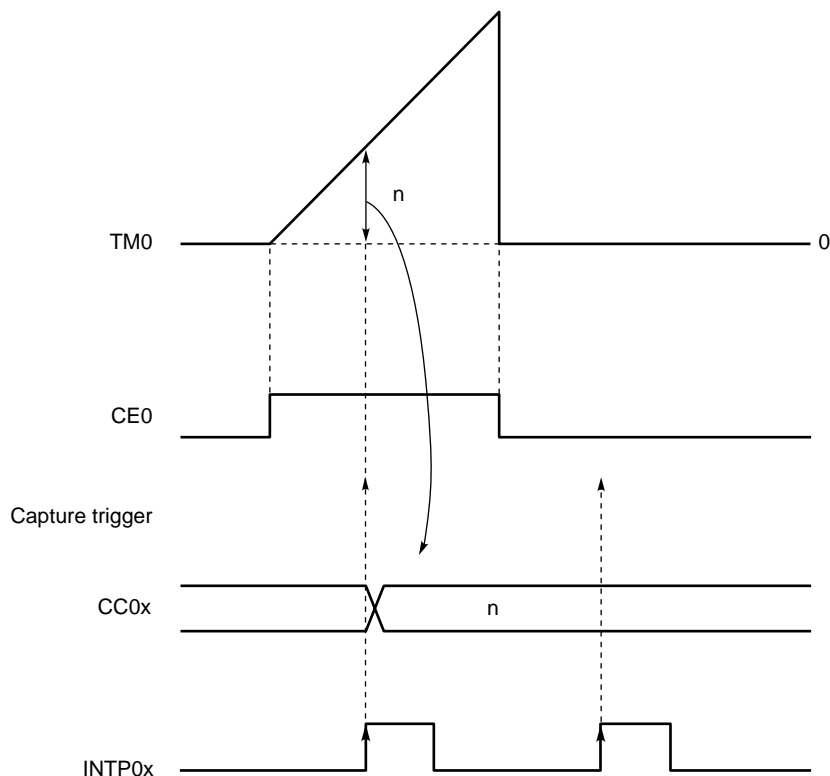
**Figure 10-5. Example of TM0 Capturing (When Both Rising and Falling Edges Are Specified)**



**Remark** D<sub>n</sub> (n = 0, 1, 2, ...): TM0 count value

If CE0 is cleared (0), inputting an interrupt signal does not trigger capturing.



**Figure 10-6. Example of TM0 Capturing****10.4.6 Comparison**

A value in a compare register is compared with the count value in TM0. This operation is called comparison.

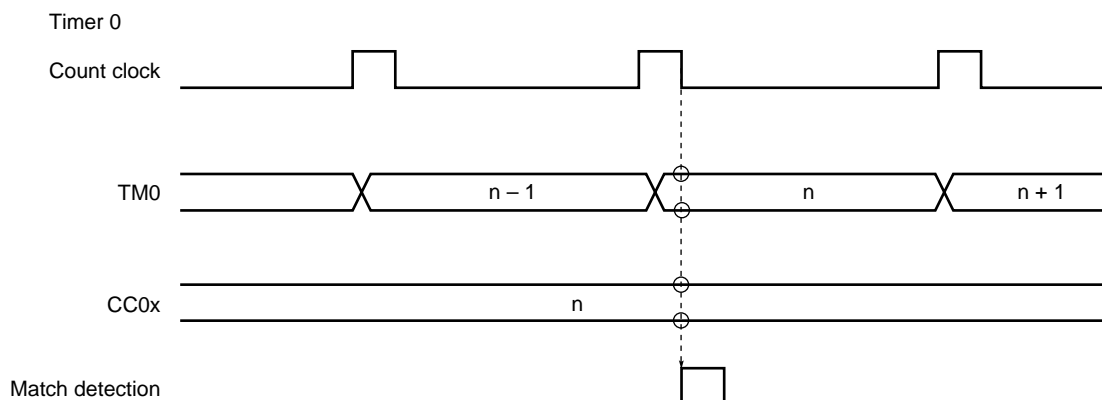
When a value previously set in the compare register matches the count value in TM0, a match signal is sent to the output control circuit. (See **Figure 10-7.**) The match signal changes the level at the timer output pins (TO00 and TO01), and generates an interrupt request signal at the same time.

**Table 10-3. Interrupt Request Signal from the 16-Bit Compare Register (TM0)**

Compare register	Interrupt request signal
CC00	INTCC00
CC01	INTCC01
CC02	INTCC02
CC03	INTCC03

**Remark** CC00 to CC03 are capture/compare registers. Timer unit mode register 0 (TUM0) is used to specify whether a capture/compare register is to be used as a capture register or compare register.

Figure 10-7. Example of Comparison



**Remark** A match is detected immediately after the timer is incremented, generating a match detection signal.

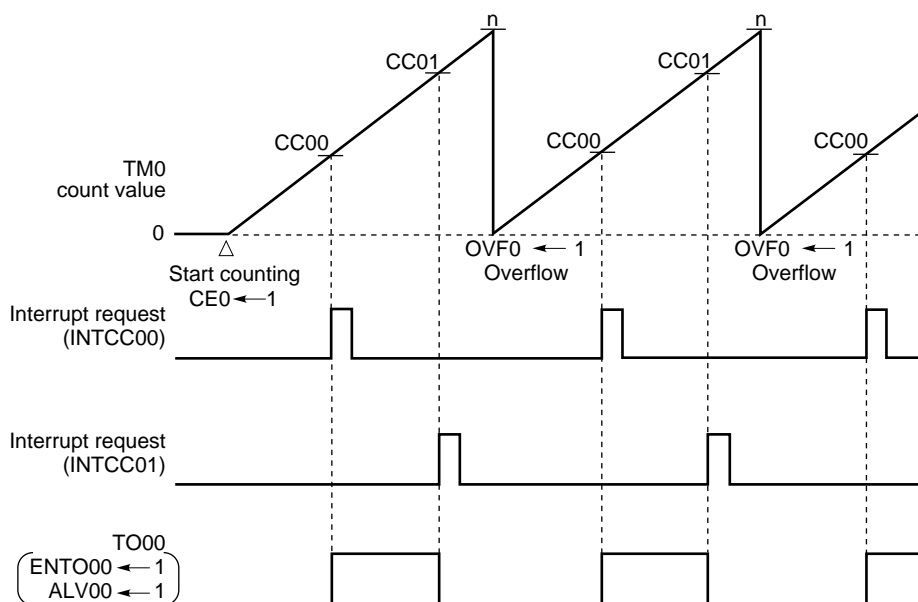
TM0 has two timer output pins (TO00 and TO01).

When the count value in TM0 is compared with the content of CC00, if they match, the output level of TO00 is set. When the count value in TM0 is compared with the content of CC01, if they match, the output level of TO00 is reset.

Similarly, when the count value in TM0 is compared with the content of CC02, if they match, the output level of TO01 is set. When the count value in TM0 is compared with the content of CC03, if they match, the output level of TO01 is reset.

The output level of TO00 and TO01 can be specified using TOC0.

Figure 10-8. Example of TM0 Comparison (Set/Reset Output Mode)



## 10.5 OPERATION OF TIMER 1

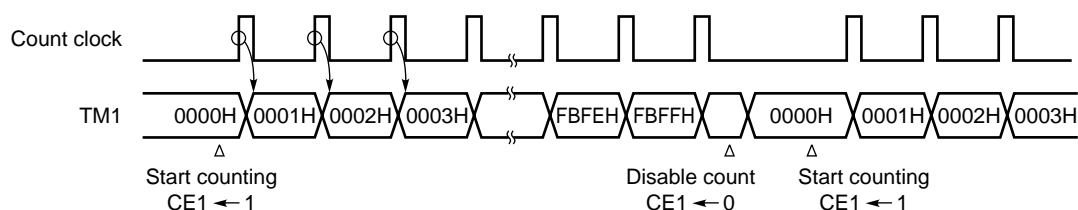
### 10.5.1 Counting

Timer 1 functions as a 16-bit interval timer. Timer control register 1 (TMC1) is used to specify the operation of timer 1.

Timer 1 counts internal count clock pulses ( $\phi/32$  to  $\phi/256$ ) specified by bits PRS0 and PRM0 of register TMC1.

When the count value in TM1 matches the content of CM1, TM1 is cleared, and a match interrupt (INTCM1) is generated simultaneously.

**Figure 10-9. Basic Operation of Timer 1**



### 10.5.2 Input Clock Selection

An internal count clock is selected from  $\phi/32$ ,  $\phi/64$ ,  $\phi/128$ , and  $\phi/256$  according to the setting of bits PRS0 and PRM0 of register TMC1.

**Caution** Do not switch the count clock when the timer is operating.

PRS0		PRM0	Count clock
0	0	0	$\phi/32$
0	0	1	$\phi/64$
0	1	0	$\phi/128$
0	1	1	RFU (reserved)
1	0	0	$\phi/64$
1	0	1	$\phi/128$
1	1	0	$\phi/256$
1	1	1	RFU (reserved)

### 10.5.3 Overflow

If TM1 overflows when it counts an internal count clock pulses, bit OV1 of the TOVS register is set.

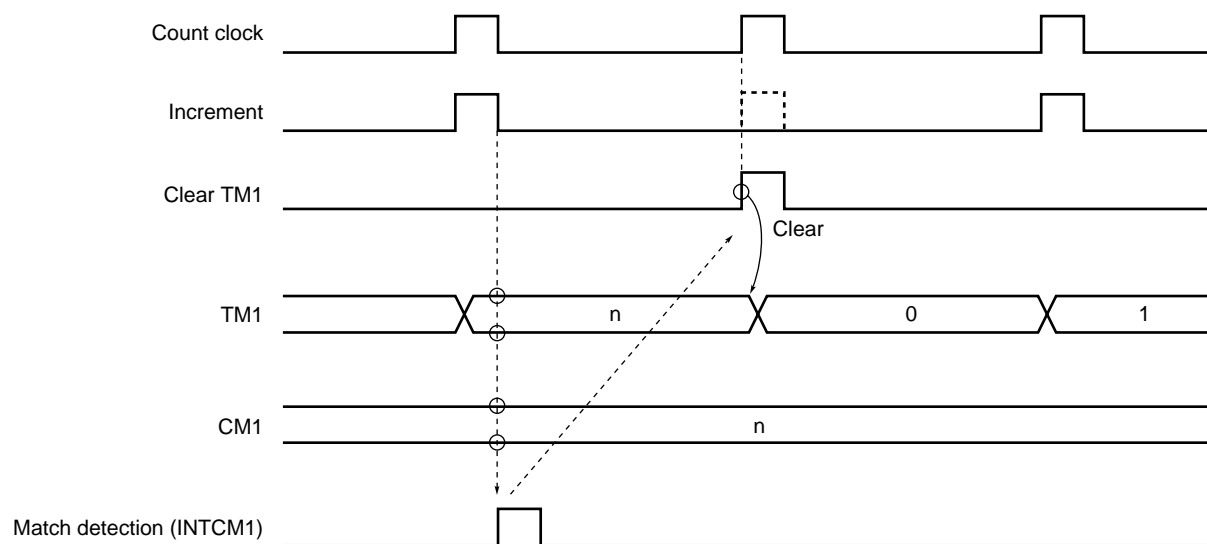
### 10.5.4 Comparison

A value in the compare register (CM1) is compared with the count value in TM1. This operation is called comparison.

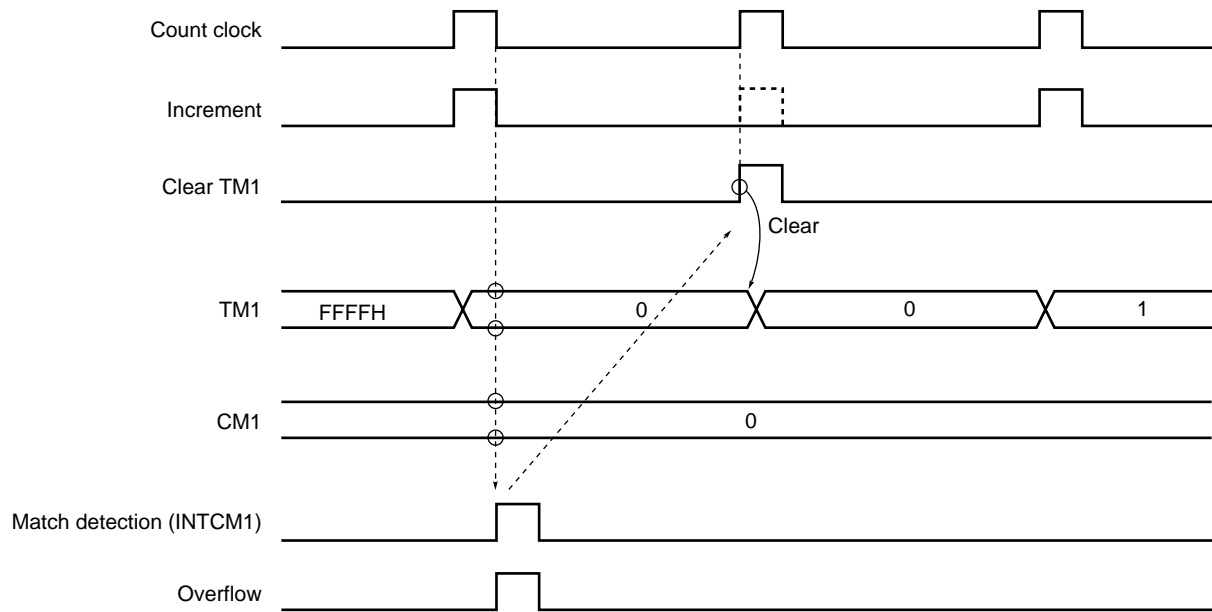
If a match is detected in comparison, an interrupt (INTCM1) occurs. After an interrupt occurs, TM1 is cleared to 0 at the next count timing. (See **Figure 10-10**.) This function enables timer 1 to be used as an interval timer.

CM1 can be set with 0. If CM1 overflows and TM1 becomes 0, a match is detected, and INTCM1 occurs. TM1 is cleared (0) at the next count timing, but INTCM1 does not occur on this match. (See **Figure 10-11**.)

**Figure 10-10. Operation from When CM1 Is 1 until It Is FFFFH**



**Remark** Interval = (n + 1) x count clock period  
 n = 1 to 65,535 (FFFFH)

**Figure 10-11. Operation with CM1 Cleared to 0**

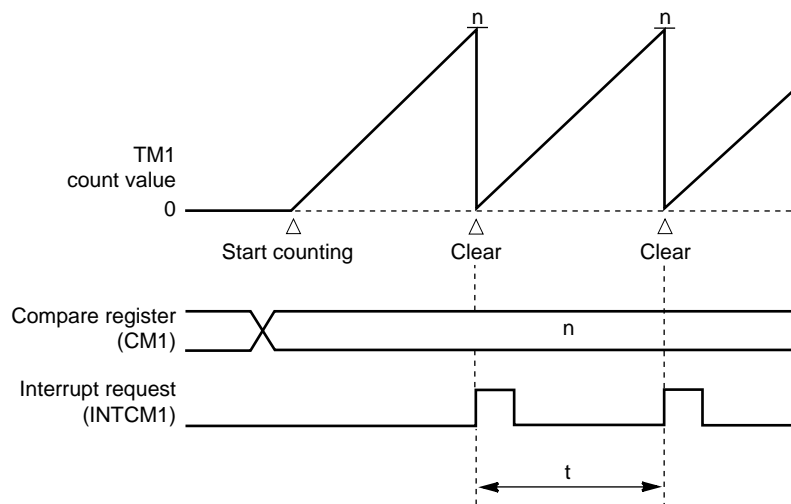
**Remark** Interval = (FFFFH + 2) x count clock period

## 10.6 EXAMPLES OF APPLICATIONS

### (1) Operation as an interval timer (timer 1)

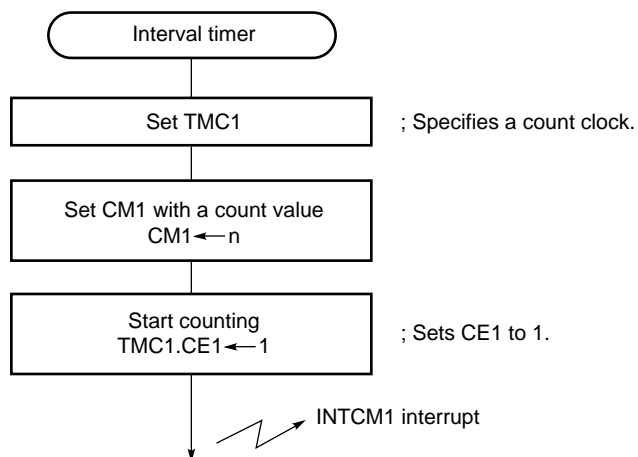
In this example, timer 1 is set with an interval previously set in compare register CM1, and used as an interval timer to generate interrupts repeatedly at the set intervals. Figure 10-12 shows the timing, and Figure 10-13 is the flowchart of the setting procedure.

**Figure 10-12. Interval Timer Operation Timing (Timer 1)**



**Remark** n : Value in CM1  
t : Interval timer = (n + 1) x count clock period

**Figure 10-13. Interval Timer Operation Setting Procedure (Timer 1)**



**(2) Pulse width measurement (timer 0)**

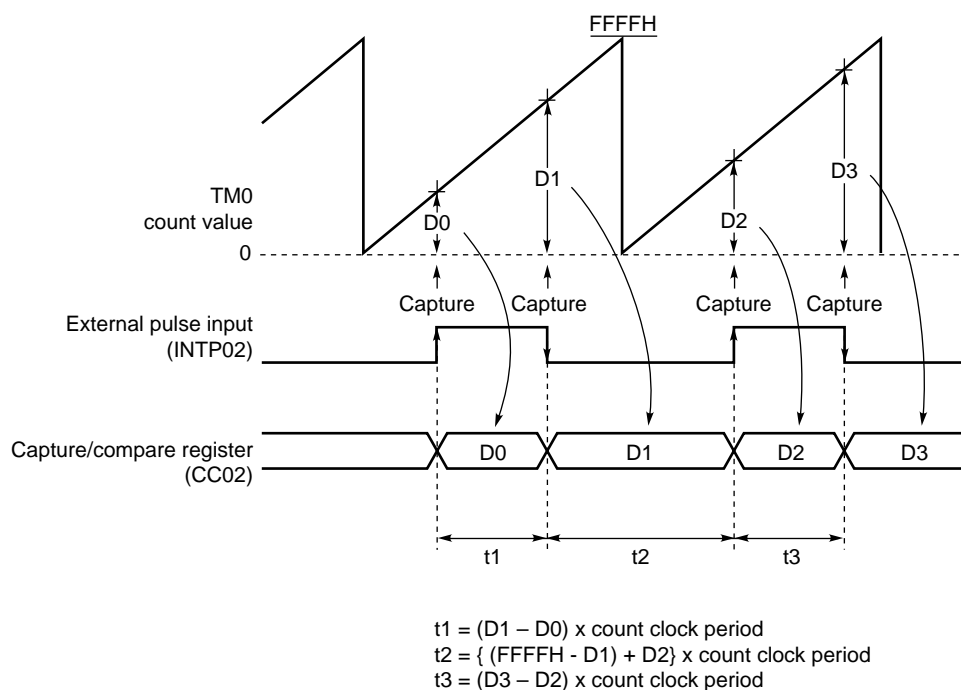
Timer 0 is used for pulse width measurement. In this example, the timer is used to measure the width of the high or low level of an external pulse input at pin INTP02.

As shown in Figure 10-14, a capture/compare register (CC02) captures the count value from timer 0 (TM0) in synchronization with an effective edge (either rising or falling edge) of an input to INTP02.

When the  $n$ th effective edge is detected, CC02 captures the count value ( $D_n$ ) from TM0 on that effective edge, and the difference between this count value ( $D_n$ ) and the count value ( $D_{n-1}$ ) captured at the previous effective edge ( $n-1$ ) is calculated. The pulse width is obtained by multiplying this difference by the count clock period.

Figure 10-15 shows the procedure for setting up pulse width measurement.

**Figure 10-14. Pulse Width Measurement Timing (Timer 0)**



**Remark**  $D_n$ : TM0 count value ( $n = 0, 1, 2, \dots$ )

Figure 10-15. Pulse Width Measurement Setting Procedure (Timer 0)

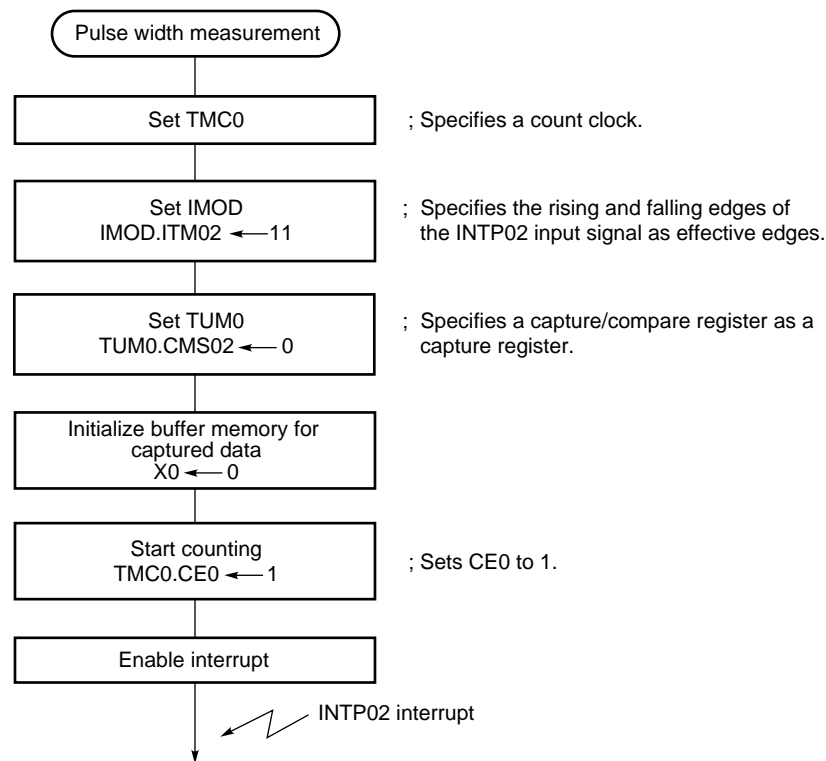
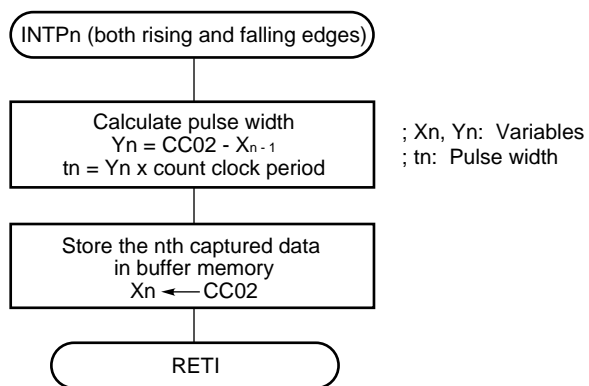


Figure 10-16. Pulse Width Calculation Interrupt Request Handling Routine (Timer 0)



**Caution** If an overflow occurs more than once between the (n-1)th capture and nth capture, it is impossible to measure the pulse width.



**(3) Operation as PWM output (timer 0)**

An arbitrary square waveform can be output from the timer output pins (TO00 and TO01) by combining timer 0 and the timer output function.

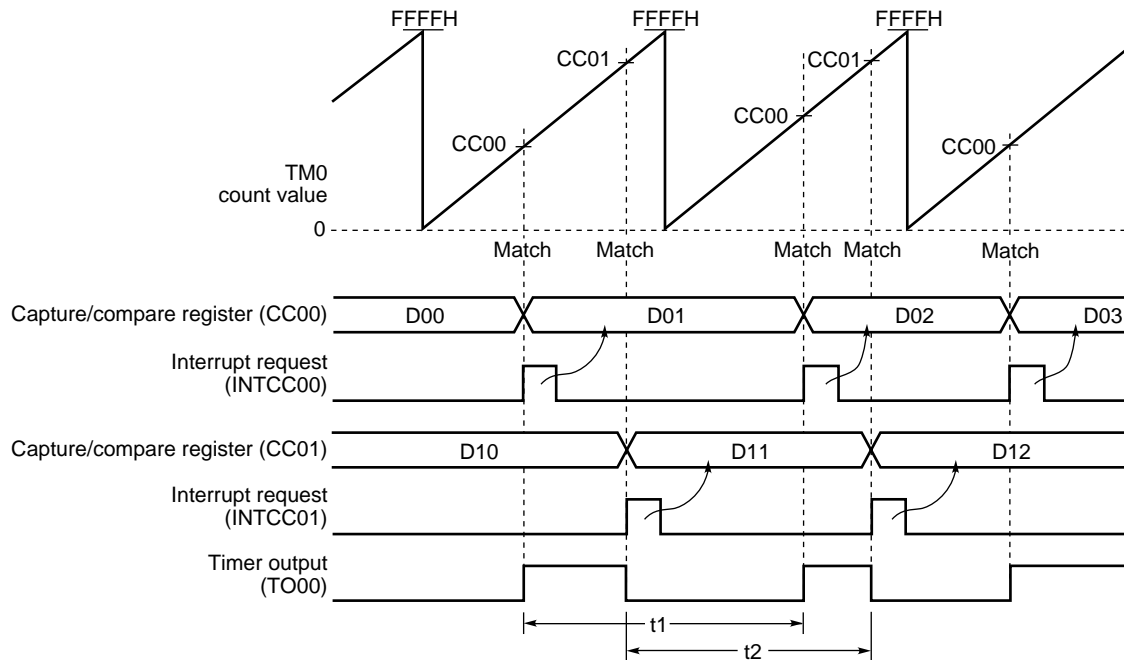
**(a) How to use timer 0**

PWM output uses two capture/compare registers (CC00 and CC01). INTP00/TO00 should be placed in the set/reset output mode. This setting enables outputting a PWM signal with 16-bit precision from pin TO00. The timing relationships are shown in Figure 10-17.

When timer 0 is used as a 16-bit timer, a value in a capture/compare register CC00 determines the timing of the rising edge of the PWM output, and a value in CC01 determines the timing of the falling edge, as shown in Figure 10-17.

Figure 10-18 shows the procedure for setting up the PWM output.

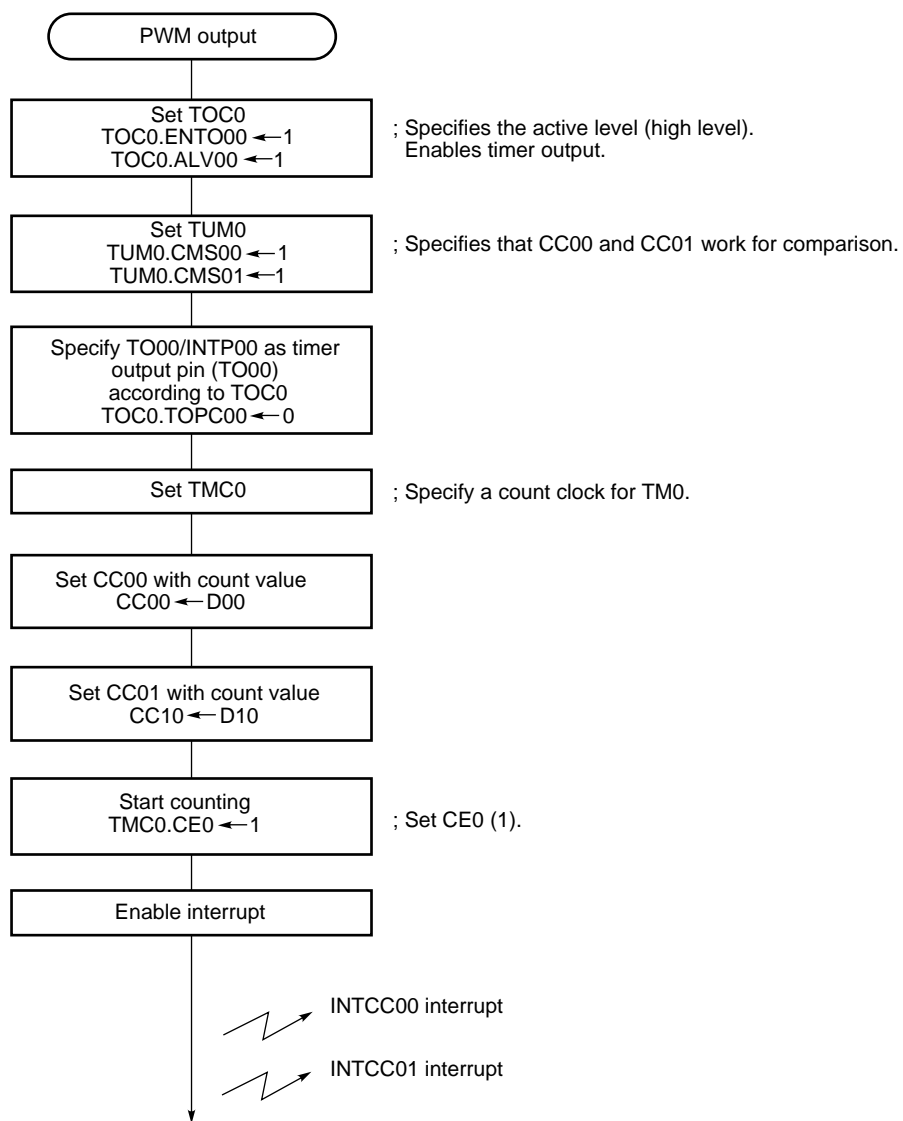
**Figure 10-17. PWM Output Timing (TM0)**

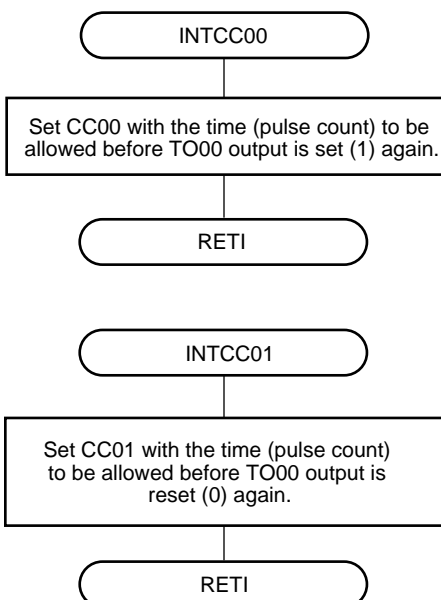


**Remark** Dxx: Value set in a compare register

$$t1 = \{(FFFFH - D00) + D01\} \times \text{count clock period}$$

$$t2 = \{(FFFFH - D10) + D11\} \times \text{count clock period}$$

**Figure 10-18. PWM Output Setting Procedure (Timer 0)**

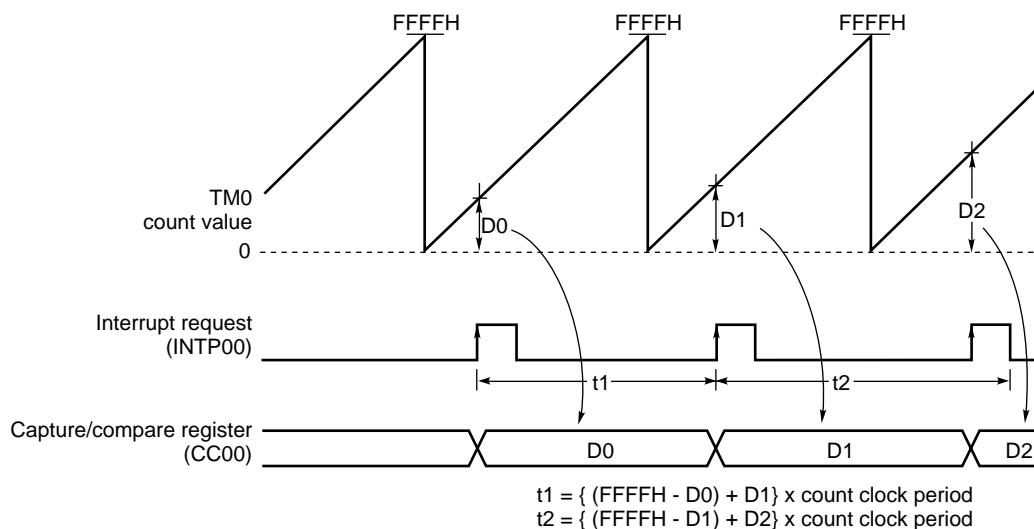
**Figure 10-19. Compare Value Rewrite Interrupt Request Handling Routine (Timer 0)****(4) Period measurement (timer 0)**

Timer 0 can be used to measure the period of an external pulse input at pin INTP<sub>n</sub> ( $n = 00$  to  $03$ ). In this example, the period of an external pulse input at INTP00 is measured with 16-bit precision by combining timer 0 and CC00 (capture/compare register).

The IMOD register is used to specify the rising edge of an input signal at INTP00 as an effective edge. When the  $n$ th rising edge is detected, CC00 captures the count value ( $D_n$ ) from TM0 on that edge, and the difference between this count value ( $D_n$ ) and the count value ( $D_{n-1}$ ) captured at the previous rising edge ( $n-1$ ) is calculated. The pulse period is obtained by multiplying this difference by the count clock period.

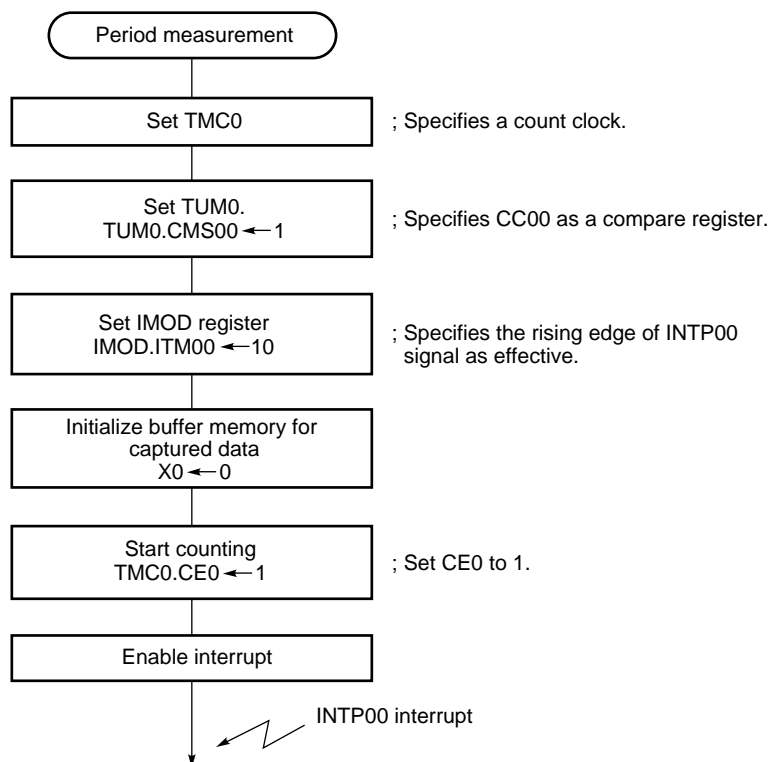
Figure 10-21 shows the procedure for setting up period measurement.

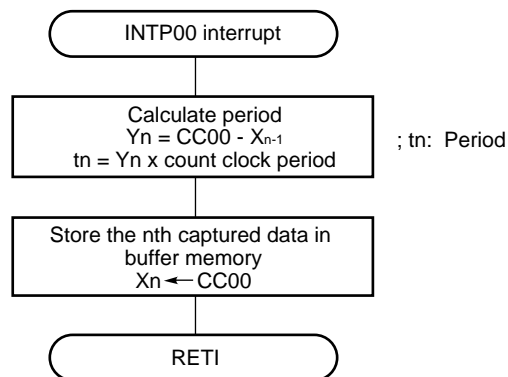
Figure 10-20. Pulse Width Measurement Timing (TM0)



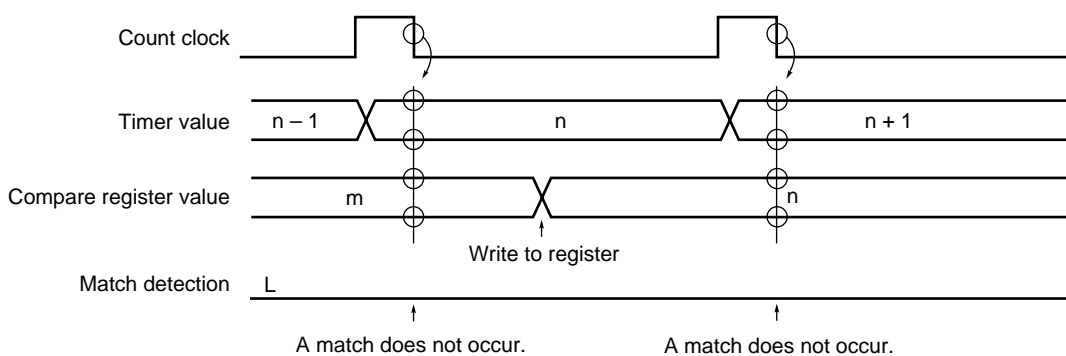
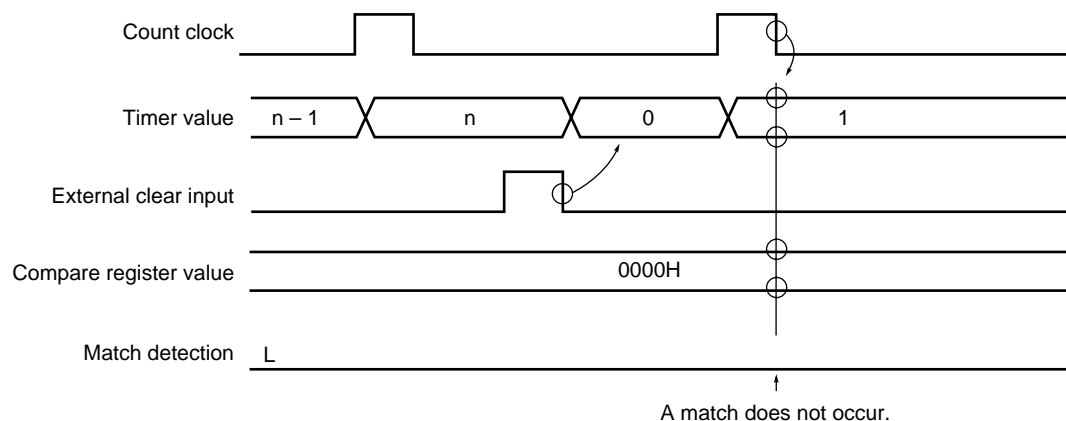
**Remark** Dn: TM0 count value (n = 0, 1, 2, ...)

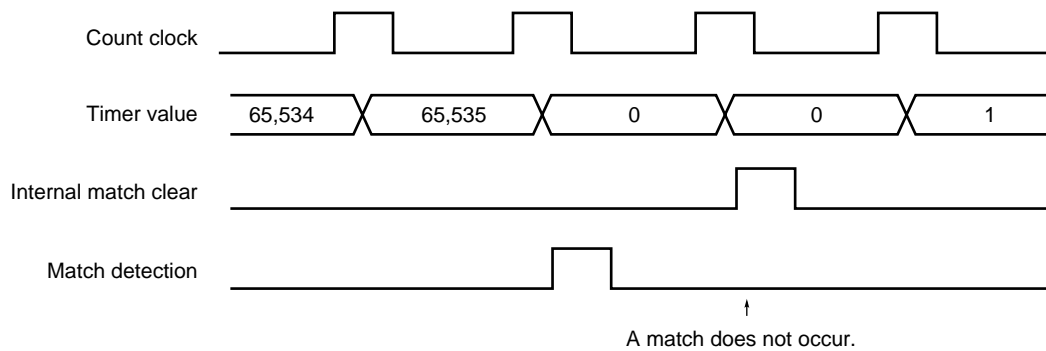
Figure 10-21. Period Measurement Setting Procedure (Timer 0)



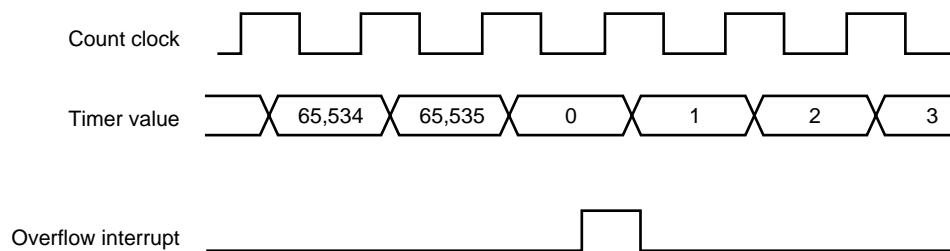
**Figure 10-22. Period Calculation Interrupt Request Handling Routine (Timer 0)****10.7 CAUTIONS**

The detection of a match using a compare register is always carried out immediately after the timer is incremented. A match is not detected in the following cases.

**(1) When writing to a compare register (TM0 and TM1)****(2) When cleared externally (TM0)**

**(3) When cleared by a timer (TM1)**

If a timer is operating in a free-running mode, when it overflows, it is cleared to 0.



## CHAPTER 11 WATCHDOG TIMER FUNCTIONS

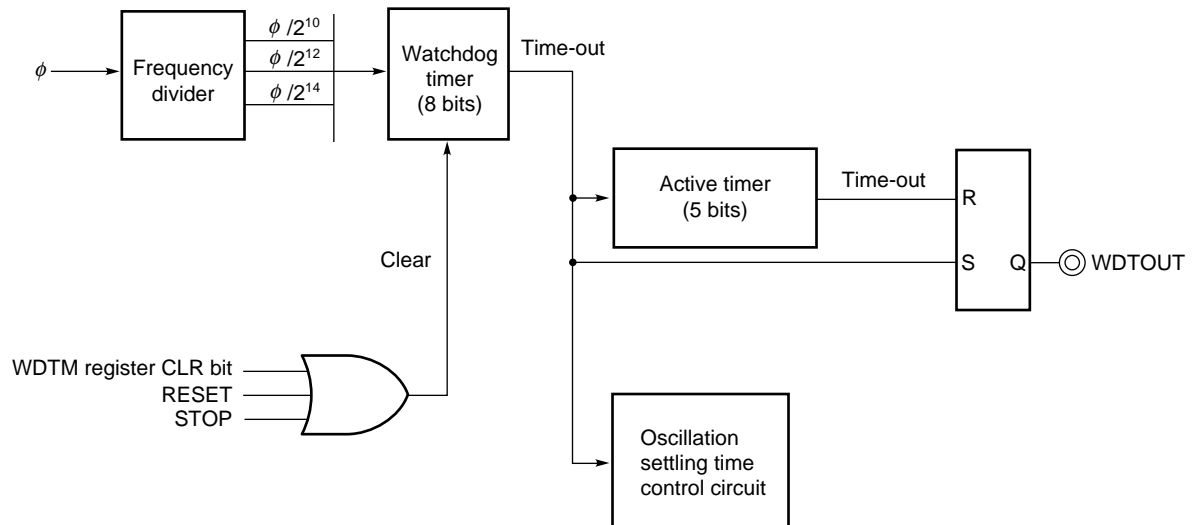
The watchdog timer is intended to prevent program crash and deadlock.

### 11.1 FEATURES

- The following three different time-out time values can be specified: 10.5 ms, 41.9 ms, and 167.8 ms (when system clock  $\phi = 25$  MHz)
- Watchdog timer time-out output (WDTOUT)

### 11.2 CONFIGURATION

**Figure 11-1. Watchdog Timer Block Diagram**



**Remark**  $\phi$ : System clock

**(1) Watchdog timer**

One of the watchdog timer functions is to secure the oscillation settling time of the system clock. When the system is reset or placed in the STOP mode, the timer is cleared to 00H.

The watchdog timer behaves in the standby modes as follows:

**(a) STOP mode**

The watchdog timer stops counting. When the system is released from the STOP mode, the timer value is cleared.

The watchdog timer starts counting at 00H, and keeps counting until a time-out occurs. A time-out signal is supplied to the oscillation settling time control circuit, thus starting to supply the system clock pulse. At this point, the WDTOUT pin does not become active. If the system is released from the STOP mode by the  $\overline{\text{NMI}}$  pin, the timer continues counting.

**(b) IDLE mode**

The watchdog timer stops counting, but it holds the count value.

When the system is released from the IDLE mode, the watchdog timer resumes counting by starting at the current count value.

**(c) HALT mode**

The watchdog timer continues counting.

**(2) Active timer**

The watchdog timer outputs the WDTOUT signal when it times out. The active timer retains this signal for 32 clock cycles.

When the watchdog timer times out, it can cause a system reset by connecting the WDTOUT and  $\overline{\text{RESET}}$  pins through an external circuit.

**11.3 OPERATION**

The watchdog timer indicates that the program or system is running normally, by keeping the WDTOUT pin from becoming active.

To use the watchdog timer, it is necessary to specify the WDTM register so that the watchdog timer is cleared (restarted to count) at constant intervals during program execution or at the beginning of a subroutine. If the watchdog timer expires because it is not cleared within a specified period of time, the WDTOUT pin becomes active, indicating a program failure. In addition, the WDT time-out flag (OV) is set. This flag is cleared by clearing the WDT counter.

To use a watchdog timer time-out as an interrupt source, it is necessary to connect the WDTOUT pin to an external interrupt request pin ( $\overline{\text{INTPn}}$  or  $\overline{\text{NMI}}$ ) through an external circuit.



## 11.4 WDT MODE REGISTER (WDTM)

The WDT mode register controls the behavior of the watchdog timer.

The WDT mode register can be read from and written to in 8-bit units.

	7	6	5	4	3	2	1	0	
WDTM	KEY		BWOS	EN	CL/OV	WDTI			Address C00000C0H
									When reset, 00H

Bit position	Bit name	Description												
7-5	KEY	KEY data Identifies data. KEY should be set as follows: KEY = 011 A value other than 011 cannot be written to the WDTM register. A value read from this register is always 000.												
4	BWOS	BLOCK/WDTOUT Output Select Selects the BLOCK or WDTOUT output signal as output. 0: WDTOUT output 1: BLOCK output												
3	EN	Enable Causes the watchdog timer to start or stop counting. 0: Stop counting 1: Start counting												
2	CL/OV	Clear/Overflow When the watchdog timer is write-accessed, this bit specifies whether the count value in the watchdog timer and the time-out flag are cleared. <table border="1"><thead><tr><th>CL</th><th>Operation</th></tr></thead><tbody><tr><td>0</td><td>Neither the count value nor the time-out flag is cleared (nothing is carried out).</td></tr><tr><td>1</td><td>The count value and the time-out flag are cleared.</td></tr></tbody></table> When the watchdog timer is read-accessed, this bit specifies that the watchdog timer time-out flag can be read-accessed. If a time-out occurs after the system is released from the STOP mode, the time-out flag is not set. <table border="1"><thead><tr><th>OV</th><th>Operation</th></tr></thead><tbody><tr><td>0</td><td>A time-out has not occurred.</td></tr><tr><td>1</td><td>A time-out has occurred.</td></tr></tbody></table>	CL	Operation	0	Neither the count value nor the time-out flag is cleared (nothing is carried out).	1	The count value and the time-out flag are cleared.	OV	Operation	0	A time-out has not occurred.	1	A time-out has occurred.
CL	Operation													
0	Neither the count value nor the time-out flag is cleared (nothing is carried out).													
1	The count value and the time-out flag are cleared.													
OV	Operation													
0	A time-out has not occurred.													
1	A time-out has occurred.													

Bit position	Bit name	Description																																				
1, 0	WDTI	<div><div>WDT Input Specifies the count clock for the watchdog timer.</div><table><tr><th colspan="2">WDTI</th><th>Count clock</th><th colspan="3">Time-out time (ms)</th></tr><tr><th colspan="2"></th><th></th><th>16 MHz</th><th>20 MHz</th><th>25 MHz</th></tr><tr><td>0</td><td>0</td><td><math>2^{10}/\phi</math></td><td>16.4</td><td>13.1</td><td>10.5</td></tr><tr><td>0</td><td>1</td><td><math>2^{12}/\phi</math></td><td>65.5</td><td>52.4</td><td>41.9</td></tr><tr><td>1</td><td>0</td><td><math>2^{14}/\phi</math></td><td>262.1</td><td>209.7</td><td>167.8</td></tr><tr><td>1</td><td>1</td><td colspan="4">This setting is inhibited.</td></tr></table><div><p><math>\phi</math>: System clock</p><p>If this bit is changed while the watchdog timer is running, normal operation is not guaranteed.</p></div></div>	WDTI		Count clock	Time-out time (ms)						16 MHz	20 MHz	25 MHz	0	0	$2^{10}/\phi$	16.4	13.1	10.5	0	1	$2^{12}/\phi$	65.5	52.4	41.9	1	0	$2^{14}/\phi$	262.1	209.7	167.8	1	1	This setting is inhibited.			
WDTI		Count clock	Time-out time (ms)																																			
			16 MHz	20 MHz	25 MHz																																	
0	0	$2^{10}/\phi$	16.4	13.1	10.5																																	
0	1	$2^{12}/\phi$	65.5	52.4	41.9																																	
1	0	$2^{14}/\phi$	262.1	209.7	167.8																																	
1	1	This setting is inhibited.																																				

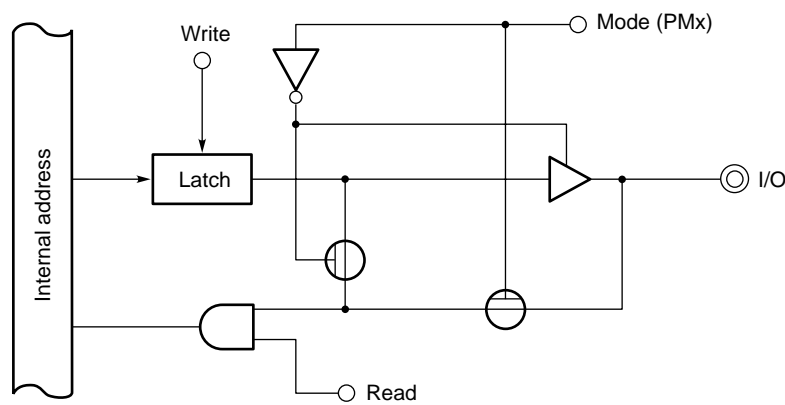
## CHAPTER 12 PORT FUNCTIONS

The V821 pins are dual-function pins that can function as both port and control pins. See **Section 2.1** for details of each pin.

### 12.1 FEATURES

- 10 input/output ports (P00 to P09)

### 12.2 CONFIGURATION



### 12.3 PIN FUNCTIONS OF PORT 0

Port register 0 (P0) is a 16-bit input/output port. For this register, it is possible to specify input/output in 16-bit units.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
P0	0	0	0	0	0	0	P09	P08	P07	P06	P05	P04	P03	P02	P01	P00	
																	Address C0000014H
																	When reset, Not defined

Bit position	Bit name	Description
9-0	P9-P0	Port data Input/output port

**Caution** If port register 0 is read when control mode is specified, the level at the corresponding pin of port 0 is read. The level of the pin, such as  $\overline{\text{DACK0}}$ , that functions as an output pin in control mode is not read.

Port mode register 0 (PM0) is used to specify whether port 0 is to work as an input or output port.

#### (1) When specified as an output port (PM0 = 0)

When the port register is write-accessed, a value is written to the output latch. When it is read-accessed, a value is read from the output latch.

#### (2) When specified as an input port (PM0 = 1)

When the port register is write-accessed, a value is written to the output latch. When it is read-accessed, the level of the corresponding pin of port 0 is read.

## 12.4 CONTROL REGISTER

The input/output ports are controlled using port mode control register 0 (PMC0) and port mode register 0 (PM0).

### 12.4.1 Port Mode Control Register 0 (PMC0)

The control mode is selected if the corresponding bit of PMC0 is set (1). The port mode is selected if the bit is reset (0).

PMC0 can be read from and written to in 16-bit units.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
PMC0	0	0	0	0	0	0	MS9	MS8	MS7	MS6	MS5	MS4	MS3	MS2	MS1	MS0	Address C0000010H	When reset, 0000H

Bit position	Bit name	Description
9-0	MS9-MS0	Mode Select 0: Port mode 1: Control mode

### 12.4.2 Port Mode Register 0 (PM0)

If the port mode is selected, PM0 specifies the port pin as either input or output. Table 12-1 lists how the port behaves when PMC0 is set to the control mode.

PM0 can be read from and written to in 16-bit units.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
PM0	0	0	0	0	0	0	PM9	PM8	PM7	PM6	PM5	PM4	PM3	PM2	PM1	PM0	Address C0000012H	When reset, 03FFH

Bit position	Bit name	Description
9-0	PM9-PM0	Port Mode 0: Output mode (output buffer on) 1: Input mode (output buffer off)

**Table 12-1. Port Operation**

Port name	Port operation			
	MSn = 0		MSn = 1	
	PMn = 0	PMn = 1	PMn = 0	PMn = 1
P00/TCLR	P00 output	P00 input	TCLR input	
P01/DREQ0	P01 output	P01 input	DREQ0 input	
P02/ $\overline{\text{DACK0}}$	P02 output	P02 input	$\overline{\text{DACK0}}$ input	
P03/DREQ1	P03 output	P03 input	DREQ1 input	
P04/ $\overline{\text{DACK1}}$	P04 output	P04 input	$\overline{\text{DACK1}}$ input	
P05/SI	P05 output	P05 input	SI input	
P06/SO	P06 output	P06 input	SO output	
P07/ $\overline{\text{SCLK}}$	P07 output	P07 input	$\overline{\text{SCLK}}$ input/output	
P08/TXD/ $\overline{\text{UBE}}$	P08 output	P08 input	$\overline{\text{UBE}}$ output	TXD output
P09/RXD/ $\overline{\text{TC}}$	P09 output	P09 input	$\overline{\text{TC}}$ output	RXD input

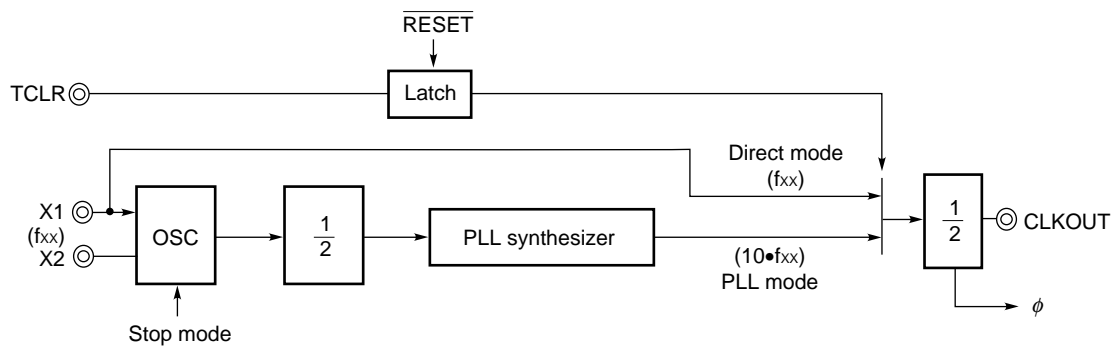
## CHAPTER 13 CLOCK GENERATION FUNCTIONS

The clock generator generates and controls the internal clock pulse ( $\phi$ ) for the CPU and other built-in hardware units.

### 13.1 FEATURES

- Frequency multiplication (5 times) using a PLL (phase locked loop) synthesizer
- Clock sources
  - Resonator-based oscillation:  $f_{xx} = 1/5 \times \phi$  (PLL mode)
  - External clock:  $f_{xx} = 1/5 \times \phi$  (PLL mode)
  - External clock:  $f_{xx} = 2 \times \phi$  (direct mode)

### ★ 13.2 CONFIGURATION



- $\phi$  : Internal clock frequency ( $\phi = 1/2 \cdot 10 \cdot f_{xx}$ : PLL mode)  
 Internal clock frequency ( $\phi = 1/2 \cdot f_{xx}$ : Direct mode)  
 OSC : Oscillator (only for the PLL mode)

### 13.3 INPUT CLOCK SELECTION

The clock generator consists of a clock oscillator and PLL synthesizer. Connecting, for example, a 5 MHz quartz or ceramic resonator to the X1 and X2 pins can generate a 25 MHz system clock pulse.

Moreover, an external clock pulse can be input directly to the oscillator. In this case, the clock signal should be supplied to the X1 pin, and the X2 pin should be left open.

The clock generator supports two basic operation modes: PLL and direct modes. Either operation mode is selected during a reset using the TCLR pin. The input signal at the TCLR pin is latched when the  $\overline{\text{RESET}}$  pin becomes inactive. However, the input signal should be kept at the same level during a reset (while the  $\overline{\text{RESET}}$  pin is active). Assume that the hold time for  $\overline{\text{RESET}}$  signal rising at the TCLR pin is one clock. A malfunction may occur if the level of the TCLR pin varies during a reset.

TCLR	Operation mode
0	PLL mode
1	Direct mode

#### 13.3.1 Direct Mode

In the direct mode, an external clock pulse having a frequency twice as high as the frequency of the system clock is input to the clock generator. Operation with low power dissipation is possible in this mode, because the OSC and PLL synthesizer do not operate. The direct mode is suitable for applications in which the V821 operates at a relatively low frequency. To minimize influence by noise, the frequency ( $f_{xx}$ ) of the external clock should be kept at about 32 MHz or less (system clock  $\phi = 16$  MHz).

To use the external clock directly, it is necessary to keep the TCLR pin at 1 during a reset.

#### 13.3.2 PLL Mode

In the PLL mode, the system clock pulse ( $\phi$ ) is generated by connecting an external resonator or supplying an external clock pulse and multiplying its frequency in the PLL synthesizer.

A frequency of up to 25 MHz can be obtained using a 3 to 5 MHz external resonator or clock. So, the PLL mode can be used to implement a low-noise, low-power dissipation system. The frequency ( $5 \times f_{xx}$ ) of the generated system clock pulse ( $\phi$ ) is five times as high as that ( $f_{xx}$ ) of the external resonator or clock.

To use the PLL mode, it is necessary to keep the TCLR pin at 0 during a reset.

#### Examples of clocks used in the PLL mode

System clock frequency ( $\phi$ )	External resonator/clock frequency ( $f_{xx}$ )
25.000 MHz	5.0000 MHz
20.000 MHz	4.0000 MHz
16.384 MHz	3.2768 MHz



**(1) Back-up mode**

In PLL mode, even if the crystal or built-in oscillator is failed for some reason and clock supply is stopped, the internal system clock ( $\phi$ ) at the self-sustained frequency of the VCO can be supplied continuously. This increases the reliability of the application system. This is called the back-up mode. This function is available only for the PLL mode. The BU bit indicates whether the clock is in the back-up mode.

**Caution** If the clock shifts to the back-up mode, it is detected after a lock-up state (UL bit = 0) is entered. This function is provided in case of failures. It does not guarantee all operations at the self-sustained frequency. It is not recommended that clock input be stopped intentionally to enter back-up mode.

**(2) Lock-up time**

Immediately after the system is switched on or released from the stop or idle mode, a frequency settling time (lock-up time) is required before the PLL enters a phase locked state at the specified frequency and becomes stable. This condition is called the unlocked state, while the condition under which the oscillation frequency is stable is called the locked state.

The stable state of the PLL frequency is indicated in the UL flag of the clock control register (CGC).

Once a factor that causes the unlocked state (such as clock stop or power break) occurs, processing that depends on software execution speed, such as real-time processing, should check the UL flag by software immediately after it starts and waits until the clock becomes stable if necessary.

On the other hand, static processing, such as setting on-chip hardware or initializing register or memory data, can be performed without waiting until the UL flag is reset.

★ The UL flag remains set to "0" as long as the lock-up state is maintained and is not initialized by means of system reset.

This flag is read-only and cannot be written.

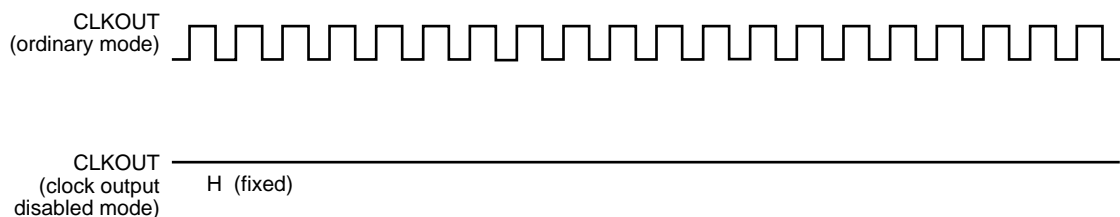
**13.4 CLOCK OUTPUT CONTROL**

The behavior of the CLKOUT pin can be specified using the COE bit of the clock control register (CGC). It is possible to save power effectively by combining the standby modes such as halt, idle, and stop.

**(1) Clock output disabled mode**

The clock output from the CLKOUT pin is disabled.

In this mode, the CLKOUT operation is kept at a complete stop, so it is possible to reduce power dissipation further and suppress noise radiation from the CLKOUT pin.



**13.5 CLOCK CONTROL REGISTER (CGC)**

The CGC register controls the output of the CLKOUT pin.

This register can be read from and written to in 8-bit units except for bits 1 and 2, which can only be read-accessed.

	7	6	5	4	3	2	1	0	
CGC	KEY				CESEL	BU	UL	COE	Address C00000E0H         When reset, 03H

Bit position	Bit name	Description
7-4	KEY	<b>KEY data</b> Identifies data. KEY should be set as follows: KEY = 0111 If KEY is set to other than 0111, it is impossible to write to the CGC register. When read-accessed, the CGC register always outputs 0000.
3	CESEL	<b>Crystal External Select</b> Specifies the function of the X1 and X2 pins. 0: A resonator is connected to the X1 and X2 pins. 1: An external clock is connected to the X1 pin. The X1 pin is provided with a pull-down transistor. During the stop mode, it keeps the X1 bit at the ground level to stop the oscillator. To use an external clock in the PLL mode, it is necessary to set CESEL to 1.
2	BU	<b>Back Up</b> Flag to indicate whether the back-up mode has been selected. 0: The back-up mode has not been selected. 1: The back-up mode has been selected. The back-up mode is indicated in the BU bit after a lock-up state is entered (UL = 0). This bit is cleared to 0 by a system reset or when the stop mode is selected. It is a read-only bit. An attempt to write to it is ignored.
1	UL	<b>Unlock</b> Flag to indicate whether the unlock state has been entered. 0: Locked state 1: Unlocked state The UL bit is set to 1 by a system reset or when the stop mode is selected. Once the clock is locked, it is kept in that state. This bit is a read-only bit. An attempt to write to it is ignored.
0	COE	<b>Clock Out Enable</b> Enables or disables the CLKOUT output. 0: Disables output (the CLKOUT pin is fixed at the high level.) 1: Enables output.

## CHAPTER 14 STANDBY FUNCTIONS

The V821 supports three standby modes to suppress power dissipation. In these standby modes, the operation of the clock is controlled. The HALT instruction is used to select a standby mode. Mode switching is controlled using the standby control register.

### 14.1 FEATURES

- HALT mode (Only the CPU clock stops.)
- IDLE mode (The CPU and peripheral operation clocks stop. The clock generator continues to operate.)
- STOP mode (The entire system, including the clock generator, stops.)

### 14.2 STANDBY MODE

The standby modes of the V821 are detailed below.

#### (1) HALT mode

In this mode, the clock generator (oscillator and PLL synthesizer) continues operation, but the CPU clock stops. Clock supply to other built-in peripheral functions continues to allow them to keep running. Intermittent operation achieved using this standby mode in conjunction with the ordinary operation mode can reduce the total power dissipation of the system.

#### (2) IDLE mode

In this mode, the clock generator (oscillator and PLL synthesizer) continues operation, but internal system clock supply is stopped to bring the entire system to a stop.

When the system is released from the IDLE mode, it is unnecessary to secure oscillation settling time for the oscillator, and therefore it is possible for the system to shift to the ordinary operation quickly.

For the oscillation settling time and current drain, the IDLE mode lies in between the stop and HALT modes. The IDLE mode is suitable for an application where it is necessary to cut the oscillation settling time using a low current drain mode.

#### (3) STOP mode

In this mode, the clock generator (oscillator and PLL synthesizer) is stopped to bring the entire system to a stop. This mode can generate an ultra-low power dissipation condition; only leak current occurs.

**(a) PLL mode**

In this mode, the PLL synthesizer clock output is stopped simultaneously with the oscillator. After the system is released from the STOP mode, it is necessary to allow oscillation settling time for the oscillator. Some programs require a PLL lock-up time.

**(b) Direct mode**

In the direct mode, it is unnecessary to secure lock-up time.

Table 14-1 lists the operation of the clock generator in the ordinary, halt, idle, and STOP modes. An effective low power dissipation system can be implemented by combining and switching these modes.

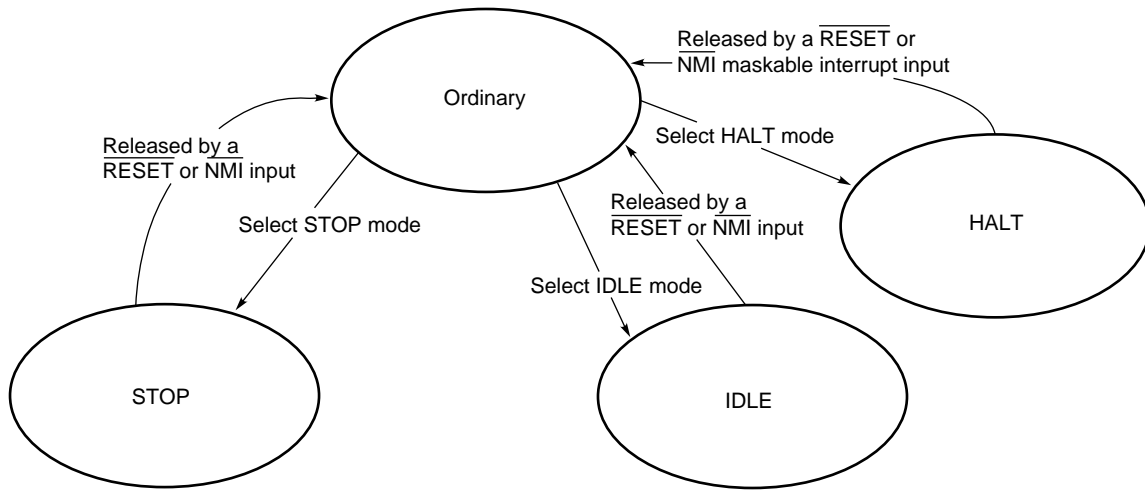
**Table 14-1. Clock Generator Operation under Standby Control**

Clock source		Standby mode	Oscillator (OSC)	PLL synthesizer	Clock supply to the peripheral I/O	Clock supply to the CPU
PLL mode	Resonator-based oscillation	Ordinary	o	o	o	o
		Halt	o	o	o	x
		Idle	o	o	x	x
		Stop	x	x	x	x
	External clock	Ordinary	x	o	o	o
		Halt	x	o	o	x
		Idle	x	o	x	x
		Stop	x	x	x	x
Direct mode		Ordinary	x	x	o	o
		Halt	x	x	o	x
		Idle	x	x	x	x
		Stop	x	x	x	x

o: Operating

x: Stop

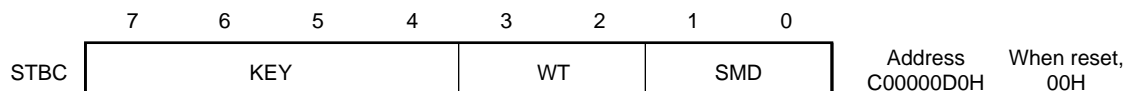
## State transition



## 14.3 STANDBY CONTROL REGISTER (STBC)

The standby control register controls standby.

The STBC register can be read from and written to in 8-bit units.



Bit position	Bit name	Description																																			
7-4	KEY	KEY data Identifies data. KEY should be set as follows: KEY = 0111 If KEY is set to other than 0111, it is impossible to write to the STBC register. When read-accessed, the STBC register always outputs 0000.																																			
3, 2	WT	Wait Time Specifies the count clock for oscillation settling time. <table border="1"><thead><tr><th colspan="2">WT</th><th rowspan="2">Count clock</th><th colspan="3">Oscillation settling time (ms)</th></tr><tr><th></th><th></th><th>16 MHz</th><th>20 MHz</th><th>25 MHz</th></tr></thead><tbody><tr><td>0</td><td>0</td><td><math>2^{19}/\phi</math></td><td>32.8</td><td>26.2</td><td>21.0</td></tr><tr><td>0</td><td>1</td><td><math>2^{18}/\phi</math></td><td>16.4</td><td>13.1</td><td>10.5</td></tr><tr><td>1</td><td>0</td><td><math>2^{17}/\phi</math></td><td>8.2</td><td>6.6</td><td>5.2</td></tr><tr><td>1</td><td>1</td><td><math>2^{16}/\phi</math></td><td>4.1</td><td>3.3</td><td>2.6</td></tr></tbody></table> <p><math>\phi</math> : System clock</p>	WT		Count clock	Oscillation settling time (ms)					16 MHz	20 MHz	25 MHz	0	0	$2^{19}/\phi$	32.8	26.2	21.0	0	1	$2^{18}/\phi$	16.4	13.1	10.5	1	0	$2^{17}/\phi$	8.2	6.6	5.2	1	1	$2^{16}/\phi$	4.1	3.3	2.6
WT		Count clock	Oscillation settling time (ms)																																		
			16 MHz	20 MHz	25 MHz																																
0	0	$2^{19}/\phi$	32.8	26.2	21.0																																
0	1	$2^{18}/\phi$	16.4	13.1	10.5																																
1	0	$2^{17}/\phi$	8.2	6.6	5.2																																
1	1	$2^{16}/\phi$	4.1	3.3	2.6																																
1, 0	SMD	Standby Mode Specifies a standby mode. The HALT instruction places the system in a standby mode selected according to the setting of SMD. <table border="1"><thead><tr><th colspan="2">SMD</th><th>Standby mode</th></tr></thead><tbody><tr><td>0</td><td>0</td><td>HALT mode</td></tr><tr><td>0</td><td>1</td><td>This setting is inhibited.</td></tr><tr><td>1</td><td>0</td><td>IDLE mode</td></tr><tr><td>1</td><td>1</td><td>STOP mode</td></tr></tbody></table>	SMD		Standby mode	0	0	HALT mode	0	1	This setting is inhibited.	1	0	IDLE mode	1	1	STOP mode																				
SMD		Standby mode																																			
0	0	HALT mode																																			
0	1	This setting is inhibited.																																			
1	0	IDLE mode																																			
1	1	STOP mode																																			

## 14.4 HALT MODE

### 14.4.1 Placing the System in the HALT Mode and Operation State

In the HALT mode, the clock generator (oscillator and PLL synthesizer) continues operation, but the CPU clock is stopped. Clock supply to other built-in peripheral functions continues to allow them to keep running. The total power dissipation of the system can be reduced by placing the CPU in the HALT mode when it is not running.

Executing the HALT instruction with the SMD bits of the standby control register (STBC) reset to 00 starts the halt acknowledge cycle, then shifting the system to the HALT mode.

In the HALT mode, program execution stops, but the contents of all registers are retained. In addition, the on-chip peripheral functions independent of CPU instruction execution continues operation.

If the HALT instruction is executed to shift the system to the HALT mode during a DMA block transfer, an actual shift to the HALT mode occurs only after the block transfer ends. If the HALT instruction is executed to shift the system to the HALT mode during a DMA single or single-step transfer, the DMAC opens the bus to the CPU at the end of each bus cycle. At this point, a halt acknowledge cycle begins, thus shifting the system to the HALT mode.

Table 14-2 lists the state in which each piece of the hardware is during HALT mode.

**Table 14-2. Operations during the HALT Mode**

Function	Operation state	
Clock generator	Operating	
Internal system clock	Operating	
CPU	Stop	
I/O line	Retained	
Peripheral function	Operating	
Internal data	All internal data such as CPU register contents retains the state in which it is before the system enters the HALT mode.	
A0-A23, $\overline{UBE}$	PC output	High impedance when $\overline{HLDAK} = 0$
D0-D15	High impedance	
$\overline{CS0}$ - $\overline{CS3}$	1	High impedance when $\overline{HLDAK} = 0$
$\overline{IORD}$ , $\overline{IOWR}$		
$\overline{WE/LMWR}$ , $\overline{UMWR}$		
$\overline{REFRQ}$ , $\overline{RAS}$ , $\overline{LCAS}$ , $\overline{UCAS}$		
	<b>1Note</b>	
$\overline{HLDRQ}$	Operating	
CLKOUT	Clock output (unless disabled)	

**Note** Except during CBR refresh

#### 14.4.2 Releasing the System from the HALT Mode

The system is released from the HALT mode by a nonmaskable interrupt request, unmasked maskable interrupt request, or a  $\overline{\text{RESET}}$  pin input.

##### (1) Release by a nonmaskable interrupt request ( $\overline{\text{NMI}}$ )

When the active level of the  $\overline{\text{NMI}}$  pin is detected, CPU clock supply is resumed. Because the  $\overline{\text{NMI}}$  input is a level input, it must be retained until a branch occurs to the  $\overline{\text{NMI}}$  handler (at this point, it is insured that the interrupt has been accepted).

##### (2) Release by a maskable interrupt request

When an unmasked maskable interrupt request occurs, CPU clock supply is resumed.

To select the HALT mode in the interrupt handling routine, it is necessary to enable an interrupt that can release the system from the HALT mode before executing the HALT instruction. More specific, it is necessary to enable an interrupt (PSW.ID = 0, EP = 0) and to specify the interrupt enable level (PSW.I3 to I0).

#### Operation after the system is released from the HALT mode by an interrupt request

Cause of release	EI state (PSW.ID = 0)	DI state (PSW.ID = 1)
NMI request	Branch to the handler address	
Maskable interrupt request	Branch to the handler address	Not released

##### (3) Release by the $\overline{\text{RESET}}$ pin input

Same as the ordinary reset operation



## 14.5 IDLE MODE

### 14.5.1 Placing the System in the IDLE Mode and Operation State

In the IDLE mode, the clock generator (oscillator and PLL synthesizer) continues operation, but internal system clock pulse supply stops, bringing the entire system to a stop.

When the system is released from the IDLE mode, it is unnecessary to secure oscillation settling time for the oscillator or lock-up time for the PLL, and therefore, the system can resume the ordinary operation quickly.

When the HALT instruction is executed with the SMD bits of the standby control register (STBC) set to 10, the halt acknowledge cycle begins. After this cycle begins, the CBR self-refresh cycle also begins, thus shifting the system to the IDLE mode.

When the system enters the IDLE mode, all programs are stopped, and the contents of all registers are retained. The built-in peripheral functions also stop operating.

A bus hold request is accepted during the IDLE mode. When a bus hold request is accepted, the bus is placed in a high-impedance state. When the hold request is canceled, the CBR self-refresh cycle restarts. Note that a bus hold request is not accepted during the halt acknowledge cycle and the first six clock cycles of the CBR self-refresh cycle after the system is put in the IDLE mode.

If the HALT instruction is executed to shift the system to the IDLE mode during a DMA block transfer, an actual shift to the IDLE mode occurs only after the block transfer ends. If the HALT instruction is executed to shift the system to the IDLE mode during a DMA single or single-step transfer, the DMAC opens the bus to the CPU at the end of each bus cycle. At this point, a halt acknowledge cycle is started, thus shifting the system to the IDLE mode.

- ★ If the bus hold function is used when the DRAM controller is used in IDLE mode, do not input  $\overline{\text{NMI}}$  while  $\overline{\text{CS0/REFRQ}}$  output is inactive or in the high-impedance state, so as not to suspend restart of the self-refresh cycle after bus hold.

Table 14-3 lists the state in which each piece of the hardware is during IDLE mode.

**Table 14-3. Operations during the IDLE Mode**

Function	Operation state	
Clock generator	Operating	
Internal system clock	Stop	
CPU	Stop	
I/O line	Retained	
Peripheral function	Stop	
Internal data	All internal data such as CPU register contents retains the state in which it is before the system enters the IDLE mode.	
A0-A23, $\overline{UBE}$	PC output	High impedance when $\overline{HLDAK} = 0$
D0-D15	High impedance	
$\overline{CS0}$ - $\overline{CS3}$	1	High impedance when $\overline{HLDAK} = 0$
$\overline{IORD}$ , $\overline{IOWR}$		
$\overline{WE/LMWR}$ , $\overline{UMWR}$		
$\overline{REFRQ}$ , $\overline{RAS}$ , $\overline{LCAS}$ , $\overline{UCAS}$	CBR self-refresh	
$\overline{HLDRQ}$	Operating	
CLKOUT	Clock output (unless disabled)	

### 14.5.2 Releasing the System from the IDLE Mode

The system is released from the IDLE mode by a nonmaskable interrupt request or a  $\overline{\text{RESET}}$  pin input.

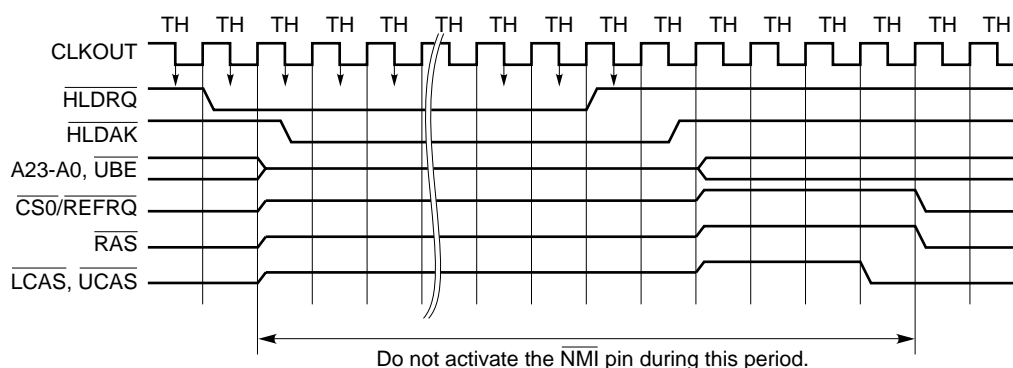
#### (1) Release by a nonmaskable interrupt request ( $\overline{\text{NMI}}$ )

When the active level of the  $\overline{\text{NMI}}$  pin is detected, CPU clock supply is resumed. Because the  $\overline{\text{NMI}}$  input is a level input, it must be retained until a branch occurs to the NMI handler (at this point, it is insured that the interrupt has been accepted).

Interrupt handling started when the system is released from the IDLE mode by an  $\overline{\text{NMI}}$  input is treated in the same way as the ordinary NMI processing performed at an emergency (because the address of the NMI handler is unique). If it is necessary to identify the two types of processing by program, a software status flag should be prepared and set before the execution of the HALT instruction. The NMI interrupt handler can check this flag to distinguish itself from the ordinary NMI processing.

For the system to enter the IDLE mode during execution of the NMI handler, it is necessary to reset PSW.NP to 0 before the HALT instruction is executed so that an NMI request becomes acceptable.

**Figure 14-1. Bus Hold in IDLE Mode and Restart of Self-Refresh**



**Remarks 1.** A broken line indicates a high impedance.

**2.** An arrow indicates a sampling timing.

**Caution** When the DRAM controller is used and the bus hold function is used in IDLE mode, restart of self-refresh cycle must not be suspended after the bus is held. Do not input  $\overline{\text{NMI}}$  while the CS0/REFRQ output is at a high impedance or inactive (high level).

#### (2) Release by the $\overline{\text{RESET}}$ pin input

Same as the ordinary reset operation

## 14.6 STOP MODE

### 14.6.1 Placing the System in the STOP Mode and Operation State

In the STOP mode, the clock generator (oscillator and PLL synthesizer) is stopped to bring the entire system to a stop. This mode can generate an ultra-low power dissipation condition; only leak current occurs.

When the HALT instruction is executed with the SMD bits of the standby control register (STBC) set to 11, the halt acknowledge cycle is started. After this cycle begins, the CBR self-refresh cycle also begins, thus shifting the system to the STOP mode.

In the PLL mode (TCLR pin = 0 during a reset) and the resonator-based mode (CESEL of the clock control register (CGC) = 0), it is necessary to secure oscillation settling time after the system is released from the STOP mode.

When the system enters the STOP mode, all programs are stopped, and the contents of all registers are retained. The built-in peripheral functions also stop operating.

A bus hold request is not accepted during the STOP mode. A bus hold request issued during the halt acknowledge cycle before the system shifts to the STOP mode is ignored.

If the HALT instruction is executed to shift the system to the STOP mode during a DMA block transfer, an actual shift to the STOP mode occurs only after the block transfer ends. If the HALT instruction is executed to shift the system to the STOP mode during a DMA single or single-step transfer, the DMAC opens the bus to the CPU at the end of each bus cycle. At this point, a halt acknowledge cycle is started, thus shifting the system to the STOP mode.

Table 14-4 lists the state in which each piece of the hardware is during the STOP mode.

**Table 14-4. Operations during the STOP Mode**

Function	Operation state
Clock generator	Stop
Internal system clock	Stop
CPU	Stop
I/O line	Retained
Peripheral function	Stop
Internal data	All internal data such as CPU register contents retains the state in which it is before the system enters the STOP mode.
A0-A23, $\overline{UBE}$	PC output
D0-D15	High impedance
$\overline{CS0}$ - $\overline{CS3}$	1
$\overline{IORD}$ , $\overline{IOWR}$	
$\overline{WE/LMWR}$ , $\overline{UMWR}$	
$\overline{REFRQ}$ , $\overline{RAS}$ , $\overline{LCAS}$ , $\overline{UCAS}$	CBR self-refresh
$\overline{HLDRQ}$	Stop
CLKOUT	1

### 14.6.2 Releasing the System from the STOP Mode

The system is released from the STOP mode by a nonmaskable interrupt request or a  $\overline{\text{RESET}}$  pin input.

To release the system from the STOP mode when the oscillator is being used (during the PLL mode (TCLR pin = 0 during a reset) in the resonator-based mode (CESEL of the CGC register = 0), it is necessary to secure oscillation settling time for the oscillator.

#### (1) Release by a nonmaskable interrupt request ( $\overline{\text{NMI}}$ input)

When the active level of the  $\overline{\text{NMI}}$  pin is detected, clock generation is resumed. After the oscillation settling time specified in the WT bit of the standby control register (STBC) elapses, internal system clock supply begins. Because the  $\overline{\text{NMI}}$  input is a level input, it must be retained until the system is released from the STOP mode and a branch occurs to the NMI handler (at this point, it is insured that the interrupt has been accepted).

Interrupt handling started when the system is released from the STOP mode by an  $\overline{\text{NMI}}$  input is treated in the same way as the ordinary NMI processing performed at an emergency (because the address of the NMI handler is unique). If it is necessary to identify the two types of processing by program, a software status flag should be prepared and set before the execution of the HALT instruction. The NMI interrupt handler can check this flag to distinguish itself from the ordinary NMI processing.

For the system to enter the STOP mode during execution of the NMI handler, it is necessary to reset PSW.NP to 0 before the HALT instruction is executed so that an NMI request becomes acceptable.

#### (2) Release by the $\overline{\text{RESET}}$ pin input

Same as the ordinary reset operation

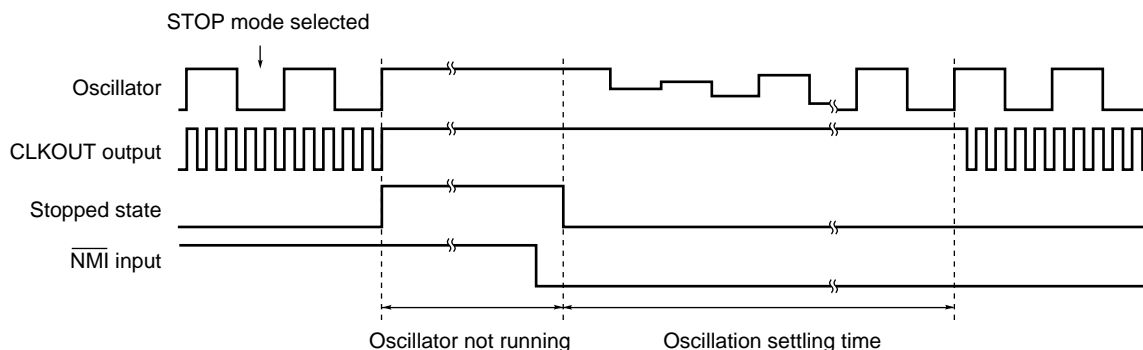
## 14.7 SECURING OSCILLATION SETTTLING TIME

There are two methods to secure oscillation settling time for the oscillator after the system is released from the STOP mode.

#### (1) Oscillation settling time secured using a timer ( $\overline{\text{NMI}}$ input)

Supplying an effective edge to the  $\overline{\text{NMI}}$  pin releases the system from the STOP mode. The oscillation settling time timer is used to determine the time required for the oscillator to become stable.

After the specified time elapses, the output of the system clock begins, causing a branch to the NMI handler address.

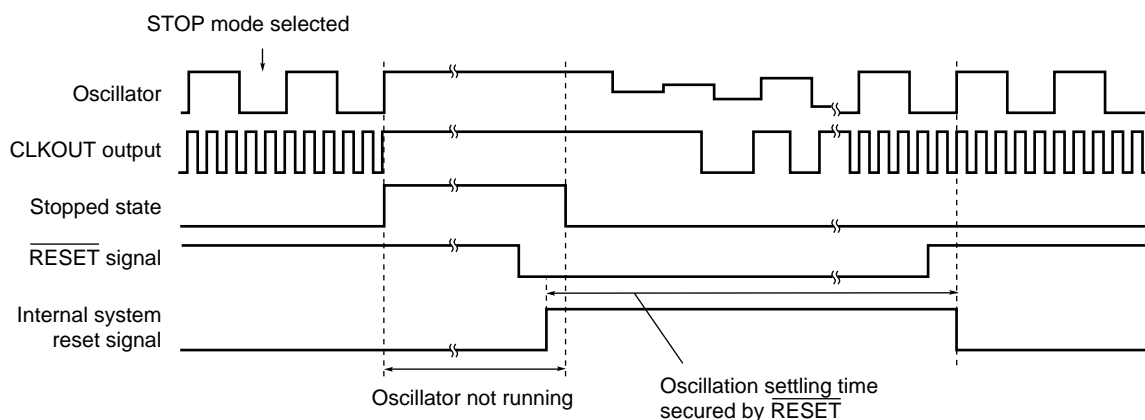


**(2) Oscillation settling time secured using a signal level width ( $\overline{\text{RESET}}$  pin input)**

Supplying a falling edge to the  $\overline{\text{RESET}}$  pin releases the system from the STOP mode.

The width of a low level supplied to the pin is used to provide the time required for the clock output of the oscillator to become stable.

After a rising edge occurs at the  $\overline{\text{RESET}}$  pin, internal system clock supply begins, causing a branch to the handler address related to a system reset.

**Oscillation settling time timer**

The oscillation settling time timer is used to secure oscillation settling time for the oscillator when the system is released from the STOP mode. This timer is used also as a watchdog timer. It is cleared to 00H when the system is put in the STOP mode.

The WT bit of the STBC register is used to select a count clock for the oscillation settling time timer as listed below:

**Table 14-5. Count Time Examples**

WT		Count clock	Time		
			$\phi = 16 \text{ MHz}$	$\phi = 20 \text{ MHz}$	$\phi = 25 \text{ MHz}$
0	0	$2^{19}/\phi$	32.8 ms	26.2 ms	21.0 ms
0	1	$2^{18}/\phi$	16.4 ms	13.1 ms	10.5 ms
1	0	$2^{17}/\phi$	8.2 ms	6.6 ms	5.2 ms
1	1	$2^{16}/\phi$	4.1 ms	3.3 ms	2.6 ms

$\phi$  : Internal system clock frequency

## CHAPTER 15 RESET FUNCTIONS

Inputting a low level to the  $\overline{\text{RESET}}$  pin triggers a system reset, thus initializing the on-chip hardware.

When the  $\overline{\text{RESET}}$  pin is driven from a low level to a high, the CPU starts program execution. The registers should be initialized in a program as required.

### 15.1 FEATURES

- The reset pin is provided with a noise suppressor circuit based on an analog delay (60 to 300 ns).

### 15.2 PIN FUNCTIONS

Table 15-1 lists the state of the output from each pin during a system reset. The output state is retained during the entire reset period.

After the  $\overline{\text{RESET}}$  pin is kept at a low level for 30 clock cycles, if the  $\overline{\text{HLDRQ}}$  signal is inactive, a memory read cycle is started to fetch an instruction.

Even during a reset period (when the  $\overline{\text{RESET}}$  pin is kept at a low level), activating the  $\overline{\text{HLDRQ}}$  signal can place the bus on hold. The state of each pin with the bus put on hold during a reset is basically the same as that with the bus put on hold during a non-reset period. (See **Section 2.2.**)

The  $\overline{\text{HLDRQ}}$  signal should be kept inactive during a power-on reset.

It is necessary to provide a pull-up or pull-down resistor to the pins that become high impedance during a reset. If no pull-up or pull-down resistor is provided to these pins, memory may be damaged when the pins are driven to high impedance.

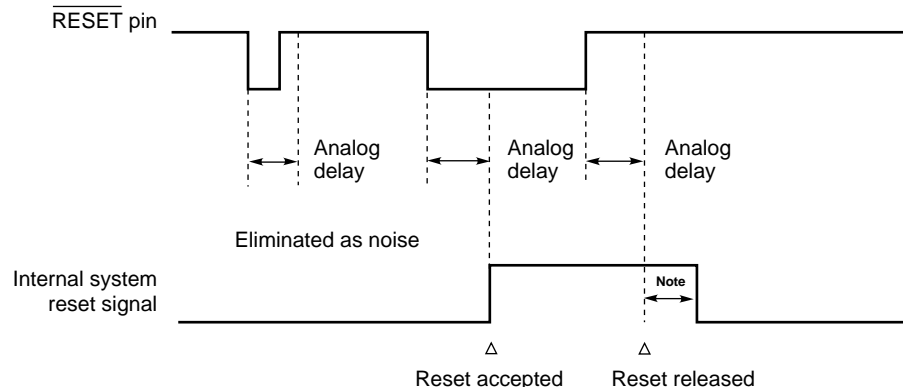
The CLKOUT pin supplies clock pulses even during a reset.

Table 15-1. Output State of Each Pin during a Reset

Pin	Operation state	Pin	Operation state
A0-A23	Not defined	$\overline{\text{HLDAK}}$	High level
D0-D15	High impedance	$\overline{\text{MRD}}$	
P00/TCLR		$\overline{\text{LMWR/WE}}$	
P01/DREQ0		$\overline{\text{UMWR}}$	
P02/ $\overline{\text{DACK0}}$		$\overline{\text{IORD}}$	
P03/DREQ1		$\overline{\text{IOWR}}$	
P04/ $\overline{\text{DACK1}}$		$\overline{\text{CS1-CS3}}$	
P05/SI		$\overline{\text{RAS}}$	
P06/SO		$\overline{\text{LCAS}}$	
P07/ $\overline{\text{SCLK}}$		$\overline{\text{UCAS}}$	
P08/TXD/ $\overline{\text{UBE}}$		$\overline{\text{CS0/REFRQ}}$	
P09/RXD/ $\overline{\text{TC}}$		BLOCK/WDTOUT	Low level



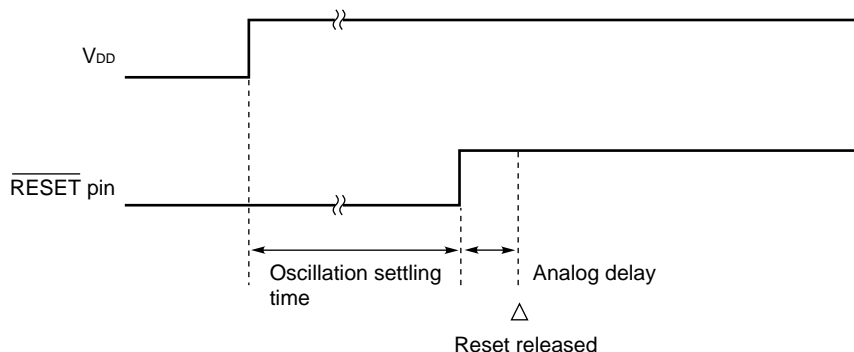
### (1) Accepting a reset signal



**Note** The internal system reset signal remains active for at least four system clock cycles, after a reset triggered by the **RESET** pin ends.

### (2) Power-on reset

When a power-on reset is used, it is necessary to provide an oscillation settling time of at least 10 ms between the power-on time and acceptance of a reset because of a low level at the **RESET** pin.



## 15.3 INITIALIZATION

Table 15-2 lists the values to which each register is initialized at a reset.

The registers should be initialized in a program as required. The following register especially should be set as required, because it is related to system setting.

- Clock control register (CGC): X1 and X2 pin function, CLKOUT pin operation, etc.

**Table 15-2. Initial Value in Each Register after a Reset (1/3)**

Register		Initial value after a reset
Program counter	PC	FFFFFFF0H
General-purpose register	r0	00000000H
	r1-r31	Not defined
System register	Register for saving the state upon the occurrence of an exception or interrupt (EIPC)	Not defined
	(EIPSW)	Not defined
	Register for saving the state upon the occurrence of an NMI or double exception (FEPC)	Not defined
	(FEPSW)	Not defined
	Exception source register (FECC)	0000H
	(EICC)	FFF0H
	Program status word (PSW)	00008000H
	Processor ID register (PIR)	0000810xH
	Task control word (TKCW)	000000E0H
	Cache control word (CHCW)	00000000H
	Address trap register (ADTRE)	Not defined
Port	Port register 0 (P0)	Not defined
	Port mode register 0 (PM0)	03FFH
	Port mode control register 0 (PMC0)	0000H
Memory management function	Bus cycle type control register (BCTC)	01H
	Programmable wait control register 0 (PWC0)	77H
	Programmable wait control register 1 (PWC1)	77H
	Programmable wait control register 2 (PWC2)	77H
	DRAM configuration register (DRC)	81H
	Refresh control register (RFC)	80H
	Page-ROM configuration register (PRC)	80H

**Table 15-2. Initial Value in Each Register after a Reset (2/3)**

Register		Initial value after a reset
DMA function	DMA source address register 0H (DSA0H)	Not defined
	DMA source address register 0L (DSA0L)	Not defined
	DMA destination address register 0H (DDA0H)	Not defined
	DMA destination address register 0L (DDA0L)	Not defined
	DMA source address register 1H (DSA1H)	Not defined
	DMA source address register 1L (DSA1L)	Not defined
	DMA destination address register 1H (DDA1H)	Not defined
	DMA destination address register 1L (DDA1L)	Not defined
	DMA byte count register 0 (DBC0)	Not defined
	DMA byte count register 1 (DBC1)	Not defined
	DMA channel control register 0 (DCHC0)	0000H
	DMA channel control register 1 (DCHC1)	0000H
Real-time pulse unit	Timer unit mode register 0 (TUM0)	0A00H
	Timer control register 0 (TMC0)	00H
	Timer control register 1 (TMC1)	00H
	Timer output control register 0 (TOC0)	03H
	Timer time-out status register (TOVS)	00H
	Timer register 0 (TM0)	00H
	Timer register 1 (TM1)	00H
	Capture/compare registers 00 to 03 (CC00 to CC03)	Not defined
	Compare register 1 (CM1)	Not defined
Serial interface	Asynchronous serial interface mode register (ASIM)	00H
	Asynchronous serial interface status register (ASIS)	00H
	Reception buffer (RXB, RXBL)	Not defined
	Transmission shift register (TXS, TXSL)	Not defined
	Synchronous serial interface mode register (CSIM)	00H
	Serial I/O shift register (SIO)	Not defined
	Baud rate generator register (BRG)	Not defined
	Baud rate generator prescaler mode register (BPRM)	00H

Table 15-2. Initial Value in Each Register after a Reset (3/3)

Register		Initial value after a reset
Interrupt/exception handling function	Interrupt group priority register (IGP)	E4H
	Interrupt clear register (ICR)	0000H
	Interrupt request register (IRR)	0000H
	Interrupt request mask register (IMR)	FFFFH
	ICU mode register (IMOD)	AAAAH
Watchdog timer function	WDT mode register (WDTM)	00H
Clock generator	Clock control register (CGC)	03H
Standby function	Standby control register (STBC)	00H

**Caution** The "not defined" state mentioned in the above table occurs at a power-on reset or when data collapses due to coincidence of the falling edge of  $\overline{\text{RESET}}$  with the data write timing. The previous state is retained on a falling edge of  $\overline{\text{RESET}}$  if a data write is not taking place.

## APPENDIX A REGISTER INDEX

### A.1 REGISTER NAMES

#### [A]

Asynchronous serial interface mode register (ASIM) .....	158
Asynchronous serial interface status register (ASIS) .....	161

#### [B]

Baud rate generator prescaler mode register (BPRM) .....	181
Baud rate generator register (BRG) .....	181
Bus cycle type control register (BCTC) .....	114

#### [C]

Capture/compare register 00 (CC00) .....	187
Capture/compare register 01 (CC01) .....	187
Capture/compare register 02 (CC02) .....	187
Capture/compare register 03 (CC03) .....	187
Clock control register (CGC) .....	224
Compare register 1 (CM1) .....	188

#### [D]

DMA byte count register 0 (DBC0) .....	143
DMA byte count register 1 (DBC1) .....	143
DMA channel control register 0 (DCHC0) .....	144
DMA channel control register 1 (DCHC1) .....	144
DMA destination address register 0H (DDA0H) .....	142
DMA destination address register 0L (DDA0L) .....	142
DMA destination address register 1H (DDA1H) .....	142
DMA destination address register 1L (DDA1L) .....	142
DMA source address register 0H (DSA0H) .....	141
DMA source address register 0L (DSA0L) .....	141
DMA source address register 1H (DSA1H) .....	141
DMA source address register 1L (DSA1L) .....	141
DRAM configuration register (DRC) .....	122

#### [I]

ICU mode register (IMOD) .....	68
Interrupt clear register (ICR) .....	66
Interrupt group priority register (IGP) .....	64
Interrupt request mask register (IMR) .....	67
Interrupt request register (IRR) .....	66

**[P]**

Page-ROM configuration register (PRC) .....	137
Port mode control register 0 (PMC0) .....	219
Port mode register 0 (PM0) .....	219
Port register 0 (P0) .....	218
Programmable wait control register 0 (PWC0) .....	115
Programmable wait control register 1 (PWC1) .....	116
Programmable wait control register 2 (PWC2) .....	117

**[R]**

Reception buffer (RXB, RXBL) .....	162
Refresh control register (RFC) .....	129

**[S]**

Serial I/O shift register (SIO) .....	172
Standby control register (STBC) .....	228
Synchronous serial interface mode register (CSIM) .....	171

**[T]**

Timer 0 (TM0) .....	186
Timer 1 (TM1) .....	188
Timer control register 0 (TMC0) .....	191
Timer control register 1 (TMC1) .....	192
Timer output control register 0 (TOC0) .....	193
Timer overflow status register (TOVS) .....	194
Timer unit mode register 0 (TUM0) .....	189
Transmission shift register (TXS, TXSL) .....	163

**[W]**

WDT mode register (WDTM) .....	215
--------------------------------	-----

## A.2 REGISTER SYMBOLS

### [A]

ASIM (asynchronous serial interface mode register) .....	158
ASIS (asynchronous serial interface status register) .....	161

### [B]

BCTC (bus cycle type control register) .....	114
BPRM (baud rate generator prescaler mode register) .....	181
BRG (baud rate generator register) .....	181

### [C]

CC00 (capture/compare register 00) .....	187
CC01 (capture/compare register 01) .....	187
CC02 (capture/compare register 02) .....	187
CC03 (capture/compare register 03) .....	187
CGC (clock control register) .....	224
CM1 (compare register 1) .....	188
CSIM (synchronous serial interface mode register) .....	171

### [I]

ICR (interrupt clear register) .....	66
IGP (interrupt group priority register) .....	64
IMR (interrupt request mask register) .....	67
IRR (interrupt request register) .....	66

### [P]

P0 (port register 0) .....	218
PM0 (port mode register 0) .....	219
PMC0 (port mode control register 0) .....	219
PWC0 (programmable wait control register 0) .....	115
PWC1 (programmable wait control register 1) .....	116
PWC2 (programmable wait control register 2) .....	117

### [R]

RFC (refresh control register) .....	129
RXB, RXBL (reception buffer) .....	162

### [S]

SIO (serial I/O shift register) .....	172
STBC (standby control register) .....	228

### [T]

TM0 (timer 0) .....	186
---------------------	-----

TM1 (timer 1) ..... 188

TMC0 (timer control register 0) ..... 191

TMC1 (timer control register 1) ..... 192

TOC0 (timer output control register 0) ..... 193

TOVS (timer overflow status register) ..... 194

TUM0 (timer unit mode register 0) ..... 189

TXS, TXSL (transmission shift register) ..... 163



## APPENDIX B GENERAL INDEX

### [A]

A0-A23 .....	35
ADC .....	122
address bus .....	35
address multiplexing function .....	121
address space .....	45
address trap .....	69
address trap register .....	51
ADTRE .....	51
ALV01, ALV00 .....	193
ASIM .....	158
ASIS .....	161
assembler-reserved register .....	50
asynchronous serial interface .....	155
asynchronous serial interface mode	
register .....	156, 158
asynchronous serial interface status	
register .....	156, 161

### [B]

back-up mode .....	223
BAU .....	29
baud rate generator .....	178
baud rate generator prescaler mode register .....	181
baud rate generator register .....	181
BC .....	143
BCTC .....	114
BIU .....	28
BLOCK .....	36
block transfer mode .....	147
BPR .....	181
BPRM .....	181
BRCE .....	181
BRG .....	178, 181
BU .....	224
built-in peripheral I/O register .....	52
built-in peripheral I/O registers .....	52
bus arbitration unit .....	29
bus control function .....	75
bus control signal .....	35

bus cycle during fly-by transfer .....	100
bus cycle during which the wait function is	
effective .....	111
bus cycle type control register .....	114
bus hold .....	106
bus interface unit .....	28
bus lock .....	108
bus priority .....	80
BWOS .....	215

### [C]

cache control word .....	51
cancellation with $\overline{\text{HLDRQ}}$ input (in IDLE mode) ...	133
cancellation with $\overline{\text{NMI}}$ input .....	132
capture/compare registers 00 to 03 .....	187
capturing (timer 0) .....	198
caution (timer/counter function) .....	211
CBR refresh cycle .....	98, 131
CC00-CC03 .....	187
CE0 .....	191
CE1 .....	192
CES0 .....	190
CESEL .....	224
CG .....	29
CGC .....	224
CHCW .....	51
CL .....	160
CL/OV .....	215
CLKOUT .....	37
clock control register .....	224
clock generation function .....	221
clock generator .....	29
clock output control .....	223
CLR14-CLR0 .....	66
CLS .....	171
CM1 .....	188
CMS00-CMS03 .....	190
COE .....	224
compare register 1 .....	188
comparison (timer 0) .....	199

comparison (timer 1) .....	202	DMA control register .....	140
connecting DRAM .....	119	DMA control signal .....	39
control register .....	64, 114, 189, 219	DMA controller .....	28
control signal timing .....	106	DMA destination address registers 0 and 1 .....	141
count clock selection (timer 0) .....	195	DMA destination address registers 0H and 1H ....	142
counting (timer 0) .....	195	DMA destination address registers 0L and 1L ....	142
counting (timer 1) .....	201	DMA function (DMA controller) .....	139
CPU bus state .....	75	DMA source address registers 0 and 1 .....	140
CPU function .....	45	DMA source address registers 0H and 1H .....	141
CPU register set .....	49	DMA source address registers 0L and 1L .....	141
CRXE .....	171	DMA transfer end interrupt .....	150
$\overline{\text{CS0}}\text{--}\overline{\text{CS3}}$ .....	37	DMA transfer end output .....	151
CSI .....	28, 169	DMA transfer request .....	150
CSI control register .....	171	DMA transfer type and transfer object .....	148
CSIM .....	169, 171	DMAC .....	28
CT0 .....	114	DMAC bus state .....	77
CT1 .....	114	double exception .....	69
CT2 .....	114	DRAM access .....	93
CT3 .....	114	DRAM configuration register .....	122
CTXE .....	171	DRAM control signal .....	38
CWS .....	122	DRAM controller .....	28, 119
		DRAM read cycle .....	93
<b>[D]</b>		DRAM read/write cycle .....	123
D0-D15 .....	35	DRAM write cycle .....	95
DA15-DA0 .....	142	DRAMC .....	28, 119
DA23-DA16 .....	142	DRC .....	122
$\overline{\text{DACK0}}$ , $\overline{\text{DACK1}}$ .....	39	DREQ0, DREQ1 .....	39
DAD .....	144	DS .....	145
data bus .....	35	DSA0, DSA1 .....	140
data flow during DMA transfer .....	152	DSA0H, DSA1H .....	141
DAW .....	122	DSA0L, DSA1L .....	141
DBC0, DBC1 .....	143	during fly-by DMA transfer .....	111
DBN .....	142	during two-cycle DMA transfer .....	111
DBT .....	142	DWS0 .....	117
DCHC0, DCHC1 .....	144	DWS1 .....	117
DDA0, DDA1 .....	141		
DDA0H, DDA1H .....	142	<b>[E]</b>	
DDA0L, DDA1L .....	142	EBS .....	160
differences between the memory and I/O maps ....	48	ECLR0 .....	189
direct mode .....	222, 226	ECR .....	51
division by zero .....	69	edge trigger .....	68
DMA byte count registers 0 and 1 .....	143	EIPC .....	51
DMA channel control registers 0 and 1 .....	144	EIPSW .....	51
DMA channel priority .....	150	EN .....	145, 215

ENTO01, ENTO00 .....	193	IMR .....	67
ETI .....	191	IMS00-IMS03 .....	190
exception handling (software exception and exception trap) .....	69	initialization .....	239
exception handling cycle .....	104	input clock selection .....	201, 222
exception source register .....	51	internal block diagram .....	27
external I/O access .....	86	internal I/O access .....	88
external I/O read cycle .....	86	internal I/O read cycle .....	88
external I/O write cycle .....	87	internal I/O write cycle .....	89
<b>[F]</b>		internal unit .....	28
FE .....	161	interrupt clear register .....	66
FEPC .....	51	interrupt control signal .....	38
FEPSW .....	51	interrupt controller .....	28
floating-point degraded precision .....	69	interrupt group priority register .....	64
floating-point division by zero .....	69	interrupt request .....	164
floating-point overflow .....	69	interrupt request mask register .....	67
floating-point reserved operand .....	69	interrupt request register .....	66
floating-point underflow .....	69	interrupt signal generation control circuit .....	169
fly-by read cycle .....	100	interrupts .....	56
fly-by transfer .....	149	INTP00-INTP03 .....	38
fly-by write cycle .....	102	INTP10-INTP13 .....	38
forcible interruption .....	152	INTSER .....	164
functional block configuration .....	27	INTSR .....	164
<b>[G]</b>		INTST .....	164
general-purpose register .....	50	invalid floating-point operation .....	69
global pointer .....	50	invalid instruction code .....	69
<b>[H]</b>		IORD .....	36
halt acknowledge cycle .....	105	IOWR .....	36
HALT mode .....	225, 229	IRR .....	66
handler stack pointer .....	50	ITM00-03, ITM10-13 .....	68
HLDK .....	36	<b>[K]</b>	
HLDRQ .....	35	KEY .....	215, 224, 228
<b>[I]</b>		<b>[L]</b>	
I/O map .....	47	LCAS .....	39
ICR .....	66	level trigger .....	68
ICU mode register .....	68	link pointer .....	50
IDLE mode .....	225, 231	LMWR .....	36
IGP .....	64	lock-up time .....	223
image .....	47	<b>[M]</b>	
IMOD .....	68, 193	MA5-MA3 .....	137
		machine fault cycle .....	104
		maskable interrupt .....	60

maskable interrupt priority .....	63	pin state .....	34
maskable interrupt (block diagram) .....	60	PIR .....	51
memory access control function .....	119	placing the system in the HALT mode and operation .....	229
memory map .....	46	placing the system in the IDLE mode and operation .....	231
MOD .....	171	placing the system in the STOP mode and operation .....	234
MRD .....	36	PLL mode .....	222, 226
MS9-MS0 .....	219	PM0 .....	219
MSK14-MSK0 .....	67	PM9-PM0 .....	219
		PMC0 .....	219
<b>[N]</b>		port control signal .....	41
NMI .....	38	port function .....	217
nonmaskable interrupt .....	58	port mode control register 0 .....	219
		port mode register 0 .....	219
<b>[O]</b>		PR3-PR0 .....	64
off-page .....	93, 95	PRC .....	137
on-page .....	93, 95	priority .....	73
on-page/off-page decision .....	134	priority of floating-point exception .....	74
operand read .....	83	priority of interrupt and exception .....	73
operand write .....	84	PRM0 .....	191
operation of timer 0 .....	195	PRM1 .....	192
operation of timer 1 .....	201	processor ID register .....	51
operations during the HALT mode .....	229	program counter .....	50
operations during the IDLE mode .....	232	program register set .....	50
operations during the STOP mode .....	234	program status word .....	51
ordering information .....	24	programmable wait control register 0 .....	115
OST .....	189	programmable wait control register 1 .....	116
OVE .....	161	programmable wait control register 2 .....	117
overflow (timer 0) .....	196	programmable wait function .....	110
overflow (timer 1) .....	201	PRS0 .....	191
OVFn .....	194	PRS1 .....	192
		PS .....	159
<b>[P]</b>		PSW .....	51
P00-P09 .....	41	PWC0 .....	115
PAE .....	122	PWC1 .....	116
page-ROM access .....	92, 135	PWC2 .....	117
page-ROM configuration register .....	137		
page-ROM read cycle .....	92	<b>[R]</b>	
PE .....	161	r0-r31 .....	50
pin configuration .....	25	$\overline{\text{RAS}}$ .....	38
pin function .....	31, 237	RCC .....	129
pin function of port 0 .....	218	$\overline{\text{READY}}$ .....	35
pin functions .....	31		
pin I/O circuit .....	43		
pin I/O circuits and processing of unused pins .....	42		

real-time pulse control signal .....	40	<b>[S]</b>	
real-time pulse unit .....	28	SA15-SA0 .....	141
reception buffer .....	156, 162	SA23-SA16 .....	141
reception completion interrupt .....	164	SAD .....	144
reception control parity check .....	156	SBN .....	141
reception error interrupt .....	164	SBT .....	141
reception shift register .....	156	SCLK .....	40
refresh control register .....	129	SCLS .....	160
refresh function .....	129	securing oscillation settling time .....	235
REFRQ .....	38	selector .....	157
register for saving the current status upon the occurrence of an exception or interrupt .....	51	self-refresh cycle .....	99
register for saving the current status upon the occurrence of an NMI or double exception .....	51	self-refresh function .....	132
relationship between external access and byte enable signal .....	82	serial clock control circuit .....	169
relationship between external access and the data bus .....	82	serial clock counter .....	169
release by a maskable interrupt request .....	230	serial clock selector .....	169
release by a nonmaskable interrupt request (NMI) .....	230, 233, 235	serial control signal .....	40
release by the RESET pin input .....	230, 233, 235	serial I/O shift register .....	172
releasing the system from the HALT mode .....	230	serial interface .....	28
releasing the system from the IDLE mode .....	233	serial interface function .....	155
releasing the system from the STOP mode .....	235	shift register .....	169
REN .....	129	SI .....	40
REQ14-REQ0 .....	66	single transfer mode .....	146
request issued by software .....	150	single-step transfer mode .....	146
request received from built-in peripheral hardware .....	150	SIO .....	169, 172
request received from the DREQ pin .....	150	SIO <sub>n</sub> (n = 7-0) .....	172
RESET .....	37	SL .....	160
reset function .....	237	SMD .....	228
return from an exception or interrupt .....	72	SO .....	40
RFC .....	129	SOT .....	161, 171
RI .....	129	SRAM (ROM) access .....	90
ROM controller .....	28, 134	SRAM (ROM) read cycle .....	90
ROMC .....	28, 134	SRAM write cycle .....	91
RPU .....	28	stack pointer .....	50
RXB, RXBL .....	156, 162	standby control register .....	228
RXB <sub>n</sub> (n = 7-0) .....	162	standby function .....	225
RXD .....	40	standby mode .....	225
RXE .....	158	STBC .....	228
RXEB .....	162	STOP mode .....	225, 234
		string destination start address register .....	50
		string destination start bit offset .....	50
		string length register .....	50
		string source start bit offset .....	50
		string start address register .....	50
		synchronous serial interface .....	169

synchronous serial interface mode		TOPC01, TOPC00 .....	193
register .....	169, 171	TOVS .....	194
system control signal .....	37	transfer mode .....	146
system register set .....	51	transfer object .....	149
<b>[T]</b>		transmission completion interrupt .....	164
T0 state .....	77	transmission control parity bit addition .....	157
T1 and T1S states .....	75	transmission shift register .....	156, 163
T1F state .....	77	TTYD .....	145
T1R state .....	77	TUM0 .....	189
T1W state .....	77	two-cycle transfer .....	148
T2 and T2S states .....	75	TXD .....	40
T2F state .....	77	TXED .....	163
T2R state .....	77	TXS, TXSL .....	156, 163
T2W state .....	77	TXSn (n = 7-0) .....	163
T3 state .....	77	<b>[U]</b>	
task control word .....	51	UART .....	28, 155
TC .....	144	UART control register .....	158
$\overline{TC}$ .....	40	$\overline{UBE}$ .....	37
TCLR .....	40	$\overline{UCAS}$ .....	39
TDIR .....	145	UL .....	224
TES0 .....	189	$\overline{UMWR}$ .....	36
text pointer .....	50	<b>[W]</b>	
TH and THS states .....	75	WA .....	137
TH state .....	78	wait control during DMA transfer .....	111
TI .....	40	wait control for DRAM cycle .....	127
TI and TIS states .....	75	wait control function .....	109
TI state .....	77	wait control unit .....	28
timer 0 .....	186	wait control using the $\overline{READY}$ pin .....	110
timer 1 .....	188	watchdog timer .....	28
timer control register 0 .....	191	watchdog timer control signal .....	41
timer control register 1 .....	192	watchdog timer function .....	213
timer output control register 0 .....	193	WCU .....	28
timer overflow status register .....	194	WDT .....	28
timer unit mode register 0 .....	189	WDT mode register .....	215
timer/counter function (real-time pulse unit) .....	183	WDTI .....	216
TKCW .....	51	WDTM .....	215
TM .....	145	WDTOUT .....	41
TM0 .....	186	$\overline{WE}$ .....	39
TM1 .....	188	WS0 .....	115
TMC0 .....	191	WS1 .....	115
TMC1 .....	192	WS2 .....	116
TO00, TO01 .....	40	WS3 .....	116
TOC0 .....	193		

WT ..... 228

**[X]**

X1, X2 ..... 37

**[Z]**

zero register ..... 50

**[Others]**

100-pin plastic LQFP ..... 24

[MEMO]



## Facsimile Message

From:

Name

Company

Tel.

FAX

Address

Although NEC has taken all possible steps to ensure that the documentation supplied to our customers is complete, bug free and up-to-date, we readily accept that errors may occur. Despite all the care and precautions we've taken, you may encounter problems in the documentation. Please complete this form whenever you'd like to report errors or suggest improvements to us.

*Thank you for your kind support.*

**North America**

NEC Electronics Inc.  
Corporate Communications Dept.  
Fax: 1-800-729-9288  
1-408-588-6130

**Hong Kong, Philippines, Oceania**

NEC Electronics Hong Kong Ltd.  
Fax: +852-2886-9022/9044

**Asian Nations except Philippines**

NEC Electronics Singapore Pte. Ltd.  
Fax: +65-250-3583

**Europe**

NEC Electronics (Europe) GmbH  
Technical Documentation Dept.  
Fax: +49-211-6503-274

**Korea**

NEC Electronics Hong Kong Ltd.  
Seoul Branch  
Fax: 02-528-4411

**Japan**

NEC Semiconductor Technical Hotline  
Fax: 044-548-7900

**South America**

NEC do Brasil S.A.  
Fax: +55-11-6465-6829

**Taiwan**

NEC Electronics Taiwan Ltd.  
Fax: 02-2719-5951

I would like to report the following error/make the following suggestion:

Document title: \_\_\_\_\_

Document number: \_\_\_\_\_ Page number: \_\_\_\_\_

If possible, please fax the referenced page or drawing.

Document Rating	Excellent	Good	Acceptable	Poor
Clarity	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Technical Accuracy	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Organization	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>