

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**

## **μPD784026 Subseries**

**16/8-bit Single-chip Microcontrollers**

### **Hardware**

---

**μPD784020**

**μPD784021**

**μPD784025**

**μPD784026**

**μPD78P4026**

[MEMO]

**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.

QTOP is a trademark of NEC Corporation.

MS-DOS and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

IBM DOS, PC/AT and PC DOS are trademarks of IBM Corporation.

SPARCstation is a trademark of SPARC International, Inc.

SunOS is a trademark of Sun Microsystems, Inc.

HP9000 series 700 and HP-UX are trademarks of Hewlett-Packard Company.

NEWS and NEWS-OS are trademarks of Sony Corporation.

Ethernet is a trademark of Xerox Corporation.

TRON is an abbreviation of The Realtime Operating system Nucleus.

ITRON is an abbreviation of Industrial TRON.

The export of these products from Japan is regulated by the Japanese government. The export of some or all of these products may be prohibited without governmental license. To export or re-export some or all of these products from a country other than Japan may also be prohibited without a license from that country. Please call an NEC sales representative.

Licence not needed :  $\mu$ PD784020GC-3B9, 784021GC-3B9, 784021GK-BE9,  
78P4026KK-T

The customer must judge the need for licence :  $\mu$ PD784025GC-xxx-3B9 784026GC-xxx-3B9,  
 $\mu$ PD78P4026GC-xxx-3B9, 78P4026GC-3B9

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: 800-366-9782  
Fax: 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: 253-8311  
Fax: 250-3583

**NEC Electronics Taiwan Ltd.**

Taipei, Taiwan  
Tel: 02-719-2377  
Fax: 02-719-5951

**NEC do Brasil S.A.**

Sao Paulo-SP, Brasil  
Tel: 011-889-1680  
Fax: 011-889-1689

**MAJOR REVISION IN THIS EDITION**

(1/3)

Page	Description
p. 2	Change of <b>78K/IV SERIES PRODUCT DEVELOPMENT DIAGRAM</b> in <b>CHAPTER 1 GENERAL</b>
p. 36 p. 40 p. 69	<b>CHAPTER 3 CPU ARCHITECTURE</b> <ul style="list-style-type: none"> <li>• Correction of <b>Figure 3-1 <math>\mu</math>PD784020 Memory Map</b></li> <li>• Addition of caution on prohibiting external wait setting for internal ROM area</li> <li>• Addition of note on reading timer register</li> </ul>
p. 176  p. 187  p. 196  p. 203	<b>CHAPTER 8 TIMER/COUNTER 0</b> <ul style="list-style-type: none"> <li>• Addition of caution on reading timer register</li> <li>• <b>8.5 EXTERNAL EVENT COUNTER FUNCTION</b> Correction of timing of TM0 in <b>Figure 8-10 Timer/Counter 0 External Event Count Timing</b></li> <li>• <b>8.7.2 Toggle Output</b> Change from “the output level at the time it was stopped is retained” to “the inactive level (<math>\overline{ALVn}</math>: n = 0, 1) is output” when timer/counter 0 is stopped by timer control register 0</li> <li>• <b>8.7.3 PWM Output (3) Stopping PWM output</b> Change from “the output level at the time it was stopped is retained” to “the active level is output” when timer/counter 0 is stopped by timer control register 0</li> </ul>
p. 237  p. 247	<b>CHAPTER 9 TIMER/COUNTER 1</b> <ul style="list-style-type: none"> <li>• Addition of caution on reading timer register</li> <li>• <b>9.5 EXTERNAL EVENT COUNTER FUNCTION</b> Correction of timing of TM1 in <b>Figure 9-10 Timing of External Event Count by Timer/Counter 1</b></li> </ul>



Page	Description
p. 275	<b>CHAPTER 10 TIMER/COUNTER 2</b> <ul style="list-style-type: none"> <li><b>10.2 CONFIGURATION</b>            Addition of caution on reading timer register to <b>(1) Timer register 2 (TM2/TM2W)</b>            Deletion of “Also, the count value can be cleared by a match (CR21/CR21W)” from description in <b>(2) Compare register (CR20/CR20W)</b> </li> </ul>
p. 276	Addition of “Also, TM2/TM2W can be cleared after capture operation” to <b>(b) When specified as capture register in (3) Capture/compare register (CR21/CR21W)</b>
p. 286	<ul style="list-style-type: none"> <li><b>10.5 EXTERNAL EVENT COUNTER FUNCTION</b>            Correction of timing of TM2 in <b>Figure 10-11 Timer/Counter 2 External Event Count Timing</b> </li> </ul>
p. 299	<ul style="list-style-type: none"> <li><b>10.8.2 Toggle Output</b>            Change from “the output level at the time it was stopped is retained” to “the inactive level (<math>\overline{ALVn}</math>: n = 0, 1) is output” when timer/counter 2 is stopped by timer control register 1         </li> </ul>
p. 307	<ul style="list-style-type: none"> <li><b>10.8.3 PWM Output (3) Stopping PWM output</b>            Change from “the output level at the time it was stopped is retained” to “the active level is output” when timer/counter 2 is stopped by timer control register 1         </li> </ul>
p. 345	Addition of caution on reading timer register to <b>CHAPTER 11 TIMER 3</b>
p. 363	<b>CHAPTER 12 WATCHDOG TIMER FUNCTION</b> Change of “If the STOP mode or IDLE mode is entered as the result of an inadvertent program loop” to “If the STOP mode, <u>HALT mode</u> , or IDLE mode is entered as the result of the an inadvertent program loop” in <5> of <b>(2) in 12.4.1 General Cautions on Use of Watchdog Timer</b>
p. 378	Low-speed conversion ( $f_{CLK} = 12.5 \text{ MHz}$ ) in <b>CHAPTER 14 A/D CONVERTER</b> Change from $240/f_{CLK}$ ( $19 \mu\text{s}$ ) to $180/f_{CLK}$ ( $14.4 \mu\text{s}$ )
p. 408, 409, 413, 415 p. 418	<b>CHAPTER 16 ASYNCHRONOUS SERIAL INTERFACE/3-WIRE SERIAL I/O</b> <ul style="list-style-type: none"> <li>Addition of note on disabling reception completion interrupt in case of receive error and wait time calculation method</li> <li>Addition of caution on selecting MSB/LSB first</li> </ul>
p. 479	<b>CHAPTER 18 CLOCK OUTPUT FUNCTION</b> Unification from CLO to CLKOUT pin
p. 564	Addition of instructions to <b>20.9 WHEN INTERRUPT REQUESTS AND MACRO SERVICE ARE TEMPORARILY HELD PENDING</b> in <b>CHAPTER 20 INTERRUPT FUNCTIONS</b>

Page	Description
p. 586	<b>CHAPTER 21 LOCAL BUS INTERFACE FUNCTION</b> <ul style="list-style-type: none"> <li>• Addition of cautions on disabling external wait setting to internal ROM area in <b>21.2.1 Wait Function Control Registers</b> and <b>21.2.3 Access Waits</b></li> </ul>
p. 587	
p. 588, 598	
p. 604	
p. 605	
p. 609	<b>CHAPTER 22 STANDBY FUNCTION</b> <ul style="list-style-type: none"> <li>• Change and addition from “The watchdog timer must not be used to release the standby mode (STOP or IDLE mode).” to “The watchdog timer must not be used to release the standby mode (STOP/<u>HALT</u>/IDLE mode).”</li> </ul>
p. 612	
p. 615	
p. 618	
p. 624	
p. 627	
p. 631	
p. 633	
p. 695	
	<b>APPENDIX B DEVELOPMENT TOOLS</b> Change from EV-9500GK-80 to TKG080SDW (Tokyo Eletech Corp.)

The mark ★ shows major revised points.

## PREFACE

### Intended Readership

This manual is intended for user engineers who understand the functions of the  $\mu$ PD784026 subseries and wish to design application systems using this subseries.

### Purpose

The purpose of this manual is to give users an understanding of the various hardware functions of the  $\mu$ PD784026 subseries.

### Organization

The  $\mu$ PD784026 subseries manual is divided into two volumes – the hardware volume (this manual) and the instruction volume.

Hardware	Instruction
Pin functions	CPU functions
Internal block functions	Addressing
Interrupts	Instruction set
Other internal peripheral functions	

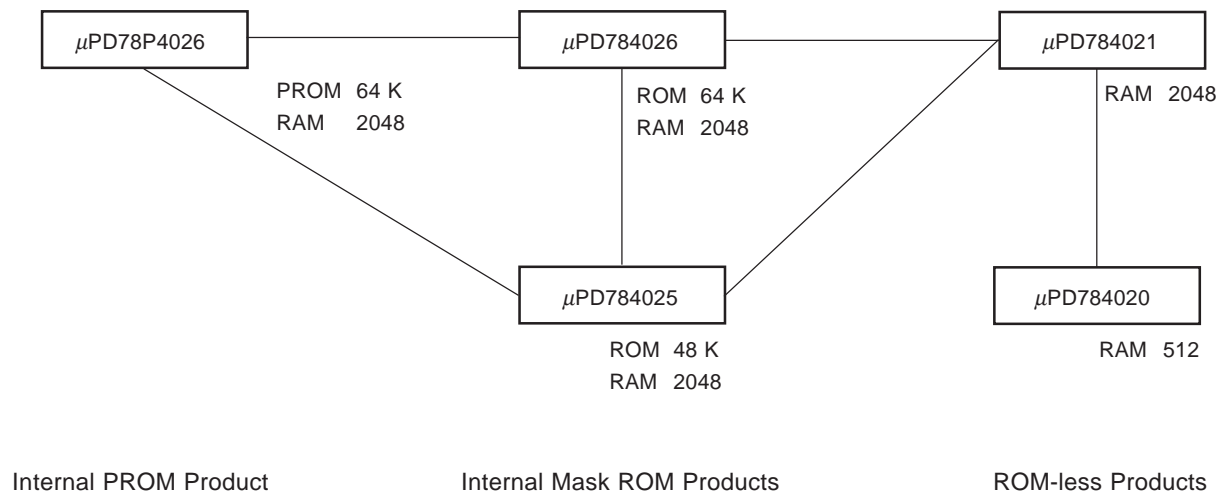
**Certain operating precautions apply to these products.**  
**These precautions are stated at the relevant points in the text of each chapter, and are also summarized at the end of each chapter. Be sure to read them.**

### How to Read This Manual

Readers are required a general knowledge of electrical and logic circuits and microcontrollers.

To readers using this manual as a  $\mu$ PD784020, 784021, 784025, 78P4026 manual:

→ This manual covers the functions of the  $\mu$ PD784020, 784021, 784025, 784026 and 78P4026. The relationship between these products is shown below.



- If there are no particular differences in terms of function:

The  $\mu$ PD784026 is treated as the representative model. Therefore, when using this manual as the  $\mu$ PD784020, 784021, 784025 or 78P4026 user's manual,  $\mu$ PD784026 should be read as  $\mu$ PD784020, 784021, 784025 or 78P4026, as appropriate.

Unless otherwise specified, the  $\mu$ PD784021 is treated as the representative ROM-less model. If you use this manual as the user's manual of the  $\mu$ PD784020, read  $\mu$ PD784021 as  $\mu$ PD784020.

- If there are functional differences:

A separate description is given, mentioning the product name.

- ◆ To find out about differences between the  $\mu$ PD784026 subseries and the  $\mu$ PD78234 subseries:

→ See **APPENDIX A DIFFERENCES BETWEEN  $\mu$ PD784026 SUBSERIES AND  $\mu$ PD78234 SUBSERIES**.

- ◆ To check the details of a register when the register name is known:

→ Use **APPENDIX D REGISTER INDEX**.

- ◆ If the device operates strangely after debugging:

→ Cautions are summarized at the end of each chapter, so refer to the cautions for the relevant function.

- ◆ For a general understanding of the functions

→ Read in accordance with the **TABLE OF CONTENTS**.

- ◆ For the details of the instruction functions:

→ Refer to the separate **78K/IV Series User's Manual-Instruction (U10905E)**.

- ◆ To find out about the electrical characteristics,

→ Refer to the individual data sheet.

- ◆ To find out about application examples of each function,

→ Refer to the application note separately available.

## Legend

Significance in data notation : High-order digit on left, low-order digit on right

Active-low notation :  $\overline{\times \times \times}$  (Line above pin or signal name)

\* : Explanation of item marked with \* in the text

**Caution** : Item to be especially noted

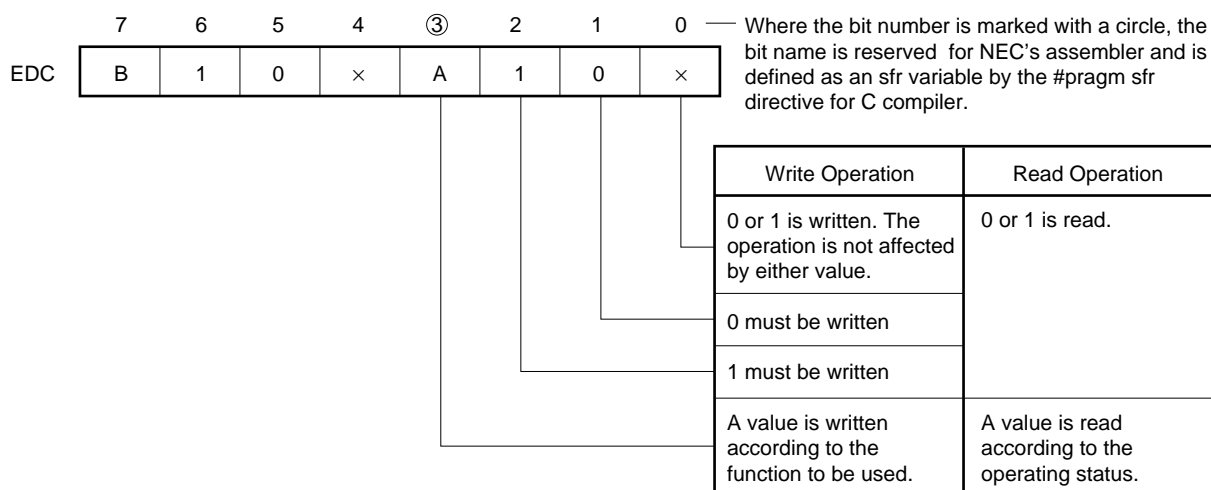
**Remark** : Supplementary information

Numeric notations : Binary .....  $\times \times \times \times$  B or  $\times \times \times \times$

Decimal .....  $\times \times \times \times$

Hexadecimal .....  $\times \times \times \times$  H

## Register Notation



**Code combinations marked “Setting prohibited” in the register notations in the text must not be written.**

Easily confused characters : 0 (Zero), O (Letter O)

: 1 (One), l (Lower-case letter L), I (Upper-case letter I)

**Related Documents** Some of the related documents in this document are preliminary versions but are not so specified.

**Device-related documents**

Document Name	Document No.	
	Japanese	English
μPD784020, 784021 Data Sheet	U11514J	U11514E
μPD784025, 784026 Data Sheet	U11605J	U11605E
μPD78P4026 Preliminary Data Sheet	U11609J	U11609E
μPD784026 Subseries Special Function Register Table	U10593J	—
μPD784026 Subseries User's Manual - Hardware	U10898J	This manual
μPD784026 Subseries Application Note - Hardware Fundamentals	U10573J	—
78K/IV Series Application Note - Software Fundamentals	U10095J	—
78K/IV Series User's Manual - Instruction	U10905J	U10905E
78K/IV Series Instruction Table	U10594J	—
78K/IV Series Instruction Set	U10595J	—

**Documents on development tools (User's Manuals)**

Document Name		Document No.	
		Japanese	English
RA78K4 Assembler Package	Operation	U11334J	U11334E
	Language	U11162J	U11162E
RA78K4 Structured Assembler Preprocessor		U11743J	U11743E
CC78K4	Operation	EEU-960	—
	Language	EEU-961	—
CC78K Series Library Source File		U12322J	—
PG-1500 PROM Programmer		U11940J	EEU-1335
PG-1500 Controller PC-9800 Series (MS-DOS™) based		EEU-704	EEU-1291
PG-1500 Controller IBM PC Series (PC DOS™) based		EEU-5008	U10540E
IE-784000-R		EEU-5004	EEU-1534
IE-784026-R-EM1		EEU-5017	EEU-1528
EP-78230		EEU-985	EEU-1515
SM78K4 System Simulator Windows™ based	Reference	U10093J	U10093E
SM78K Series System Simulator	External component user open interface specification	U10092J	U10092E
ID78K4 Integrated Debugger Windows based	Reference	U10440J	U10440E
ID78K4 Integrated Debugger HP9000 Series 700™ (HP-UX™) based	Reference	U11960J	On preparation

**Caution** The contents of the above documents are subject to change without notice. Be sure to use the latest edition of each document for designing.

**Documents on embedded software (User's Manuals)**

Document Name		Document No.	
		Japanese	English
78K/IV Series Real-Time OS	Fundamental	U10603J	U10603E
	Installation	U10604J	U10604E
	Debugger	U10364J	—
78K/IV Series OS MX78K4	Fundamental	U11779J	—

**Other documents**

Document Name	Document No.	
	Japanese	English
IC Package Manual	C10943X	
Semiconductor Device Mounting Technology Manual	C10535J	C10535E
Quality Grades on NEC's Semiconductor Devices	C11531J	C11531E
Reliability and Quality Control of NEC's Semiconductor Devices	C10983J	C10983E
Electrostatic Discharge Test	MEM-539	—
Quality Assurance Guide to Semiconductor Devices	C11893J	MEI-1202
Guide to Microcontroller-Related Products - by third parties	C11416J	—

**Caution** The contents of the above documents are subject to change without notice. Be sure to use the latest edition of each document for designing.

[MEMO]



## TABLE OF CONTENTS

<b>CHAPTER 1 GENERAL .....</b>	<b>1</b>
<b>1.1 FEATURES .....</b>	<b>3</b>
<b>1.2 ORDERING INFORMATION AND QUALITY GRADES .....</b>	<b>4</b>
1.2.1 Ordering Information .....	4
1.2.2 Quality Grades .....	4
<b>1.3 PIN CONFIGURATION (TOP VIEW) .....</b>	<b>5</b>
1.3.1 Normal Operating Mode .....	5
1.3.2 PROM Programming Mode .....	7
<b>1.4 APPLICATION SYSTEM CONFIGURATION EXAMPLE (PPC) .....</b>	<b>8</b>
<b>1.5 BLOCK DIAGRAM .....</b>	<b>9</b>
<b>1.6 LIST OF FUNCTIONS .....</b>	<b>10</b>
<b>1.7 DIFFERENCES AMONG <math>\mu</math>PD784026 SERIES PRODUCTS .....</b>	<b>12</b>
1.7.1 Functional Differences .....	12
1.7.2 Main Differences between $\mu$ PD784026 Subseries and $\mu$ PD78234 Subseries .....	13
<b>CHAPTER 2 PIN FUNCTIONS .....</b>	<b>15</b>
<b>2.1 PIN FUNCTION TABLES .....</b>	<b>15</b>
2.1.1 Normal Operating Mode .....	15
2.1.2 PROM Programming Mode .....	19
<b>2.2 PIN FUNCTIONS .....</b>	<b>20</b>
2.2.1 Normal Operating Mode .....	20
2.2.2 PROM Programming Mode ( $\mu$ PD78P4026) .....	29
<b>2.3 INPUT/OUTPUT CIRCUITS AND CONNECTION OF UNUSED PINS .....</b>	<b>30</b>
<b>2.4 CAUTIONS .....</b>	<b>33</b>
<b>CHAPTER 3 CPU ARCHITECTURE .....</b>	<b>35</b>
<b>3.1 MEMORY SPACE .....</b>	<b>35</b>
<b>3.2 INTERNAL ROM AREA .....</b>	<b>40</b>
<b>3.3 BASE AREA .....</b>	<b>41</b>
3.3.1 Vector Table Area .....	42
3.3.2 CALLT Instruction Table Area .....	43
3.3.3 CALLF Instruction Entry Area .....	43
<b>3.4 INTERNAL DATA AREA .....</b>	<b>44</b>
3.4.1 Internal RAM Area .....	45
3.4.2 Special Function Register (SFR) Area .....	48
3.4.3 External SFR Area .....	48
<b>3.5 EXTERNAL MEMORY SPACE .....</b>	<b>48</b>
<b>3.6 <math>\mu</math>PD78P4026 MEMORY MAPPING .....</b>	<b>49</b>
<b>3.7 CONTROL REGISTERS .....</b>	<b>50</b>
3.7.1 Program Counter (PC) .....	50
3.7.2 Program Status Word (PSW) .....	50

3.7.3	Use of RSS Bit.....	54
3.7.4	Stack Pointer (SP) .....	57
<b>3.8</b>	<b>GENERAL-PURPOSE REGISTERS .....</b>	<b>61</b>
3.8.1	Configuration.....	61
3.8.2	Functions.....	63
<b>3.9</b>	<b>SPECIAL FUNCTION REGISTERS (SFRS).....</b>	<b>66</b>
<b>3.10</b>	<b>CAUTIONS.....</b>	<b>72</b>
<b>CHAPTER 4</b>	<b>CLOCK GENERATOR .....</b>	<b>73</b>
<b>4.1</b>	<b>CONFIGURATION AND FUNCTION.....</b>	<b>73</b>
<b>4.2</b>	<b>CONTROL REGISTERS .....</b>	<b>75</b>
4.2.1	Standby Control Register (STBC).....	75
4.2.2	Oscillation Stabilization Time Specification Register (OSTS) .....	77
<b>4.3</b>	<b>CLOCK GENERATOR OPERATION.....</b>	<b>78</b>
4.3.1	Clock Oscillator.....	78
4.3.2	Frequency Divider .....	79
<b>4.4</b>	<b>CAUTIONS.....</b>	<b>80</b>
4.4.1	When an External Clock Is Input .....	80
4.4.2	When Crystal/Ceramic Oscillation is Used.....	81
<b>CHAPTER 5</b>	<b>PORT FUNCTIONS.....</b>	<b>85</b>
<b>5.1</b>	<b>DIGITAL INPUT/OUTPUT PORTS .....</b>	<b>85</b>
<b>5.2</b>	<b>PORT 0.....</b>	<b>87</b>
5.2.1	Hardware Configuration .....	87
5.2.2	Input/Output Mode/Control Mode Setting .....	88
5.2.3	Operating Status .....	89
5.2.4	Internal Pull-Up Resistors .....	91
5.2.5	Transistor Drive .....	93
<b>5.3</b>	<b>PORT 1.....</b>	<b>94</b>
5.3.1	Hardware Configuration .....	95
5.3.2	Input/Output Mode/Control Mode Setting .....	99
5.3.3	Operating Status .....	101
5.3.4	Internal Pull-Up Resistors .....	104
5.3.5	Direct LED Drive .....	105
<b>5.4</b>	<b>PORT 2.....</b>	<b>106</b>
5.4.1	Hardware Configuration .....	108
5.4.2	Input Mode/Control Mode Setting .....	110
5.4.3	Operating Status .....	110
5.4.4	Internal Pull-Up Resistors .....	110
<b>5.5</b>	<b>PORT 3.....</b>	<b>112</b>
5.5.1	Hardware Configuration .....	114
5.5.2	Input/Output Mode/Control Mode Setting .....	118
5.5.3	Operating Status .....	120
5.5.4	Internal Pull-Up Resistors .....	123
<b>5.6</b>	<b>PORT 4 .....</b>	<b>125</b>
5.6.1	Hardware Configuration .....	125

5.6.2	Input/Output Mode / Control Mode Setting .....	126
5.6.3	Operating Status .....	127
5.6.4	Internal Pull-Up Resistors .....	129
5.6.5	Direct LED Drive .....	131
<b>5.7</b>	<b>PORT 5 .....</b>	<b>132</b>
5.7.1	Hardware Configuration .....	132
5.7.2	Input/Output Mode/Control Mode Setting .....	133
5.7.3	Operating Status .....	134
5.7.4	Internal Pull-Up Resistors .....	136
5.7.5	Direct LED Drive .....	138
<b>5.8</b>	<b>PORT 6 .....</b>	<b>139</b>
5.8.1	Hardware Configuration .....	140
5.8.2	Input/Output Mode/Control Mode Setting .....	144
5.8.3	Operating Status .....	146
5.8.4	Internal Pull-Up Resistors .....	148
<b>5.9</b>	<b>PORT 7 .....</b>	<b>150</b>
5.9.1	Hardware Configuration .....	150
5.9.2	Input/Output Mode/Control Mode Setting .....	151
5.9.3	Operating Status .....	152
5.9.4	Internal Pull-Up Resistors .....	153
5.9.5	Caution .....	153
<b>5.10</b>	<b>PORT OUTPUT CHECK FUNCTION .....</b>	<b>154</b>
<b>5.11</b>	<b>CAUTIONS .....</b>	<b>155</b>
<b>CHAPTER 6</b>	<b>REAL-TIME OUTPUT FUNCTION .....</b>	<b>157</b>
6.1	CONFIGURATION AND FUNCTION .....	157
6.2	REAL-TIME OUTPUT PORT CONTROL REGISTER (RTPC) .....	159
6.3	REAL-TIME OUTPUT PORT ACCESSES .....	160
6.4	OPERATION .....	162
6.5	EXAMPLE OF USE .....	165
6.6	CAUTIONS .....	167
<b>CHAPTER 7</b>	<b>OUTLINE OF TIMER/COUNTER .....</b>	<b>169</b>
<b>CHAPTER 8</b>	<b>TIMER/COUNTER 0 .....</b>	<b>171</b>
8.1	FUNCTIONS .....	171
8.2	CONFIGURATION .....	174
8.3	TIMER/COUNTER 0 CONTROL REGISTERS .....	178
8.4	16-BIT TIMER REGISTER 0 (TM0) OPERATION .....	183
8.4.1	Basic Operation .....	183
8.4.2	Clear Operation .....	185
8.5	EXTERNAL EVENT COUNTER FUNCTION .....	187
8.6	COMPARE REGISTER AND CAPTURE REGISTER OPERATION .....	191
8.6.1	Compare Operations .....	191
8.6.2	Capture Operations .....	193
8.7	BASIC OPERATION OF OUTPUT CONTROL CIRCUIT .....	194

8.7.1	Basic Operation .....	196
8.7.2	Toggle Output .....	196
8.7.3	PWM Output .....	198
8.7.4	PPG Output.....	204
8.7.5	Software Triggered One-Shot Pulse Output.....	210
<b>8.8</b>	<b>EXAMPLES OF USE.....</b>	<b>211</b>
8.8.1	Operation as Interval Timer (1) .....	211
8.8.2	Operation as Interval Timer (2) .....	213
8.8.3	Pulse Width Measurement Operation .....	215
8.8.4	Operation as PWM Output .....	217
8.8.5	Operation as PPG Output .....	221
8.8.6	Example of Software Triggered One-Shot Pulse Output .....	225
<b>8.9</b>	<b>CAUTIONS .....</b>	<b>228</b>
<b>CHAPTER 9</b>	<b>TIMER/COUNTER 1.....</b>	<b>233</b>
<b>9.1</b>	<b>FUNCTIONS.....</b>	<b>233</b>
<b>9.2</b>	<b>CONFIGURATION .....</b>	<b>235</b>
<b>9.3</b>	<b>TIMER/COUNTER 1 CONTROL REGISTERS .....</b>	<b>239</b>
<b>9.4</b>	<b>TIMER REGISTER 1 (TM1) OPERATION .....</b>	<b>242</b>
9.4.1	Basic Operation .....	242
9.4.2	Clear Operation .....	245
<b>9.5</b>	<b>EXTERNAL EVENT COUNTER FUNCTION .....</b>	<b>247</b>
<b>9.6</b>	<b>COMPARE REGISTER, CAPTURE/COMPARE REGISTER, AND CAPTURE REGISTER OPERATION .....</b>	<b>251</b>
9.6.1	Compare Operations .....	251
9.6.2	Capture Operations .....	253
<b>9.7</b>	<b>EXAMPLES OF USE.....</b>	<b>256</b>
9.7.1	Operation as Interval Timer (1) .....	256
9.7.2	Operation as Interval Timer (2) .....	259
9.7.3	Pulse Width Measurement Operation .....	261
<b>9.8</b>	<b>CAUTIONS .....</b>	<b>264</b>
<b>CHAPTER 10</b>	<b>TIMER/COUNTER 2.....</b>	<b>269</b>
<b>10.1</b>	<b>FUNCTIONS.....</b>	<b>269</b>
<b>10.2</b>	<b>CONFIGURATION .....</b>	<b>273</b>
<b>10.3</b>	<b>TIMER/COUNTER 2 CONTROL REGISTERS .....</b>	<b>277</b>
<b>10.4</b>	<b>TIMER REGISTER 2 (TM2) OPERATION .....</b>	<b>281</b>
10.4.1	Basic Operation .....	281
10.4.2	Clear Operation .....	284
<b>10.5</b>	<b>EXTERNAL EVENT COUNTER FUNCTION .....</b>	<b>286</b>
<b>10.6</b>	<b>ONE-SHOT TIMER FUNCTION.....</b>	<b>290</b>
<b>10.7</b>	<b>COMPARE REGISTER, CAPTURE/COMPARE REGISTER AND CAPTURE REGISTER OPERATION .....</b>	<b>291</b>
10.7.1	Compare Operations .....	291
10.7.2	Capture Operations .....	294
<b>10.8</b>	<b>BASIC OPERATION OF OUTPUT CONTROL CIRCUIT .....</b>	<b>297</b>

10.8.1	Basic Operation .....	299
10.8.2	Toggle Output .....	299
10.8.3	PWM Output .....	301
10.8.4	PPG Output .....	308
<b>10.9</b>	<b>EXAMPLES OF USE .....</b>	<b>315</b>
10.9.1	Operation as Interval Timer (1) .....	315
10.9.2	Operation as Interval Timer (2) .....	318
10.9.3	Pulse Width Measurement Operation .....	321
10.9.4	Operation as PWM Output .....	324
10.9.5	Operation as PPG Output .....	328
10.9.6	Operation as External Event Counter .....	333
10.9.7	Operation as One-Shot Timer .....	335
<b>10.10</b>	<b>CAUTIONS .....</b>	<b>338</b>
<b>CHAPTER 11</b>	<b>TIMER 3 .....</b>	<b>343</b>
<b>11.1</b>	<b>FUNCTION .....</b>	<b>343</b>
<b>11.2</b>	<b>CONFIGURATION .....</b>	<b>344</b>
<b>11.3</b>	<b>TIMER 3 CONTROL REGISTERS .....</b>	<b>346</b>
<b>11.4</b>	<b>TIMER REGISTER 3 (TM3) OPERATION .....</b>	<b>348</b>
11.4.1	Basic Operation .....	348
11.4.2	Clear Operation .....	351
<b>11.5</b>	<b>COMPARE REGISTER OPERATION .....</b>	<b>353</b>
<b>11.6</b>	<b>EXAMPLE OF USE .....</b>	<b>354</b>
<b>11.7</b>	<b>CAUTIONS .....</b>	<b>356</b>
<b>CHAPTER 12</b>	<b>WATCHDOG TIMER FUNCTION .....</b>	<b>359</b>
<b>12.1</b>	<b>CONFIGURATION .....</b>	<b>359</b>
<b>12.2</b>	<b>WATCHDOG TIMER MODE REGISTER (WDM) .....</b>	<b>360</b>
<b>12.3</b>	<b>OPERATION .....</b>	<b>362</b>
12.3.1	Count Operation .....	362
12.3.2	Interrupt Priorities .....	362
<b>12.4</b>	<b>CAUTIONS .....</b>	<b>363</b>
12.4.1	General Cautions on Use of Watchdog Timer .....	363
12.4.2	Cautions on $\mu$ PD784026 Subseries Watchdog Timer .....	364
<b>CHAPTER 13</b>	<b>PWM OUTPUT UNIT .....</b>	<b>365</b>
<b>13.1</b>	<b>PWM OUTPUT UNIT CONFIGURATION .....</b>	<b>365</b>
<b>13.2</b>	<b>PWM OUTPUT UNIT CONTROL REGISTERS .....</b>	<b>367</b>
13.2.1	PWM Control Register (PWMC) .....	367
13.2.2	PWM Prescaler Register (PWPR) .....	368
13.2.3	PWM Modulo Registers (PWM0, PWM1) .....	368
<b>13.3</b>	<b>PWM OUTPUT UNIT OPERATION .....</b>	<b>369</b>
13.3.1	Basic PWM Output Operation .....	369
13.3.2	PWM Pulse Output Enabling/Disabling .....	370
13.3.3	PWM Pulse Active Level Specification .....	370

13.3.4	PWM Pulse Width Rewrite Cycle Specification .....	371
<b>13.4</b>	<b>CAUTION .....</b>	<b>372</b>
<b>CHAPTER 14</b>	<b>A/D CONVERTER .....</b>	<b>373</b>
<b>14.1</b>	<b>CONFIGURATION .....</b>	<b>373</b>
<b>14.2</b>	<b>A/D CONVERTER MODE REGISTER (ADM).....</b>	<b>377</b>
<b>14.3</b>	<b>OPERATION .....</b>	<b>380</b>
14.3.1	Basic A/D Converter Operation .....	380
14.3.2	Select Mode .....	384
14.3.3	Scan Mode .....	385
14.3.4	A/D Conversion Operation Start by Software .....	387
14.3.5	A/D Conversion Operation Start by Hardware .....	389
<b>14.4</b>	<b>EXTERNAL CIRCUIT OF A/D CONVERTER .....</b>	<b>392</b>
<b>14.5</b>	<b>CAUTIONS .....</b>	<b>392</b>
<b>CHAPTER 15</b>	<b>D/A CONVERTER .....</b>	<b>395</b>
<b>15.1</b>	<b>CONFIGURATION .....</b>	<b>395</b>
<b>15.2</b>	<b>D/A CONVERTER MODE REGISTER (DAM).....</b>	<b>397</b>
<b>15.3</b>	<b>D/A CONVERTER OPERATION.....</b>	<b>398</b>
15.3.1	Basic Operation .....	398
15.3.2	D/A Converter Standby Operation .....	398
<b>15.4</b>	<b>CAUTIONS .....</b>	<b>399</b>
<b>CHAPTER 16</b>	<b>ASYNCHRONOUS SERIAL INTERFACE/3-WIRE SERIAL I/O .....</b>	<b>401</b>
<b>16.1</b>	<b>SWITCHING BETWEEN ASYNCHRONOUS SERIAL INTERFACE MODE AND 3-WIRE SERIAL I/O MODE.....</b>	<b>402</b>
<b>16.2</b>	<b>ASYNCHRONOUS SERIAL INTERFACE MODE .....</b>	<b>403</b>
16.2.1	Configuration in Asynchronous Serial Interface Mode.....	403
16.2.2	Asynchronous Serial Interface Control Registers .....	406
16.2.3	Data Format .....	410
16.2.4	Parity Types and Operations.....	411
16.2.5	Transmission .....	412
16.2.6	Reception .....	413
16.2.7	Receive Errors .....	414
<b>16.3</b>	<b>3-WIRE SERIAL I/O MODE .....</b>	<b>415</b>
16.3.1	Configuration in 3-Wire Serial I/O Mode .....	415
16.3.2	Clocked Serial Interface Mode Registers (CSIM1, CSIM2) .....	418
16.3.3	Basic Operation Timing .....	419
16.3.4	Operation When Transmission Only is Enabled .....	421
16.3.5	Operation When Reception Only is Enabled .....	422
16.3.6	Operation When Transmission/Reception is Enabled .....	423
16.3.7	Corrective Action in Case of Slippage of Serial Clock and Shift Operations .....	424
<b>16.4</b>	<b>BAUD RATE GENERATOR .....</b>	<b>424</b>
16.4.1	Baud Rate Generator Configuration .....	424
16.4.2	Baud Rate Generator Control Register (BRGC, BRGC2) .....	426

16.4.3	Baud Rate Generator Operation .....	428
16.4.4	Baud Rate Setting in Asynchronous Serial Interface Mode .....	430
<b>16.5</b>	<b>CAUTIONS .....</b>	<b>433</b>
<b>CHAPTER 17</b>	<b>CLOCKED SERIAL INTERFACE .....</b>	<b>435</b>
<b>17.1</b>	<b>FUNCTIONS .....</b>	<b>435</b>
<b>17.2</b>	<b>CONFIGURATION .....</b>	<b>435</b>
<b>17.3</b>	<b>CONTROL REGISTERS .....</b>	<b>438</b>
17.3.1	Clocked Serial Interface Mode Register (CSIM) .....	438
17.3.2	Serial Bus Interface Control Register (SBIC) .....	439
<b>17.4</b>	<b>3-WIRE SERIAL I/O MODE .....</b>	<b>441</b>
17.4.1	Basic Operation Timing .....	442
17.4.2	Operation When Transmission Only Is Enabled .....	444
17.4.3	Operation When Reception Only Is Enabled .....	445
17.4.4	Operation When Transmission/Reception Is Enabled .....	446
17.4.5	Corrective Action in Case of Slippage of Serial Clock and Shift Operations .....	447
<b>17.5</b>	<b>SBI MODE .....</b>	<b>447</b>
17.5.1	SBI Features .....	448
17.5.2	Serial Interface Configuration .....	450
17.5.3	Address Match Detection .....	453
17.5.4	SBI Mode Control Registers .....	453
<b>17.6</b>	<b>SBI COMMUNICATION OPERATIONS AND VARIOUS SIGNALS .....</b>	<b>458</b>
17.6.1	Bus Release Signal (REL) .....	459
17.6.2	Command Signal (CMD) .....	459
17.6.3	Address .....	460
17.6.4	Command and Data .....	461
17.6.5	Acknowledge Signal ( $\overline{\text{ACK}}$ ) .....	462
17.6.6	Busy Signal ( $\overline{\text{BUSY}}$ ) and Ready Signal (READY) .....	463
17.6.7	Various Signals .....	464
17.6.8	Communication Operations .....	471
17.6.9	Busy Release .....	476
17.6.10	Wake-Up Setting Operation .....	476
17.6.11	Starting Transmit/Receive Operation .....	476
<b>17.7</b>	<b>CAUTIONS .....</b>	<b>477</b>
<b>CHAPTER 18</b>	<b>CLOCK OUTPUT FUNCTION .....</b>	<b>479</b>
<b>18.1</b>	<b>CONFIGURATION .....</b>	<b>479</b>
<b>18.2</b>	<b>CLOCK OUTPUT MODE REGISTER (CLOM) .....</b>	<b>481</b>
<b>18.3</b>	<b>OPERATION .....</b>	<b>482</b>
18.3.1	Clock Output .....	482
18.3.2	One-Bit Output Port .....	483
18.3.3	Operation in Standby Mode .....	483
<b>18.4</b>	<b>CAUTIONS .....</b>	<b>483</b>

<b>CHAPTER 19</b>	<b>EDGE DETECTION FUNCTION .....</b>	<b>485</b>
<b>19.1</b>	<b>EDGE DETECTION FUNCTION CONTROL REGISTERS .....</b>	<b>485</b>
19.1.1	External Interrupt Mode Registers (INTM0, INTM1) .....	485
19.1.2	Sampling Clock Selection Register (SCS0) .....	488
<b>19.2</b>	<b>EDGE DETECTION FOR PINS P20, P25 AND P26 .....</b>	<b>489</b>
<b>19.3</b>	<b>P21 PIN EDGE DETECTION .....</b>	<b>490</b>
<b>19.4</b>	<b>PIN EDGE DETECTION FOR PINS P22 TO P24 .....</b>	<b>491</b>
<b>19.5</b>	<b>CAUTIONS .....</b>	<b>492</b>
<b>CHAPTER 20</b>	<b>INTERRUPT FUNCTIONS .....</b>	<b>493</b>
<b>20.1</b>	<b>INTERRUPT REQUEST SOURCES .....</b>	<b>494</b>
20.1.1	Software Interrupts .....	496
20.1.2	Operand Error Interrupts .....	496
20.1.3	Non-Maskable Interrupts .....	496
20.1.4	Maskable Interrupts .....	496
<b>20.2</b>	<b>INTERRUPT PROCESSING MODES .....</b>	<b>497</b>
20.2.1	Vectored Interrupt Processing .....	497
20.2.2	Macro Service .....	497
20.2.3	Context Switching .....	497
<b>20.3</b>	<b>INTERRUPT PROCESSING CONTROL REGISTERS .....</b>	<b>498</b>
20.3.1	Interrupt Control Registers .....	501
20.3.2	Interrupt Mask Registers (MK0/MK1L) .....	506
20.3.3	In-Service Priority Register (ISPR) .....	508
20.3.4	Interrupt Mode Control Register (IMC) .....	509
20.3.5	Watchdog Timer Mode Register (WDM) .....	510
20.3.6	Program Status Word (PSW) .....	511
<b>20.4</b>	<b>SOFTWARE INTERRUPT ACKNOWLEDGMENT OPERATIONS .....</b>	<b>511</b>
20.4.1	BRK Instruction Software Interrupt Acknowledgment Operation .....	511
20.4.2	BRKCS Instruction Software Interrupt (Software Context Switching) Acknowledgment Operation .....	512
<b>20.5</b>	<b>OPERAND ERROR INTERRUPT ACKNOWLEDGMENT OPERATION .....</b>	<b>513</b>
<b>20.6</b>	<b>NON-MASKABLE INTERRUPT ACKNOWLEDGMENT OPERATION .....</b>	<b>514</b>
<b>20.7</b>	<b>MASKABLE INTERRUPT ACKNOWLEDGMENT OPERATION .....</b>	<b>518</b>
20.7.1	Vectored Interrupt .....	520
20.7.2	Context Switching .....	520
20.7.3	Maskable Interrupt Priority Levels .....	522
<b>20.8</b>	<b>MACRO SERVICE FUNCTION .....</b>	<b>528</b>
20.8.1	Outline of Macro Service Function .....	528
20.8.2	Types of Macro Service .....	528
20.8.3	Basic Macro Service Operation .....	531
20.8.4	Operation at End of Macro Service .....	532
20.8.5	Macro Service Control Registers .....	535
20.8.6	Macro Service Type A .....	538
20.8.7	Macro Service Type B .....	543
20.8.8	Macro Service Type C .....	548
20.8.9	Counter Mode .....	562



<b>20.9 WHEN INTERRUPT REQUESTS AND MACRO SERVICE ARE TEMPORARILY HELD PENDING .....</b>	<b>564</b>
<b>20.10 INSTRUCTIONS WHOSE EXECUTION IS TEMPORARILY SUSPENDED BY AN INTERRUPT OR MACRO SERVICE .....</b>	<b>566</b>
<b>20.11 INTERRUPT AND MACRO SERVICE OPERATION TIMING .....</b>	<b>566</b>
20.11.1 Interrupt Acknowledge Processing Time .....	567
20.11.2 Processing Time of Macro Service .....	568
<b>20.12 RESTORING INTERRUPT FUNCTION TO INITIAL STATE .....</b>	<b>569</b>
<b>20.13 CAUTIONS .....</b>	<b>570</b>
 <b>CHAPTER 21 LOCAL BUS INTERFACE FUNCTION.....</b>	 <b>573</b>
<b>21.1 MEMORY EXTENSION FUNCTION .....</b>	<b>573</b>
21.1.1 Memory Extension Mode Register (MM) .....	573
21.1.2 Memory Map with External Memory Extension .....	575
21.1.3 Basic Operation of Local Bus Interface .....	584
<b>21.2 WAIT FUNCTION .....</b>	<b>585</b>
21.2.1 Wait Function Control Registers .....	585
21.2.2 Address Waits .....	588
21.2.3 Access Waits .....	591
<b>21.3 PSEUDO-STATIC RAM REFRESH FUNCTION.....</b>	<b>598</b>
21.3.1 Control Registers .....	599
21.3.2 Operations .....	600
<b>21.4 BUS HOLD FUNCTION .....</b>	<b>604</b>
21.4.1 Hold Mode Register (HLDM) .....	604
21.4.2 Operation .....	605
<b>21.5 CAUTIONS .....</b>	<b>607</b>
 <b>CHAPTER 22 STANDBY FUNCTION .....</b>	 <b>609</b>
<b>22.1 CONFIGURATION AND FUNCTION .....</b>	<b>609</b>
<b>22.2 CONTROL REGISTERS .....</b>	<b>611</b>
22.2.1 Standby Control Register (STBC).....	611
22.2.2 Oscillation Stabilization Time Specification Register (OSTS) .....	613
<b>22.3 HALT MODE .....</b>	<b>615</b>
22.3.1 HALT Mode Setting and Operating States .....	615
22.3.2 HALT Mode Release .....	616
<b>22.4 STOP MODE .....</b>	<b>624</b>
22.4.1 STOP Mode Setting and Operating States .....	624
22.4.2 STOP Mode Release .....	626
<b>22.5 IDLE MODE .....</b>	<b>631</b>
22.5.1 IDLE Mode Setting and Operating States .....	631
22.5.2 IDLE Mode Release .....	632
<b>22.6 CHECK ITEMS WHEN STOP MODE/IDLE MODE IS USED .....</b>	<b>636</b>
<b>22.7 CAUTIONS .....</b>	<b>639</b>

<b>CHAPTER 23</b>	<b>RESET FUNCTION .....</b>	<b>641</b>
23.1	RESET FUNCTION .....	641
23.2	CAUTIONS .....	646
<b>CHAPTER 24</b>	<b>μPD78P4026 PROGRAMMING .....</b>	<b>647</b>
24.1	OPERATING MODES .....	647
24.2	PROM WRITE PROCEDURE .....	649
24.3	PROM READING PROCEDURE .....	653
24.4	SCREENING OF ONE-TIME PROM PRODUCT .....	654
24.5	ERASING PROCEDURE (μPD78P4026KK-T ONLY) .....	655
24.6	STICKER ON ERASURE WINDOW (μPD78P4026KK-T ONLY) .....	655
24.7	QUALITY .....	655
24.8	CAUTIONS .....	655
<b>CHAPTER 25</b>	<b>INSTRUCTION OPERATIONS .....</b>	<b>657</b>
25.1	LEGEND .....	657
25.2	LIST OF OPERATIONS .....	661
25.3	INSTRUCTIONS LISTED BY TYPE OF ADDRESSING .....	686
<b>APPENDIX A</b>	<b>DIFFERENCES BETWEEN μPD784026 SUBSERIES AND μPD78234 SUBSERIES .....</b>	<b>691</b>
<b>APPENDIX B</b>	<b>DEVELOPMENT TOOLS .....</b>	<b>695</b>
B.1	LANGUAGE PROCESSING SOFTWARE .....	697
B.2	DEBUGGING TOOLS .....	699
B.2.1	Hardware .....	699
B.2.2	Software .....	700
B.3	PROM WRITING TOOLS .....	701
B.4	OS OF IBM PC .....	702
B.5	CONVERSION SOCKET (EV-9200GC-80) AND CONVERSION ADAPTER (TGK-080SDW) .....	703
B.6	CHECK SHEET FOR μPD784026 SUBSERIES DEVELOPMENT TOOLS .....	706
<b>APPENDIX C</b>	<b>SOFTWARE FOR EMBEDDING .....</b>	<b>709</b>
C.1	REAL-TIME OS .....	709
<b>APPENDIX D</b>	<b>REGISTER INDEX .....</b>	<b>711</b>
<b>APPENDIX E</b>	<b>RIVISION HISTORY .....</b>	<b>715</b>

# LIST OF FIGURES (1/11)

Figure No.	Title	Page
2-1	Pin Input/Output Circuits.....	32
3-1	μPD784020 Memory Map .....	36
3-2	μPD784021 Memory Map .....	37
3-3	μPD784025 Memory Map .....	38
3-4	μPD784026 Memory Map .....	39
3-5	Internal RAM Memory Map .....	46
3-6	Internal Memory Size Switching Register (IMS) Format .....	49
3-7	Program Counter (PC) Format .....	50
3-8	Program Status Word (PSW) Format .....	51
3-9	Stack Pointer (SP) Format .....	57
3-10	Data Saved to Stack Area .....	58
3-11	Data Restored from Stack Area .....	59
3-12	General-Purpose Register Format .....	61
3-13	General-Purpose Register Addresses .....	62
4-1	Clock Generator Block Diagram .....	73
4-2	Clock Oscillator External Circuitry.....	74
4-3	Standby Control Register (STBC) Format .....	76
4-4	Oscillation Stabilization Time Specification Register (OSTS) Format .....	77
4-5	Signal Extraction with External Clock Input .....	80
4-6	Cautions on Resonator Connection .....	81
4-7	Incorrect Example of Resonator Connection .....	82
5-1	Port Configuration .....	85
5-2	Port 0 Block Diagram .....	87
5-3	Port 0 Mode Register (PM0) Format .....	88
5-4	Port Specified as Output Port .....	89
5-5	Port Specified as Input Port .....	90
5-6	Pull-Up Resistor Option Register (PUO) Format .....	91
5-7	Pull-Up Resistor Specification (Port 0) .....	92
5-8	Example of Transistor Drive .....	93
5-9	Block Diagram of P10 and P11 (Port 1) .....	95
5-10	Block Diagram of P12 (Port 1) .....	96
5-11	Block Diagram of P13 (Port 1) .....	97
5-12	Block Diagram of P14 (Port 1) .....	98
5-13	Block Diagram of P15 to P17 (Port 1) .....	98
5-14	Port 1 Mode Register (PM1) Format .....	99
5-15	Port 1 Mode Control Register (PMC1) Format .....	100
5-16	Port Specified as Output Port .....	101
5-17	Port Specified as Input Port .....	102
5-18	Control Specification .....	103
5-19	Pull-Up Resistor Option Register (PUO) Format .....	104

# LIST OF FIGURES (2/11)

Figure No.	Title	Page
5-20	Pull-Up Resistor Specification (Port 1) .....	105
5-21	Example of Direct LED Drive .....	105
5-22	Block Diagram of P20 to P24, P26 and P27 (Port 2).....	108
5-23	Block Diagram of P25 (Port 2) .....	109
5-24	Port Specified as Input Port .....	110
5-25	Pull-Up Resistor Option Register (PUO) Format.....	110
5-26	Pull-Up Specification (Port 2) .....	111
5-27	Block Diagram of P30 (Port 3) .....	114
5-28	Block Diagram of P31 and P34 to P37 (Port 3) .....	115
5-29	Block Diagram of P32 (Port 3) .....	116
5-30	Block Diagram of P33 (Port 3) .....	117
5-31	Port 3 Mode Register (PM3) Format.....	118
5-32	Port 3 Mode Control Register (PMC3) Format .....	119
5-33	Port Specified as Output Port .....	120
5-34	Port Specified as Input Port .....	121
5-35	Control Specification .....	122
5-36	Pull-Up Resistor Option Register (PUO) Format.....	123
5-37	Pull-Up Specification (Port 3) .....	124
5-38	Port 4 Block Diagram .....	125
5-39	Port 4 Mode Register (PM4) Format.....	126
5-40	Port Specified as Output Port .....	127
5-41	Port Specified as Input Port .....	128
5-42	Pull-Up Resistor Option Register (PUO) Format.....	129
5-43	Pull-Up Specification (Port 4) .....	130
5-44	Example of Direct LED Drive .....	131
5-45	Port 5 Block Diagram .....	132
5-46	Port 5 Mode Register (PM5) Format.....	133
5-47	Port Specified as Output Port .....	134
5-48	Port Specified as Input Port .....	135
5-49	Pull-Up Resistor Option Register (PUO) Format.....	136
5-50	Pull-Up Specification (Port 5) .....	137
5-51	Example of Direct LED Drive .....	138
5-52	Block Diagram of P60 to P63 (Port 6) .....	140
5-53	Block Diagram of P64 and P65 (Port 6) .....	141
5-54	Block Diagram of P66 (Port 6) .....	142
5-55	Block Diagram of P67 (Port 6) .....	143
5-56	Port 6 Mode Register (PM6) Format.....	146
5-57	Port Specified as Output Port .....	146
5-58	Port Specified as Input Port .....	147
5-59	Pull-Up Resistor Option Register (PUO) Format.....	148
5-60	Pull-Up Specification (Port 6) .....	149
5-61	Port 7 Block Diagram .....	150
5-62	Port 7 Mode Register (PM7) Format.....	151

# LIST OF FIGURES (3/11)

Figure No.	Title	Page
5-63	Port Specified as Output Port .....	152
5-64	Port Specified as Input Port .....	153
6-1	Real-Time Output Port Block Diagram .....	158
6-2	Real-Time Output Port Control Register (RTPC) Format .....	159
6-3	Port 0 Buffer Register (P0H, P0L) Configuration .....	160
6-4	Real-Time Output Port Operation Timing .....	163
6-5	Real-Time Output Port Operation Timing (2 Channel Independent Control Example) .....	164
6-6	Real-Time Output Port Operation Timing .....	165
6-7	Real-Time Output Function Control Register Settings .....	166
6-8	Real-Time Output Function Setting Procedure .....	166
6-9	Interrupt Request Processing when Real-Time Output Function is Used .....	167
7-1	Timer/Counter Block Diagram .....	170
8-1	Timer/Counter 0 Block Diagram .....	175
8-2	Timer Control Register 0 (TMC0) Format .....	178
8-3	Prescaler Mode Register 0 (PRM0) Format .....	179
8-4	Capture/Compare Control Register 0 (CRC0) Format .....	180
8-5	Timer Output Control Register (TOC) Format .....	181
8-6	One-Shot Pulse Output Control Register (OSPC) Format .....	182
8-7	Basic Operation of Timer Register 0 (TM0) .....	184
8-8	TM0 Clearance by Match with Compare Register (CR01) .....	185
8-9	Clear Operation When CE0 Bit is Cleared (0) .....	186
8-10	Timer/Counter 0 External Event Count Timing .....	187
8-11	Example of the Case Where the External Event Counter Does Not Distinguish Between One Valid Edge Input and No Valid Edge Input .....	189
8-12	To Distinguish Whether One or No Valid Edge Has Been Input with External Event Counter .....	190
8-13	Compare Operation .....	191
8-14	TM0 Clearance After Match Detection .....	192
8-15	Capture Operation .....	193
8-16	Toggle Output Operation .....	196
8-17	PWM Pulse Output .....	198
8-18	Example of PWM Output Using TM0 .....	199
8-19	Example of PWM Output When CR00 = FFFFH .....	200
8-20	Example of Compare Register (CR00) Rewrite .....	201
8-21	Example of 100 % Duty With PWM Output .....	202
8-22	When Timer/Counter 0 is Stopped During PWM Signal Output .....	203
8-23	Example of PPG Output Using TM0 .....	204
8-24	Example of PPG Output When CR00 = CR01 .....	205
8-25	Example of Compare Register (CR00) Rewrite .....	206
8-26	Example of 100 % Duty With PPG Output .....	207

# LIST OF FIGURES (4/11)

Figure No.	Title	Page
8-27	Example of Extended PPG Output Cycle .....	208
8-28	When Timer/Counter 0 is Stopped During PPG Signal Output .....	209
8-29	Example of Software Triggered One-Shot Pulse Output .....	210
8-30	Interval Timer Operation (1) Timing .....	211
8-31	Control Register Settings for Interval Timer Operation (1) .....	212
8-32	Interval Timer Operation (1) Setting Procedure .....	212
8-33	Interval Timer Operation (1) Interrupt Request Processing .....	212
8-34	Interval Timer Operation (2) Timing .....	213
8-35	Control Register Settings for Interval Timer Operation (2) .....	214
8-36	Interval Timer Operation (2) Setting Procedure .....	214
8-37	Pulse Width Measurement Timing .....	215
8-38	Control Register Settings for Pulse Width Measurement .....	216
8-39	Pulse Width Measurement Setting Procedure .....	216
8-40	Interrupt Request Processing that Calculates Pulse Width .....	217
8-41	Example of Timer/Counter 0 PWM Signal Output .....	217
8-42	Control Register Settings for PWM Output Operation .....	218
8-43	PWM Output Setting Procedure .....	219
8-44	Changing PWM Output Duty .....	220
8-45	Example of Timer/Counter 0 PPG Signal Output .....	221
8-46	Control Register Settings for PPG Output Operation .....	222
8-47	PPG Output Setting Procedure .....	223
8-48	Changing PPG Output Duty .....	224
8-49	Example of Timer/Counter 0 One-Shot Pulse Output .....	225
8-50	Control Register Settings for One-Shot Pulse Output .....	226
8-51	One-Shot Pulse Output Setting Procedure .....	227
8-52	Operation When Counting Is Started .....	229
8-53	Example Where Whether One or No Valid Edge Has Been Input Cannot Be Distinguished with External Event Counter .....	230
8-54	To Distinguish Whether One or No Valid Edge Has Been Input with External Event Counter .....	231
9-1	Timer/Counter 1 Block Diagram .....	236
9-2	Timer Control Register 1 (TMC1) Format .....	239
9-3	Prescaler Mode Register 1 (PRM1) Format .....	240
9-4	Capture/Compare Control Register 1 (CRC1) Format .....	241
9-5	Basic Operation in 8-Bit Operating Mode (BW1 = 0) .....	243
9-6	Basic Operation in 16-Bit Operating Mode (BW1 = 1) .....	244
9-7	TM1 Clearance by Match With Compare Register (CR10, CR11) .....	245
9-8	TM1 Clearance after Capture Operation .....	245
9-9	Clear Operation When CE1 Bit is Cleared (0) .....	246
9-10	Timing of External Event Count by Timer/Counter 1 .....	247
9-11	Example of the Case Where the External Event Counter Does Not Distinguish Between One Valid Edge Input and No Valid Edge Input .....	249

# LIST OF FIGURES (5/11)

Figure No.	Title	Page
9-12	To Distinguish Whether One or No Valid Edge Has Been Input with External Event Counter .....	250
9-13	Compare Operation in 8-Bit Operating Mode .....	251
9-14	Compare Operation in 16-Bit Operating Mode .....	252
9-15	TM1 Clearance after Match Detection .....	252
9-16	Capture Operation in 8-Bit Operating Mode .....	253
9-17	Capture Operation in 16-Bit Operating Mode .....	254
9-18	TM1 Clearance after Capture Operation .....	255
9-19	Interval Timer Operation (1) Timing .....	256
9-20	Control Register Settings for Interval Timer Operation (1) .....	257
9-21	Interval Timer Operation (1) Setting Procedure .....	258
9-22	Interval Timer Operation (1) Interrupt Request Processing .....	258
9-23	Interval Timer Operation (2) Timing (When CR11 is Used as Compare Register) .....	259
9-24	Control Register Settings for Interval Timer Operation (2) .....	260
9-25	Interval Timer Operation (2) Setting Procedure .....	260
9-26	Pulse Width Measurement Timing (When CR11 is Used as Capture Register) .....	261
9-27	Control Register Settings for Pulse Width Measurement .....	262
9-28	Pulse Width Measurement Setting Procedure .....	263
9-29	Interrupt Request Processing that Calculates Pulse Width .....	263
9-30	Operation When Counting Is Started .....	265
9-31	Example Where Whether One or No Valid Edge Has been Input Cannot Be Distinguished with External Event Counter .....	266
9-32	To Distinguish Whether One or No Valid Edge Has Been Input with External Event Counter .....	267
10-1	Timer/Counter 2 Block Diagram .....	274
10-2	Timer Control Register 1 (TMC1) Format .....	277
10-3	Prescaler Mode Register 1 (PRM1) Format .....	278
10-4	Capture/Compare Control Register 2 (CRC2) Format .....	279
10-5	Timer Output Control Register (TOC) Format .....	280
10-6	Basic Operation in 8-Bit Operating Mode (BW2 = 0) .....	282
10-7	Basic Operation in 16-Bit Operating Mode (BW2 = 1) .....	283
10-8	TM2 Clearance by Match With Compare Register (CR20/CR21) .....	284
10-9	TM2 Clearance after Capture Operation .....	284
10-10	Clear Operation When CE2 Bit is Cleared (0) .....	285
10-11	Timer/Counter 2 External Event Count Timing .....	286
10-12	Example of the case where the external event counter does not distinguish between one valid edge input and no valid edge input .....	288
10-13	To Distinguish Whether One or No Valid Edge Has Been Input with External Event Counter .....	289
10-14	One-Shot Timer Operation .....	290
10-15	Compare Operation in 8-Bit Operating Mode .....	292
10-16	Compare Operation in 16-Bit Operating Mode .....	293

# LIST OF FIGURES (6/11)

Figure No.	Title	Page
10-17	TM2 Clearance after Match Detection .....	294
10-18	Capture Operation in 8-Bit Operating Mode .....	295
10-19	Capture Operation in 16-Bit Operating Mode .....	296
10-20	TM2 Clearance after Capture Operation .....	297
10-21	Toggle Output Operation .....	299
10-22	PWM Pulse Output (BW2 = 0) .....	302
10-23	PWM Pulse Output (BW2 = 1) .....	303
10-24	Example of PWM Output Using TM2W .....	304
10-25	Example of PWM Output When CR20W = FFFFH .....	304
10-26	Example of Compare Register (CR20W) Rewrite .....	305
10-27	Example of 100 % Duty With PWM Output .....	306
10-28	When Timer/Counter 2 is Stopped During PWM Signal Output .....	307
10-29	Example of PPG Output Using TM2 .....	309
10-30	Example of PPG Output When CR20 = CR21 .....	310
10-31	Example of Compare Register Rewrite .....	311
10-32	Example of 100 % Duty With PPG Output .....	312
10-33	Example of Extended PPG Output Cycle .....	313
10-34	When Timer/Counter 2 is Stopped During PPG Signal Output .....	314
10-35	Interval Timer Operation (1) Timing .....	315
10-36	Control Register Settings for Interval Timer Operation (1) .....	316
10-37	Interval Timer Operation (1) Setting Procedure .....	317
10-38	Interval Timer Operation (1) Interrupt Request Processing .....	317
10-39	Interval Timer Operation (2) Timing .....	318
10-40	Control Register Settings for Interval Timer Operation (2) .....	319
10-41	Interval Timer Operation (2) Setting Procedure .....	320
10-42	Pulse Width Measurement Timing .....	321
10-43	Control Register Settings for Pulse Width Measurement .....	322
10-44	Pulse Width Measurement Setting Procedure .....	323
10-45	Interrupt Request Processing that Calculates Pulse Width .....	323
10-46	Example of Timer/Counter 2 PWM Signal Output .....	324
10-47	Control Register Settings for PWM Output Operation .....	325
10-48	PWM Output Setting Procedure .....	326
10-49	Changing PWM Output Duty .....	327
10-50	Example of Timer/Counter 2 PPG Signal Output .....	328
10-51	Control Register Settings for PPG Output Operation .....	329
10-52	PPG Output Setting Procedure .....	331
10-53	Changing PPG Output Duty .....	332
10-54	External Event Counter Operation (Single Edge) .....	333
10-55	Control Register Settings for External Event Counter Operation .....	334
10-56	External Event Counter Operation Setting Procedure .....	334
10-57	One-Shot Timer Operation .....	335
10-58	Control Register Settings for One-Shot Timer Operation .....	336
10-59	One-Shot Timer Operation Setting Procedure .....	337



# LIST OF FIGURES (7/11)

Figure No.	Title	Page
10-60	One-Shot Timer Operation Start Procedure from Second Time Onward .....	337
10-61	Operation When Counting Is Started .....	339
10-62	Example Where Whether One or No Valid Edge Has been Input Cannot Be Distinguished with External Event Counter .....	340
10-63	To Distinguish Whether One or No Valid Edge Has Been Input with External Event Counter .....	341
11-1	Timer 3 Block Diagram .....	344
11-2	Timer Control Register 0 (TMC0) Format .....	346
11-3	Prescaler Mode Register 0 (PRM0) Format .....	347
11-4	Basic Operation in 8-Bit Operating Mode (BW3 = 0) .....	349
11-5	Basic Operation in 16-Bit Operating Mode (BW3 = 1) .....	350
11-6	TM3 Clearance by Match with Compare Register (CR30) .....	351
11-7	Clear Operation When CE3 Bit is Cleared (0) .....	352
11-8	Compare Operation .....	353
11-9	Interval Timer Operation Timing .....	354
11-10	Control Register Settings for Interval Timer Operation .....	355
11-11	Interval Timer Operation Setting Procedure .....	355
11-12	Operation When Count Starts .....	357
12-1	Watchdog Timer Block Diagram .....	359
12-2	Watchdog Timer Mode Register (WDM) Format .....	361
13-1	PWM Output Unit Configuration .....	365
13-2	PWM Control Register (PWMC) Format .....	367
13-3	PWM Prescaler Register (PWPR) Format .....	368
13-4	Basic PWM Output Operation .....	369
13-5	PWM Output Active Level Setting .....	370
13-6	PWM Output Timing Example 1 (PWM Pulse Width Rewrite Cycle = $2^{12}/f_{PWMC}$ ) .....	371
13-7	PWM Output Timing Example 2 (PWM Pulse Width Rewrite Cycle = $2^8/f_{PWMC}$ ) .....	372
14-1	A/D Converter Block Diagram .....	374
14-2	Example of Capacitor Connection on A/D Converter Pins .....	375
14-3	A/D Converter Mode Register (ADM) Format .....	378
14-4	Basic A/D Converter Operation .....	381
14-5	Relationship Between Analog Input Voltage and A/D Conversion Result .....	382
14-6	Select Mode Operation Timing .....	384
14-7	Scan Mode 0 Operation Timing .....	385
14-8	Scan Mode 1 Operation Timing .....	386
14-9	Software Start Select Mode A/D Conversion Operation .....	387
14-10	Software Start Scan Mode A/D Conversion Operation .....	388
14-11	Hardware Start Select Mode A/D Conversion Operation .....	390
14-12	Hardware Start Scan Mode A/D Conversion Operation .....	391
14-13	Example of Capacitor Connection on A/D Converter Pins .....	393

# **LIST OF FIGURES (8/11)**

Figure No.	Title	Page
15-1	D/A Converter Block Diagram .....	395
15-2	D/A Converter Mode Register (DAM) Format .....	397
15-3	Example of Connecting Capacitors to Reference Voltage Input Pins of D/A Converter .....	398
15-4	Example of Buffer Amp Insertion .....	399
16-1	Switching Between Asynchronous Serial Interface Mode and 3-Wire Serial I/O Mode .....	402
16-2	Asynchronous Serial Interface Block Diagram .....	404
16-3	Format of Asynchronous Serial Interface Mode Register (ASIM) and Asynchronous Serial Interface Mode Register 2 (ASIM2) .....	407
16-4	Format of Asynchronous Serial Interface Status Register (ASIS) and Asynchronous Serial Interface Status Register 2 (ASIS2) .....	409
16-5	Asynchronous Serial Interface Transmit/Receive Data Format .....	410
16-6	Asynchronous Serial Interface Transmission Completion Interrupt Timing .....	412
16-7	Asynchronous Serial Interface Reception Completion Interrupt Timing .....	413
16-8	Receive Error Timing .....	414
16-9	Example of 3-Wire Serial I/O System Configuration .....	415
16-10	3-Wire Serial I/O Mode Block Diagram .....	416
16-11	Format of Clocked Serial Interface Mode Register 1 (CSIM1) and Clocked Serial Interface Mode Register 2 (CSIM2) .....	418
16-12	3-Wire Serial I/O Mode Timing .....	419
16-13	Example of Connection to 2-Wire Serial I/O .....	420
16-14	Baud Rate Generator block Diagram .....	425
16-15	Baud Rate Generator Control Register (BRGC) Format and Baud Rate Generator Control Register 2 (BRGC2) Format .....	427
17-1	Clocked Serial Interface (CSI) Block Diagram .....	436
17-2	Clocked Serial Interface Mode Register (CSIM) Format .....	438
17-3	Serial Bus Interface Control Register (SBIC) Reading/Writing .....	439
17-4	Serial Bus Interface Control Register (SBIC) Format .....	440
17-5	Example of 3-Wire Serial I/O System Configuration .....	441
17-6	3-Wire Serial I/O Mode Timing .....	442
17-7	Example of Connection to 2-Wire Serial I/O Device .....	443
17-8	Example of SBI Serial Bus Configuration .....	449
17-9	Clocked Serial Interface (CSI) Block Diagram .....	451
17-10	Pin Configuration Diagram .....	452
17-11	Clocked Serial Interface Mode Register (CSIM) Format .....	454
17-12	SBIC Format .....	455
17-13	Configuration Around Shift Register (SIO) .....	457
17-14	SBI Transfer Timing .....	458
17-15	Bus Release Signal .....	459
17-16	Command Signal .....	459
17-17	Address .....	460
17-18	Slave Selection by Address .....	460

# LIST OF FIGURES (9/11)

Figure No.	Title	Page
17-19	Command .....	461
17-20	Data .....	461
17-21	Acknowledge Signal.....	462
17-22	Busy Signal and Ready Signal.....	463
17-23	RELT, CMDT, RELD, CMDD Operation .....	464
17-24	ACKT Operation .....	464
17-25	ACEK Operation.....	465
17-26	ACKD Operation .....	466
17-27	BSYE Operation .....	467
17-28	Address Transmission from Master Device to Slave Device .....	472
17-29	Command Transmission from Master Device to Slave Device.....	473
17-30	Data Transmission from Master Device to Slave Device.....	474
17-31	Data Transmission from Slave Device to Master Device.....	475
18-1	Clock Output Function Configuration .....	479
18-2	Clock Output Mode Register (CLOM) Format .....	481
18-3	Clock Output Operation Timing .....	482
18-4	One-Bit Output Port Operation .....	483
19-1	External Interrupt Mode Register 0 (INTM0) Format .....	486
19-2	External Interrupt Mode Register 1 (INTM1) Format .....	487
19-3	Sampling Clock Selection Register (SCS0) Format .....	488
19-4	Edge Detection for Pins P20, P25 and P26 .....	489
19-5	P21 Pin Edge Detection .....	490
19-6	Edge Detection for Pins P22 to P24 .....	491
20-1	Interrupt Control Registers (xICn) .....	503
20-2	Interrupt Mask Register (MK0, MK1L) Format .....	506
20-3	In-Service Priority Register (ISPR) Format .....	508
20-4	Interrupt Mode Control Register (IMC) Format .....	509
20-5	Watchdog Timer Mode Register (WDM) Format .....	510
20-6	Program Status Word (PSWL) Format .....	511
20-7	Context Switching Operation by Execution of a BRKCS Instruction .....	512
20-8	Return from BRKCS Instruction Software Interrupt (RETCSB Instruction Operation) .....	513
20-9	Non-Maskable Interrupt Request Acknowledgment Operations .....	515
20-10	Interrupt Acknowledgment Processing Algorithm .....	519
20-11	Context Switching Operation by Generation of an Interrupt Request .....	520
20-12	Return from Interrupt that Uses Context Switching by Means of RETCS Instruction .....	521
20-13	Examples of Processing When Another Interrupt Request is Generated During Interrupt Processing .....	523
20-14	Examples of Processing of Simultaneously Generated Interrupts .....	526
20-15	Differences in Level 3 Interrupt Acknowledgment According to IMC Register Setting .....	527
20-16	Differences between Vectored Interrupt and Macro Service Processing .....	528

# LIST OF FIGURES (10/11)

Figure No.	Title	Page
20-17	Macro Service Processing Sequence .....	531
20-18	Operation at End of Macro Service When VCIE = 0 .....	533
20-19	Operation at End of Macro Service When VCIE = 1 .....	534
20-20	Macro Service Control Word Format .....	535
20-21	Macro Service Mode Register Format .....	536
20-22	Macro Service Data Transfer Processing Flow (Type A) .....	539
20-23	Type A Macro Service Channel .....	541
20-24	Asynchronous Serial Reception .....	542
20-25	Macro Service Data Transfer Processing Flow (Type B) .....	544
20-26	Type B Macro Service Channel .....	545
20-27	Parallel Data Input Synchronized with External Interrupts .....	546
20-28	Parallel Data Input Timing .....	547
20-29	Macro Service Data Transfer Processing Flow (Type C) .....	549
20-30	Type C Macro Service Channel .....	552
20-31	Stepping Motor Open Loop Control by Real-Time Output Port .....	554
20-32	Data Transfer Control Timing .....	555
20-33	Single-Phase Excitation of 4-Phase Stepping Motor .....	557
20-34	1-2-Phase Excitation of 4-Phase Stepping Motor .....	557
20-35	Automatic Addition Control + Ring Control Block Diagram 1 (When Output Timing Varies with 1-2-Phase Excitation) .....	558
20-36	Automatic Addition Control + Ring Control Timing Diagram 1 (When Output Timing Varies with 1-2-Phase Excitation) .....	559
20-37	Automatic Addition Control + Ring Control Block Diagram 2 (1-2-Phase Excitation Constant-Velocity Operation) .....	560
20-38	Automatic Addition Control + Ring Control Timing Diagram 2 (1-2-Phase Excitation Constant-Velocity Operation) .....	561
20-39	Macro Service Data Transfer Processing Flow (Counter Mode) .....	562
20-40	Counter Mode .....	563
20-41	Counting Number of Edges .....	563
20-42	Interrupt Request Generation and Acknowledgment (Unit: Clocks = 1/f <sub>CLK</sub> ) .....	566
21-1	Memory Extension Mode Register (MM) Format .....	574
21-2	μPD784025 Memory Map .....	576
21-3	μPD784026 Memory Map .....	578
21-4	μPD784020 Memory Map .....	580
21-5	μPD784021 Memory Map .....	582
21-6	Read Timing .....	584
21-7	Write Timing .....	584
21-8	Memory Extension Mode Register (MM) Format .....	585
21-9	Programmable Wait Control Register (PWC1/PWC2) Format .....	587
21-10	Address Wait Function Read/Write Timing .....	588
21-11	Wait Control Spaces .....	592
21-12	Access Wait Function Read Timing .....	593

# LIST OF FIGURES (11/11)

Figure No.	Title	Page
21-13	Access Wait Function Write Timing.....	595
21-14	Timing with External Wait Signal.....	597
21-15	Refresh Mode Register (RFM) Format .....	599
21-16	Refresh Area Specification Register (RFA) Format .....	600
21-17	Pulse Refresh Operation in Internal Memory Access .....	601
21-18	Refresh Pulse Output Operation .....	602
21-19	Timing for Return from Self-Refresh Operation.....	603
21-20	Hold Mode Register (HLDM) Format .....	604
21-21	Hold Mode Timing .....	606
22-1	Standby Mode Transition Diagram .....	609
22-2	Standby Function Block Diagram .....	610
22-3	Standby Control Register (STBC) Format .....	612
22-4	Oscillation Stabilization Time Specification Register (OSTS) Format .....	614
22-5	Operation After HALT Mode Has Been Released .....	618
22-6	Operation After STOP Mode Has Been Released .....	627
22-7	STOP Mode Release by NMI Input.....	629
22-8	STOP Mode Release by INTP4/INTP5 Input .....	630
22-9	Operation After IDLE Mode Has Been Released .....	633
22-10	Example of Address/Data Bus Processing .....	637
23-1	Reset Signal Acknowledgment.....	641
23-2	Power-On Reset Operation .....	642
23-3	Reset Input Timing .....	645
24-1	Page Program Mode Flowchart.....	649
24-2	Page Program Mode Timing .....	650
24-3	Byte Program Mode Flowchart.....	651
24-4	Byte Program Mode Timing .....	652
24-5	PROM Read Timing .....	653
B-1	Development Tool Configuration .....	696
B-2	Dimensions of EV-9200GC-80 (reference) .....	703
B-3	Recommended Board Mounting Pattern of EV-9200GC-80 (reference) .....	704
B-4	Dimensions of T GK-080SDW (reference).....	705

# LIST OF TABLES (1/3)

Table No.	Title	Page
2-1	Port 1 Operating Modes .....	20
2-2	Port 2 Operating Modes .....	22
2-3	Port 3 Operating Modes (n = 0 to 7) .....	23
2-4	Port 6 Operating Modes .....	25
2-5	Pin Input/Output Circuit Types and Recommended Connection When Not Used .....	30
3-1	Vector Table .....	42
3-2	Internal RAM Area .....	45
3-3	Register Bank Selection .....	53
3-4	Correspondence between Function Names and Absolute Names .....	65
3-5	List of Special Function Registers (SFRs) .....	67
4-1	Time Required to Change Division Ratio .....	79
5-1	Port Functions .....	86
5-2	Number of Input/Output Ports .....	86
5-3	Port 1 Operating Modes .....	94
5-4	Method of Setting P10 & P11 PWM Signal Output Function .....	99
5-5	Port 2 Operating Modes .....	106
5-6	Port 3 Operating Modes .....	112
5-7	Port 4 Operating Modes .....	126
5-8	Port 5 Operating Modes .....	133
5-9	Port 6 Operating Modes .....	139
5-10	Port 6 Control Pin Functions and Operations .....	144
5-11	P60 to P65 Control Pin Specification .....	144
6-1	Operations When Port 0 and Port 0 Buffer Registers (P0H, P0L) are Manipulated .....	160
6-2	Real-Time Output Port Output Triggers (When P0MH = P0ML = 1) .....	162
7-1	Operations of Timer/Counters .....	169
8-1	Timer/Counter 0 Interval Time .....	171
8-2	Timer/Counter 0 Programmable Square-Wave Output Setting Range .....	172
8-3	Timer/Counter 0 Pulse Width Measurement Range .....	173
8-4	Timer/Counter 0 Pulse Width Measurement Time .....	174
8-5	Timer Output (TO0/TO1) Operations .....	195
8-6	TO0, TO1 Toggle Output (f <sub>xx</sub> = 25 MHz) .....	197
8-7	TO0, TO1 PWM Cycle (f <sub>xx</sub> = 25 MHz) .....	199
8-8	TO0 PPG Output (f <sub>xx</sub> = 25 MHz) .....	205
9-1	Timer/Counter 1 Intervals .....	233
9-2	Timer/Counter 1 Pulse Width Measurement Range .....	234

# LIST OF TABLES (2/3)

Table No.	Title	Page
9-3	Timer/Counter 1 Pulse Width Measurement Time .....	235
9-4	Maximum Input Frequency and Minimum Input Pulse Width That Can be Counted as Events .....	247
10-1	Timer/Counter 2 Intervals .....	270
10-2	Timer/Counter 2 Programmable Square-Wave Output Setting Range .....	271
10-3	Timer/Counter 2 Pulse Width Measurement Range .....	272
10-4	Clocks Enabled to be Input to Timer/Counter 2 .....	272
10-5	Timer Output (TO2/TO3) Operations .....	298
10-6	TO2/TO3 Toggle Output ( $f_{xx} = 25 \text{ MHz}$ ) .....	300
10-7	TO2/TO3 PWM Cycle ( $f_{xx} = 25 \text{ MHz}$ , $BW2 = 0$ ) .....	302
10-8	TO2/TO3 PWM Cycle ( $f_{xx} = 25 \text{ MHz}$ , $BW2 = 1$ ) .....	303
10-9	TO2 PPG Output ( $f_{xx} = 25 \text{ MHz}$ ) .....	309
11-1	Timer 3 Intervals .....	343
14-1	A/D Conversion Time .....	383
16-1	Differences Between UART/IOE1 and UART2/IOE2 Names .....	401
16-2	Receive Error Causes .....	414
16-3	Baud Rate Setting Methods .....	430
16-4	Examples of BRGC Settings When Baud Rate Generator is Used .....	431
16-5	Examples of Settings When External Baud Rate Input (ASCK) is Used .....	432
17-1	Signals in SBI Mode .....	468
17-2	BUSY Release Conditions .....	476
19-1	Pins P20 to P26 and Use of Detected Edge .....	485
20-1	Interrupt Request Service Modes .....	493
20-2	Interrupt Request Sources .....	494
20-3	Control Registers .....	498
20-4	Interrupt Control Register Flags Corresponding to Interrupt Requests .....	500
20-5	Nesting Interrupt .....	522
20-6	Interrupts for Which Macro Service Can be Used .....	529
20-7	Interrupt Acknowledge Processing Time .....	567
20-8	Macro Service Processing Time .....	568
21-1	System Clock Frequency and Refresh Pulse Output Cycle When Pseudo-Static RAM is Used .....	600
22-1	Operating States in HALT Mode .....	615
22-2	HALT Mode Release and Operations after Release .....	617

# LIST OF TABLES (3/3)

Table No.	Title	Page
22-3	HALT Mode Release by Maskable Interrupt Request .....	623
22-4	Operating States in STOP Mode .....	624
22-5	STOP Mode Release and Operations after Release .....	626
22-6	Operating States in IDLE Mode .....	631
22-7	DLE Mode Release and Operations after Release .....	632
23-1	Pin Statuses During Reset Input and After Reset Release .....	642
23-2	Hardware States After Reset .....	643
24-1	PROM Programming Operating Modes .....	647
25-1	List of Instructions by 8-Bit Addressing .....	686
25-2	List of Instructions by 16-Bit Addressing .....	687
25-3	List of Instructions by 24-Bit Addressing .....	688
25-4	List of Instructions by Bit Manipulation Instruction Addressing .....	688
25-5	List of Instructions by Call/Return Instruction / Branch Instruction Addressing .....	689
A-1	Differences Between $\mu$ PD784026 Subseries and $\mu$ PD78234 Subseries .....	691



## CHAPTER 1 GENERAL

The  $\mu$ PD784026 subseries comprises 78K/IV series products that can perform input/output directly with analog signals. The 78K/IV series comprises 16-bit single-chip microcontrollers equipped with a high-performance CPU that has a function such as accessing a 1M-byte memory space. The  $\mu$ PD784026 subseries is upward-compatible with the 78K/II series, and has pin compatibility with  $\mu$ PD78234 subseries of the 78K/II series.

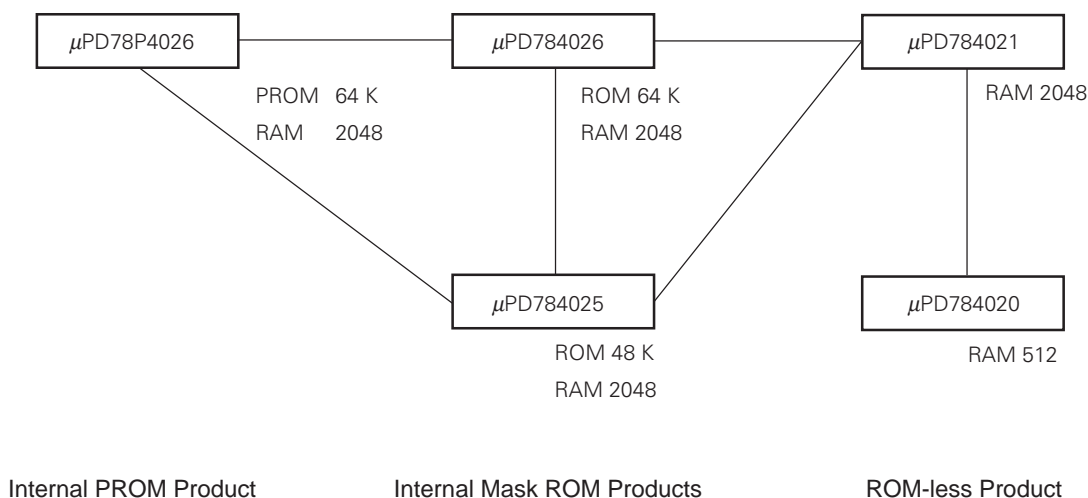
The  $\mu$ PD784026 incorporates 64K-byte mask ROM and 2048-byte RAM, plus high-performance timer/counters, an 8-bit A/D converter, 8-bit D/A converter, PWM output function, two independent serial interface channels, etc.

The  $\mu$ PD784020 is a version of the  $\mu$ PD784026 with no mask ROM, and with 512-byte RAM.

The  $\mu$ PD784021 is a version of the  $\mu$ PD784026 with no mask ROM.

The  $\mu$ PD784025 is a version of the  $\mu$ PD784026 with 48K-byte mask ROM.

The  $\mu$ PD78P4026 has PROM instead of the mask ROM of the  $\mu$ PD784026.



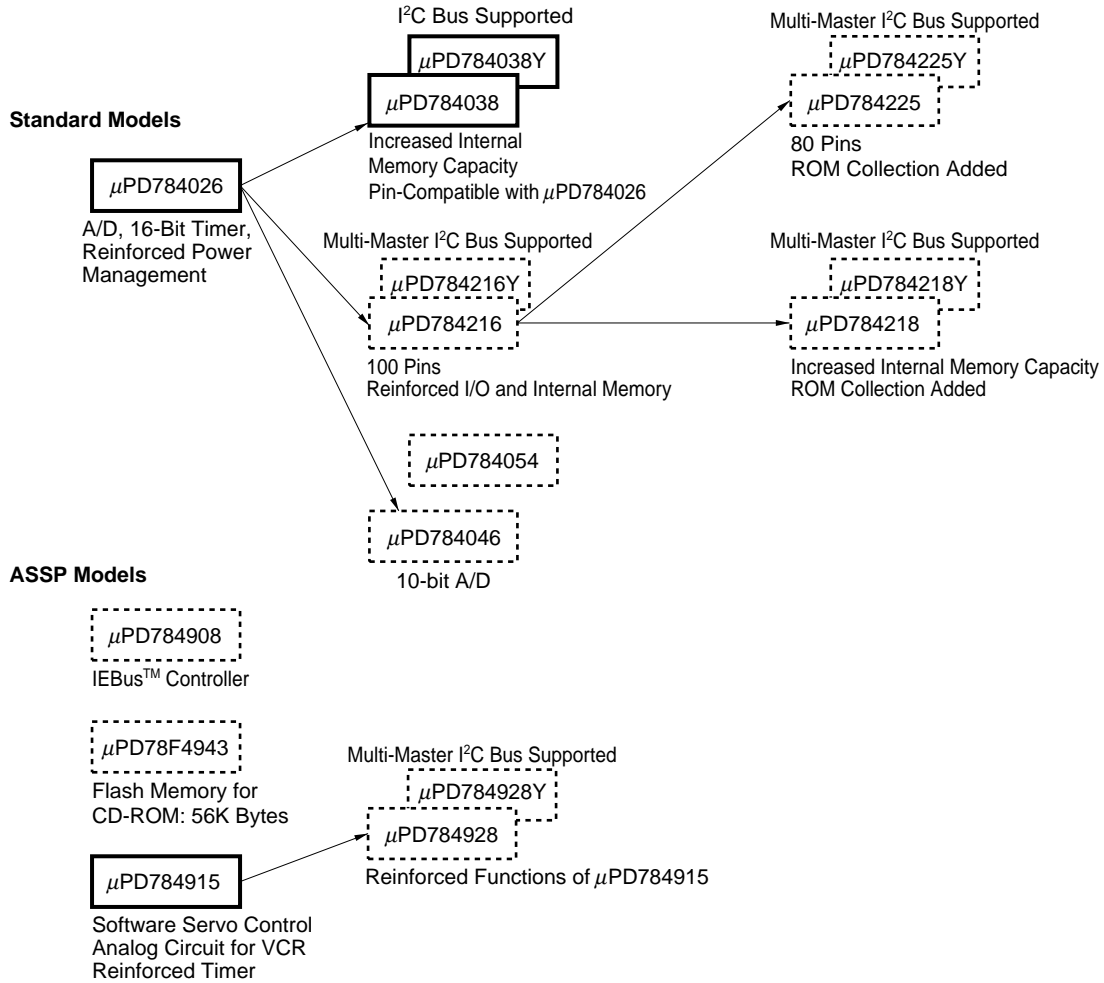
These products can be used in the following application areas:

- LBPs
- Auto-focus cameras
- PPCs
- Printers
- Electric typewriters
- Air conditioners
- Electronic musical instruments
- Cellular phones, etc

## ★ 78K/IV SERIES PRODUCT DEVELOPMENT DIAGRAM

: Under Mass Production

: Under Development



## 1.1 FEATURES

- 78K/IV series
- Pin-compatible with  $\mu$ PD78234 subseries
- High-speed instruction execution
  - Minimum instruction execution time (25 MHz operation): 160 ns/320 ns/640 ns/1280 ns
- Instruction set suitable for control applications
- Data memory extension function (1 M-byte memory space: 2 bank specification pointers)
- Interrupt controller (4-level priority system)
  - Vectored interrupt service/macro service/context switching
- Standby functions: HALT/STOP/IDLE modes
- Internal memory:
  - ROM
    - Mask ROM : 64 K bytes ( $\mu$ PD784026)
    - 48 K bytes ( $\mu$ PD784025)
    - Not incorporated ( $\mu$ PD784020, 784021)
  - PROM : 64 K bytes ( $\mu$ PD78P4026)
  - RAM : 2048 bytes ( $\mu$ PD784021, 784025, 784026, 78P4026)
  - 512 bytes ( $\mu$ PD784020)
- I/O pins : 64 ( $\mu$ PD784025, 784026, 78P4026)
- 46 ( $\mu$ PD784020, 784021)
  - Software programmable pull-up : 54 inputs ( $\mu$ PD784025, 784026, 78P4026)
  - 34 inputs ( $\mu$ PD784020, 784021)
  - Direct LED drive capability : 24 outputs ( $\mu$ PD784025, 784026, 78P4026)
  - 8 outputs ( $\mu$ PD784020, 784021)
  - Direct transistor drive capability : 8 outputs
- Serial interface
  - UART/IOE (3-wire serial I/O): 2 channels (with internal baud rate generator)
  - CSI (3-wire serial I/O, serial bus interface): 1 channel
- Real-time output ports (combination with timer/counter allows independent control of 2-system stepping motors)
- A/D converter (8-bit resolution  $\times$  8 channels)
- D/A converter (8-bit resolution  $\times$  2 channels)
- PWM outputs (12-bit resolution  $\times$  2 channels)
- High-performance timer/counter
  - Timer/counter (16 bits)  $\times$  3 units
  - Timer (16 bits)  $\times$  1 unit
- Watchdog timer: 1 channel
- Clock output function: fCLK, fCLK/2, fCLK/4, fCLK/8, fCLK/16 can be selected (except for  $\mu$ PD784020, 784021)

## 1.2 ORDERING INFORMATION AND QUALITY GRADES

### 1.2.1 Ordering Information

Part Number	Package	Internal ROM
$\mu$ PD784020GC-3B9	80-pin plastic QFP (14 × 14 mm)	None
$\mu$ PD784021GC-3B9	80-pin plastic QFP (14 × 14 mm)	None
$\mu$ PD784021GK-BE9	80-pin plastic TQFP (fine pitch) (12 × 12 mm)	None
$\mu$ PD784025GC-xxx-3B9	80-pin plastic QFP (14 × 14 mm)	Mask ROM
$\mu$ PD784026GC-xxx-3B9	80-pin plastic QFP (14 × 14 mm)	Mask ROM
$\mu$ PD78P4026GC-3B9	80-pin plastic QFP (14 × 14 mm)	One-time PROM
$\mu$ PD78P4026GC-xxx-3B9*	80-pin plastic QFP (14 × 14 mm)	One-time PROM (QTOP™ microcontroller)
$\mu$ PD78P4026KK-T	80-pin ceramic WQFN (14 × 14 mm)	EPROM

\* This is a QTOP microcontroller, which is a microcontroller with one-time PROM totally supported by NEC's writing service (writing, marking, screening, and inspection).

**Remark** xxx indicates a ROM code suffix.

### 1.2.2 Quality Grades

Part Number	Package	Quality Grade
$\mu$ PD784020GC-3B9	80-pin plastic QFP (14 × 14 mm)	Standard
$\mu$ PD784021GC-3B9	80-pin plastic QFP (14 × 14 mm)	Standard
$\mu$ PD784021GK-BE9	80-pin plastic TQFP (fine pitch) (12 × 12 mm)	Standard
$\mu$ PD784025GC-xxx-3B9	80-pin plastic QFP (14 × 14 mm)	Standard
$\mu$ PD784026GC-xxx-3B9	80-pin plastic QFP (14 × 14 mm)	Standard
$\mu$ PD78P4026GC-3B9	80-pin plastic QFP (14 × 14 mm)	Standard
$\mu$ PD78P4026GC-xxx-3B9*	80-pin plastic QFP (14 × 14 mm)	Standard
$\mu$ PD78P4026KK-T	80-pin ceramic WQFN (14 × 14 mm)	Not applied (for function evaluation)

\* This is a QTOP microcontroller, which is a microcontroller with one-time PROM totally supported by NEC's writing service (writing, marking, screening, and inspection).

Refer to **Quality Grades on NEC Semiconductor Devices (Document No. C11531E)** published by NEC for details of quality grade on devices and its application areas.

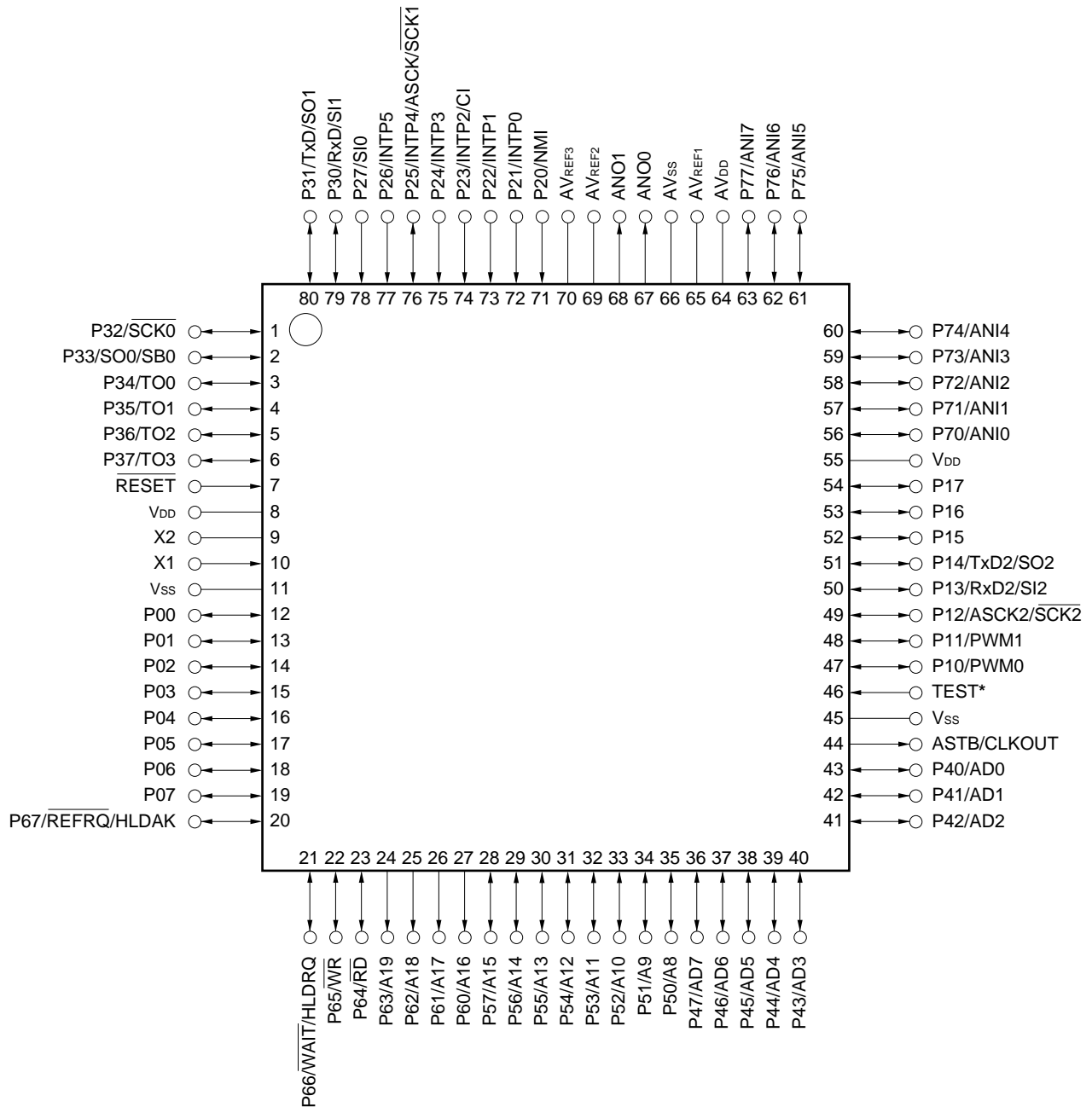
**Caution** The EPROM version of the  $\mu$ PD78P4026 does not provide a level of reliability intended for mass production of user systems, and should only be used for experimental or preproduction function evaluation.

**Remark** xxx indicates a ROM code suffix.

## 1.3 PIN CONFIGURATION (TOP VIEW)

## 1.3.1 Normal Operating Mode

- 80-pin plastic QFP (14 × 14 mm)  
 $\mu$ PD784020GC-3B9, 784021GC-3B9, 784025GC-xxx-3B9,  
 $\mu$ PD784026GC-xxx-3B9, 78P4026GC-3B9, 78P4026GC-xxx-3B9
- 80-pin plastic TQFP (fine pitch) (12 × 12 mm)  
 $\mu$ PD784021GK-BE9
- 80-pin ceramic WQFN (14 × 14 mm)  
 $\mu$ PD78P4026KK-T

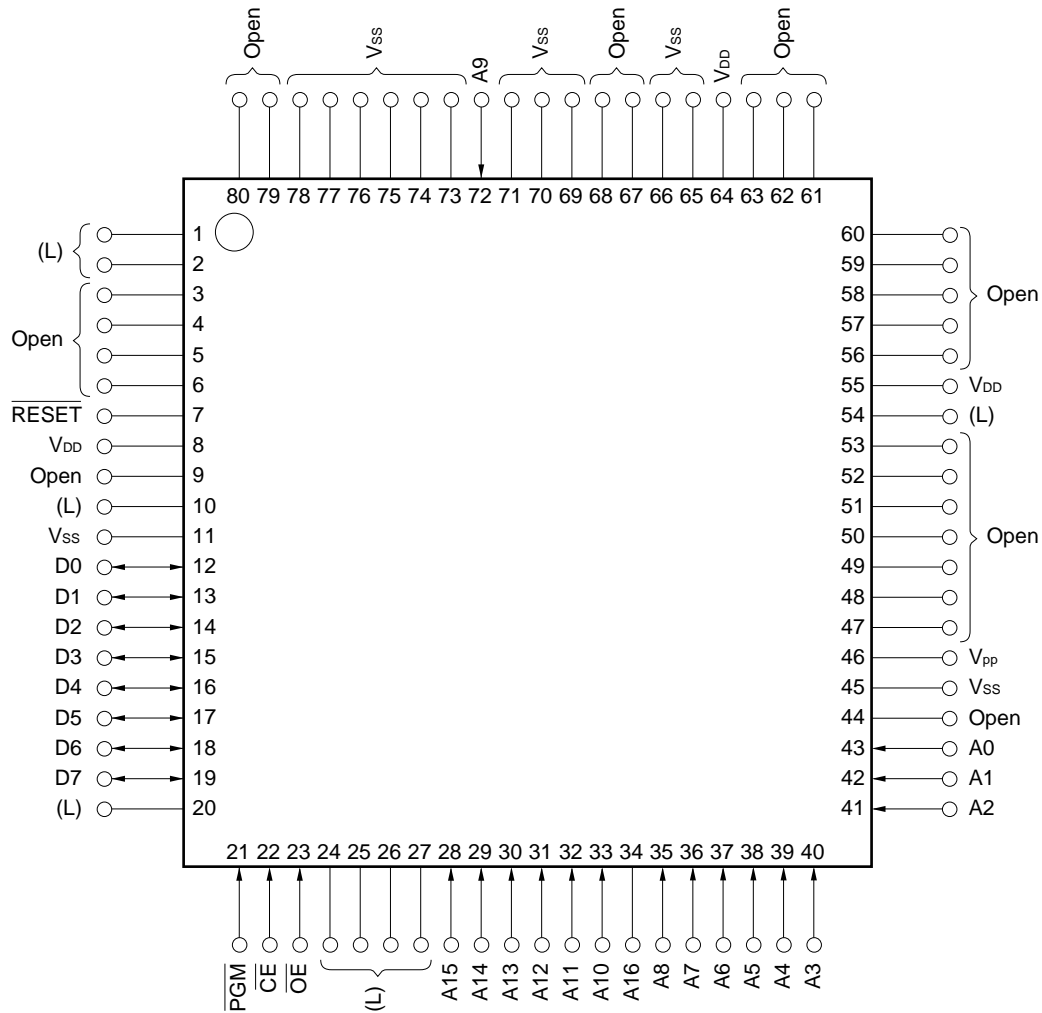


\* Connect the TEST pin directly to Vss.

P00-P07	: Port0	AD0-AD7	: Address/Data Bus
P10-P17	: Port1	A8-A19	: Address Bus
P20-P27	: Port2	$\overline{\text{RD}}$	: Read Strobe
P30-P37	: Port3	$\overline{\text{WR}}$	: Write Strobe
P40-P47	: Port4	$\overline{\text{WAIT}}$	: Wait
P50-P57	: Port5	HLDREQ	: Hold Request
P60-P67	: Port6	HLDACK	: Hold Acknowledge
P70-P77	: Port7	CLKOUT	: Clock Out
TO0-TO3	: Timer Output	ASTB	: Address Strobe
CI	: Clock Input	$\overline{\text{REFRQ}}$	: Refresh Request
RxD, RxD2	: Receive Data	$\overline{\text{RESET}}$	: Reset
TxD, TxD2	: Transmit Data	X1, X2	: Crystal
$\overline{\text{SCK0-SCK2}}$	: Serial Clock	ANI0-ANI7	: Analog Input
ASCK, ASCK2	: Asynchronous Serial Clock	ANO0, ANO1	: Analog Output
SB0	: Serial Bus	AVREF1-AVREF3	: Reference Voltage
SI0-SI2	: Serial Input	AVDD	: Analog Power Supply
SO0-SO2	: Serial Output	AVss	: Analog Ground
PWM0, PWM1	: Pulse Width Modulation Output	VDD	: Power Supply
NMI	: Non-maskable Interrupt	Vss	: Ground
INTP0-INTP5	: Interrupt from Peripherals	TEST	: Test

**1.3.2 PROM Programming Mode ( $V_{PP} \geq +5\text{ V}/+12.5\text{ V}$ ,  $\overline{\text{RESET}} = \text{L}$ )**

- 80-pin plastic QFP ( $14 \times 14\text{ mm}$ )  
 $\mu\text{PD78P4026GC-3B9}$ ,  $\mu\text{PD78P4026GC-}\times\times\times\text{-3B9}$
- 80-pin ceramic WQFN ( $14 \times 14\text{ mm}$ )  
 $\mu\text{PD78P4026KK-T}$



**Caution L** : Connect to  $V_{ss}$  individually via a  $10\text{ k}\Omega$  pull-down resistor.

**$V_{ss}$**  : Connect to ground.

**Open** : Do not make any connection.

**$\overline{\text{RESET}}$**  : Drive low.

$V_{PP}$  : Programming Power Supply

$\overline{\text{RESET}}$  : Reset

A0-A16 : Address Bus

D0-D7 : Data Bus

$\overline{\text{CE}}$  : Chip Enable

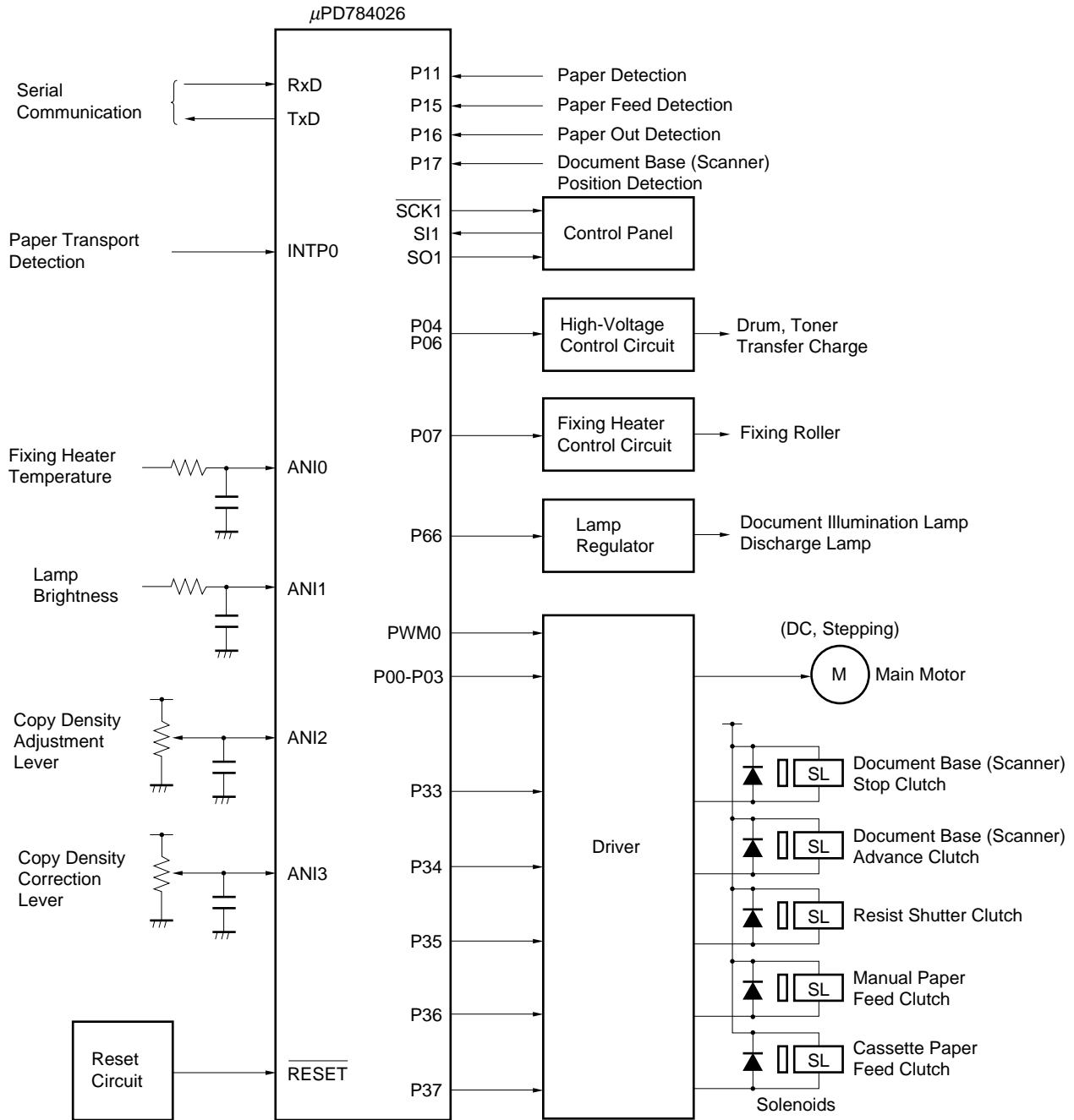
$\overline{\text{PGM}}$  : Program

$\overline{\text{OE}}$  : Output Enable

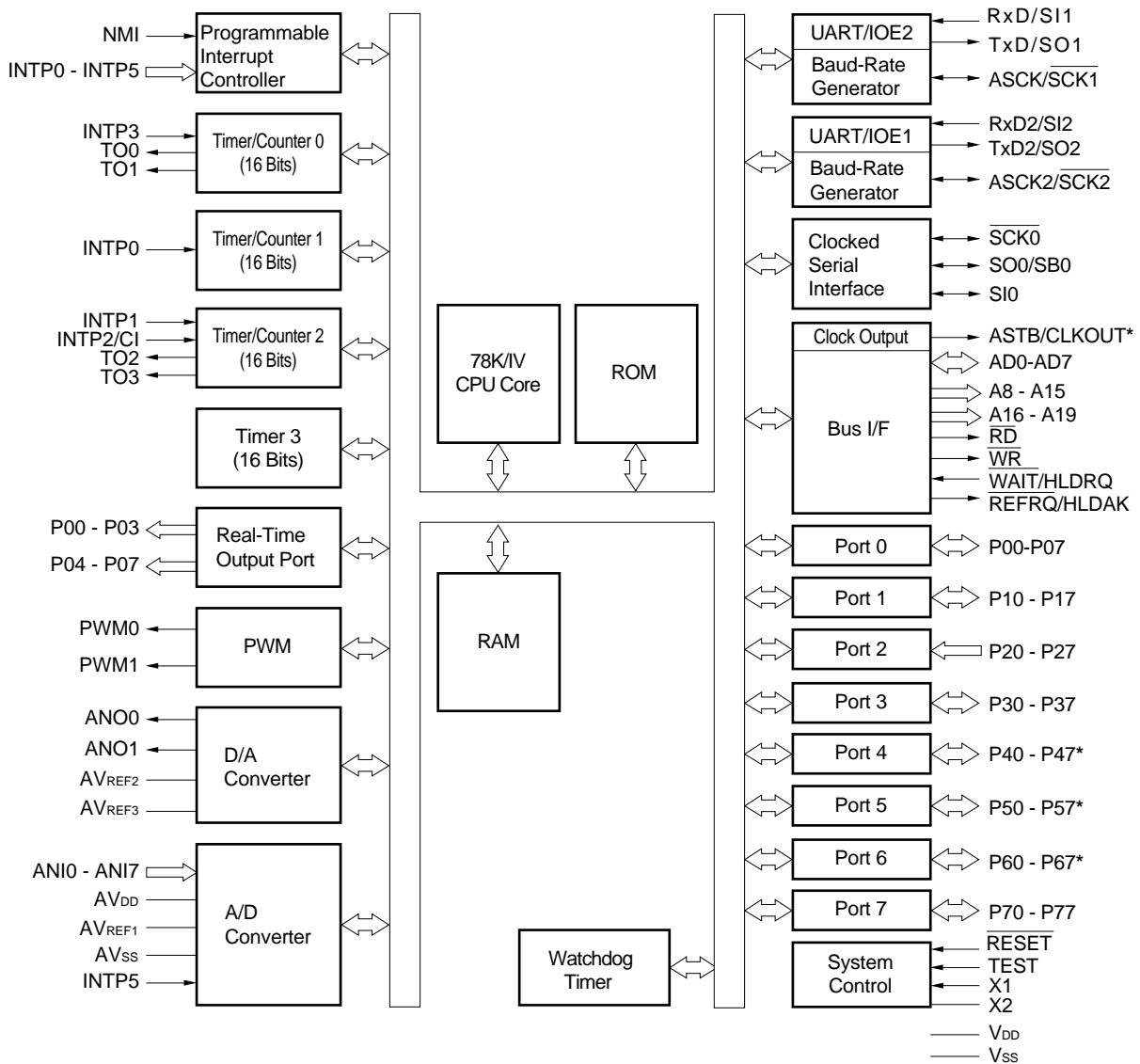
$V_{DD}$  : Power Supply

$V_{ss}$  : Ground

## 1.4 APPLICATION SYSTEM CONFIGURATION EXAMPLE (PPC)





**1.5 BLOCK DIAGRAM**

\* CLKOUT, P40 through P47, P50 through P57, P64, and P65 cannot be used with the  $\mu$ PD784020 and 784021.

**Remark** The capacities of the internal ROM and RAM differ depending on the product.

## 1.6 LIST OF FUNCTIONS

(1/2)

Part Number		$\mu$ PD784020	$\mu$ PD784021	$\mu$ PD784025	$\mu$ PD784026	$\mu$ PD78P4026
Item						
Number of basic instructions (mnemonics)		113				
General-purpose register		8 bits $\times$ 16 registers $\times$ 8 banks, or 16 bits $\times$ 8 registers $\times$ 8 banks (memory mapping)				
Minimum instruction execution time		160 ns/320 ns/640 ns/1280 ns (at 25 MHz operation)				
Internal memory	ROM	None		48 KB (mask ROM)	64 KB (mask ROM)	64 KB (one-time PROM or EPROM)
	RAM	512 B	2048 B			
Memory space		1 MB with program and data memories combined				
I/O port	Total	46 lines		64 lines		
	Input	8 lines		8 lines		
	I/O	34 lines		56 lines		
	Output	4 lines		0 line		
Pins with ancillary functions*	Pin with pull-up resistor	32 pins		54 pins		
	LED direct drive output	8 pins		24 pins		
	Transistor direct drive	8 pins				
Real-time output port		4 bits $\times$ 2, or 8 bits $\times$ 1				
Timer/counter		Timer/counter 0 (16 bits) : Timer register $\times$ 1 Capture register $\times$ 1 Compare register $\times$ 2			Pulse output <ul style="list-style-type: none"><li>• Toggle output</li><li>• PWM/PPG output</li><li>• One-shot pulse output</li></ul>	
		Timer/counter 1 (8/16 bits): Timer register $\times$ 1 Capture register $\times$ 1 Capture/compare register $\times$ 1 Compare register $\times$ 1			Pulse output <ul style="list-style-type: none"><li>• Real-time output (4 bits <math>\times</math> 2)</li></ul>	
		Timer/counter 2 (8/16 bits): Timer register $\times$ 1 Capture register $\times$ 1 Capture/compare register $\times$ 1 Compare register $\times$ 1			Pulse output <ul style="list-style-type: none"><li>• Toggle output</li><li>• PWM/PPG output</li></ul>	
		Timer 3 (8/16 bits) : Timer register $\times$ 1 Compare register $\times$ 1				
PWM output		12-bit resolution $\times$ 2 channels				
Serial interface		UART/IOE (3-wire serial I/O): 2 channels (with baud rate generator) CSI (3-wire serial I/O, SBI) : 1 channel				
A/D converter		8-bit resolution $\times$ 8 channels				
D/A converter		8-bit resolution $\times$ 2 channels				

\* The pins with ancillary functions are included in the I/O pins.

(2/2)

Part Number		$\mu$ PD784020	$\mu$ PD784021	$\mu$ PD784025	$\mu$ PD784026	$\mu$ PD78P4026
Item						
Clock output		—			Selectable from f <sub>CLK</sub> , f <sub>CLK</sub> /2, f <sub>CLK</sub> /4, f <sub>CLK</sub> /8, f <sub>CLK</sub> /16 (also can be used as 1-bit output port)	
Watchdog timer		1 channel				
Standby		HALT/STOP/IDLE mode				
Interrupt	Hardware source	23 (internal: 16, external: 7 (variable sampling clock input: 1))				
	Software source	BRK instruction, BRKCS instruction, operand error				
	Non-maskable	Internal: 1, external : 1				
	Maskable	Internal: 15, external: 6				
		<ul style="list-style-type: none"><li>• 4 priority levels</li><li>• Three processing formats: vectored interrupt/macro service/context switching</li></ul>				
Supply voltage		V <sub>DD</sub> = 2.7 to 5.5 V				
Package		80-pin plastic QFP (14 × 14 mm) 80-pin plastic TQFP (fine pitch) (12 × 12 mm): $\mu$ PD784021 only 80-pin ceramic WQFN (14 × 14 mm) : $\mu$ PD78P4026 only				

**1.7 DIFFERENCES AMONG  $\mu$ PD784026 SERIES PRODUCTS****1.7.1 Functional Differences**

Item \ Part Number	$\mu$ PD784020	$\mu$ PD784021	$\mu$ PD784025	$\mu$ PD784026	$\mu$ PD78P4026
Internal ROM	None		48 K bytes (Mask ROM)	64 K bytes (Mask ROM)	64 K bytes (One-time PROM or EPROM)
Internal RAM	512 bytes	2048 bytes			
P40 to P47	Operate as address/data bus only		Switchable between general-purpose port and address/ data bus by software		
P50 to P57	Operate as address bus only				
P60 to P63	Switchable between output-only port and address bus in 2-bit units by software				
P64, P65	Operate as $\overline{\text{RD}}$ and $\overline{\text{WR}}$ pins only		Operate as $\overline{\text{RD}}$ and $\overline{\text{WR}}$ pins when local bus interface is used, and as general-purpose ports when not used		
ASTB/CLKOUT	Operates as ASTB pin only		Switchable by software		

**1.7.2 Main Differences between  $\mu$ PD784026 Subseries and  $\mu$ PD78234 Subseries**

Series Name		$\mu$ PD784026 Subseries	$\mu$ PD78234 Subseries
Item			
Number of basic instructions (mnemonics)		113	65
Minimum instruction execution time		160 ns (25 MHz operation)	333 ns (12 MHz operation)
Memory space (program/data)		1 M bytes in total	64 K bytes/1 M bytes
Timer/counter		<ul style="list-style-type: none"> <li>• 16-bit timer/counter <math>\times</math> 1</li> <li>• 8/16-bit timer/counter <math>\times</math> 2</li> <li>• 8/16-bit timer <math>\times</math> 1</li> </ul>	<ul style="list-style-type: none"> <li>• 16-bit timer/counter <math>\times</math> 1</li> <li>• 8-bit timer/counters <math>\times</math> 2</li> <li>• 8-bit timer <math>\times</math> 1</li> </ul>
Clock output function		Provided	Not provided
Watchdog timer		Provided	Not provided
Serial interface		<ul style="list-style-type: none"> <li>• UART/IOE <math>\times</math> 2 channels</li> <li>• CSI (IOE/SBI) <math>\times</math> 1 channel</li> </ul>	<ul style="list-style-type: none"> <li>• UART <math>\times</math> 1 channel</li> <li>• CSI (IOE/SBI) <math>\times</math> 1 channel</li> </ul>
Interrupts	Context switching	Provided	Not provided
	Priority	4 levels	2 levels
	Counter mode	Provided	Not provided
Standby functions		3 modes: HALT/STOP/IDLE	2 modes: HALT/STOP
Operating clock switching		$f_{xx}/2$ , $f_{xx}/4$ , $f_{xx}/8$ , $f_{xx}/16$ can be selected. (except for $\mu$ PD784020, 784021)	$f_{xx}/2$ only
Pin functions	MODE pin	None	ROM-less mode specified Always high in $\mu$ PD78233/78237
	TEST pin	Device test pin	None Always low in normal operation
Packages		<ul style="list-style-type: none"> <li>• 80-pin plastic QFP (14 <math>\times</math> 14 mm)</li> <li>• 80-pin plastic TQFP (fine-pitch) (12 <math>\times</math> 12 mm): <math>\mu</math>PD784021 only</li> <li>• 80-pin ceramic WQFN (14 <math>\times</math> 14 mm) : <math>\mu</math>PD78P4026 only</li> </ul>	<ul style="list-style-type: none"> <li>• 80-pin plastic QFP (14 <math>\times</math> 14 mm)</li> <li>• 94-pin plastic QFP (20 <math>\times</math> 20 mm)</li> <li>• 84-pin plastic QFJ (1150 <math>\times</math> 1150 mil)</li> <li>• 94-pin ceramic WQFN (14 <math>\times</math> 14 mm) : <math>\mu</math>PD78P238 only</li> </ul>

**Remark** See **APPENDIX A DIFFERENCES BETWEEN  $\mu$ PD784026 SUBSERIES AND  $\mu$ PD78234 SUBSERIES** for details.

[MEMO]

## CHAPTER 2 PIN FUNCTIONS

### 2.1 PIN FUNCTION TABLES

#### 2.1.1 Normal Operating Mode

##### (1) Port pins (1/2)

Pin Name	Input/Output	Dual-Function Pins	Functions
P00 to P07	Input/output	—	Port 0 (P0): <ul style="list-style-type: none"> <li>8-bit input/output port</li> <li>Can be used as real-time output ports (4 bits × 2)</li> <li>Input/output specifiable bit-wise</li> <li>For input mode pins, internal pull-up resistor connection can be specified at once in software</li> <li>Transistor drive capability</li> </ul>
P10	Input/output	PWM0	Port 1 (P1): <ul style="list-style-type: none"> <li>8-bit input/output port</li> <li>Input/output specifiable bit-wise.</li> <li>For input mode pins, internal pull-up resistor connection can be specified at once in software</li> <li>LED drive capability</li> </ul>
P11		PWM1	
P12		ASCK2/SCK2	
P13		RxD2/SI2	
P14		TxD2/SO2	
P15 to P17		—	
P20	Input	NMI	Port 2 (P2): <ul style="list-style-type: none"> <li>8-bit input/output port</li> <li>P20 cannot be used as a general-purpose port (non-maskable interrupt) Input level can be confirmed in the interrupt routine.</li> <li>For P22 to P27, internal pull-up resistor connection can be specified in software in 6-bit units</li> <li>The P25/INTP4/ASCK/SCK1 pin operates as the SCK1 output pin in accordance with the CSIM1 register specification</li> </ul>
P21		INTP0	
P22		INTP1	
P23		INTP2/CI	
P24		INTP3	
P25		INTP4/ASCK/SCK1	
P26		INTP5	
P27		SI0	
P30	Input/output	RxD/SI1	Port 3 (P3): <ul style="list-style-type: none"> <li>8-bit input/output port</li> <li>Input/output specifiable bit-wise</li> <li>For input mode pins, internal pull-up resistor connection can be specified at once in software</li> </ul>
P31		TxD/SO1	
P32		SCK0	
P33		SO0/SB0	
P34 to P37		TO0 to TO3	

**(1) Port pins (2/2)**

Pin Name	Input/Output	Dual-Function Pins	Functions
P40 to P47 <sup>*1</sup>	Input/output	AD0 to AD7	Port 4 (P4): <ul style="list-style-type: none"> <li>8-bit input/output port</li> <li>Input/output specifiable bit-wise</li> <li>For input mode pins, internal pull-up resistor connection can be specified at once in software</li> <li>LED drive capability</li> </ul>
P50 to P57 <sup>*1</sup>	Input/output	A8 to A15	Port 5 (P5): <ul style="list-style-type: none"> <li>8-bit input/output port</li> <li>Input/output specifiable bit-wise</li> <li>For input mode pins, internal pull-up resistor connection can be specified at once in software</li> <li>LED drive capability</li> </ul>
P60 to P63 <sup>*2</sup>	Input/output	A16 to A19	Port 6 (P6): <ul style="list-style-type: none"> <li>8-bit input/output port</li> <li>Input/output specifiable bit-wise</li> <li>For input mode pins, internal pull-up resistor connection can be specified at once in software</li> </ul>
P64 <sup>*1</sup>		$\overline{RD}$	
P65 <sup>*1</sup>		$\overline{WR}$	
P66		$\overline{WAIT}/HLDRQ$	
P67		$\overline{REFRQ}/HLDAK$	
P70 to P77	Input/output	ANI0 to ANI7	Port 7 (P7): <ul style="list-style-type: none"> <li>8-bit input/output port</li> <li>Input/output specifiable bit-wise</li> </ul>

\* 1. Does not operate as a port in the  $\mu$ PD784020 and 784021.

2. An output-only port in the  $\mu$ PD784020 and 784021.



## (2) Pins other than port pins (1/2)

Pin Name	Input/Output	Dual-Function Pins	Functions	
TO0/TO3	Output	P34 to P37	Timer output	
CI	Input	P23/INTP2	Count clock input to timer/counter 2	
RxD	Input	P30/SI1	Serial data input (UART0)	
RxD2		P13/SI2	Serial data input (UART2)	
TxD	Output	P31/SO1	Serial data output (UART0)	
TxD2		P14/SO2	Serial data output (UART2)	
ASCK	Input	P25/INTP4/SCK1	Baud rate clock input (UART0)	
ASCK2		P12/SCK2	Baud rate clock input (UART2)	
SB0	Input/output	P33/SO0	Serial data input/output (SBI)	
SI0	Input	P27	Serial data input (3-wire serial I/O0)	
SI1		P30/RxD	Serial data input (3-wire serial I/O1)	
SI2		P13/RxD2	Serial data input (3-wire serial I/O2)	
SO0	Output	P33/SB0	Serial data output (3-wire serial I/O0)	
SO1		P31/TxD	Serial data output (3-wire serial I/O1)	
SO2		P14/TxD2	Serial data output (3-wire serial I/O2)	
SCK0	Input/output	P32	Serial clock input/output (SBI, 3-wire serial I/O0)	
SCK1		P25/INTP4/ASCK	Serial clock input/output (3-wire serial I/O1)	
SCK2		P12/ASCK2	Serial clock input/output (3-wire serial I/O2)	
NMI	Input	P20	External interrupt requests	—
INTP0		P21		<ul style="list-style-type: none"> <li>Count clock input to timer/counter 1</li> <li>CR11 or CR12 capture trigger signal</li> </ul>
INTP1		P22		<ul style="list-style-type: none"> <li>Count clock input to timer/counter 2</li> <li>CR22 capture trigger signal</li> </ul>
INTP2		P23/CI		<ul style="list-style-type: none"> <li>Count clock input to timer/counter 2</li> <li>CR21 capture trigger signal</li> </ul>
INTP3		P24		<ul style="list-style-type: none"> <li>Count clock input to timer/counter 0</li> <li>CR02 capture trigger signal</li> </ul>
INTP4		P25/ASCK/SCK1		—
INTP5		P26		A/D converter conversion start trigger input
AD0 to AD7	Input/output	P40 to P47	Time division address/data bus (external memory connection)	
A8 to A15	Output	P50 to P57	Upper address bus (external memory connection)	
A16 to A19	Output	P60 to P63	Upper address with address extension (external memory connection)	
RD	Output	P64	External memory read strobe	
WR	Output	P65	External memory write strobe	
WAIT	Input	P66/HLDRQ	Wait insertion	
REFRQ	Output	P67/HLDAK	External pseudo-static memory refresh pulse output	
HLDRQ	Input	P66/WAIT	Bus hold request input	
HLDAK	Output	P67/REFRQ	Bus hold response output	

**(2) Pins other than port pins (2/2)**

Pin Name	Input/Output	Dual-Function Pins	Functions
ASTB	Output	CLKOUT	Time division address (A0 to A7) latch timing output (during external memory access)
CLKOUT	Output	ASTB	Clock output
$\overline{\text{RESET}}$	Input	—	Chip reset
X1	Input	—	System clock oscillation crystal connections (clock can also be input to X1)
X2	—		
ANI0-ANI7	Input	P70 to P77	A/D converter analog voltage inputs
ANO0, ANO1	Output	—	D/A converter analog voltage outputs
AV <sub>REF1</sub>	—	—	A/D converter reference voltage application
AV <sub>REF2</sub> , AV <sub>REF3</sub>			D/A converter reference voltage application
AV <sub>DD</sub>			A/D converter positive power supply
AV <sub>SS</sub>			A/D converter GND
V <sub>DD</sub>			Positive power supply
V <sub>SS</sub>			GND
TEST			Connect directly to V <sub>SS</sub> (IC test pin).

**2.1.2 PROM Programming Mode ( $\mu$ PD78P4026 Only:  $V_{PP} \geq +5\text{ V}/+12.5\text{ V}$ ,  $\overline{\text{RESET}} = \text{L}$ )**

Pin Name	Input/Output	Functions
$V_{PP}$	Input	PROM programming mode setting High-voltage application pin in program write/verify
$\overline{\text{RESET}}$		PROM programming mode setting
A0 to A16		Address bus
D0 to D7	Input/output	Data bus
$\overline{\text{CE}}$	Input	PROM enable input/program pulse input
OE		PROM read strobe input
$\overline{\text{PGM}}$		PROM program/program inhibit input
$V_{DD}$	—	Positive power supply
$V_{SS}$		GND

## 2.2 PIN FUNCTIONS

### 2.2.1 Normal Operating Mode

#### (1) P00 to P07 (Port 0) ... 3-state input/output

Port 0 is an 8-bit input/output port with an output latch, and has direct transistor drive capability. Input/output can be specified bit-wise by means of the port 0 mode register (PM0). Each pin incorporates a software programmable pull-up resistor. P00 to P03 and P04 to P07 can output the port 0 buffer register (P0L, P0H) contents at any time interval as 4-bit or 8-bit real-time output port. The real-time output port control register (RTPC) is used to select whether this port is used as a normal output port or a real-time output port.

When  $\overline{\text{RESET}}$  is input, port 0 is set as an input port (output high-impedance state), and the output latch contents are undefined.

#### (2) P10 to P17 (Port 1) ... 3-state input/output

Port 1 is an 8-bit input/output port with an output latch. Input/output can be specified bit-wise by means of the port 1 mode register (PM1). Each pin incorporates a software programmable pull-up resistor. This port has direct LED drive capability. Pins P10 and P11 are also made to function as PWM output pins by means of the PWM control register (PWMC), and pins P12 to P14 can also be made to function as serial input/output pins by means of the port 1 mode control register (PMC1). When  $\overline{\text{RESET}}$  is input, port 1 is set as an input port (output high-impedance state), and the output latch contents are undefined.

**Table 2-1 Port 1 Operating Modes**

Pin Name	Port Mode	Control Signal Input/Output Mode	Operation to Operate as Control Pin
P10	Input/output port	PWM0 output	Set (1) EN0 bit of PWMC
P11		PWM1 output	Set (1) EN1 bit of PWMC
P12		ASCK2/ $\overline{\text{SCK2}}$ input/output	Set (1) PMC12 bit of PMC1
P13		RxD2/SI2 input	Set (1) PMC13 bit of PMC1
P14		TxD2/SO2 output	Set (1) PMC14 bit of PMC1
P15 to P17		—	—

**(a) Port mode**

P10 and P11 operate as port mode pins when the EN0 and EN1 bits of the PWM control (PWMC) register are cleared (0), and P12 to P14 do the same when the relevant bits of the port 1 mode control (PMC1) register are cleared (0), and P15 to P17 always operate as port mode pins. Input/output can be specified bit-wise by means of the port 1 mode register (PM1).

**(b) Control signal output mode**

P10 and P11 operate as PWM signal output pins when the EN0 and EN1 bits, respectively, of the PWM control (PWMC) register are set (1).

P12 to P14 can be set as control pins bit-wise by setting the port 1 mode control (PMC1) register.

**(i) ASCK2/ $\overline{\text{SCK2}}$** 

ASCK2 is the asynchronous serial interface baud rate clock input pin.

$\overline{\text{SCK2}}$  is the serial clock input/output pin (in 3-wire serial I/O2 mode).

**(ii) RxD2/SI2**

RxD2 is the asynchronous serial interface serial data input pin.

SI2 is the serial data input pin (in 3-wire serial I/O2 mode).

**(iii) TxD2/SO2**

TxD2 is the asynchronous serial interface serial data output pin.

SO2 is the serial data output pin (in 3-wire serial I/O2 mode).

**(3) P20 to P27 (Port 2) ... Input**

Port 2 is an 8-bit input-only port. P22 to P27 incorporate a software programmable pull-up resistor. As well as operating as an input port, port 2 pins also operate as control signal input pins, such as external interrupt signal pins (see **Table 2-2**). All 8 pins are Schmitt-triggered inputs to prevent misoperation due to noise.

Also, pin P25 can also be made to function as a serial clock output pin by selecting the external clock as "serial operation enabled" with the clocked serial interface mode register 1 (CSIM1).

**Table 2-2 Port 2 Operating Modes**

Port	Functions
P20	Input port / NMI input*
P21	Input port / INTP0 input / CR11 capture trigger input / timer/counter 1 count clock / real-time output port trigger signal
P22	Input port / INTP1 input / CR22 capture trigger input
P23	Input port / INTP2 input / CI input
P24	Input port / INTP3 input / CR02 capture trigger timer/input/counter 0 count clock
P25	Input port / INTP4 input / ASCK input / $\overline{\text{SCK1}}$ input/output
P26	Input port / INTP5 input / A/D converter external trigger input
P27	Input port / SI0 input

\* NMI input is acknowledged regardless of whether interrupts are enabled or disabled.

**(a) Function as port pins**

The pin level can always be read or tested regardless of the dual-function pin operation.

**(b) Functions as control signal input pins**

**(i) NMI (Non-maskable Interrupt)**

The external non-maskable interrupt request input pin. Rising edge detection or falling edge detection can be specified by means of the external interrupt mode register 0 (INTM0).

**(ii) INTP0 to INTP5 (Interrupt from Peripherals)**

External interrupt request input pins. When the valid edge specified by the external interrupt mode register 0, (INTM0/INTM1) is detected by pins INTP0 to INTP5, an interrupt is generated (see **CHAPTER 19 EDGE DETECTION FUNCTION**).

In addition, pins INTP0 to INTP3 and INTP5 are also used as external trigger input pins with the various functions shown below.

- INTP0 ..... Timer/counter 1 capture trigger input pin  
Timer/counter 1 external count clock input pin  
Real-time output port trigger input pin
- INTP1 ..... Timer/counter 2 capture trigger input pin to capture register (CR22)
- INTP2 ..... Timer/counter 2 external count clock input pin  
Capture trigger input pin to capture/compare register (CR21)
- INTP3 ..... Timer/counter 0 capture trigger input pin  
Timer/counter 0 external count clock input pin
- INTP5 ..... A/D converter external trigger input pin

**(iii) CI (Clock Input)**

The timer/counter 2 external clock input pin.

**(iv) ASCK (Asynchronous Serial Clock)**

The external baud rate clock input pin.

**(v)  $\overline{\text{SCK1}}$  (Serial Clock)**

The serial clock input/output pin (in 3-wire serial I/O1 mode).

**(vi) SI0 (Serial Input 0)**

The serial data input pin (in 3-wire serial I/O0 mode).

**(4) P30 to P37 (Port 3) ... 3-state input/output**

Port 3 is an 8-bit input/output port with an output latch. Input/output can be specified bit-wise by means of the port 3 mode register (PM3). Each pin incorporates a software programmable pull-up resistor.

In addition to its function as an input/output port, port 3 also has various dual-function control signal pin functions.

The operating mode can be specified bit-wise by means of the port 3 mode control register (PMC3), as shown in Table 2-3. The pin level of any pin can always be read or tested regardless of the dual-function pin operation.

When  $\overline{\text{RESET}}$  is input, port 3 is set as an input port (output high-impedance state), and the output latch contents are undefined.

**Table 2-3 Port 3 Operating Modes (n = 0 to 7)**

Mode	Port Mode	Control Signal Input/Output Mode
Setting Condition	PMC3n = 0	PMC3n = 1
P30	Input/output port	RxD input / SI1 input
P31		TxD output / SO1 output
P32		$\overline{\text{SCK0}}$ input/output
P33		SO0 output / SB0 input/output
P34		TO0 output
P35		TO1 output
P36		TO2 output
P37		TO3 output

**(a) Port mode**

Each port specified as port mode by the port 3 mode control (PMC3) register can be specified as input/output bit-wise by means of the port 3 mode register (PM3).

**(b) Control signal input/output mode**

Pins can be set as control pins bit-wise by setting the port 3 mode control (PMC3) register.

**(i) RxD (Receive Data)/SI1 (Serial Input 1)**

RxD is the asynchronous serial interface serial data input pin.

SI1 is the serial data input pin (in 3-wire serial I/O1 mode).

**(ii) TxD (Transmit Data)/SO1 (Serial Output 1)**

TxD is the asynchronous serial interface serial data output pin.

SO1 is the serial data output pin (in 3-wire serial I/O1 mode).

**(iii)  $\overline{\text{SCK0}}$  (Serial Clock 0)**

The clocked serial interface serial clock input/output pin.

**(iv) SO0 (Serial Output 0)/SB0 (Serial Bus)**

SO0 is the serial data output pin (in 3-wire serial I/O mode), and SB0 is the serial bus input/output pin (in SBI mode).

**(v) TO0 to TO3 (Timer Output)**

The timer output pins.

**(5) P40 to P47 (Port 4) ... 3-state input/output**

Port 4 is an 8-bit input/output port with an output latch. Input/output can be specified bit-wise by means of the port 4 mode register (PM4). Each pin incorporates a software programmable pull-up resistor. This port has direct LED drive capability. Port 4 also functions as the time division address/data bus (AD0 to AD7) by the memory extension mode register (MM) when external memory or I/Os are extended.

In the  $\mu\text{PD784021}$ , this port only functions as the time division address/data bus (AD0 to AD7).

When RESET is input, port 4 is set as an input port (output high-impedance state), and the output latch contents are undefined.



**(6) P50 to P57 (Port 5) ... 3-state input/output**

Port 5 is an 8-bit input/output port with an output latch. Input/output can be specified bit-wise by means of the port 5 mode register (PM5). Each pin incorporates a software programmable pull-up resistor. This port has direct LED drive capability. In addition, P50 to P57 can be selected by means of the memory extension mode register (MM) in 2-bit units as pins that function as the address bus (A8 to A15) when external memory or I/Os are extended.

In the  $\mu$ PD784021, these pins only function as the address bus (A8 to A15).

When  $\overline{\text{RESET}}$  is input, port 5 is set as an input port (output high-impedance state), and the output latch contents are undefined.

**(7) P60 to P67 (Port 6) ... 3-state input/output**

Port 6 is an 8-bit input/output port with an output latch (in the  $\mu$ PD784021, P60 to P63 are output-only port pins). P60 to P67 (P64 to P67 in the  $\mu$ PD784021) incorporate a software programmable pull-up resistor.

In addition to its function as a port, port 6 also has various dual-function control signal pin functions, as shown in Table 2-4. Operations as control pins are performed by the respective function operations.

In the  $\mu$ PD784021, P64 and P65 only function as the  $\overline{\text{RD}}$  output and  $\overline{\text{WR}}$  output, respectively.

When  $\overline{\text{RESET}}$  is input, P60 to P67 are set as input port pins (output high-impedance state), and the output latch contents are undefined. In the  $\mu$ PD784021, P60 to P63 are set in output high-impedance state when  $\overline{\text{RESET}}$  is input, and then the pins are driven low after  $\overline{\text{RESET}}$  release. P64 to P67 are set as input port pins (output high-impedance state). In the  $\mu$ PD784021, the high-order 4 bits of the output latch contents are undefined, and the low-order 4 bits are 0H.

**Table 2-4 Port 6 Operating Modes**

Pin Name	Port Mode	Control Signal Input/Output Mode	Operation to Operate as Control Pin
P60 to P63	Input/output ports*	A16 to A19 output	Specified in 2-bit units by bits MM3 to MM0 of the MM
P64	Input/output ports	$\overline{\text{RD}}$ output	In the $\mu$ PD784021, or external memory extension mode specified by bits MM3 to MM0 of the MM
P65		$\overline{\text{WR}}$ output	
P66		$\overline{\text{WAIT}}$ input	Specified by setting bits PWN1 & PWN0 (n = 0 to 7) of the PWC1 & PWC2 and P66 to input mode
		HLDRQ input	
P67		HLDAR output	Set (1) the RFEN bit of the RFM
		$\overline{\text{REFRQ}}$ output	

\* Output port in the case of the  $\mu$ PD784021

**Caution** In the  $\mu$ PD784021, P60 to P63 are in the output high-impedance state during  $\overline{\text{RESET}}$  input, but output a low level after  $\overline{\text{RESET}}$  release. Therefore, external circuitry should be designed so that output of a low-level signal in the initial state is no problem.

**Remark** See CHAPTER 21 LOCAL BUS INTERFACE FUNCTION for details.

**(a) Port mode****(i) With  $\mu$ PD784026**

Each port not set in the control mode can be set in the input or output mode in 1-bit units by using the port 6 mode register (PM6).

**(ii) With  $\mu$ PD784021**

Of the port not set in the control mode, P60 through P63 function as output port pins, and P64 through P67 can be set in the input or output mode in 1-bit units by using the port 6 mode register (PM6).

**(b) Control signal input/output mode****(i) A16 to A19 (Address Bus)**

Upper address bus output pins in case of external memory space extension (10000H to FFFFFH).  
These pins operate in accordance with the memory extension mode register (MM).

**(ii)  $\overline{\text{RD}}$  (Read Strobe)**

Pin that outputs the strobe signal for an external memory read operation.  
Operates in accordance with the memory extension mode register (MM).  
In the  $\mu$ PD784021, this pin always operates as the  $\overline{\text{RD}}$  pin.

**(iii)  $\overline{\text{WR}}$  (Write Strobe)**

Pin that outputs the strobe signal for an external memory write operation.  
Operates in accordance with the memory extension mode register (MM).  
In the  $\mu$ PD784021, this pin always operates as the  $\overline{\text{WR}}$  pin.

**(iv)  $\overline{\text{WAIT}}$  (Wait)**

Wait signal input pin. Operates in accordance with the programmable wait control registers (PWC1, PWC2).

**(v)  $\overline{\text{REFRQ}}$  (Refresh Request)**

This pin outputs refresh pulses to pseudo-static memory when this memory is connected externally. Operates in accordance with the refresh mode register (RFM).

**(vi) HLDRQ (Hold Request)**

External bus hold request signal input pin. Operates in accordance with the hold mode register (HLDM).

**(vii) HLDAK (Hold Acknowledge)**

Bus hold acknowledge signal output pin. Operates in accordance with the hold mode register (HLDM).

**(8) P70 to P77 (Port 7) ... 3-state input/output**

Port 7 is an 8-bit input/output port. In addition to operating as an input/output port, it also operates as the A/D converter analog input pins (ANI0 to ANI7).

Input/output can be specified bit-wise by means of the port 7 mode register (PM7).

The levels of these pins can always be read or tested, regardless of the operation of the multiplexed pins.

When  $\overline{\text{RESET}}$  is input, port 7 is set as an input port (output high-impedance state), and the output latch contents are undefined.

**(9) ASTB (Address Strobe)/CLKOUT (Clock Output) ... Output**

This pin outputs the timing signal that latches address information externally in order to access an external address. It also operates as the pin that supplies the clock to an external device.

**(10) X1, X2 (Crystal)**

The internal clock oscillation crystal connection pins. When the clock is supplied externally, it is input to the X1 pin. Usually signal with the inverse phase of the X1 pin signal phase is input to the X2 pin (see **4.3.1 Clock oscillation circuit**).

**(11)  $\overline{\text{RESET}}$  (Reset) ... Input**

The active-low reset input.

**(12) ANO0, ANO1 ... Output**

The D/A converter analog voltage output pins.

**(13)  $\text{AV}_{\text{REF1}}$** 

The A/D converter reference voltage input pin.

**(14)  $\text{AV}_{\text{REF2}}$** 

The D/A converter reference voltage input (+ side) pin.

**(15)  $\text{AV}_{\text{REF3}}$** 

The D/A converter reference voltage input (– side) pin.

**(16)  $\text{AV}_{\text{DD}}$** 

The A/D converter power supply pin. This should be made at the same potential as the  $\text{V}_{\text{DD}}$  pin.

**(17)  $\text{AV}_{\text{SS}}$** 

The A/D converter GND pin. This should be made at the same potential as the  $\text{V}_{\text{SS}}$  pin.

**(18)  $\text{V}_{\text{DD}}$** 

Positive power supply pins. All  $\text{V}_{\text{DD}}$  pins should be connected to the positive power supply.

**(19)  $\text{V}_{\text{SS}}$** 

GND potential pins. All  $\text{V}_{\text{SS}}$  pins should be connected to ground.

**(20) TEST**

Pin used by NEC for IC testing. Must be directly connected to  $\text{V}_{\text{SS}}$ .

**Caution** In the  $\mu$ PD78233 and 78237, the TEST pin is the MODE pin and is fixed high. When changing over from the  $\mu$ PD78233, 78237, the circuitry can be modified so that this pin is directly connected to V<sub>ss</sub>. Any modification is needed if the  $\mu$ PD78234, 78238 was used with switching between the internal ROM mode and ROM-less mode performed by MODE pin switching (the TEST pin must be directly connected to V<sub>ss</sub>).

**Modification examples:**

- Incorporate all programs in ROM.
- Store all programs in external ROM.
- Change the location address of a program previously held in external ROM, shift the address to avoid overlapping internal ROM, and execute this program from the program internal ROM.

**2.2.2 PROM Programming Mode ( $\mu$ PD78P4026)****(1)  $V_{PP}$  (Programming Power Supply) ... Input**

Input pin that sets the  $\mu$ PD78P4026 to the PROM programming mode. When the input voltage of this pin is +5 V or more and the  $\overline{\text{RESET}}$  input is driven low, the  $\mu$ PD78P4026 switches to the PROM programming mode.

If  $\overline{\text{CE}} = \text{L}$  is set when  $V_{PP} = +12.5 \text{ V}$  and  $\overline{\text{OE}} = \text{H}$ , the program data on D0 to D7 can be written in the internal PROM cell selected by A0 to A16.

**(2)  $\overline{\text{RESET}}$  (Reset) ... Input**

Input pin that sets the  $\mu$ PD78P4026 to the PROM programming mode. When the input voltage of the  $V_{PP}$  pin reaches +5 V or more and the input of this pin is low, the  $\mu$ PD78P4026 switches to the PROM programming mode.

**(3) A0 to A16 (Address Bus) ... Input**

The address bus. Selects an internal PROM address (0000H to 7FFFH).

**(4) D0 to D7 (Data Bus) ... Input/Output**

The data bus. Internal PROM program reads and writes are performed via this bus.

**(5)  $\overline{\text{CE}}$  (Chip Enable) ... Input**

Inputs the internal PROM enable signal. When this signal is active, program writing/reading is enabled.

**(6)  $\overline{\text{OE}}$  (Output Enable) ... Input**

Inputs the internal PROM read strobe signal. When this signal is activated while  $\overline{\text{CE}} = \text{L}$ , the program data (1 byte) in the internal PROM cell selected by A0 to A16 can be read onto D0 to D7.

**(7)  $\overline{\text{PGM}}$  (Program) ... Input**

The internal PROM operating mode control signal input pin.

When this signal is active, it is possible to write to internal PROM.

When this signal is inactive, it is possible to read from internal PROM.

**(8)  $V_{DD}$** 

Positive power supply pins.

**(9)  $V_{SS}$** 

GND potential pins.

### 2.3 INPUT/OUTPUT CIRCUITS AND CONNECTION OF UNUSED PINS

Table 2-5 shows the input/output circuit types of the pins that have functions, and the connection method when that function is not used.

Each input/output circuit type is shown in Figure 2-1.

**Table 2-5 Pin Input/Output Circuit Types and Recommended Connection When Not Used (1/2)**

Pin Name		Input/Output Circuit Type	Input/Output	Recommended Connection When Not Used
P00 to P07		5-A	Input/output	Input : Connect to V <sub>DD</sub> Output : Leave open
P10/PWM0 P11/PWM1		5-A	Input/output	Input : Connect to V <sub>DD</sub> Output : Leave open
P12/ASCK2/ $\overline{\text{SCK2}}$		8-A		
P13/RxD2/SI2		5-A		
P14/TxD2/SO2		5-A		
P15 to P17				
P20/NMI		2	Input	Connect to V <sub>DD</sub> or V <sub>SS</sub>
P21/INTP0				
P22/INTP1		2-A		Connect to V <sub>DD</sub>
P23/INTP2/CI				
P24/INTP3				
P25/INTP4/ASCK/ $\overline{\text{SCK1}}$		8-A	Input/output	Input : Connect to V <sub>DD</sub> Output : Leave open
P26/INTP5		2-A	Input	Connect to V <sub>DD</sub>
P27/SI0				
P30/RxD/SI1		5-A	Input/output	Input : Connect to V <sub>DD</sub> Output : Leave open
P31/TxD/SO1				
P32/ $\overline{\text{SCK0}}$		8-A		
P33/SB0/SO0		10-A		
P34/TO0 to P37/TO3		5-A		
P40/AD0 to P47/AD7				
P50/A8 to P57/A15				
P60/A16 to	ROM-less product	4	Output	Leave open
P63/A19	Mask ROM product	5-A	Input/output	Input : Connect to V <sub>DD</sub> Output : Leave open
P64/ $\overline{\text{RD}}$		5-A	Input/output	Input : Connect to V <sub>DD</sub> Output : Leave open
P65/ $\overline{\text{WR}}$				
P66/ $\overline{\text{WAIT}}$ /HLDRQ				
P67/REFRQ/HLDAK				

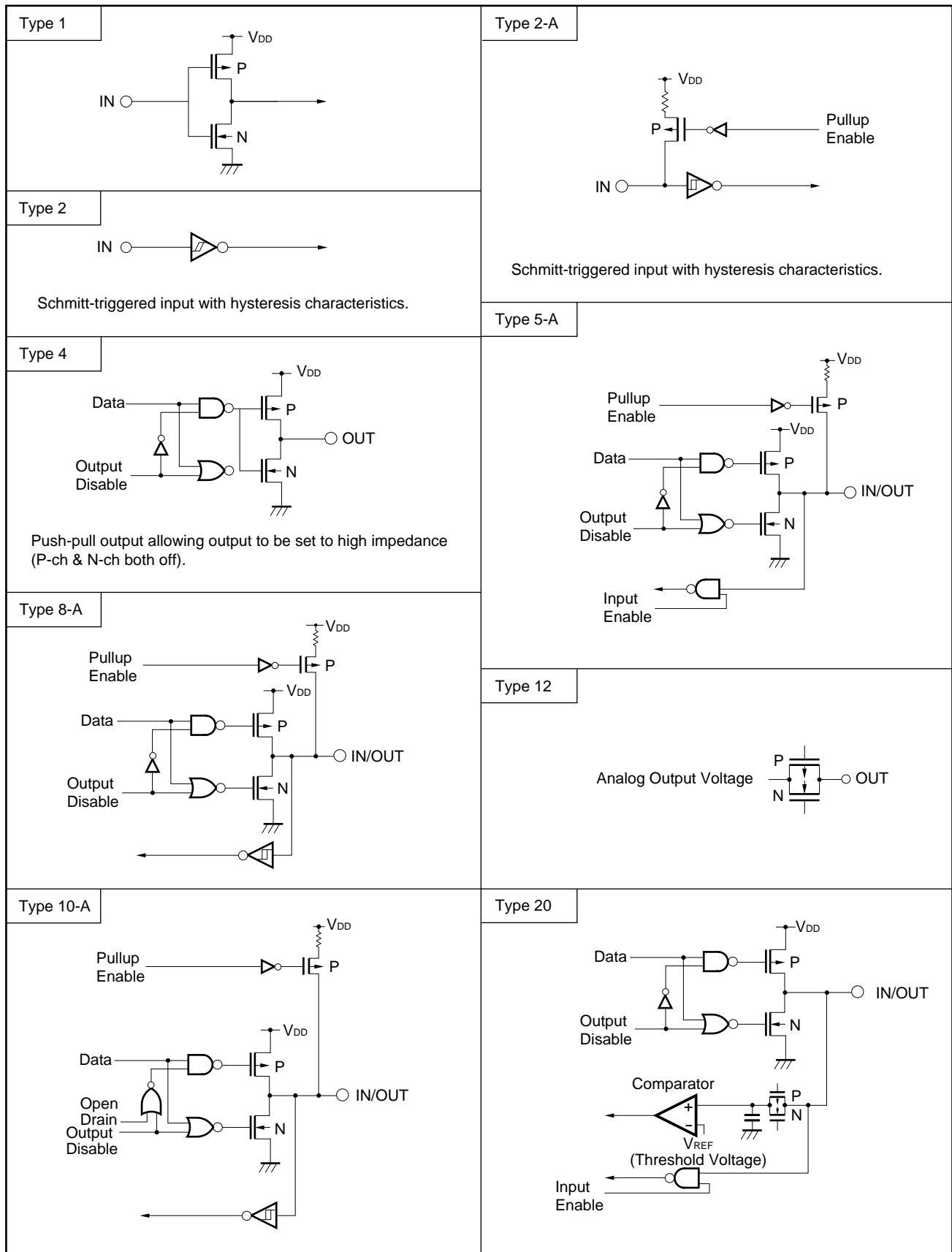
**Table 2-5 Pin Input/Output Circuit Types and Recommended Connection When Not Used (2/2)**

Pin Name	Input/Output Circuit Type	Input/Output	Recommended Connection When Not Used
P70/ANI0 to P77/ANI7	20	Input/output	Input : Connect to V <sub>DD</sub> or V <sub>SS</sub> Output : Leave open
ANO0, ANO1	12	Output	Leave open
ASTB/CLKOUT	4		
RESET	2	Input	—
TEST	1		Directly connect to V <sub>SS</sub>
AV <sub>REF1</sub> – AV <sub>REF3</sub>	—		Connect to V <sub>SS</sub>
AV <sub>SS</sub>			
AV <sub>DD</sub>			Connect to V <sub>DD</sub>

**Caution** If the input/output mode is undefined for an input/output dual-function pin, it should be connected to  $V_{DD}$  via a resistor of several tens of  $k\Omega$  (especially when the reset input pin goes to the low-level input voltage or over upon powering on, and when input/output is switched in software.)

**Remark** The type numbers are standard for the 78K series, and therefore are not necessarily serial numbers within each product (there are non-incorporated circuits).

Figure 2-1 Pin Input/Output Circuits





## 2.4 CAUTIONS

- (1) When connecting unused pins, if the input/output mode is undefined for an input/output dual-function pin, it should be connected to  $V_{DD}$  with a resistor of several tens of  $k\Omega$  (especially when the reset input pin becomes the low-level input voltage or over upon powering on, and when input/output is switched by software.)
- (2) In the  $\mu$ PD784021, P60 to P63 are in the output high-impedance state during  $\overline{RESET}$  input, but output low level after  $\overline{RESET}$  release. Therefore, external circuitry should be designed so that output of low-level signal in the initial state is no problem.
- (3) In the  $\mu$ PD78233 and 78237, the TEST pin is the MODE pin and is fixed high. When changing over from the  $\mu$ PD78233/78237, the circuitry must be modified so that this pin is directly connected to  $V_{SS}$ .  
Some modification is needed if the  $\mu$ PD78234/78238 was used with switching between the internal ROM mode and ROM-less mode performed by MODE pin switching (the TEST pin must be directly connected to  $V_{SS}$ ).

Modifications examples:

- Incorporate all programs in ROM
- Store all programs in external ROM
- Change the location address of a program previously held in external ROM, shift the address to avoid overlapping internal ROM, and execute this program from the program in internal ROM

[MEMO]

## CHAPTER 3 CPU ARCHITECTURE

### 3.1 MEMORY SPACE

The  $\mu$ PD784026 can access a 1 M-byte memory space. The mapping of the internal data area (special function registers and internal RAM) depends on the LOCATION instruction. A LOCATION instruction must be executed after reset release, and can only be used once.

The program after reset release must be as follows:

```

RSTVCT  CSEG  AT 0
        DW    RSTSTRT
        to
INITSEG  CSEG  BASE
RSTSTRT: LOCATION 0H; or LOCATION 0FH
        MOVG  SP, #STKBGN

```

#### (1) When LOCATION 0 instruction is executed

The internal data area is mapped onto addresses 0F700H to 0FFFFH.

Internal ROM is mapped onto addresses 0 to 0F6FFH in the  $\mu$ PD784026, and onto addresses 0 to 0BFFFH in the  $\mu$ PD784025. The 2304 bytes of the 64 K-byte ROM incorporated in the  $\mu$ PD784026 that overlap the internal data area cannot be used when the LOCATION 0 instruction is executed.

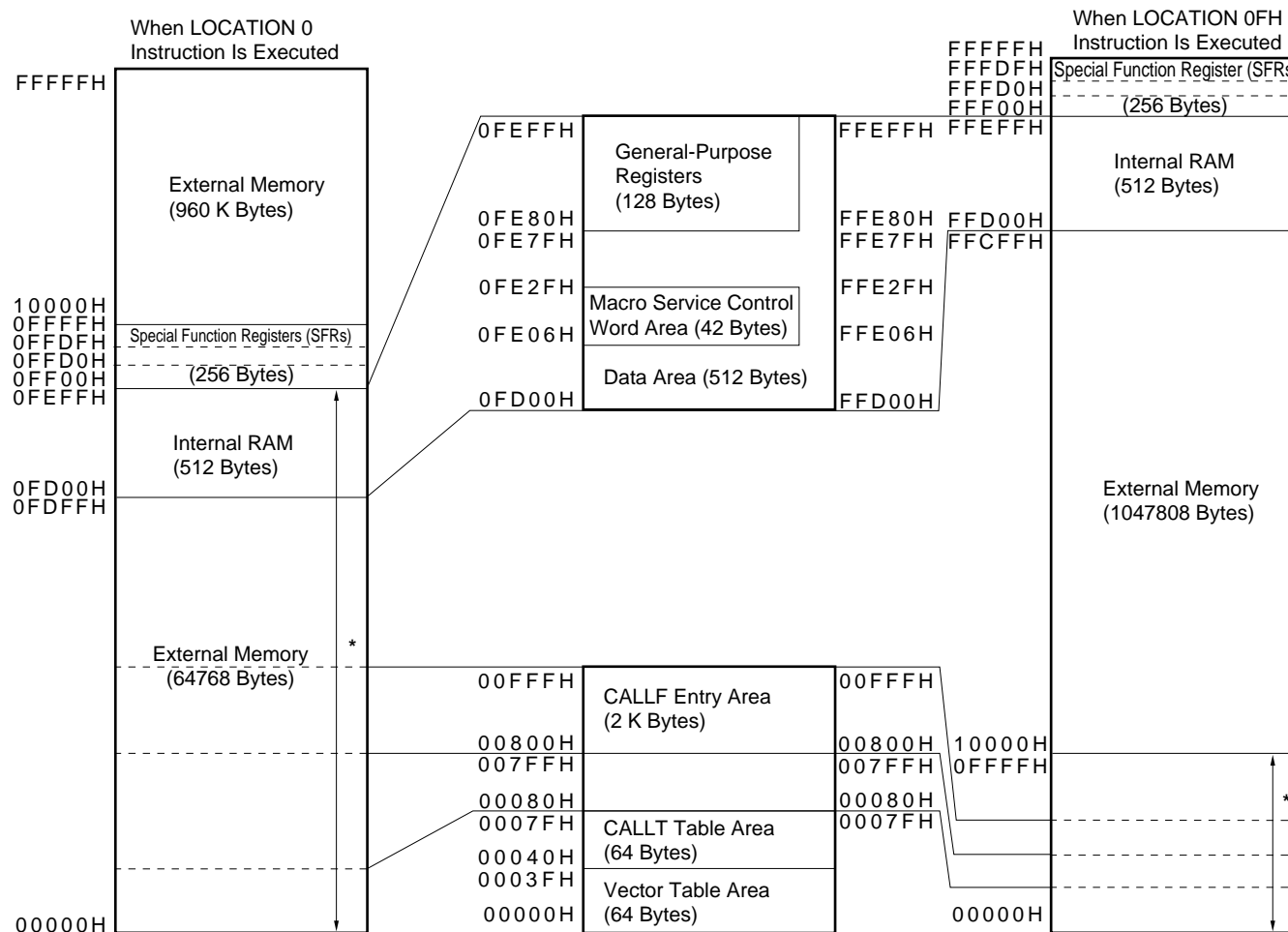
External memory is accessed in external memory extension mode.

#### (2) When LOCATION 0FH instruction is executed

The internal data area is mapped onto addresses FF700H to FFFFFH.

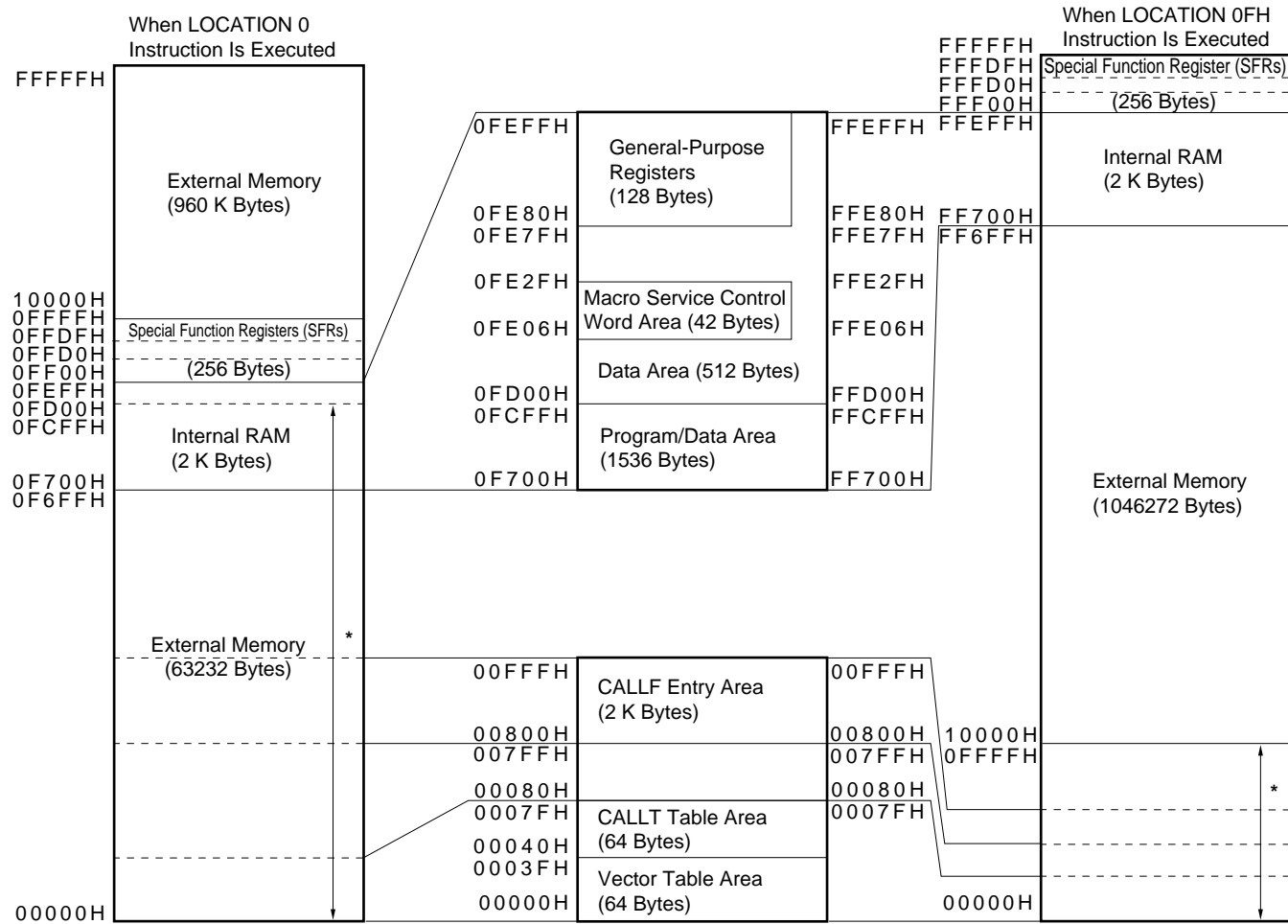
Internal ROM is mapped onto addresses 0 to 0FFFFH in the  $\mu$ PD784026, and onto addresses 0 to 0BFFFH in the  $\mu$ PD784025.

External memory is accessed in external memory extension mode.

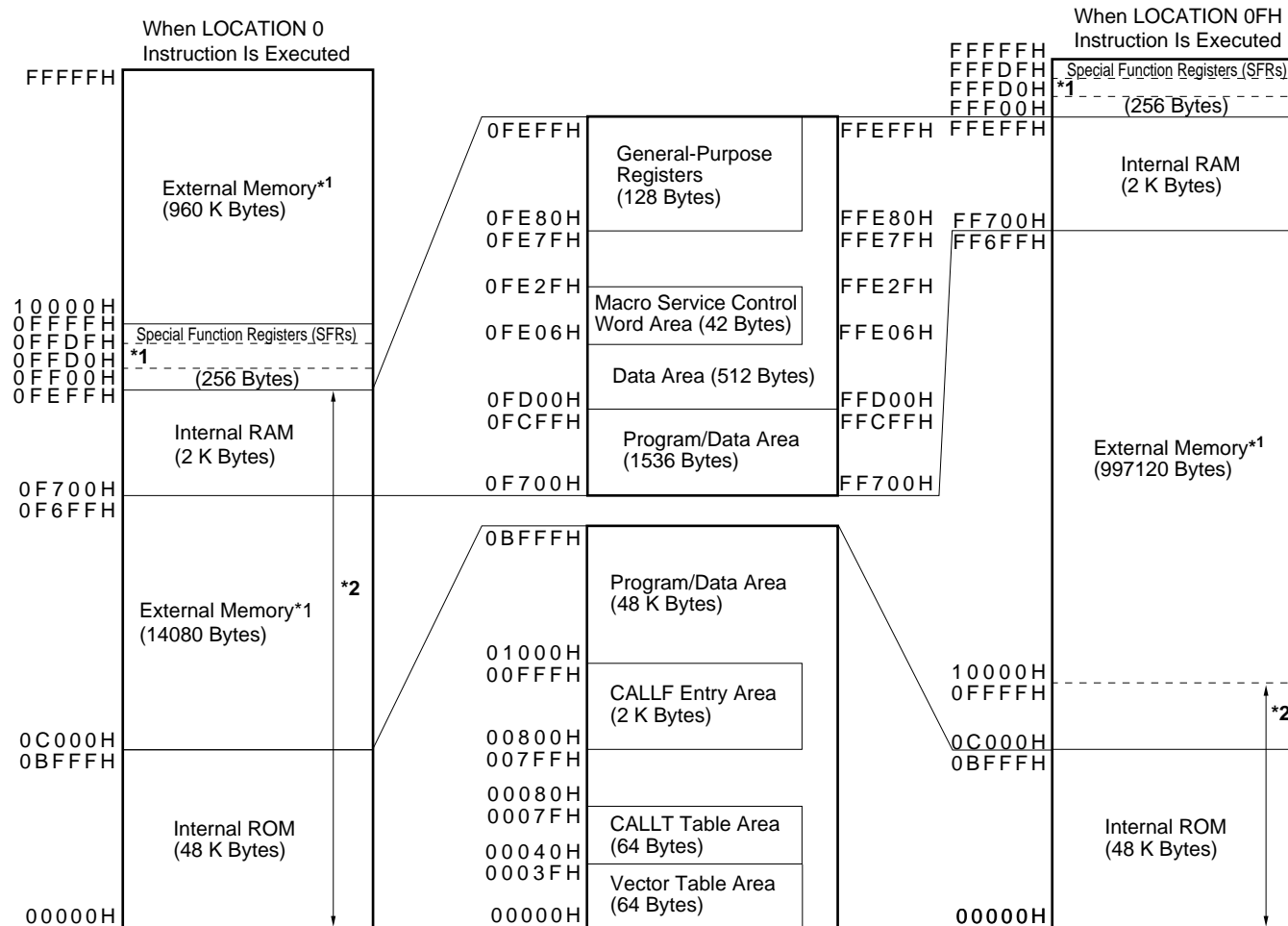
★ Figure 3-1  $\mu$ PD784020 Memory Map

\* Base area, reset or interrupt entry area, excluding internal RAM in the case of reset.

Figure 3-2  $\mu$ PD784021 Memory Map

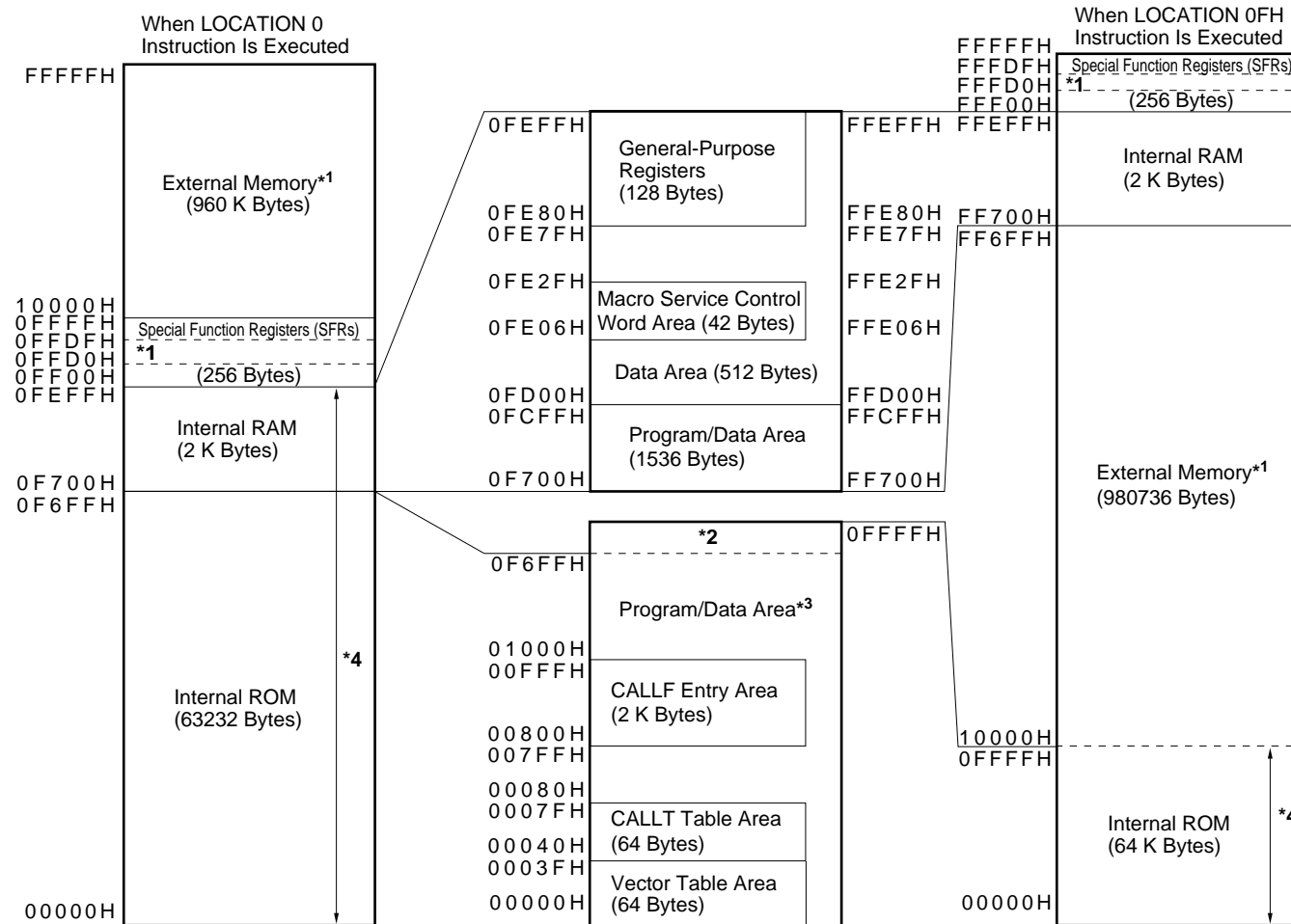


\* Base area, reset or interrupt entry area, excluding internal RAM in the case of reset.

Figure 3-3  $\mu$ PD784025 Memory Map

- \* 1. Accessed in external memory extension mode.  
 \* 2. Base area, reset or interrupt entry area, excluding internal RAM in the case of reset.

Figure 3-4  $\mu$ PD784026 Memory Map



- \* 1. Accessed in external memory extension mode.
- \* 2. The 2304 bytes of this area can be used as internal ROM only when the LOCATION 0FH instruction is executed.
- \* 3. 63232 bytes when the LOCATION 0 instruction is executed, and 65536 bytes when the LOCATION 0FH instruction is executed.
- \* 4. Base area, reset or interrupt entry area, excluding internal RAM in the case of reset.

### 3.2 INTERNAL ROM AREA

The  $\mu$ PD784026 subseries products shown below incorporate ROM which is used to store programs, table data, etc.

If the internal ROM area and internal data area overlap when the LOCATION 0 instruction is executed, the internal data area is accessed, and the overlapping part of the internal ROM area cannot be accessed.

Product Name	Internal ROM	Address Space	
		Location 0 Instruction	Location 0FH Instruction
$\mu$ PD784025	48 K $\times$ 8 bits	00000H-0BFFFFH	00000H-0BFFFFH
$\mu$ PD784026	64 K $\times$ 8 bits $\mu$ PD78P4026	00000H-0F6FFFH	00000H-0FFFFFH

The internal ROM can be accessed at high speed. Normally, fetches are performed at the same speed as external ROM, but if the IFCH bit of the memory extension mode register (MM) is set (1), the high-speed fetch function is used and internal ROM fetches are performed at high speed (2-byte fetch performed in 2 system clocks).

When the instruction execution cycle equal to an external ROM fetch is selected, wait insertion is performed by the wait function, but when high-speed fetches are used, wait insertion is not performed for internal ROM. However, do not set external wait for the internal ROM area. If external wait is set for the internal ROM area, the CPU is deadlocked. This deadlock status can be cleared only by reset input.

RESET input sets the instruction execution cycle equal to the external ROM fetch cycle.

**Remark** In the case of the ROM-less product ( $\mu$ PD784020, 784021), this address space is external memory.



### 3.3 BASE AREA

The space from 0 to FFFFH comprises the base area. The base area is the object for the following uses:

- Reset entry address
- Interrupt entry address
- CALLT instruction entry address
- 16-bit immediate addressing mode (with instruction address addressing)
- 16-bit direct addressing mode
- 16-bit register addressing mode (with instruction address addressing)
- 16-bit register indirect addressing mode
- Short direct 16-bit memory indirect addressing mode

The vector table area, CALLT instruction table area and CALLF instruction entry area are allocated to the base area.

When the LOCATION 0 instruction is executed, the internal data area is located in the base area. Note that, in the internal data area, program fetches cannot be performed from the internal high-speed RAM area or special function register (SFR) area. Also, internal RAM area data should only be used after initialization has been performed.

### 3.3.1 Vector Table Area

The 64-byte area from 00000H to 0003FH is reserved as the vector table area. The vector table area stores the program start addresses used when a branch is made as the result of  $\overline{\text{RESET}}$  input or generation of an interrupt request. When context switching is used by an interrupt, the number of the register bank to be switched to is stored here.

Any portion not used as the vector table can be used as program memory or data memory.

16-bit values can be written to the vector table. Therefore, branches can only be made within the base area.

**Table 3-1 Vector Table**

Vector Table Address	Interrupt Source
0003CH	Operand error
0003EH	BRK
00000H	Reset ( $\overline{\text{RESET}}$ input)
00002H	NMI
00004H	WDT
00006H	INTP0
00008H	INTP1
0000AH	INTP2
0000CH	INTP3
0000EH	INTC00
00010H	INTC01
00012H	INTC10
00014H	INTC11
00016H	INTC20
00018H	INTC21
0001AH	INTC30
0001CH	INTP4
0001EH	INTP5
00020H	INTAD
00022H	INTSER1
00024H	INTSR1/INTCSI1
00026H	INTST1
00028H	INTCSI
0002AH	INTSER2
0002CH	INTSR2/INTCSI2
0002EH	INTST2

### 3.3.2 CALLT Instruction Table Area

The 1-byte call instruction (CALLT) subroutine entry addresses can be stored in the 64-byte area from 00040H to 0007FH.

The CALLT instruction references this table, and branches to a base area address written in the table as a subroutine. As the CALLT instruction is one byte in length, use of the CALLT instruction for subroutine calls written frequently throughout the program enables the program object size to be reduced. The table can contain up to 32 subroutine entry addresses, and therefore it is recommended that they be recorded in order of frequency.

If this area is not used as the CALLT instruction table, it can be used as ordinary program memory or data memory.

### 3.3.3 CALLF Instruction Entry Area

A subroutine call can be made directly to the area from 00800H to 00FFFFH with the 2-byte call instruction (CALLF).

As the CALLF instruction is a two-byte call instruction, it enables the object size to be reduced compared with use of the direct subroutine call CALL instruction (3 or 4 bytes).

Writing subroutines directly in this area is an effective means of exploiting the high-speed capability of the device.

If you wish to reduce the object size, writing an unconditional branch (BR) instruction in this area and locating the subroutine itself outside this area will result in a reduced object size for subroutines that are called from five or more points. In this case, only the 4 bytes of the BR instruction are occupied in the CALLF entry area, enabling the object size to be reduced with a large number of subroutines.

### 3.4 INTERNAL DATA AREA

The internal data area consists of the internal RAM area and special function register area (see **Figures 3-1 to 3-4**).

The last address of the internal data area can be specified by means of the LOCATION instruction as either 0FFFFH (when a LOCATION 0 instruction is executed) or FFFFFH (when a LOCATION 0FH instruction is executed). Selection of the addresses of the internal data area by means of the LOCATION instruction must be executed once immediately after reset release, and once the selection is made, it cannot be changed. The program after reset release must be as shown in the example below. If the internal data area and another area are allocated to the same addresses, the internal data area is accessed and the other area cannot be accessed.

```

Example  RSTVCT  CSEG  AT 0
           DW      RSTSTRT
           to
           INITSEG CSEG  BASE
           RSTSTRT: LOCATION 0H; or LOCATION 0FH
           MOVG   SP, #STKBGN

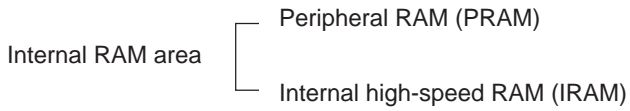
```

**Caution** When the LOCATION 0 instruction is executed, it is necessary to ensure that the program after reset release does not overlap the internal data area. It is also necessary to make sure that the entry addresses of the processing routines for non-maskable interrupts such as NMI do not overlap the internal data area. Also, initialization must be performed for maskable interrupt entry areas, etc., before the internal data area is referenced.

**3.4.1 Internal RAM Area**

The  $\mu$ PD784026 incorporates general-purpose static RAM.

This area is configured as follows:

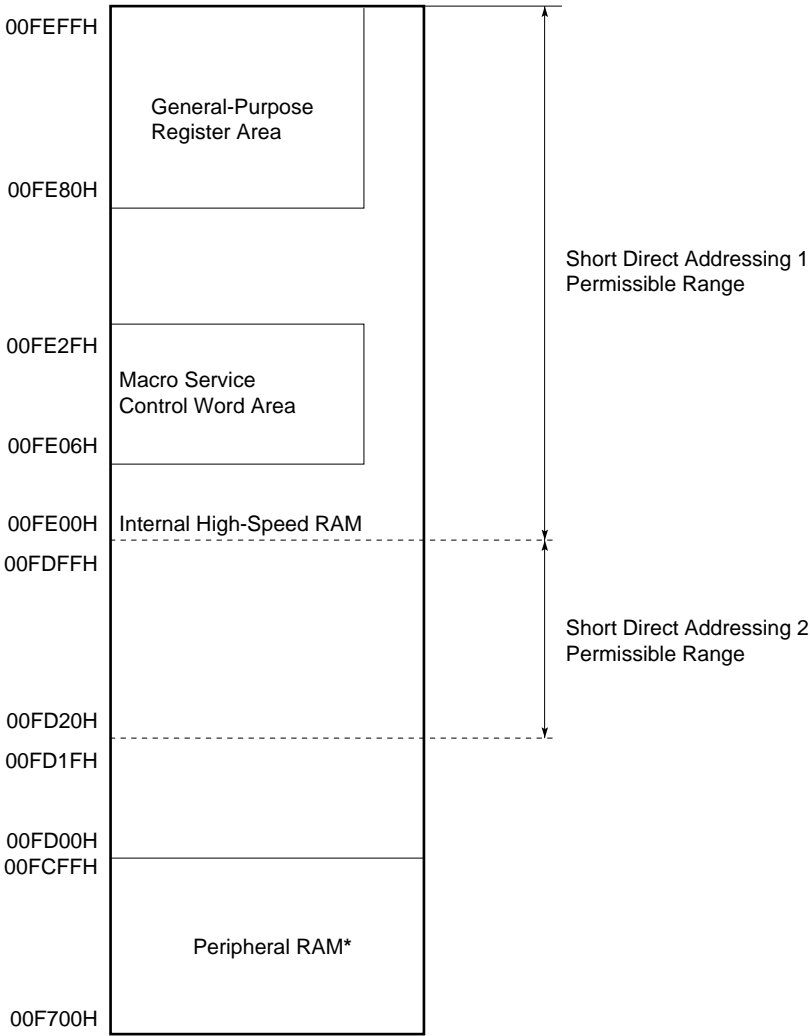
**Table 3-2 Internal RAM Area**

Internal RAM Product Name	Internal RAM Area		
		Peripheral RAM: PRAM	Internal High-Speed RAM: IRAM
μPD784020	512 bytes (0FD00H-0FEFFH)	0 bytes	512 bytes (0FD00H-0FEFFH)
μPD784021	2048 bytes (0F700H-0FEFFH)	1536 bytes (0F700H-0FCFFH)	
μPD784025			
μPD784026			
μPD78P4026			

**Remark** The addresses in the table are the values that apply when the LOCATION 0 instruction is executed. When the LOCATION 0FH instruction is executed, 0F0000H should be added to the values shown above.

The internal RAM memory map is shown in Figure 3-5.

**Figure 3-5 Internal RAM Memory Map**



\* This area is not available on the  $\mu$ PD784020.

**Remark** The addresses in the figure are the values that apply when the LOCATION 0 instruction is executed. When the LOCATION 0FH instruction is executed, 0F0000H should be added to the values shown above.

**(1) Internal high-speed RAM (IRAM)**

The internal high-speed RAM (IRAM) allows high-speed accesses to be made. The short direct addressing mode for high-speed accesses can be used on FD20H to FEFFH in this area. There are two kinds of short direct addressing mode, short direct addressing 1 and short direct addressing 2, according to the target address. The function is the same in both of these addressing modes. With some instructions, the word length is shorter with short direct addressing 2 than with short direct addressing 1. See the **78K/IV Series User's Manual-Instruction** for details.

A program fetch cannot be performed from IRAM. If a program fetch is performed from an address onto which IRAM is mapped, CPU inadvertent loop will result.

The following areas are reserved in IRAM.

- General-purpose register area : FE80H to FEFFH
- Macro service control word area : FE06H to FE2FH (excluding 0FE22H, 0FE23H, 0FE2AH, 0FE2BH)
- Macro service channel area : FE00H to FEFFH (the address is specified by the macro service control word)

If the reserved function is not used in these areas, they can be used as ordinary data memory.

**Remark** The addresses in this text are those that apply when the LOCATION 0 instruction is executed. When the LOCATION 0FH instruction is executed, 0F0000H should be added to the values shown in the text.

**(2) Peripheral RAM (PRAM)**

The peripheral RAM (PRAM) is used as ordinary program memory or data memory. When used as program memory, the program must be written to the peripheral RAM beforehand by a program.

Program fetches from peripheral RAM are fast, with a 2-byte fetch being executed in 2 clocks.

### 3.4.2 Special Function Register (SFR) Area

The internal peripheral hardware special function registers (SFRs) are mapped onto the area from 0FF00H to 0FFFFH (see **Figures 3-1 to 3-4**).

The area from 0FFD0H to 0FFDFH is mapped as an external SFR area, and allows externally connected peripheral I/Os, etc., to be accessed in external memory extension mode (specified by the memory extension mode register (MM)) by the ROM-less product or internal ROM products.

**Caution** Addresses onto which SFRs are not mapped should not be accessed in this area. If such an address is accessed by mistake, the CPU may become deadlocked. A deadlock can only be released by reset input.

**Remark** The addresses in this text are those that apply when the LOCATION 0 instruction is executed. When the LOCATION 0FH instruction is executed, 0F0000H should be added to the values shown in the text.

### 3.4.3 External SFR Area

In  $\mu$ PD784026 subseries products, the 16-byte area from 0FFD0H to 0FFDFH in the SFR area (when the LOCATION 0 is executed; 0FFFD0H to 0FFFDH when the LOCATION 0FH instruction is executed) is mapped as an external SFR area. When the external memory extension mode is set in a ROM-less product or internal ROM product, externally connected peripheral I/Os, etc., can be accessed using the address bus or address/data bus, etc.

As the external SFR area can be accessed by SFR addressing, peripheral I/O and similar operations can be performed easily, the object size can be reduced, and macro service can be used.

Bus operations for accesses to the external SFR area are performed in the same way as for ordinary memory accesses.

## 3.5 EXTERNAL MEMORY SPACE

The external memory space is a memory space that can be accessed in accordance with the setting of the memory extension mode register (MM). It can store programs, table data, etc., and can have peripheral I/O devices allocated to it.



### 3.6 $\mu$ PD78P4026 MEMORY MAPPING

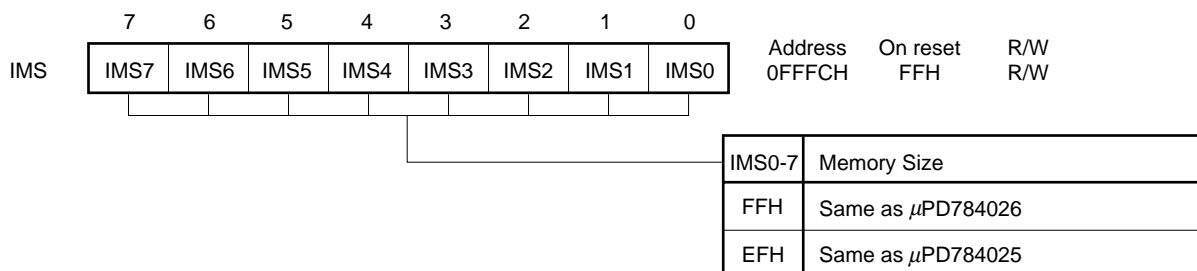
The  $\mu$ PD78P4026 incorporates 64 K-byte internal ROM and 2048-byte internal RAM, and its memory map is therefore slightly different from that of the  $\mu$ PD784025. In order to mask this difference, the  $\mu$ PD78P4026 has a function (the memory size switching function) which prevents part of the internal memory from being used in software.

Memory size switching is performed by means of the internal memory size switching register (IMS). To set the same memory map as in the  $\mu$ PD784025, this register must be written to immediately after a reset. Do not change the written value.

The IMS can be written to with an 8-bit manipulation instruction. The IMS format is shown in **Figure 3-6**.

RESET input sets the IMS register to FFH.

**Figure 3-6 Internal Memory Size Switching Register (IMS) Format**



The IMS register is not provided in the  $\mu$ PD784025 and  $\mu$ PD784026, but executing an instruction that writes to IMS register in the  $\mu$ PD784025 or  $\mu$ PD784026 will not affect the operation of the device.

★ **Caution** If the  $\mu$ PD78P4026 is selected by the in-circuit emulator as the emulation CPU, the memory size is always the same as that of the  $\mu$ PD784026 even if an instruction that writes data other than FFH (EFH) to IMS is executed.

### 3.7 CONTROL REGISTERS

Control registers consist of the program counter (PC), program status word (PSW), and stack pointer (SP).

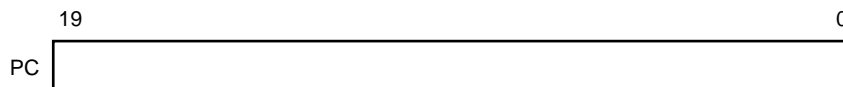
#### 3.7.1 Program Counter (PC)

This is a 20-bit binary counter that holds address information on the next program to be executed (see **Figure 3-7**).

Normally, the PC is incremented automatically by the number of bytes in the fetched instruction. When an instruction associated with a branch is executed, the immediate data or register contents are set in the PC.

Upon RESET input, the 16-bit data in address 0 and 1 is set in the low-order 16 bits, and 0000 in the high-order 4 bits of the PC.

**Figure 3-7 Program Counter (PC) Format**



#### 3.7.2 Program Status Word (PSW)

The program status word (PSW) is a 16-bit register comprising various flags that are set or reset according to the result of instruction execution.

Read accesses and write accesses are performed in high-order 8-bit (PSWH) and low-order 8-bit (PSWL) units. Individual flags can be manipulated by bit-manipulation instructions.

The contents of the PSW are automatically saved to the stack when a vectored interrupt request is acknowledged or a BRK instruction is executed, and automatically restored when an RETI or RETB instruction is executed. When context switching is used, the contents are automatically saved in RP3, and automatically restored when an RETCS or RETCSB instruction is executed.

RESET input resets (0) all bits.

“0” must always be written to the bits written as “0” in **Figure 3-8**. The contents of bits written as “-” are undefined when read.

**Figure 3-8 Program Status Word (PSW) Format**

Symbol	7	6	5	4	3	2	1	0
PSWH	UF	RBS2	RBS1	RBS0	–	–	–	–

PSWL	7	6	5	4	3	2	1	0
	S	Z	RSS	AC	IE	P/V	0	CY

The flags are described below.

#### (1) Carry flag (CY)

The carry flag records a carry or borrow resulting from an operation.

This flag also records the shifted-out value when a shift/rotate instruction is executed, and functions as a bit accumulator when a bit-manipulation instruction is executed.

The status of the CY flag can be tested with a conditional branch instruction.

#### (2) Parity/overflow flag (P/V)

The P/V flag performs the following two kinds of operation associated with execution of an operation instruction.

The status of the P/V flag can be tested with a conditional branch instruction.

- Parity flag operation

Set (1) when the number of bits set (1) as the result of execution of a logical operation instruction, shift/rotate instruction, or a CHKL or CHKLA instruction is even, and reset (0) if odd. When a 16-bit shift instruction is executed, however, only the low-order 8 bits of the operation result are valid for the parity flag.

- Overflow flag operation

Set (1) only when the numeric range expressed as a two's complement is exceeded as the result of execution of an arithmetic operation instruction, and reset (0) otherwise. More specifically, the value of this flag is the exclusive OR of the carry into the MSB and the carry out of the MSB. For example, the two's complement range in an 8-bit arithmetic operation is 80H (–128) to 7FH (+127), and the flag is set (1) if the operation result is outside this range, and reset (0) if within this range.

**Example** The operation of the overflow flag when an 8-bit addition instruction is executed is shown below.

When the addition of 78H (+120) and 69H (+105) is performed, the operation result is E1H (+225), and the two's complement limit is exceeded, with the result that the P/V flag is set (1). Expressed as a two's complement, E1H is -31.

$$\begin{array}{rcl}
 78\text{H } (+120) & = & 0111 \ 1000 \\
 +) \ 69\text{H } (+105) & = & +) \ 0110 \ 1001 \\
 \hline
 & & 0 \ 1110 \ 0001 = -31 \quad \text{P/V} = 1 \\
 & \uparrow & \\
 & \text{CY} &
 \end{array}$$

When the following two negative numbers are added together, the operation result is within the two's complement range, and therefore the P/V flag is reset (0).

$$\begin{array}{rcl}
 \text{FBH } (-5) & = & 1111 \ 1011 \\
 +) \ \text{F0H } (-16) & = & +) \ 1111 \ 0000 \\
 \hline
 & & 1 \ 1110 \ 1011 = -21 \quad \text{P/V} = 0 \\
 & \uparrow & \\
 & \text{CY} &
 \end{array}$$

### (3) Interrupt request enable flag (IE)

This flag controls CPU interrupt request acknowledgment operations.

When "0", interrupts are disabled, and only non-maskable interrupts and unmasked macro service can be acknowledged. All other interrupts are disabled.

When "1", the interrupt enabled state is set, and enabling of interrupt request acknowledgment is controlled by the interrupt mask flags corresponding to the individual interrupt requests and the priority of the individual interrupts.

The IE flag is set (1) by execution of an EI instruction, and reset (0) by execution of a DI instruction or acknowledgment of an interrupt.

### (4) Auxiliary carry flag (AC)

The AC flag is set (1) when there is a carry out of bit 3 or a borrow into bit 3 as the result of an operation, and reset (0) otherwise.

This flag is used when the ADJBA or ADJBS instruction is executed.

**(5) Register set selection flag (RSS)**

The RSS flag specifies the general-purpose registers that function as X, A, C and B, and the general-purpose register pairs (16-bit) that function as AX and BC.

This flag is provided to maintain compatibility with the 78K/III series, and must be set to 0 except when using a 78K/III series program.

**(6) Zero flag (Z)**

The Z flag records the fact that the result of an operation is "0".

It is set (1) when the result of an operation is "0", and reset (0) otherwise. The status of the Z flag can be tested with a conditional branch instruction.

**(7) Sign flag (S)**

The S flag records the fact that the MSB is "1" as the result of an operation.

It is set (1) when the MSB is "1" as the result of an operation, and reset (0) otherwise. The status of the S flag can be tested with a conditional branch instruction.

**(8) Register bank selection flag (RBS0 to RBS2)**

This is a 3-bit flag used to select one of the 8 register banks (register bank 0 to register bank 7) (see **Table 3-3**).

It stores 3-bit information which indicates the register bank selected by execution of a SEL RBn instruction, etc.

**Table 3-3 Register Bank Selection**

RBS2	RBS1	RBS0	Specified Register Bank
0	0	0	Register bank 0
0	0	1	Register bank 1
0	1	0	Register bank 2
0	1	1	Register bank 3
1	0	0	Register bank 4
1	0	1	Register bank 5
1	1	0	Register bank 6
1	1	1	Register bank 7

**(9) User flag (UF)**

This flag can be set and reset in the user program, and used for program control.

### 3.7.3 Use of RSS Bit

Basically, the RSS bit should be fixed at 0 at all times.

The following explanation refers to the case where a 78K/III series program is used, and the program used sets the RSS bit to 1. This explanation can be skipped if the RSS bit is fixed at 0.

The RSS bit is provided to allow the functions of A (R1), X (R0), B (R3), C (R2), AX (RP0) and BC (RP1) to be used by registers R4 to R7 (RP2, RP3) as well. Effective use of this bit enables efficient programs to be written in terms of program size and program execution.

However, careless use can result in unforeseen problems. Therefore, the RSS bit should always be set to 0. The RSS bit should only be set to 1 when a 78K/III series program is used.

Use of the RSS bit set to 0 in all programs will improve programming and debugging efficiency.

Even when using a program in which the RSS bit set to 1 is used, it is recommended that the program be amended if possible so that it does not set the RSS bit to 1.

#### (1) RSS bit recommendations

- Registers used by instructions for which the A, X, B, C and AX registers are directly entered in the operand column of the operation list (see 25.2.)
- Registers specified as implied by instructions that use the A, AX, B and C registers by means of implied addressing
- Registers used in addressing by instructions that use the A, B and C registers in indexed addressing and based indexed addressing

The registers used in these cases are switched as follows according to the RSS bit.

- When RSS = 0  
A→R1, X→R0, B→R3, C→R2, AX→RP0, BC→RP1
- When RSS = 1  
A→R5, X→R4, B→R7, C→R6, AX→RP2, BC→RP3

Registers used other than those mentioned above are always the same irrespective of the value of the RSS bit. With the NEC assembler (RA78K4), the register operation code generated when the A, X, B, C, AX and BC registers are described by those names is determined by the assembler RSS pseudo-instruction.

When the RSS bit is set or reset, an RSS pseudo-instruction must be written immediately before (or immediately after) the relevant instruction (see example below).

**<Program example>**

- When RSS is set to 0

```
RSS 0      ; RSS pseudo-instruction
CLR1 PSWL.5
MOV B, A    ; This code is equivalent to "MOV R3, R1".
```

- When RSS is set to 1

```
RSS 1      ; RSS pseudo-instruction
SET1 PSWL.5
MOV B, A    ; This code is equivalent to "MOV R7, R5".
```

**(2) Operation code generation method with RA78K4**

- With RA78K4, if there is an instruction with the same function as an instruction for which A or AX is directly entered in the operand column of the instruction operation list, the operation code for which A or AX is directly entered in the operand column is generated first.

**Example** The function is the same when B is used as r in a MOV A,r instruction, and when A is used as r and B is used as r' in a MOVr,r' instruction, and the same code (MOV,A,B) is used in the assembler source program. In this case, RA78K4 generates code equivalent to the MOV A, r instruction.

- If A, X, B, C, AX or BC is written in an instruction for which r, r', rp and rp' are specified in the operand column, the A, X, B, C, AX and BC instructions generate an operation code that specifies the following registers according to the operand of the RA78K4 RSS pseudo-instruction.

Register	RSS = 0	RSS = 1
A	R1	R5
X	R0	R4
B	R3	R7
C	R2	R6
AX	RP0	RP2
BC	RP1	RP3

- If R0 to R7 or RP0 to RP4 is written as r, r', rp or rp' in the operand column, an operation code in accordance with that specification is output (an operation code for which A or AX is directly entered in the operand column is not output.)
- R1, R3, R2 or R5, R7, R6 cannot be used for registers A, B and C used in indexed addressing and based indexed addressing.

### (3) Operating precautions

Switching the RSS bit has the same effect as having two register sets. However, when writing a program, care must be taken to ensure that the static program code and dynamic RSS bit changes at the time of program execution always coincide. Also, a program that sets RSS to 1 cannot be used by a program that uses the context switching function, and therefore program usability is poor. Moreover, since different registers are used with the same name, program readability is poor and debugging is difficult. Therefore, if it is necessary to set RSS to 1, these disadvantages must be fully taken into consideration when writing a program.

A register not specified by the RSS bit can be accessed by writing its absolute name.



**3.7.4 Stack Pointer (SP)**

The stack pointer is a 24-bit register that holds the start address of the stack area (LIFO type: 00000H to FFFFFFFH) (see **Figure 3-9**). It is used to address the stack area when subroutine processing or interrupt processing is performed. Be sure to write "0" in the high-order 4 bits.

The contents of the SP are decremented before a write to the stack area and incremented after a read from the stack area (see **Figures 3-10 and 3-11**).

The SP is accessed by dedicated instructions.

The SP contents are undefined after  $\overline{\text{RESET}}$  input, and therefore the SP must always be initialized by an initialization program directly after reset release (before a subroutine call or interrupt acknowledgment).

**Example** SP initialization

MOVG SP, #0FEE0H; SP  $\leftarrow$  0FEE0H (when used from FEDFH)

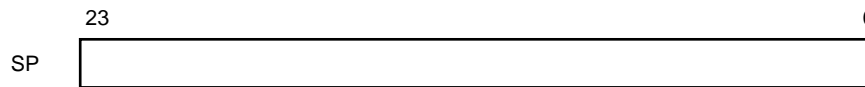
**Figure 3-9 Stack Pointer (SP) Format**

Figure 3-10 Data Saved to Stack Area

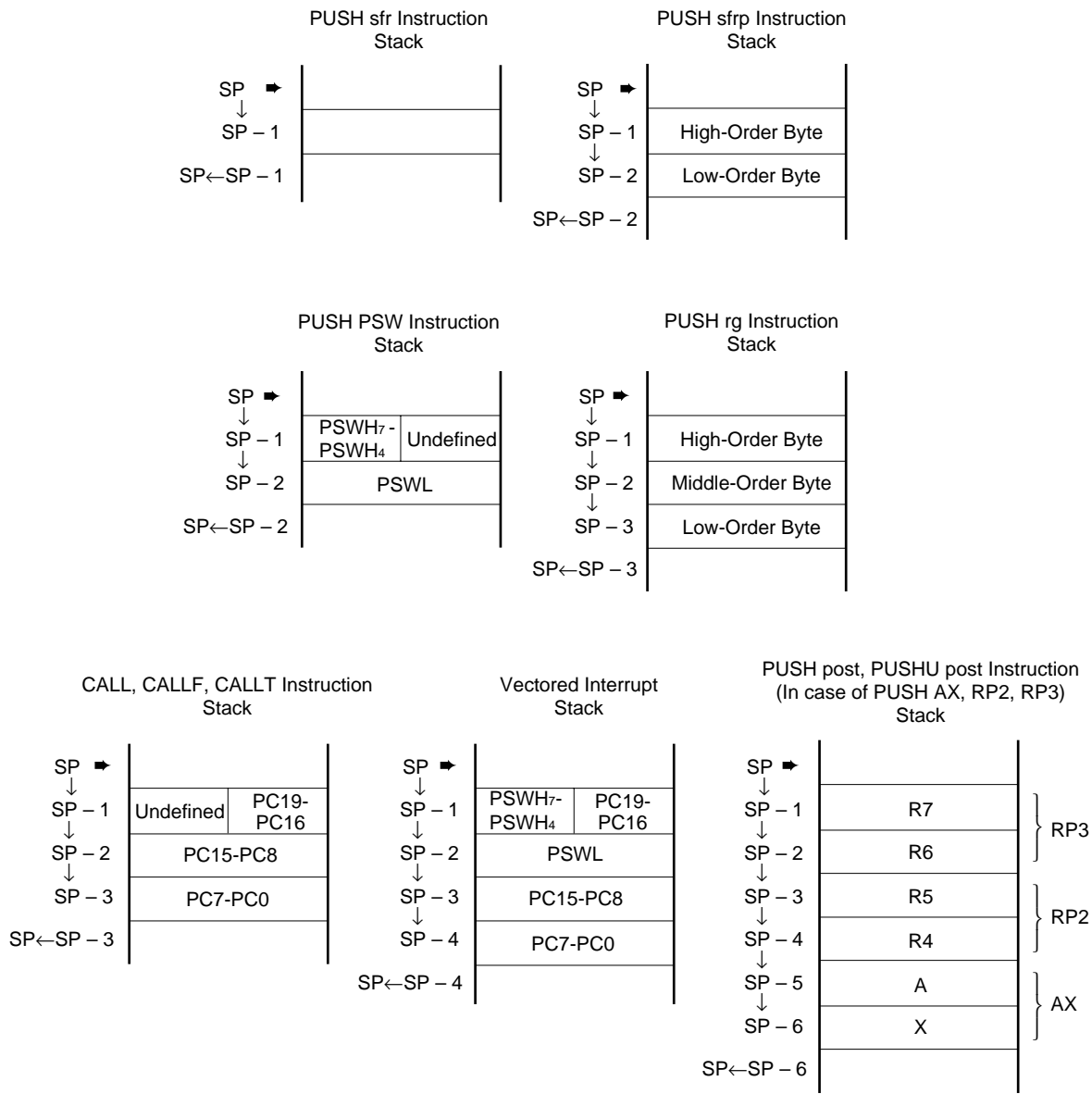
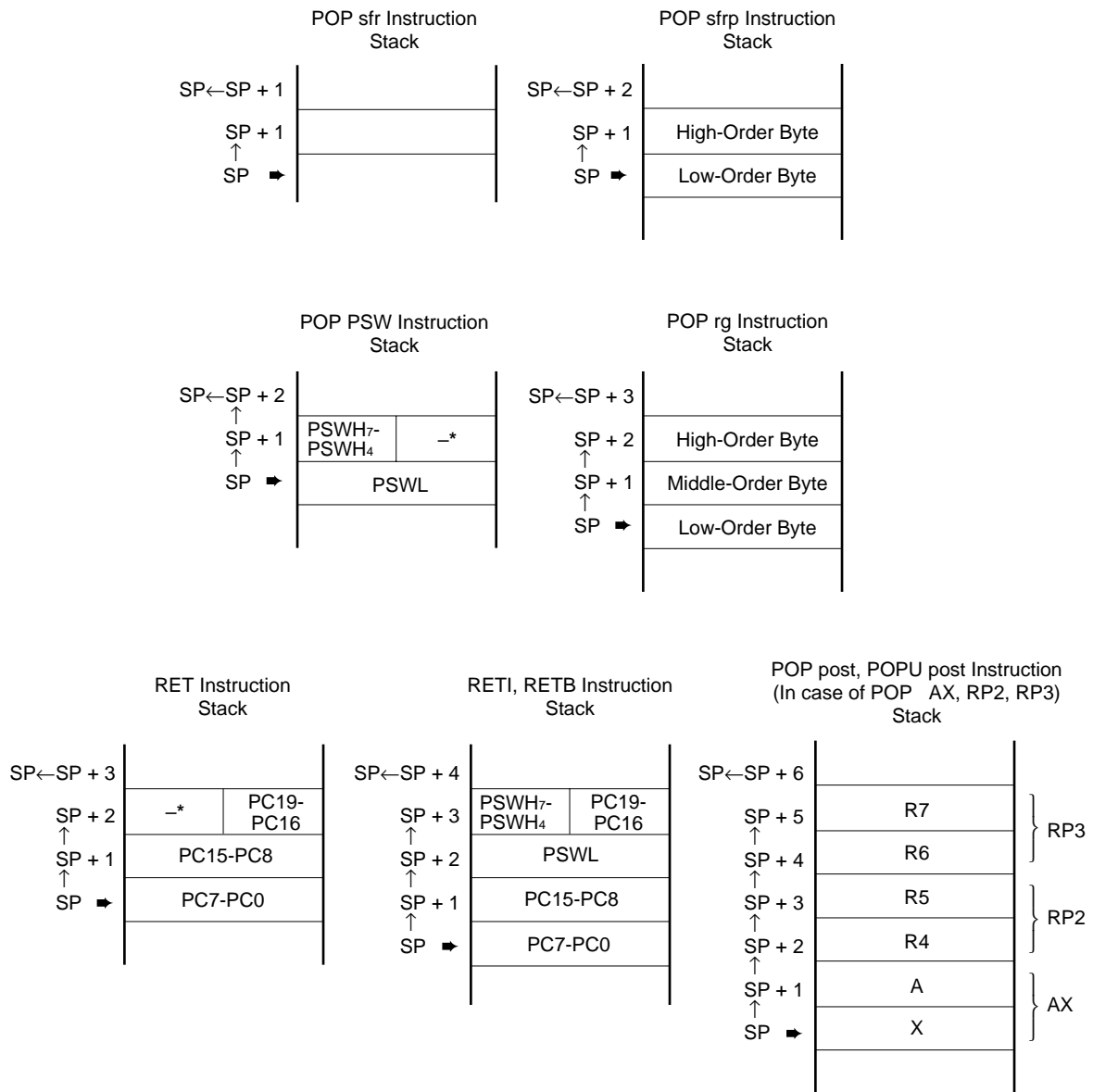


Figure 3-11 Data Restored from Stack Area



\* This 4-bit data is ignored.

- Cautions**
1. With stack addressing, the entire 1 M-byte space can be accessed but a stack area cannot be reserved in the SFR area or internal ROM area.
  2. The stack pointer (SP) is undefined after RESET input. Moreover, non-maskable interrupts can still be acknowledged when the SP is in an undefined state. An unanticipated operation may therefore be performed if a non-maskable interrupt request is generated when the SP is in the undefined state directly after reset release. To avoid this risk, the program after reset release must be written as follows.

```

RSTVCT    CSEG    AT      0
           DW      RSTSTRT
           to
INITSEG    CSEG    BASE
RSTSTRT : LOCATION 0H ; or LOCATION 0FH
           MOVG SP, #STKBGN

```

### 3.8 GENERAL-PURPOSE REGISTERS

#### 3.8.1 Configuration

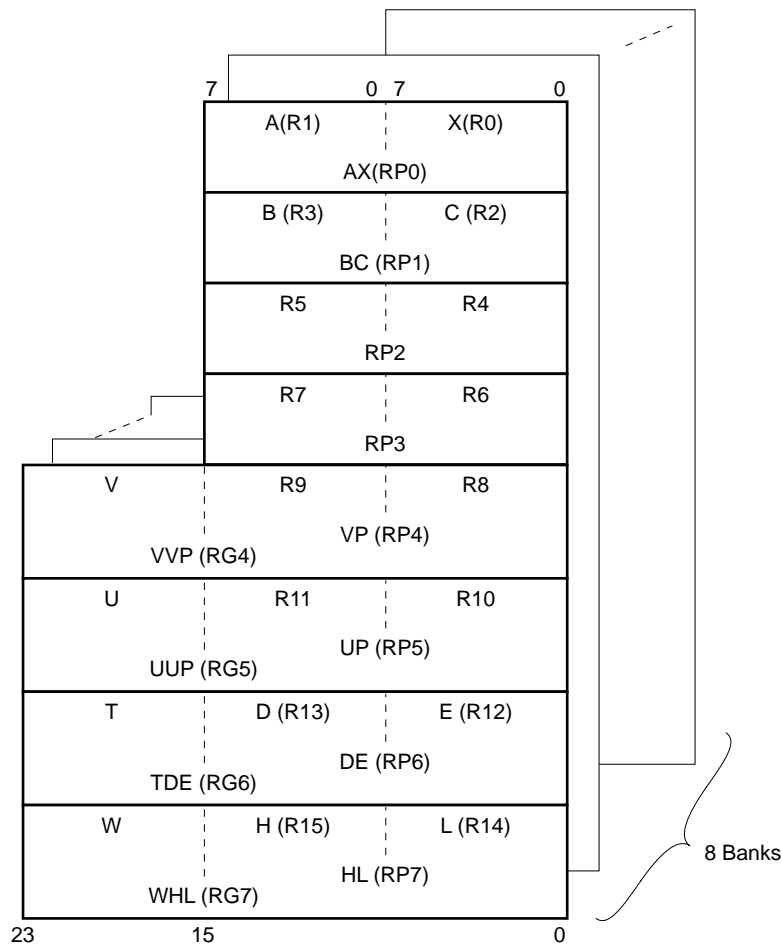
There are sixteen 8-bit general-purpose registers, and two 8-bit general-purpose registers can be used together as a 16-bit general-purpose register. In addition, four of the 16-bit general-purpose registers can be combined with an 8-bit register for address extension, and used as 24-bit address specification registers.

General-purpose registers other than the V, U, T and W registers for address extension are mapped onto internal RAM.

These register sets are provided in 8 banks, and can be switched by means of software or the context switching function.

Upon  $\overline{\text{RESET}}$  input, register bank 0 is selected. The register bank used during program execution can be checked by reading the register bank selection flag (RBS0, RBS1, RBS2) in the PSW.

**Figure 3-12 General-Purpose Register Format**



**Remark** Absolute names are shown in parentheses.

Figure 3-13 General-Purpose Register Addresses

		8-Bit Processing		16-Bit Processing	
FEFFH*	RBNK0	H(R15) <sub>(FH)</sub>	L(R14) <sub>(EH)</sub>	HL(RP7) <sub>(EH)</sub>	
	RBNK1	D(R13) <sub>(DH)</sub>	E(R12) <sub>(CH)</sub>	DE(RP6) <sub>(CH)</sub>	
	RBNK2	R11 <sub>(BH)</sub>	R10 <sub>(AH)</sub>	UP(RP5) <sub>(AH)</sub>	
	RBNK3	R9 <sub>(9H)</sub>	R8 <sub>(8H)</sub>	VP(RP4) <sub>(8H)</sub>	
	RBNK4	R7 <sub>(7H)</sub>	R6 <sub>(6H)</sub>	RP3 <sub>(6H)</sub>	
	RBNK5	R5 <sub>(5H)</sub>	R4 <sub>(4H)</sub>	RP2 <sub>(4H)</sub>	
	RBNK6	B(R3) <sub>(3H)</sub>	C(R2) <sub>(2H)</sub>	BC(RP1) <sub>(2H)</sub>	
FE80H*	RBNK7	A(R1) <sub>(1H)</sub>	X(R0) <sub>(0H)</sub>	AX(RP0) <sub>(0H)</sub>	
		7	0 7	15	0

\* When the LOCATION 0 instruction is executed. When the LOCATION 0FH instruction is executed, 0F0000H should be added to the address values shown above.

**Caution** R4, R5, R6, R7, RP2 and RP3 can be used as the X, A, C, B, AX and BC registers respectively by setting the RSS bit of the PSW to 1, but this function should only be used when using a 78K/III series program.

**Remark** When the register bank is changed, and it is necessary to return to the original register bank, an SEL RBN instruction should be executed after saving the PSW to the stack with a PUSH PSW instruction. When returning to the original register bank, if the stack location does not change the POP PSW instruction should be used. When the register bank is changed by a vectored interrupt service program, etc., the PSW is automatically saved to the stack when an interrupt is acknowledged and restored by an RETI or RETB instruction, so that, if only one register bank is used in the interrupt service routine, only an SEL RBN instruction needs be executed, and execution of a PUSH PSW and POP PSW instruction is not necessary.

**Example** When register bank 2 is specified

```

      ⋮
      ⋮
      ⋮
PUSH PSW
SEL RB2
      ⋮
      ⋮
      ⋮
POP PSW
      ⋮
      ⋮
      ⋮

```

Operations in register bank 2

Operations in original register bank

### 3.8.2 Functions

In addition to being manipulated in 8-bit units, the general-purpose registers can also be manipulated in 16-bit units by pairing two 8-bit registers. Also, four of the 16-bit registers can be combined with an 8-bit register for address extension and manipulated in 24-bit units.

Each register can be used in a general-purpose way for temporary storage of an operation result and as the operand of an inter-register operation instruction.

The area from 0FE80H to 0FEFFH (when the LOCATION 0 instruction is executed; 0FFE80H to 0FFEFFH when the LOCATION 0FH instruction is executed) can be given an address specification and accessed as ordinary data memory irrespective of whether or not it is used as the general-purpose register area.

As 8 register banks are provided in the 78K/IV series, efficient programs can be written by using different register banks for normal processing and processing in the event of an interrupt.

The registers have the following specific functions.

#### A (R1):

- Register mainly used for 8-bit data transfers and operation processing. Can be used in combination with all addressing modes for 8-bit data.
- Can also be used for bit data storage.
- Can be used as the register that stores the offset value in indexed addressing and based indexed addressing.

#### X (R0):

- Can be used for bit data storage.

#### AX (RP0):

- Register mainly used for 16-bit data transfers and operation processing. Can be used in combination with all addressing modes for 16-bit data.

#### AXDE:

- Used for 32-bit data storage when a DIVUX, MACW or MACSW instruction is executed.

#### B (R3):

- Has a loop counter function, and can be used by the DBNZ instruction.
- Can be used as the register that stores the offset value in indexed addressing and based indexed addressing.
- Used as the MACW and MACSW instruction data pointer.

#### C (R2):

- Has a loop counter function, and can be used by the DBNZ instruction.
- Can be used as the register that stores the offset value in based indexed addressing.
- Used as the counter in a string instruction and the SACW instruction.
- Used as the MACW and MACSW instruction data pointer.

#### RP2:

- Used to save the low-order 16 bits of the program counter (PC) when context switching is used.

#### RP3:

- Used to save the high-order 4 bits of the program counter (PC) and the program status word (PSW) (excluding bits 0 to 3 of PSWH) when context switching is used.

**VVP (RG4):**

- Has a pointer function, and operates as the register that specifies the base address in register indirect addressing, based addressing and based indexed addressing.

**UUP (RG5):**

- Has a user stack pointer function, and enables a stack separate from the system stack to be implemented by means of the PUSHU and POPU instructions.
- Has a pointer function, and operates as the register that specifies the base address in register indirect addressing and based addressing.

**DE (RP6), HL (RP7):**

- Operate as the registers that store the offset value in indexed addressing and based indexed addressing.

**TDE (RG6):**

- Has a pointer function, and operates as the register that specifies the base address in register indirect addressing and based addressing.
- Used as the pointer in a string instruction and the SACW instruction.

**WHL (RG7):**

- Register used mainly for 24-bit data transfers and operation processing.
- Has a pointer function, and operates as the register that specifies the base address in register indirect addressing and based addressing.
- Used as the pointer in a string instruction and the SACW instruction.



In addition to the function name that emphasizes the specific function of the register (X, A, C, B, E, D, L, H, AX, BC, VP, UP, DE, HL, VVP, UUP, TDE, WHL), each register can also be written by its absolute name (R0 to R15, RP0 to RP7, RG4 to RG7). The correspondence between these names is shown in Table 3-4.

**Table 3-4 Correspondence between Function Names and Absolute Names**

**(a) 8-bit registers**

Absolute Name	Function Name	
	RSS = 0	RSS = 1*
R0	X	
R1	A	
R2	C	
R3	B	
R4		X
R5		A
R6		C
R7		B
R8		
R9		
R10		
R11		
R12	E	E
R13	D	D
R14	L	L
R15	H	H

**(b) 16-bit registers**

Absolute Name	Function Name	
	RSS = 0	RSS = 1*
RP0	AX	
RP1	BC	
RP2		AX
RP3		BC
RP4	VP	VP
RP5	UP	UP
RP6	DE	DE
RP7	HL	HL

**(c) 24-bit registers**

Absolute Name	Function Name
RG4	VVP
RG5	UUP
RG6	TDE
RG7	WHL

\* RSS should only be set to 1 when a 78K/III series program is used.

**Remark** R8 to R11 have no function name.

### 3.9 SPECIAL FUNCTION REGISTERS (SFRS)

These are registers to which a special function is assigned, such as internal peripheral hardware mode registers, control registers, etc. They are mapped onto the 256-byte space from 0FF00H to 0FFFFH\*.

- \* When the LOCATION 0 instruction is executed. When the LOCATION 0FH instruction is executed, the area is FFF00H to FFFFFH.

**Caution** Addresses onto which SFRs are not assigned should not be accessed in this area. If such an address is as accessed by mistake, the  $\mu$ PD784026 may become deadlocked. A deadlock can only be released by reset input.

A list of special function registers (SFRs) is given in Table 3-5. The meaning of the items in the table is as explained below.

- Symbol ..... Symbol that indicates the incorporated SFR. This is a reserved word in the NEC assembler (RA78K4). With the C compiler (CC78K4), this symbol can be used as an sfr variable by means of a #pragma sfr command.
- R/W ..... Indicates whether the corresponding SFR is read/write enabled.
  - R/W: Read/write enabled
  - R : Read-only
  - W : Write-only
- Manipulable Bit Units ..... Indicates the applicable manipulation bit units when the corresponding SFR is manipulated.
  - A 16-bit-manipulable SFR can be written in the operand "sfrp", and when specified by an address, an even address is specified.
  - A bit-manipulable SFR can be written in a bit manipulation instruction.
- On Reset ..... Indicates the status of the register after RESET input.

Table 3-5 List of Special Function Registers (SFRs) (1/5)

Address*1	Special Function Register (SFR) Name		Symbol	R/W	Manipulable Bit Units			On Reset
					1 Bit	8 Bits	16 Bits	
0FF00H	Port 0		P0	R/W	○	○	—	Undefined
0FF01H	Port 1		P1		○	○	—	
0FF02H	Port 2		P2	R	○	○	—	
0FF03H	Port 3		P3	R/W	○	○	—	
0FF04H	Port 4*2		P4		○	○	—	
0FF05H	Port 5*2		P5		○	○	—	
0FF06H	Port 6		P6		○	○	—	00H
0FF07H	Port 7		P7	R/W	○	○	—	Undefined
0FF0EH		Port 0 buffer register L	P0L		○	○	—	
0FF0FH	Port 0 buffer register H		P0H		○	○	—	
0FF10H	Compare register (timer/counter 0)		CR00		—	—	○	
0FF12H	Capture/compare register (timer/counter 0)		CR01		—	—	○	
0FF14H	Compare register L (timer/counter 1)		CR10		CR10W	—	○	○
0FF15H	Compare register H (timer/counter 1)		—			—	—	
0FF16H	Capture/compare register L (timer/counter 1)		CR11		CR11W	—	○	○
0FF17H	Capture/compare register H (timer/counter 1)		—			—	—	
0FF18H	Compare register L (timer/counter 2)		CR20		CR20W	—	○	○
0FF19H	Compare register H (timer/counter 2)		—			—	—	
0FF1AH	Capture/compare register L (timer/counter 2)		CR21		CR21W	—	○	○
0FF1BH	Capture/compare register H (timer/counter 2)		—			—	—	
0FF1CH	Compare register L (timer 3)		CR30		CR30W	—	○	○
0FF1DH	Compare register H (timer 3)		—			—	—	

\*1. When the LOCATION 0 instruction is executed. When the LOCATION 0FH instruction is executed, "F0000H" should be added to the value shown.

2. These ports are not available on the  $\mu$ PD784020 and 784021.

Table 3-5 List of Special Function Registers (SFRs) (2/5)

Address*1	Special Function Register (SFR) Name	Symbol	R/W	Manipulable Bit Units			On Reset
				1 Bit	8 Bits	16 Bits	
0FF20H	Port 0 mode register	PM0	R/W	○	○	—	FFH
0FF21H	Port 1 mode register	PM1		○	○	—	
0FF23H	Port 3 mode register	PM3		○	○	—	
0FF24H	Port 4 mode register*2	PM4		○	○	—	
0FF25H	Port 5 mode register*2	PM5		○	○	—	
0FF26H	Port 6 mode register	PM6		○	○	—	
0FF27H	Port 7 mode register	PM7		○	○	—	
0FF2EH	Real-time output port control register	RTPC		○	○	—	00H
0FF30H	Capture/compare control register 0	CRC0		—	○	—	10H
0FF31H	Timer output control register	TOC		○	○	—	00H
0FF32H	Capture/compare control register 1	CRC1		—	○	—	
0FF33H	Capture/compare control register 2	CRC2		—	○	—	10H
0FF36H	Capture register (timer/counter 0)	CR02	R	—	—	○	0000H
0FF38H	Capture register L (timer/counter 1)	CR12		—	○	○	
0FF39H	Capture register H (timer/counter 1)	—		—	—	—	
0FF3AH	Capture register L (timer/counter 2)	CR22		—	○	○	
0FF3BH	Capture register H (timer/counter 2)	—		—	—	—	
0FF41H	Port 1 mode control register	PMC1	R/W	○	○	—	00H
0FF43H	Port 3 mode control register	PMC3		○	○	—	
0FF4EH	Pull-up resistor option register	PUO		○	○	—	

\* 1. When the LOCATION 0 instruction is executed. When the LOCATION 0FH instruction is executed, "F0000H" should be added to the value shown.

2. These ports are not available on the  $\mu$ PD784020 and 784021.

**Table 3-5 List of Special Function Registers (SFRs) (3/5)**

Address*1	Special Function Register (SFR) Name	Symbol		R/W	Manipulable Bit Units			On Reset
					1 Bit	8 Bits	16 Bits	
0FF50H	Timer register 0	TM0*2		R	—	—	○	0000H
0FF51H					—	—		
0FF52H	Timer register 1	TM1*2	TM1W		—	○	○	
0FF53H		—	*2		—	—		
0FF54H	Timer register 2	TM2*2	TM2W		—	○	○	
0FF55H		—	*2		—	—		
0FF56H	Timer register 3	TM3*2	TM3W		—	○	○	
0FF57H		—	*2		—	—		
0FF5CH	Prescaler mode register 0	PRM0		R/W	—	○	—	11H
0FF5DH	Timer control register 0	TMC0			○	○	—	00H
0FF5EH	Prescaler mode register 1	PRM1W			—	○	—	11H
0FF5FH	Timer control register 1	TMC1			○	○	—	00H
0FF60H	D/A conversion value setting register 0	DACS0			—	○	—	
0FF61H	D/A conversion value setting register 1	DACS1			—	○	—	
0FF62H	D/A converter mode register	DAM			—	○	—	03H
0FF68H	A/D converter mode register	ADM			○	○	—	00H
0FF6AH	A/D conversion result register	ADCR		R	—	○	—	Undefined
0FF70H	PWM control register	PWMC		R/W	—	○	—	05H
0FF71H	PWM prescaler register	PWPR			—	○	—	00H
0FF72H	PWM modulo register 0	PWM0			—	—	○	Undefined
0FF74H	PWM modulo register 1	PWM1			—	—	○	
0FF7DH	One-shot pulse output control register	OSPC			○	○	—	00H
0FF80H	Serial bus interface control register	SBIC			○	○	—	
0FF82H	Clocked serial interface mode register	CSIM			○	○	—	
0FF84H	Clocked serial interface mode register 1	CSIM1			○	○	—	
0FF85H	Clocked serial interface mode register 2	CSIM2			○	○	—	

\* 1. When the LOCATION 0 instruction is executed. When the LOCATION 0FH instruction is executed, "F0000H" should be added to the value shown.

2. Although the timer register can be read, if counting up the timer and reading the timer register conflict, the read value of the timer register may be illegal. Therefore, do not read the timer register if at all possible.

If it is necessary to read the timer register, read it two or more times in a row in software, and assume the value of the timer register is correct only when all the read values match.

Table 3-5 List of Special Function Registers (SFRs) (4/5)

Address*1	Special Function Register (SFR) Name	Symbol	R/W	Manipulable Bit Units			On Reset
				1 Bit	8 Bits	16 Bits	
0FF86H	Serial shift register	SIO	R/W	—	○	—	Undefined
0FF88H	Asynchronous serial interface mode register	ASIM		○	○	—	00H
0FF89H	Asynchronous serial interface mode register 2	ASIM2		○	○	—	
0FF8AH	Asynchronous serial interface status register	ASIS	R	○	○	—	
0FF8BH	Asynchronous serial interface status register 2	ASIS2		○	○	—	
0FF8CH	Serial receive buffer: UART0	RXB		—	○	—	Undefined
	Serial transmit shift register: UART0	TXS	W	—	○	—	
	Serial shift register: IOE1	SIO1	R/W	—	○	—	
0FF8DH	Serial receive buffer: UART2	RXB2	R	—	○	—	Undefined
	Serial transmit shift register: UART2	TXS2	W	—	○	—	
	Serial shift register: IOE2	SIO2	R/W	—	○	—	
0FF90H	Baud rate generator control register	BRGC		—	○	—	00H
0FF91H	Baud rate generator control register 2	BRGC2		—	○	—	
0FFA0H	External interrupt mode register 0	INTM0		○	○	—	
0FFA1H	External interrupt mode register 1	INTM1		○	○	—	
0FFA4H	Sampling clock selection register	SCS0		—	○	—	
0FFA8H	In-service priority register	ISPR	R	○	○	—	80H
0FFAAH	Interrupt mode control register	IMC	R/W	○	○	—	
0FFACH	Interrupt mask register 0L	MK0L		○	○	○	
0FFADH	Interrupt mask register 0H	MK0H		○	○	—	FFFFH
0FFAEH	Interrupt mask register 1L	MK1L		○	○	—	FFH
0FFC0H	Standby control register	STBC		—	○*2	—	30H
0FFC2H	Watchdog timer mode register	WDM		—	○*2	—	00H

\*1. When the LOCATION 0 instruction is executed. When the LOCATION 0FH instruction is executed, "F0000H" should be added to the value shown.

- The write operation is possible by using the dedicated instruction "MOV STBC, #byte instruction" or "MOV WDM, #byte instruction" only. Instructions other than these cannot perform the write operation.

Table 3-5 List of Special Function Registers (SFRs) (5/5)

Address*1	Special Function Register (SFR) Name	Symbol	R/W	Manipulable Bit Units			On Reset
				1 Bit	8 Bits	16 Bits	
0FFC4H	Memory extension mode register	MM	R/W	○	○	—	20H
0FFC5H	Hold mode register	HLDM		○	○	—	00H
0FFC6H	Clock output mode register	CLOM		○	○	—	
0FFC7H	Programmable wait control register 1	PWC1		—	○	—	AAH
0FFC8H	Programmable wait control register 2	PWC2		—	—	○	AAAAH
0FFCCH	Refresh mode register	RFM		○	○	—	00H
0FFCDH	Refresh area specification register	RFA		○	○	—	
0FFCEH	Oscillation stabilization time specification register	OSTS		—	○	—	
0FFD0H to 0FFDFH	External SFR area	—		○	○	—	—
0FFE0H	Interrupt control register (INTP0)	PIC0		○	○	—	43H
0FFE1H	Interrupt control register (INTP1)	PIC1		○	○	—	
0FFE2H	Interrupt control register (INTP2)	PIC2		○	○	—	
0FFE3H	Interrupt control register (INTP3)	PIC3		○	○	—	
0FFE4H	Interrupt control register (INTC00)	CIC00		○	○	—	
0FFE5H	Interrupt control register (INTC01)	CIC01		○	○	—	
0FFE6H	Interrupt control register (INTC10)	CIC10		○	○	—	
0FFE7H	Interrupt control register (INTC11)	CIC11		○	○	—	
0FFE8H	Interrupt control register (INTC20)	CIC20		○	○	—	
0FFE9H	Interrupt control register (INTC21)	CIC21		○	○	—	
0FFEAH	Interrupt control register (INTC30)	CIC30		○	○	—	
0FFEBH	Interrupt control register (INTP4)	PIC4		○	○	—	
0FFECH	Interrupt control register (INTP5)	PIC5		○	○	—	
0FFEDH	Interrupt control register (INTAD)	ADIC		○	○	—	
0FFEEH	Interrupt control register (INTSER)	SERIC		○	○	—	
0FFEFH	Interrupt control register (INTSR)	SRIC		○	○	—	
	Interrupt control register (INTCSI1)	CSIIC1		○	○	—	
0FFF0H	Interrupt control register (INTST)	STIC		○	○	—	
0FFF1H	Interrupt control register (INTCSI)	CSIIC		○	○	—	
0FFF2H	Interrupt control register (INTSER2)	SERIC2		○	○	—	
0FFF3H	Interrupt control register (INTSR2)	SRIC2		○	○	—	
	Interrupt control register (INTCSI2)	CSIIC2		○	○	—	
0FFF4H	Interrupt control register (INTST2)	STIC2		○	○	—	
0FFFCH	Internal memory size switching register*2	IMS		—	○	—	FFH

\*1. When the LOCATION 0 instruction is executed. When the LOCATION 0FH instruction is executed, "F0000H" should be added to the value shown.

\*2. Writes to this register are only meaningful in the case of the  $\mu$ PD78P4026.

**3.10 CAUTIONS**

- (1) Program fetches cannot be performed from the internal high-speed RAM area (0FD00H to 0FEFFH when the LOCATION 0 instruction is executed; FFD00H to FFEFFH when the LOCATION 0FH instruction is executed).

- (2) Special function registers (SFRs)

Addresses onto which SFRs are not assigned should not be accessed in the area 0FF00H to 0FFFFH\*. If such an address is accessed by mistake, the  $\mu$ PD784026 may become deadlocked. A deadlock can only be released by reset input.

\* When the LOCATION 0 instruction is executed; FFF00H to FFFFFH when the LOCATION 0FH instruction is executed.

- (3) Stack pointer (SP) operation

With stack addressing, the entire 1 M-byte space can be accessed, but a stack area cannot be reserved in the SFR area or internal ROM area.

- (4) Stack pointer (SP) initialization

The SP is undefined after RESET input, while non-maskable interrupts can be acknowledged directly after reset release. Therefore, an unforeseen operation may be performed if a non-maskable interrupt request is generated while the SP is in the undefined state directly after reset release. To minimize this risk, the following program should be coded without fail after reset release.

```

RSTVCT    CSEG  AT    0
          DW    RSTSTRT
          to
INITSEG    CSEG  BASE
RSTSTRT :  LOCATION 0H ; or LOCATION 0FH
          MOVG SP, #STKBGN

```

- (5) Do not set external wait for the internal ROM area. If external wait is set for the internal ROM area, the CPU is deadlocked. This deadlock status can be cleared only by reset input.
- (6) If the  $\mu$ PD78P4026 is selected by the in-circuit emulator as the emulation CPU, the memory size is always the same as that of the  $\mu$ PD784026 even if an instruction that writes data other than FFH (EFH) to IMS is executed.

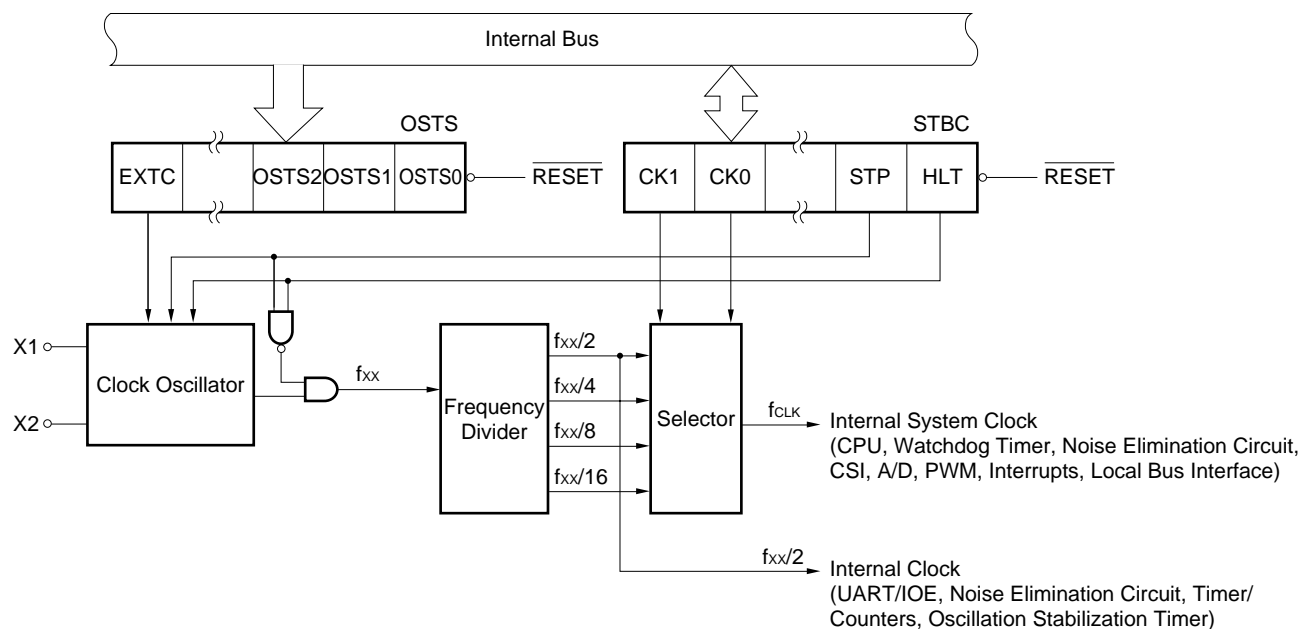


## CHAPTER 4 CLOCK GENERATOR

### 4.1 CONFIGURATION AND FUNCTION

The clock generator generates and controls the internal clock and internal system clock supplied to the CPU and internal hardware. The clock generator block diagram is shown in Figure 4-1.

**Figure 4-1 Clock Generator Block Diagram**



**Remark**  $f_{xx}$  : Crystal/ceramic oscillation frequency or internal clock frequency  
 $f_{CLK}$  : Internal system clock frequency

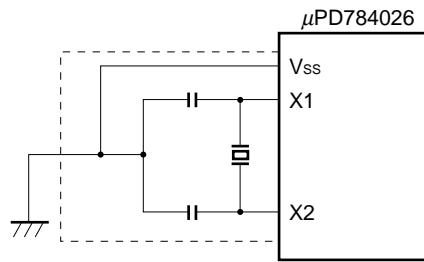
The clock oscillator oscillates by means of a crystal resonator/ceramic resonator connected to the X1 and X2 pins. When standby mode (STOP) is set, oscillation stops (see **CHAPTER 22 STANDBY FUNCTION**).

It is also possible to input an external clock. In this case, the clock signal is input to the X1 pin, and the inverse phase signal to the X2 pin.

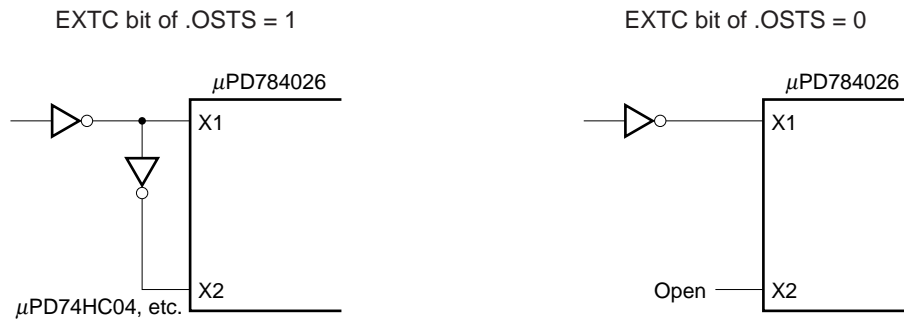
The frequency divider generates an internal system clock by 1/2, 1/4, 1/8 or 1/16 scaling of the clock oscillator output ( $f_{xx}$ ) according to the setting of the standby control register (STBC).

Figure 4-2 Clock Oscillator External Circuitry

## (a) Crystal/ceramic oscillation



## (b) External clock



- Cautions**
1. The oscillator should be as close as possible to the X1 and X2 pins.
  2. No other signal lines should pass through the area enclosed by the dotted line.

**Remark** Use of crystal resonator and ceramic resonator

Generally speaking, the oscillation frequency of a crystal resonator is extremely stable. It is therefore ideal for performing high-precision time management (in clocks, frequency meters, etc.).

A ceramic resonator is inferior to a crystal resonator in terms of oscillation frequency stability, but it has three advantages: a fast oscillation start-up time, small size, and low price. It is therefore suitable for general use (when high-precision time management is not required). In addition, there are products with an internal capacitor, etc., which enable the number of parts and mounting area to be reduced.

## 4.2 CONTROL REGISTERS

### 4.2.1 Standby Control Register (STBC)

STBC is a register used to set the standby mode and select the internal system clock. See **CHAPTER 22 STANDBY FUNCTION** for details of the standby modes.

To prevent erroneous entry into standby mode due to an inadvertent program loop, the STBC register can only be written to by a dedicated instruction. This instruction is the MOV STBC, #byte instruction, and has a special code configuration (4 bytes). A write is only performed if the 3rd and 4th bytes of the op code are mutual complements. If the 3rd and 4th bytes of the op code are not mutual complements, a write is not performed, and an op error interrupt is generated. In this case, the return address saved in the stack area is the address of the instruction which is the source of the error. The error source address can thus be found from the return address saved on the stack area.

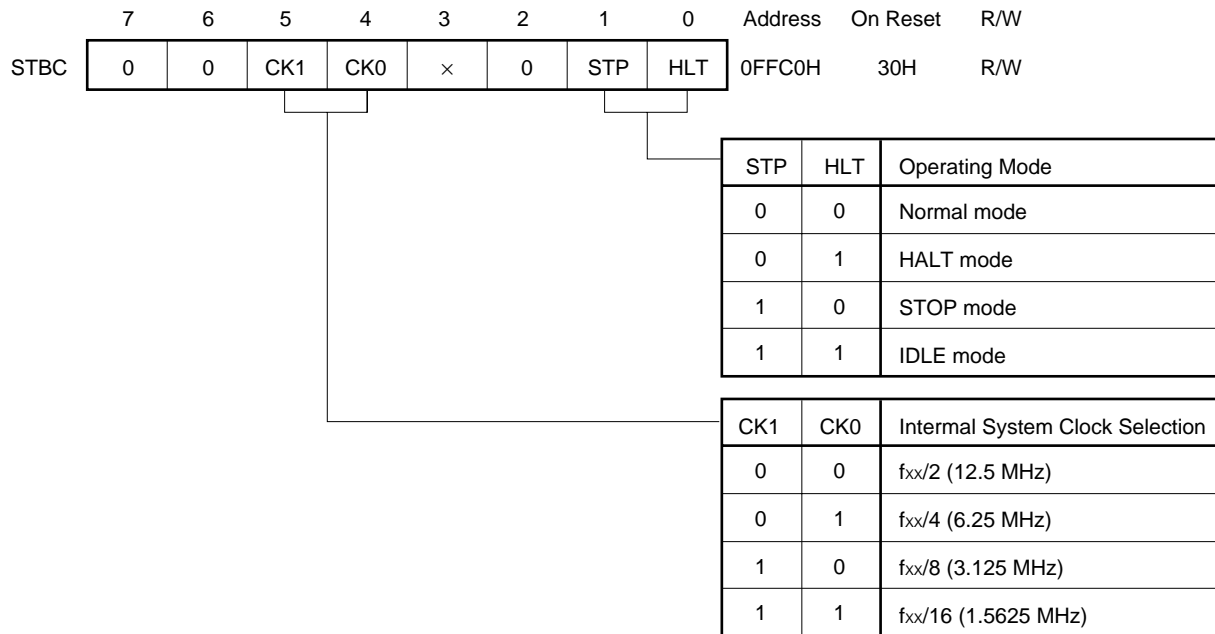
An endless loop will result if restore from an operand error is simply performed with an RETB instruction.

Since an operand error interrupt is only generated in the event of an inadvertent program loop (with the NEC assembler RA78K4, only the correct dedicated instruction is generated when the MOV STBC, #byte instruction is written), system initialization should be performed in software.

Other write instructions ("MOV STBC, A", "AND STBC, # byte", "SET1 STBC.7", etc.) are ignored, and no operation is performed. That is, a write is not performed on the STBC, and an interrupt such as an operand error interrupt is not generated. The STBC can be read at any time with a data transfer instruction.

$\overline{\text{RESET}}$  input sets the STBC register contents to 30H.

The format of the STBC is shown in Figure 4-3.

**Figure 4-3 Standby Control Register (STBC) Format**

**Caution** If the STOP mode is used when external clock input is used, the EXTC bit of the oscillation stabilization time specification register (OSTS) must be set (1) before setting the STOP mode. If the STOP mode is used when the EXTC bit of the OSTS is in the cleared (0) state when external clock input is used, the  $\mu$ PD784026 may be damaged or suffer reduced reliability.

When setting the EXTC bit to 1, be sure to input a clock in phase reverse to that of the clock input to the X1 pin, to the X2 pin.

**4.2.2 Oscillation Stabilization Time Specification Register (OSTS)**

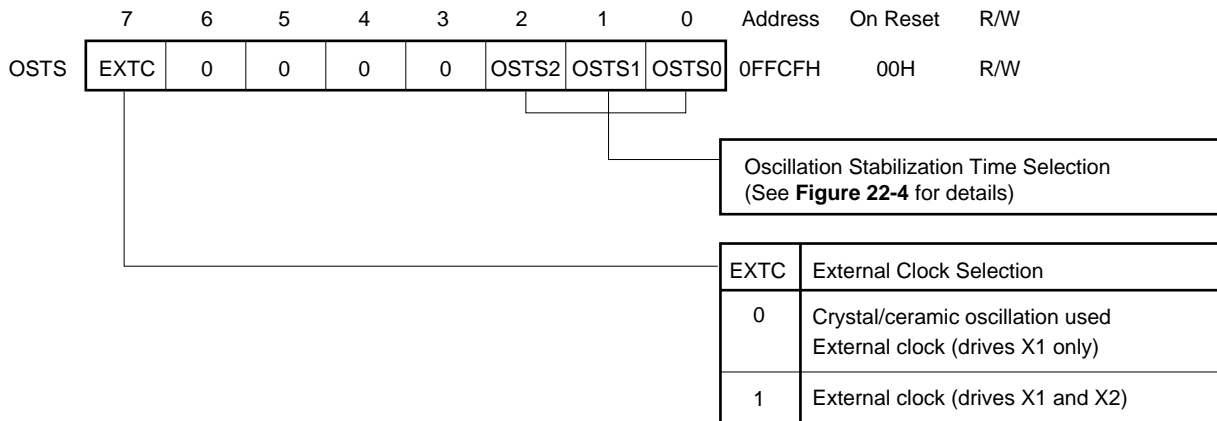
OSTS is a register used to specify the operation of the oscillator. The EXTC bit of the OSTS specifies whether a crystal/ceramic resonator or an external clock is used. The STOP mode can be set during use of external clock input, only when the EXTC bit is set (1).

The OSTS can be written to only by an 8-bit transfer instruction.

RESET input clears the OSTS register contents to 00H.

The format of the OSTS is shown in Figure 4-4.

**Figure 4-4 Oscillation Stabilization Time Specification Register (OSTS) Format**



- Cautions**
1. When using a crystal/ceramic oscillation, the EXTC bit must be cleared (0). If the EXTC bit is set (1), oscillation will stop.
  2. If the STOP mode is used with external clock input, the EXTC bit must be set (1) before setting the STOP mode. If the STOP mode is used when the EXTC bit is in the cleared (0) state, the  $\mu$ PD784026 may be damaged or suffer reduced reliability.
  3. When setting the EXTC bit to 1 during external clock input, be sure to input a clock in phase reverse to that of the clock input to the X1 pin, to the X2 pin. When the EXTC bit is set to 1, the  $\mu$ PD784026 operates on only the clock input to the X2 pin.

### 4.3 CLOCK GENERATOR OPERATION

#### 4.3.1 Clock Oscillator

##### (1) When using crystal/ceramic oscillation

The clock oscillator starts oscillating when the  $\overline{\text{RESET}}$  signal is input, and stops oscillation when the STOP mode is set by the standby control register (STBC). Oscillation is resumed when the STOP mode is released.

##### (2) When using external clock

The clock oscillator supplies the clock input from the X1 pin to the internal circuitry when the  $\overline{\text{RESET}}$  signal is input.

The oscillation circuit operates as follows when the EXTC bit of the oscillation stabilization time specification register (OSTS) is set to 1.

- The clock oscillator is set in the external clock input mode.
- The clock oscillator supplies the clock input to the X2 pin to the internal circuitry.
- The necessary circuit stops operating during the crystal/ceramic oscillation of the clock oscillator, to reduce the power dissipation.
- The STOP mode can be used even when the external clock is input.

- Cautions**
1. When using a crystal/ceramic oscillation, the EXTC bit of the Oscillation stabilization time specification register (OSTS) must be cleared (0). If the EXTC bit is set (1), oscillation will stop.
  2. If the STOP mode is used with external clock input, the EXTC bit of the OSTS must be set (1) before setting the STOP mode. If the STOP mode is used when the EXTC bit is in the cleared (0) state, not only will the clock generator consumption current not be reduced, but the  $\mu\text{PD784026}$  may also be damaged or suffer reduced reliability.
  3. When setting the EXTC bit of OSTS to 1, be sure to input a clock in phase reverse to that of the clock input to the X1 pin, to the X2 pin.

### 4.3.2 Frequency Divider

The frequency divider performs 1/2, 1/4, 1/8 or 1/16 scaling of the clock oscillator output, and supplies the resulting clock to the CPU, watchdog timer, noise elimination circuit, clocked serial interface (CSI), A/D converter, PWM, interrupt control circuit, and local bus interface. The division ratio is specified by the CK0 and CK1 bits of the standby control register (STBC).

Controlling the division ratio to match the speed required by the CPU enables the overall power consumption to be reduced. Also, the operating speed can be selected to match the supply voltage.

When  $\overline{\text{RESET}}$  is input, the lowest speed (1/16) is selected.

If the division ratio of the frequency divider circuit is changed, the maximum time shown in Table 4-1 is required to change the division ratio, depending on the clock selected before change.

Instruction execution continues even while the division ratio is changed, and the clock is supplied with the previous division ratio until the division ratio has been completely changed.

**Table 4-1 Time Required to Change Division Ratio**

Previous Division Ratio	Maximum Time Required for Change
1/2	$22/f_{xx}$
1/4	$24/f_{xx}$
1/8	$16/f_{xx}$
1/16	$16/f_{xx}$

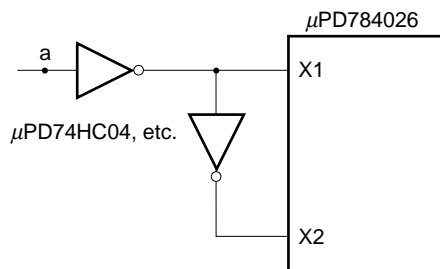
## 4.4 CAUTIONS

The following cautions apply to the clock generator.

### 4.4.1 When an External Clock Is Input

- (1) If the STOP mode is used with external clock input, the EXTC bit of the oscillation stabilization time specification register (OSTS) must be set (1). If the STOP mode is used when the EXTC bit is in the cleared (0) state, the  $\mu$ PD784026 may be damaged or suffer reduced reliability.
- (2) When setting the EXTC bit of the OSTS to 1, be sure to input a clock in phase reverse to that of the clock input to the X1 pin, to the X2 pin.
- (3) Even when inputting the external clock by clearing the EXTC bit of the oscillation stabilization time specification register (OSTS) to 0, input a signal in phase reverse to that of the signal input to the X1 pin, to the X2 pin, whenever possible. Otherwise, more malfunctioning may occur due to noise.
- (4) When an external clock is input, this should be performed with a HCMOS device, or a device with the equivalent drive capability.
- (5) A signal should not be extracted from the X1 and X2 pins. If a signal is extracted, it should be extracted from point a in Figure 4-5.

**Figure 4-5 Signal Extraction with External Clock Input**



- (6) The wiring connecting the X1 pin to the X2 pin via an inverter, in particular, should be made as short as possible.



**4.4.2 When Crystal/Ceramic Oscillation is Used**

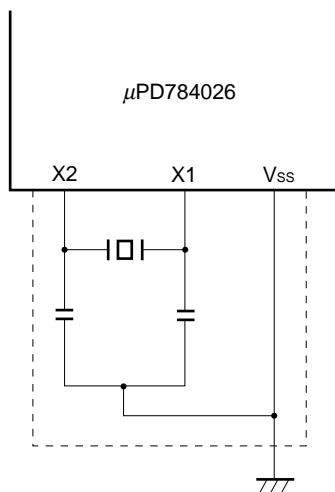
(1) As the oscillator is a high-frequency analog circuit, considerable care is required.

The following points, in particular, require attention.

- The wiring should be kept as short as possible.
- No other signal lines should be crossed.
- Avoid lines carrying a high fluctuating current.
- The oscillator capacitor grounding point should always be at the same potential as the V<sub>ss</sub> pin. Do not ground to a ground pattern carrying a high current.
- A signal should not be taken from the oscillator.

If oscillation is not performed normally and stably, the microcontroller will not be able to operate normally and stably, either. Also, if a high-precision oscillation frequency is required, consultation with the oscillator manufacturer is recommended.

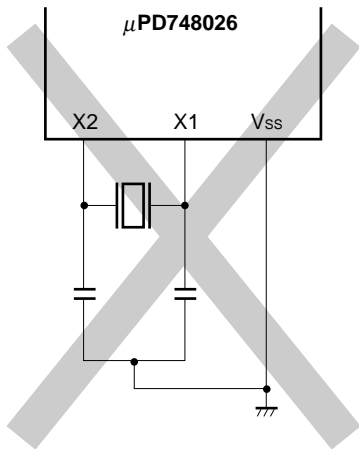
**Figure 4-6 Cautions on Resonator Connection**



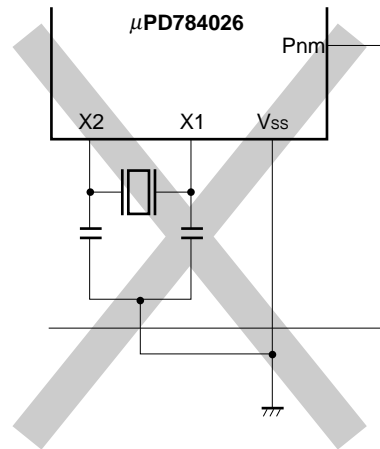
- Cautions**
1. The oscillator should be as close as possible to the X1 and X2 pins.
  2. No other signal lines should pass through the area enclosed by the dotted line.

Figure 4-7 Incorrect Example of Resonator Connection

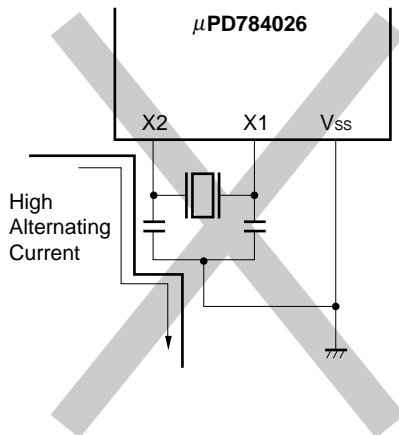
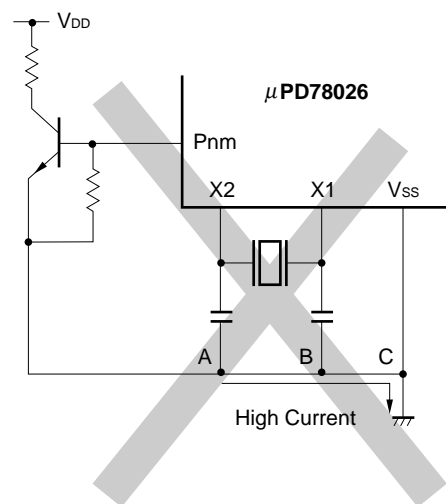
(a) Wiring of conneted circuits is too long



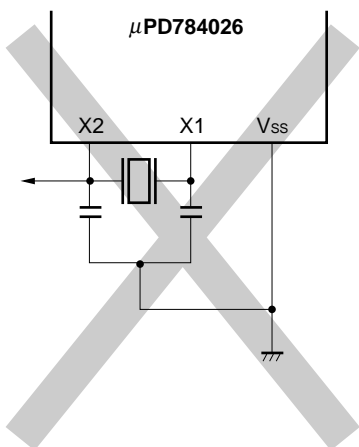
(b) Crossed signal lines



(c) Wiring near high alternating current

(d) Current flowing through ground line of oscillator  
(Potentials at points A, B, and C fluctuate)

(e) Signal extracted



- (2) When the device is powered on, and when restoring from the STOP mode, sufficient time must be allowed for the oscillation to stabilize. Generally speaking, the time required for oscillation stabilization is several milliseconds when a crystal resonator is used, and several hundred microseconds when a ceramic resonator is used.

An adequate oscillation stabilization period should be secured by the following means:

- <1> When powering-on :  $\overline{\text{RESET}}$  input (reset period)
- <2> When returning from STOP mode :
  - (i)  $\overline{\text{RESET}}$  input (reset period)
  - (ii) Time of the oscillation stabilization timer that automatically starts at the valid edge of NMI, INTP4, or INTP5 signal\* (set by the oscillation stabilization time specification register (OSTS))

\* For INTP4 and INTP5, when masking is released and macro service is disabled.

- (3) The EXTC bit of the oscillation stabilization time specification register (OSTS) must be cleared (0). If the EXTC bit is set (1), oscillation will stop.

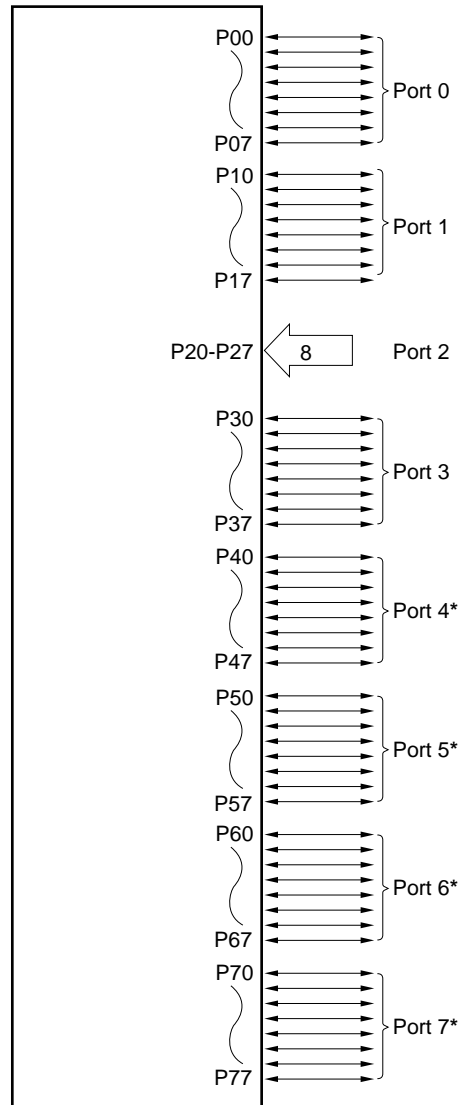
[MEMO]

## CHAPTER 5 PORT FUNCTIONS

### 5.1 DIGITAL INPUT/OUTPUT PORTS

The  $\mu$ PD784026 is provided with the ports shown in Figure 5-1, enabling various kinds of control to be performed. The function of each port is shown in Table 5-1. For ports 0 to 6, use of an internal pull-up resistor can be specified in software when used as input ports.

**Figure 5-1 Port Configuration**



- \* In the  $\mu$ PD784021, P40-47, P50-57, P64 and P65 operate only as the address/data bus, address bus, and  $\overline{\text{RD}}$  and  $\overline{\text{WR}}$  pins, respectively, and P60 to P63 are output-only port pins.

Table 5-1 Port Functions

Port Name	Pin Names	Functions	SoftwarePull-up Specification
Port 0	P00 to P07	<ul style="list-style-type: none"> <li>Input or output specifiable bit-wise.</li> <li>Can also operate as 4-bit real-time output ports (P00 to P03, P04 to P07).</li> <li>Transistor drive capability.</li> </ul>	Input mode pins specified at once
Port 1	P10 to P17	<ul style="list-style-type: none"> <li>Input or output specifiable bit-wise.</li> <li>LED drive capability.</li> </ul>	
Port 2	P20 to P27	Input port	6-bit unit (P22 to P27)
Port 3	P30 to P37	Input or output specifiable bit-wise.	Input mode pins specified at once
Port 4	P40 to P47	<ul style="list-style-type: none"> <li>Input or output specifiable bit-wise.</li> <li>LED drive capability.</li> </ul>	
Port 5	P50 to P57	<ul style="list-style-type: none"> <li>Input or output specifiable bit-wise.</li> <li>LED drive capability.</li> </ul>	
Port 6	P60 to P67*	Input or output specifiable bit-wise.	
Port 7	P70 to P77	Input or output specifiable bit-wise.	—

\* In the  $\mu$ PD784021, P60 to P63 operate only as output-only port pins.

**Remark** In the  $\mu$ PD784021, P40-47, P50-57, P64, and P65 operate as the address/data bus, address bus, and  $\overline{RD}$  and  $\overline{WR}$  pins, respectively. Therefore, LED direct drive and internal software pull-up resistor specification are not possible for these pins.

Table 5-2 Number of Input/Output Ports

Input/Output Ports	Total	Input Mode	Output Mode	
		Software Pull-up Resistor	Direct LED Drive	Direct Transistor Drive
Input ports	8 (8)	6 (6)	—	—
Input/output ports	56 (34)	48 (26)	24 (8)	0 (0)
Output ports	0 (4)	—	0 (0)	8 (8)
Total	64 (46)	54 (32)	24 (8)	8 (8)

**Remark** ( ):  $\mu$ PD784021

## 5.2 PORT 0

Port 0 is an 8-bit input/output port with an output latch, and has direct transistor drive capability. Input/output can be specified bit-wise by means of the port 0 mode register (PM0). Each pin incorporates a software programmable pull-up resistor.

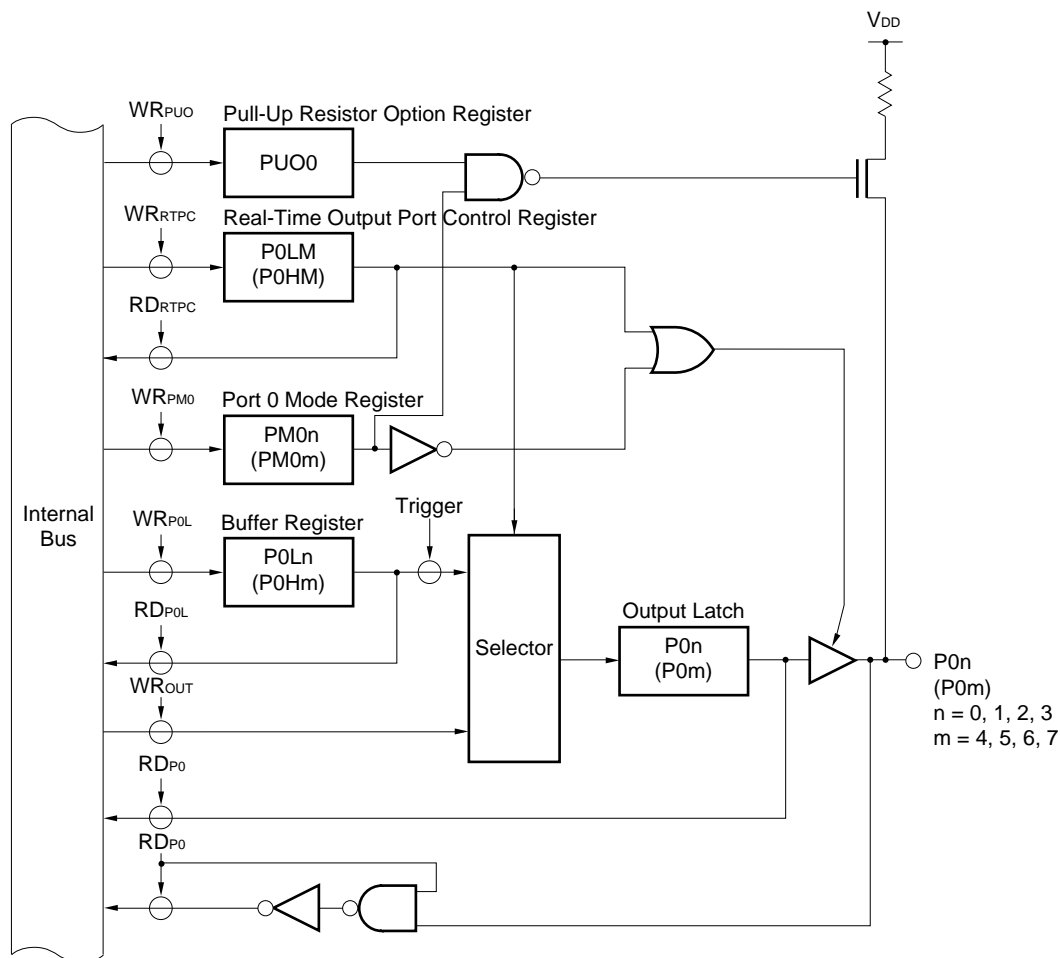
P00 to P03 and P04 to P07 can output the buffer register (P0L, P0H) contents at any time interval as 4-bit real-time output ports or one 8-bit real-time output port. The real-time output port control register (RTPC) is used to select whether this port is used as a normal output port or a real-time output port.

When  $\overline{\text{RESET}}$  is input, port 0 is set as an input port (output high-impedance state), and the output latch contents are undefined.

### 5.2.1 Hardware Configuration

The port 0 hardware configuration is shown in Figure 5-2.

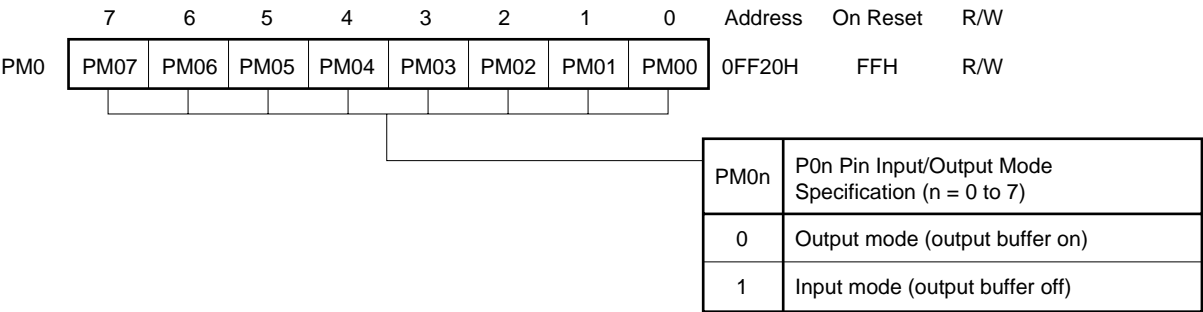
**Figure 5-2 Port 0 Block Diagram**



5.2.2 Input/Output Mode/Control Mode Setting

The port 0 input/output mode is set by means of the port 0 mode register (PM0) as shown in Figure 5-3.

Figure 5-3 Port 0 Mode Register (PM0) Format



When port 0 is used as a real-time output port, the P0LM and P0HM bits of the real-time output port control register (RTPC) should be set (1).

When P0LM and P0HM are set, the respective pin output buffer is turned on and the output latch contents are output to the pin irrespective of the contents of PM0.



### 5.2.3 Operating Status

Port 0 is an input/output port

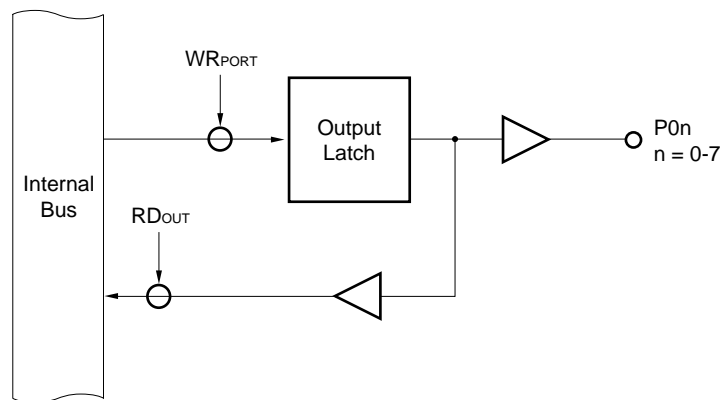
#### (1) When set as an output port

The output latch is enabled, and data transfers between the output latch and accumulator are performed by means of transfer instructions. The output latch contents can be freely set by means of logical operation instructions. Once data has been written to the output latch, it is retained until data is next written to the output latch\*.

Writes cannot be performed to the output latch of a port specified as a real-time output port. However, the output latch contents can be read even if it is set to the real-time output port mode.

\* Including the case where another bit of the same port is manipulated by a bit manipulation instruction.

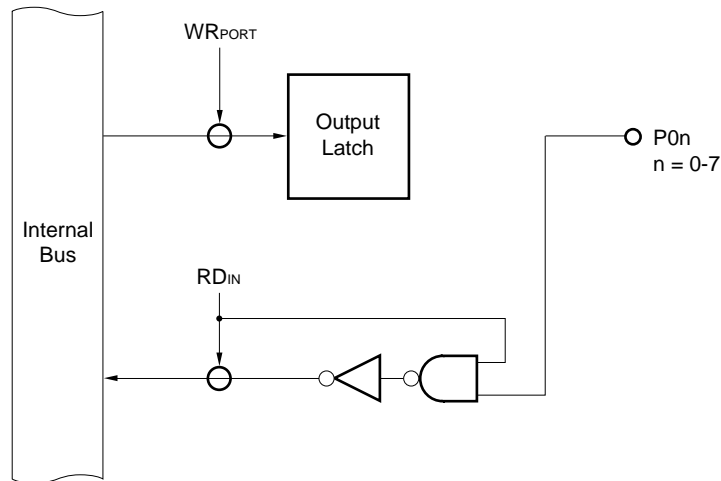
**Figure 5-4 Port Specified as Output Port**



**(2) When set as an input port**

The port pin level can be loaded into an accumulator by means of a transfer instruction, etc. In this case, too, writes can be performed to the output latch, and data transferred from the accumulator by a transfer instruction, etc., is stored in all output latches irrespective of the port input/output specification. However, since the output buffer of a bit specified as an input port is high-impedance, the data is not output to the port pin (when a bit specified as input is switched to an output port, the output latch contents are output to the port pin). Also, the contents of the output latch of a bit specified as an input port cannot be loaded into an accumulator.

### Figure 5-5 Port Specified as Input Port



**Caution** A bit manipulation instruction manipulates one bit as the result, but accesses the port in 8-bit units. Therefore, if a bit manipulation instruction is used on a port with a mixture of input and output pins, the contents of the output latch of pins specified as inputs will be undefined (excluding bits manipulated with a SET1 or CLR1 instruction, etc.). Particular care is required when there are bits which are switched between input and output.

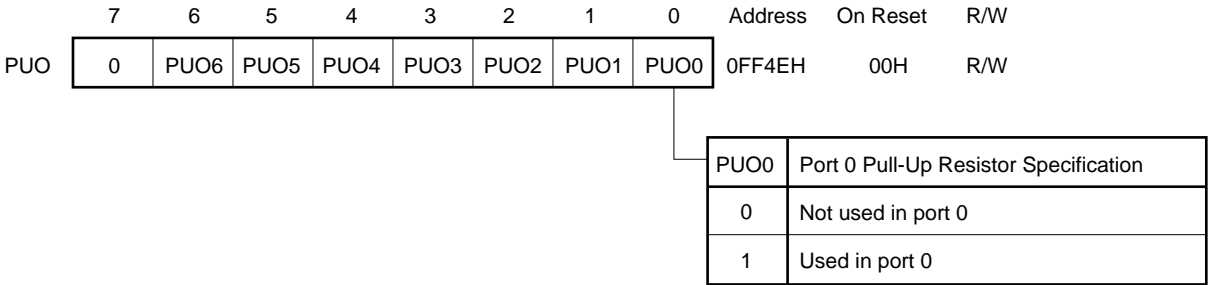
**Caution** is also required when manipulating the port with other 8-bit manipulation instructions.

5.2.4 Internal Pull-Up Resistors

Port 0 incorporates pull-up resistors. Use of these internal resistors when pull-up is necessary enables the number of parts and the mounting area to be reduced.

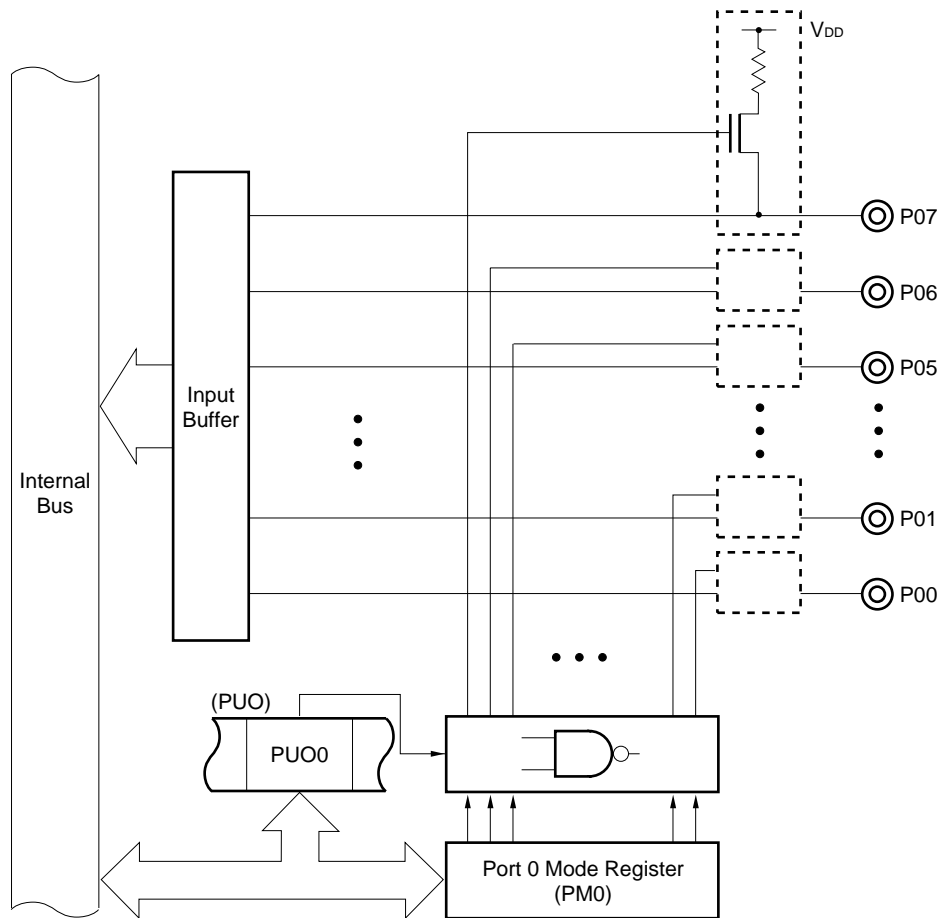
Whether or not an internal pull-up resistor is to be used can be specified for each pin by means of the PU00 bit of the pull-up resistor option register (PUO) and the port 0 mode register (PM0). When PU00 is 1, the internal pull-up resistors of the pins for which input is specified by PM0 are enabled (PM0n = 1, n = 0 to 7).

Figure 5-6 Pull-Up Resistor Option Register (PUO) Format



**Remark** When STOP mode is entered, assigning 00H to PUO is effective in reducing the current consumption.

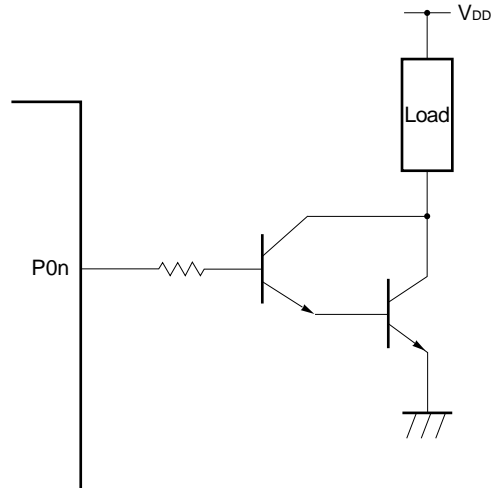
Figure 5-7 Pull-Up Resistor Specification (Port 0)



### 5.2.5 Transistor Drive

In port 0, the output buffer high-level side drive capability has been increased, allowing active-high direct transistor drive. An example of the connection is shown in Figure 5-8.

**Figure 5-8 Example of Transistor Drive**



### 5.3 PORT 1

Port 1 is an 8-bit input/output port with an output latch. Input/output can be specified bit-wise by means of the port 1 mode register (PM1). Each pin incorporates a programmable pull-up resistor. This port has direct LED drive capability.

In addition to their input/output port function, P10 to P14 also have a dual function as PWM output pins and serial interface pins. The operating mode can be specified bit-wise by means of the PWM control register (PWMC) and the port 1 mode control register (PMC1), as shown in Table 5-3. The level of any pin can be read and tested at any time irrespective of the dual-function pin operation.

When  $\overline{\text{RESET}}$  is input, port 1 is set as an input port (output high-impedance state), and the output latch contents are undefined.

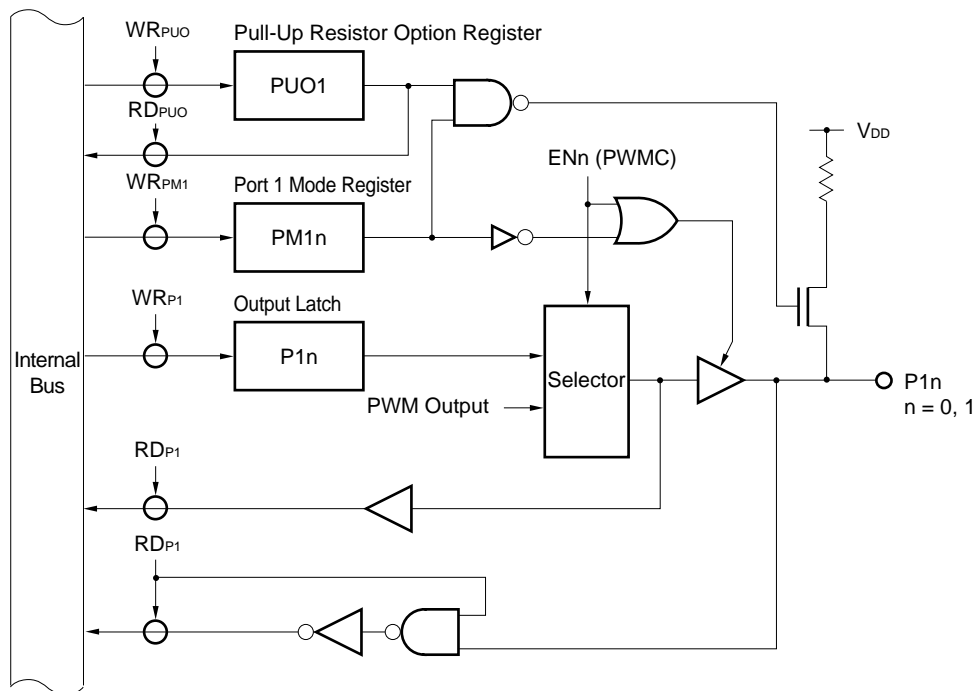
**Table 5-3 Port 1 Operating Modes**

Pin Name	Port Mode	Control Signal I/O Mode	Operation to Operate Control Pin
P10	I/O port	PWM0 output	Setting of EN0 bit of PWMC to 1
P11		PWM1 output	Setting of EN1 bit of PWMC to 1
P12		ASCK2 I/O/ $\overline{\text{SCK2}}$ I/O	Setting PMC12 bit of PMC1 to 1
P13		RxD2 input/SI2 input	Setting PMC13 bit of PMC1 to 1
P14		TxD2 output/SO2 output	Setting PMC14 bit of PMC1 to 1
P15-P17		—	—

**5.3.1 Hardware Configuration**

The port 1 hardware configuration is shown in Figures 5-9 to 5-13.

**Figure 5-9 Block Diagram of P10 and P11 (Port 1)**



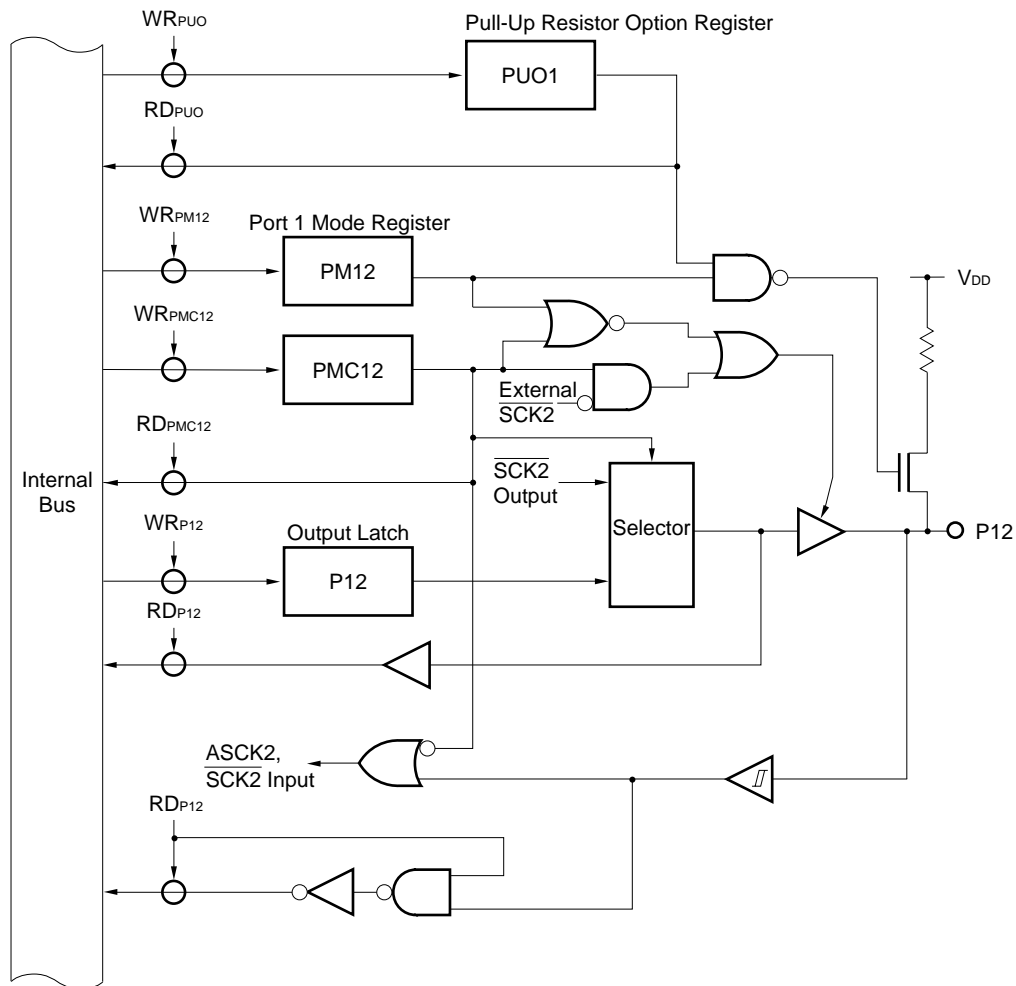
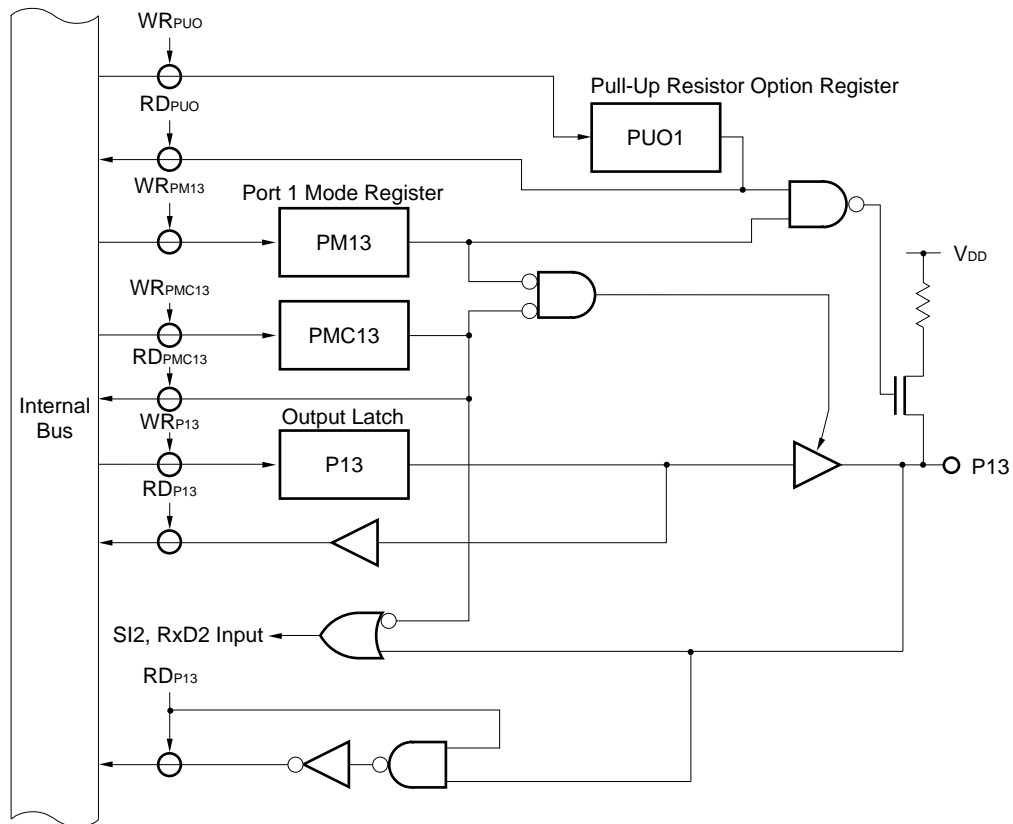
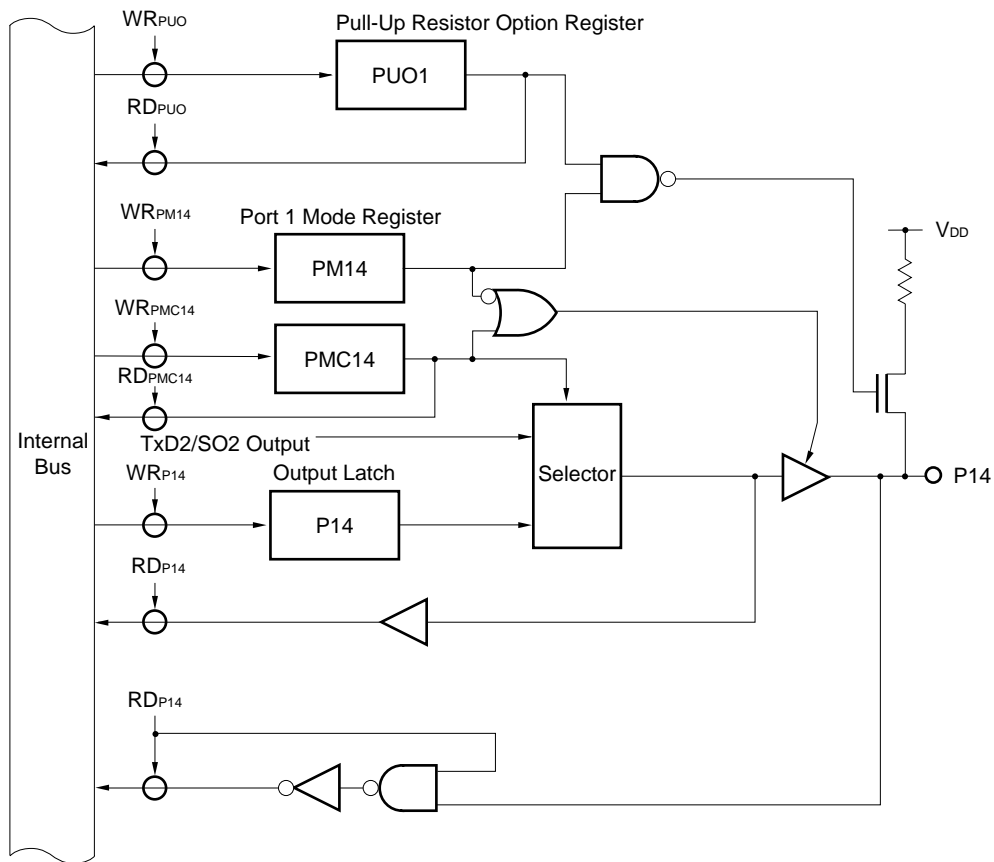
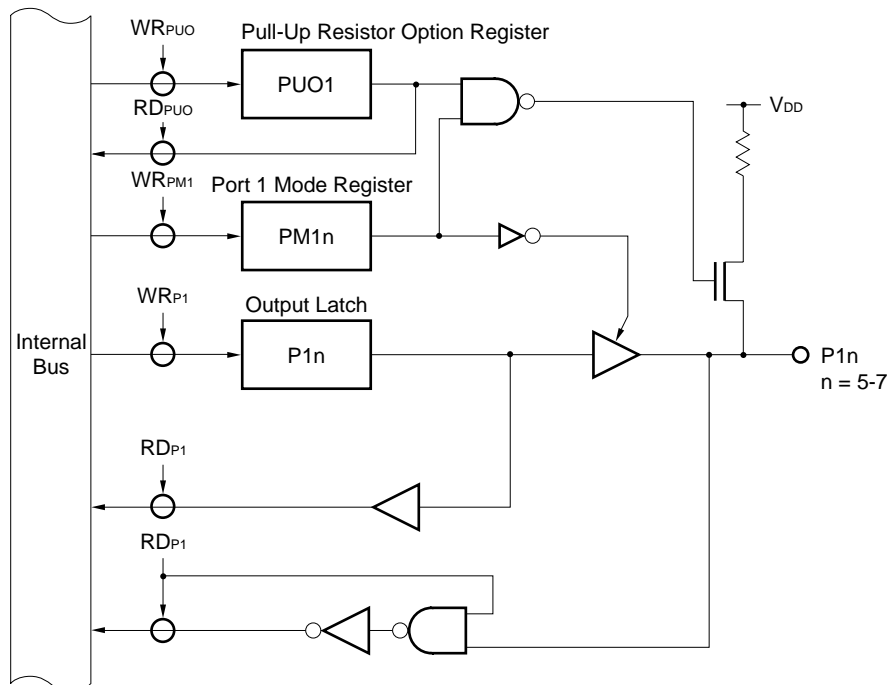
**Figure 5-10 Block Diagram of P12 (Port 1)**



Figure 5-11 Block Diagram of P13 (Port 1)



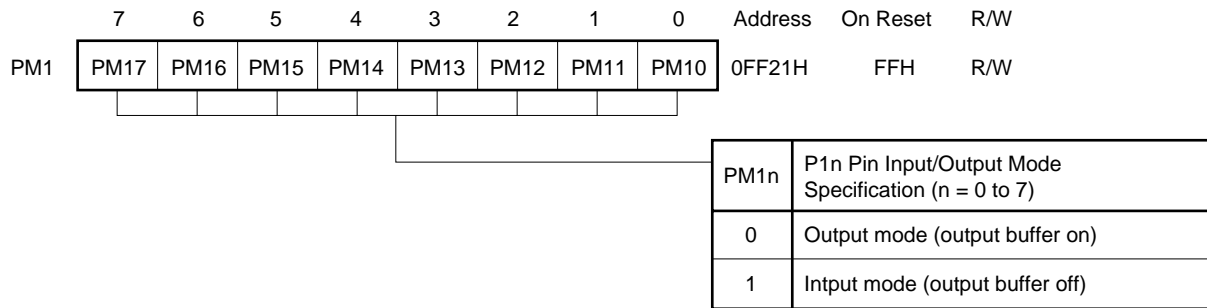
**Figure 5-12 Block Diagram of P14 (Port 1)****Figure 5-13 Block Diagram of P15 to P17 (Port 1)**

**5.3.2 Input/Output Mode/Control Mode Setting**

The port 1 input/output mode is set for each pin by means of the port 1 mode register (PM1) as shown in Figure 5-14.

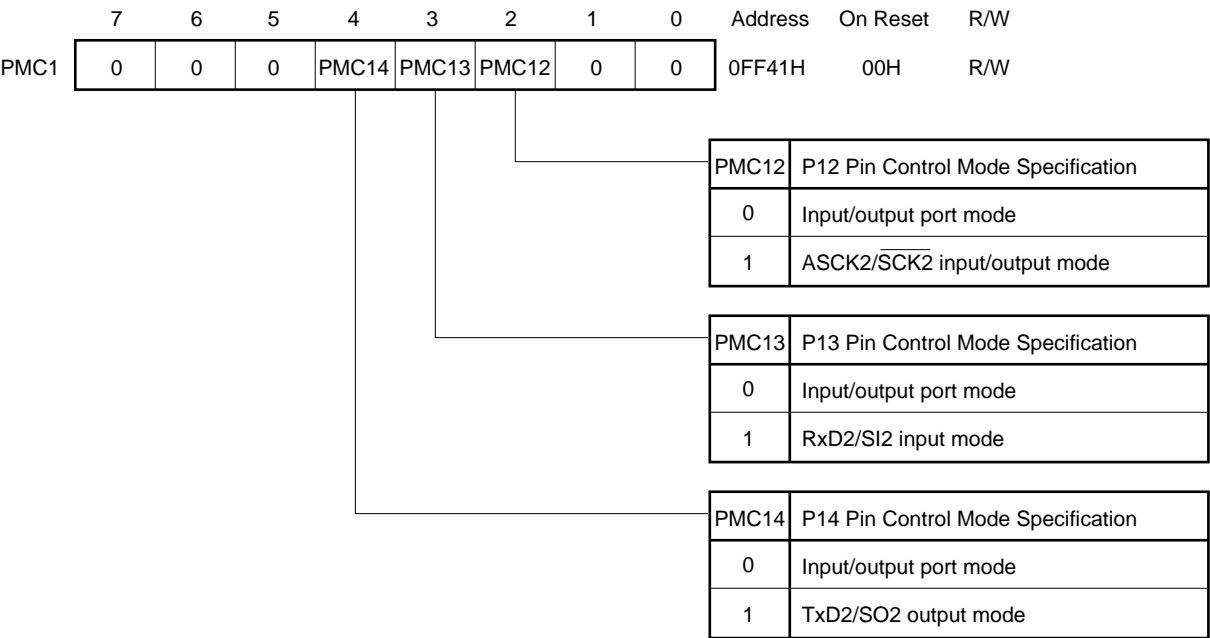
In addition to their input/output port function, P10 and P11 also have a dual function as PWM signal output pins, and the control mode is specified by means of the PWM control register (PWMC) as shown in Table 5-4.

In addition to their input/output port function, P12 to P14 also have a dual function as serial interface pins, and the control mode is specified by means of the port 1 mode control register (PMC1) as shown in Figure 5-15.

**Figure 5-14 Port 1 Mode Register (PM1) Format****Table 5-4 Method of Setting P10 & P11 PWM Signal Output Function**

Pin Name	Function	Method of Setting PWM Signal Output Function
P10	PWM0	Set (1) EN0 bit of PWMC
P11	PWM1	Set (1) EN1 bit of PWMC

Figure 5-15 Port 1 Mode Control Register (PMC1) Format



### 5.3.3 Operating Status

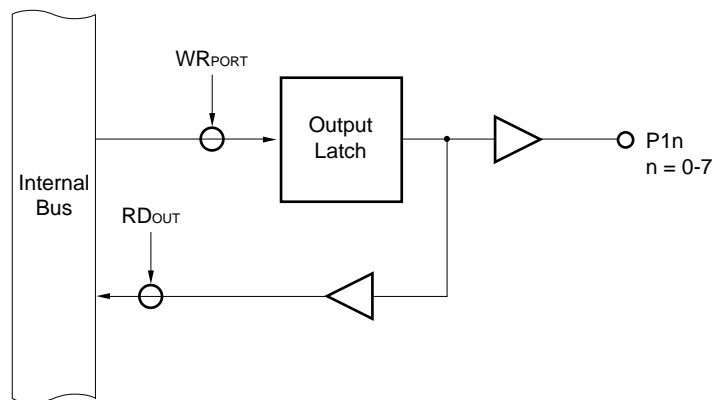
Port 1 is an input/output port. Pins P10 and P11 have a dual function as PWM signal output pins, and pins P12 to P14 have a dual function as serial interface pins.

#### (1) When set as an output port

The output latch is enabled, and data transfers between the output latch and accumulator are performed by means of transfer instructions. The output latch contents can be freely set by means of logical operation instructions. Once data has been written to the output latch, it is retained until data is next written to the output latch\*.

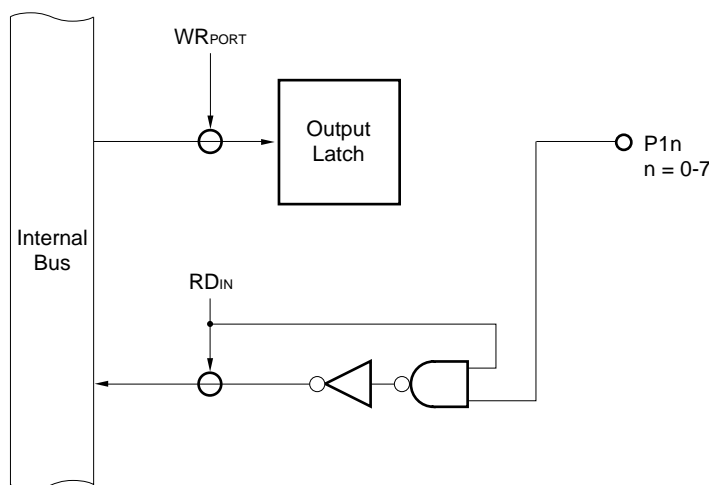
\* Including the case where another bit of the same port is manipulated by a bit manipulation instruction.

**Figure 5-16 Port Specified as Output Port**



**(2) When set as an input port**

The port pin level can be loaded into an accumulator by means of a transfer instruction, etc. In this case, too, writes can be performed to the output latch, and data transferred from the accumulator by a transfer instruction, etc., is stored in all output latches irrespective of the port input/output specification. However, since the output buffer of a bit specified as an input port is high-impedance, the data is not output to the port pin (when a bit specified as input is switched to an output port, the output latch contents are output to the port pin). Also, the contents of the output latch of a bit specified as an input port cannot be loaded into an accumulator.

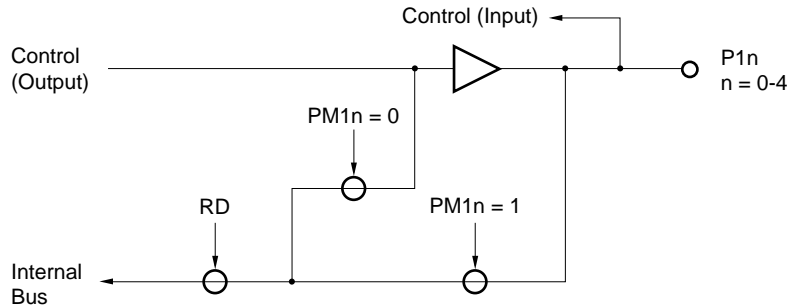
**Figure 5-17 Port Specified as Input Port**

**Caution** A bit manipulation instruction manipulates one bit as the result, but accesses the port in 8-bit units. Therefore, if a bit manipulation instruction is used on a port that has the I/O mode or port mode and control mode, the contents of the output latch of the pin set in the input mode or control mode become undefined (excluding bits manipulated with a SET1 or CLR1 instruction, etc.). Particular care is required when there are bits which are switched between input and output.

Caution is also required when manipulating the port with other 8-bit manipulation instructions.

**(3) When specified as control signal input/output**

P10 and P11 (by setting (1) the ENn bit (n = 0 or 1) of the PWM control register(PWMC)) and P12 to P14 (by setting (1) bits of the port 1 mode control register (PMC1)) can be used as control signal inputs or outputs bit-wise irrespective of the setting of the port 1 mode register (PM1). When a pin is used as a control signal, the control signal status can be seen by executing a port read instruction.

**Figure 5-18 Control Specification****(a) When port is control signal output**

When the port 1 mode register (PM1) is set (1), the control signal pin level can be read by executing a port read instruction.

When PM1 is reset (0), the  $\mu$ PD784026 internal control signal status can be read by executing a port read instruction.

**(b) When port is control signal input**

When the port 1 mode register (PM1) is set (1), control signal pin level can be read by executing a port read instruction.

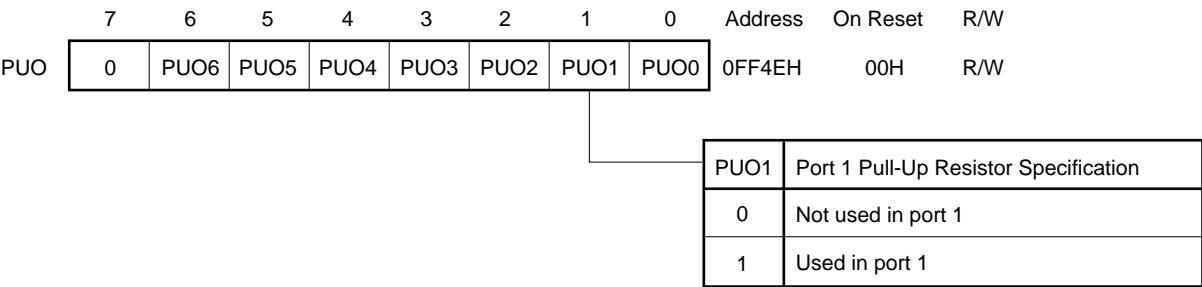
5.3.4 Internal Pull-Up Resistors

Port 1 incorporates pull-up resistors. Use of these internal resistors when pull-up is necessary enables the number of parts and the mounting area to be reduced.

Whether or not an internal pull-up resistor is to be used can be specified for each pin by means of the PUO1 bit of the pull-up resistor option register (PUO) and the port 1 mode register (PM1). When PUO1 is 1, the internal pull-up resistors of the pins for which input is specified by PM1 are enabled ( $PM1n = 1$ ,  $n = 0$  to 7).

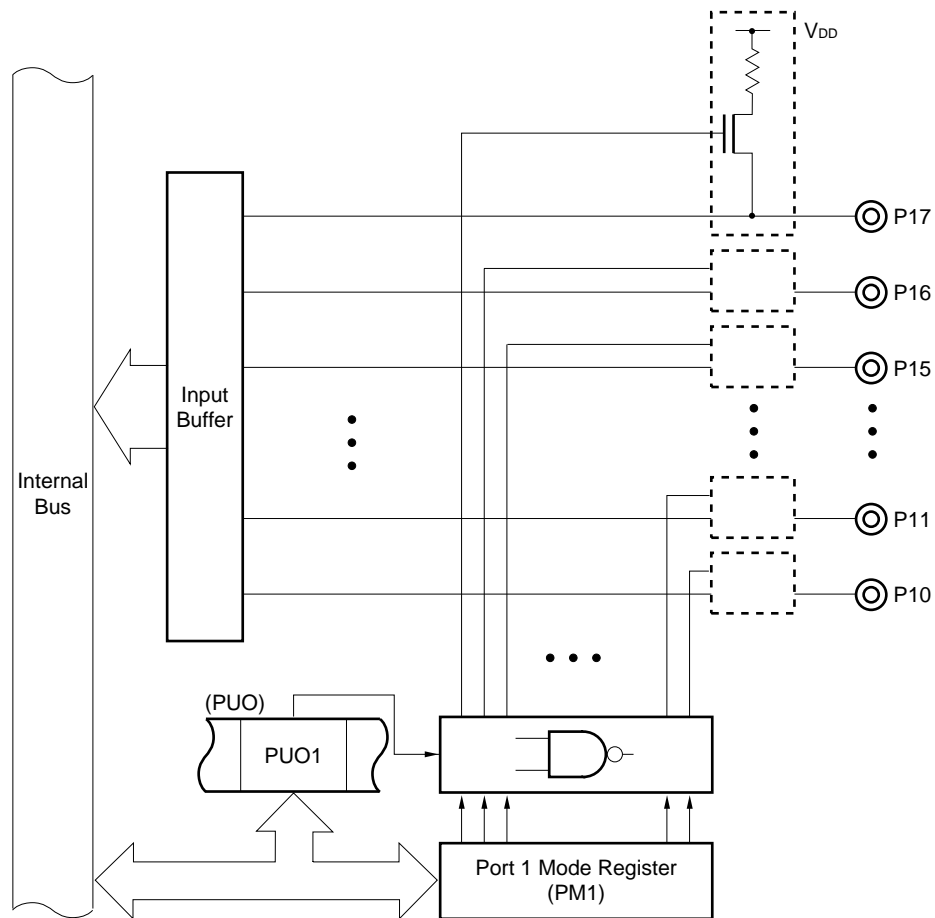
Also, the specification for use of the pull-up resistor is also valid for pins specified as control signal output pins (pull-up resistors are also connected to pins that function as control signal output pins). Therefore, if you do not want to connect the pull-up resistors with the control signal output pin, the contents of the corresponding bits of PM1 should be set to 0 (output mode).

Figure 5-19 Pull-Up Resistor Option Register (PUO) Format



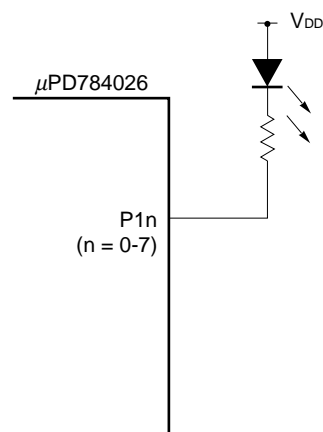
**Remark** When STOP mode is entered, assigning 00H to PUO is effective in reducing the current consumption.



**Figure 5-20 Pull-Up Resistor Specification (Port 1)**

### 5.3.5 Direct LED Drive

In port 1, the output buffer low-level side drive capability has been reinforced allowing active-low direct LED drive. An example of such use is shown in Figure 5-21.

**Figure 5-21 Example of Direct LED Drive**

**5.4 PORT 2**

Port 2 is an 8-bit input-only port. P22 to P27 incorporate a software programmable pull-up resistor. As well as operating as input ports, port 2 pins also operate as control signal input pins, such as external interrupt signal pins (see **Table 5-5**). All 8 pins are Schmitt-triggered inputs to prevent misoperation due to noise.

**Table 5-5 Port 2 Operating Modes**

Port Name	Function
P20	Input port/NMI input*
P21	Input port/INTP0 input/CR11 capture trigger input/ Timer/counter 1 count clock/real-time output port trigger signal
P22	Input port/INTP1 input/CR22 capture trigger input
P23	Input port/INTP2 input/CI input
P24	Input port/INTP3 input/CR02 capture trigger input/ Timer/count 0 count clock
P25	Input port/INTP4 input/ASCK input/ $\overline{\text{SCK1}}$ input/output
P26	Input port/INTP5 input/A/D converter external trigger input
P27	Input port/SIO input

\* NMI input is acknowledged regardless of whether interrupts are enabled or disabled.

**(a) Function as port pins**

The pin level can always be read or tested regardless of the dual-function pin operation.

**(b) Functions as control signal input pins****(i) NMI (Non-maskable Interrupt)**

The external non-maskable interrupt request input pin. Rising edge detection or falling edge detection can be specified by means of the external interrupt mode register 0 (INTM0).

**(ii) INTP0 to INTP5 (Interrupt From Peripherals)**

External interrupt request input pins. When the valid edge specified by the external interrupt mode registers 0, 1 (INTM0/INTM1) is detected an interrupt is generated (see **CHAPTER 19 EDGE DETECTION FUNCTION**). In addition, pins INTP0 to INTP3 and INTP5 are also used as external trigger input pins with the various functions shown below.

- INTP0 ..... Timer/counter 1 capture trigger input pin  
External count clock input pin  
Real-time output port trigger input pin
- INTP1 ..... Timer/counter 2 capture register (CR22) capture trigger input pin
- INTP2 ..... Timer/counter 2 external count clock input pin  
Capture/compare register (CR21) capture trigger input pin
- INTP3 ..... Timer/counter 0 capture trigger input pin  
Timer/counter 0 external count clock input pin
- INTP5 ..... A/D converter external trigger input pin

**(iii) CI (Clock Input)**

The timer/counter 2 external clock input pin.

**(iv) ASCK (Asynchronous Serial Clock)**

The external baud rate clock input pin.

**(v)  $\overline{\text{SCK1}}$  (Serial Clock 1)**

The serial clock input/output pin (in 3-wire serial I/O 1 mode).

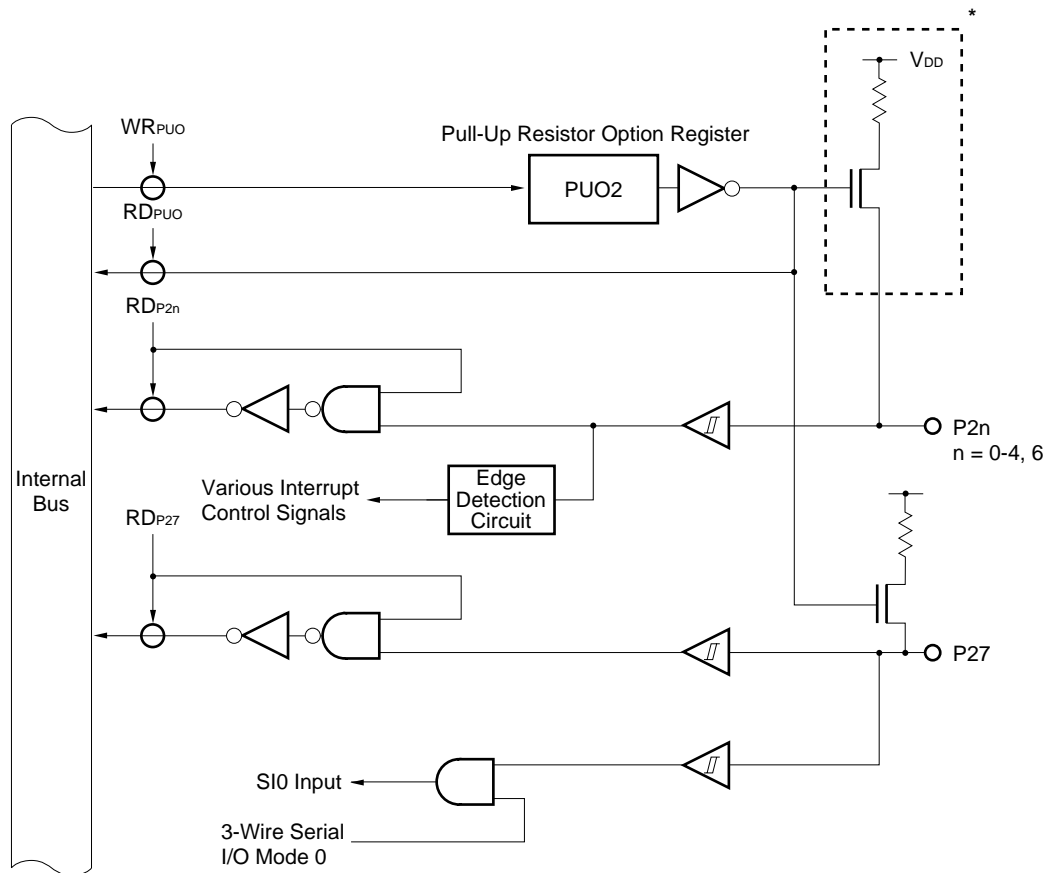
**(vi) SIO (Serial Input 0)**

The serial data input pin (in 3-wire serial I/O 0 mode).

**5.4.1 Hardware Configuration**

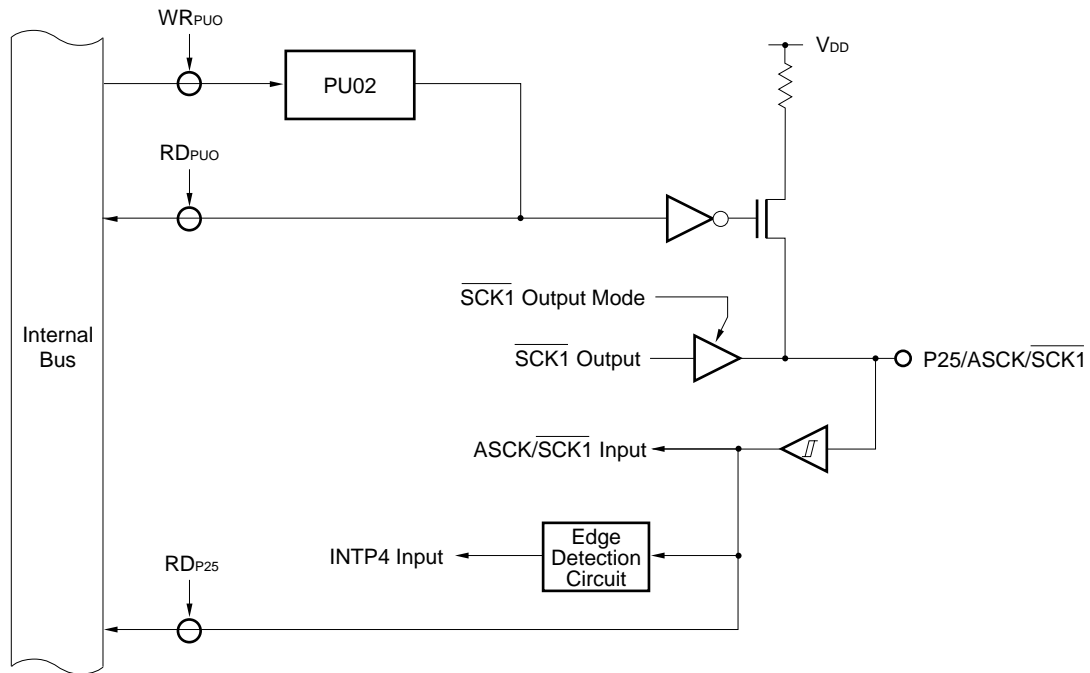
The port 2 hardware configuration is shown in Figure 5-22.

**Figure 5-22 Block Diagram of P20 to P24, P26 and P27 (Port 2)**



\* P20 and P21 do not have the circuitry enclosed by the dotted line.

Figure 5-23 Block Diagram of P25 (Port 2)



### 5.4.2 Input Mode/Control Mode Setting

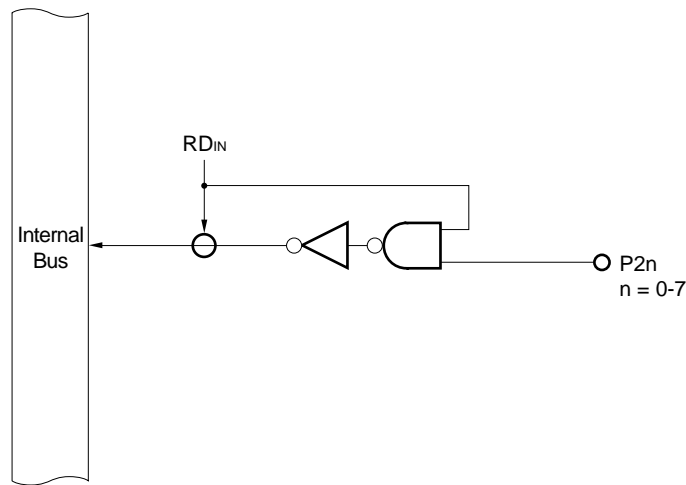
Port 2 is an input-only port, and there is no register for setting the input mode.

Also, control signal input is always possible, and therefore the signal to be used is determined by the control registers for individual internal hardware items.

### 5.4.3 Operating Status

Port 2 is an input-only port, and pin levels can always be read or tested.

**Figure 5-24 Port Specified as Input Port**



### 5.4.4 Internal Pull-Up Resistors

P22 to P27 incorporate pull-up resistors. Use of these internal resistors when pull-up is necessary enables the number of parts and the mounting area to be reduced.

Whether or not an internal pull-up resistor is to be used can be specified for all six pins, P22 to P27, together by means of the PUO2 bit of the pull-up resistor option register (PUO) (bit-wise specification is not possible).

P20 and P21 do not incorporate a pull-up resistor.

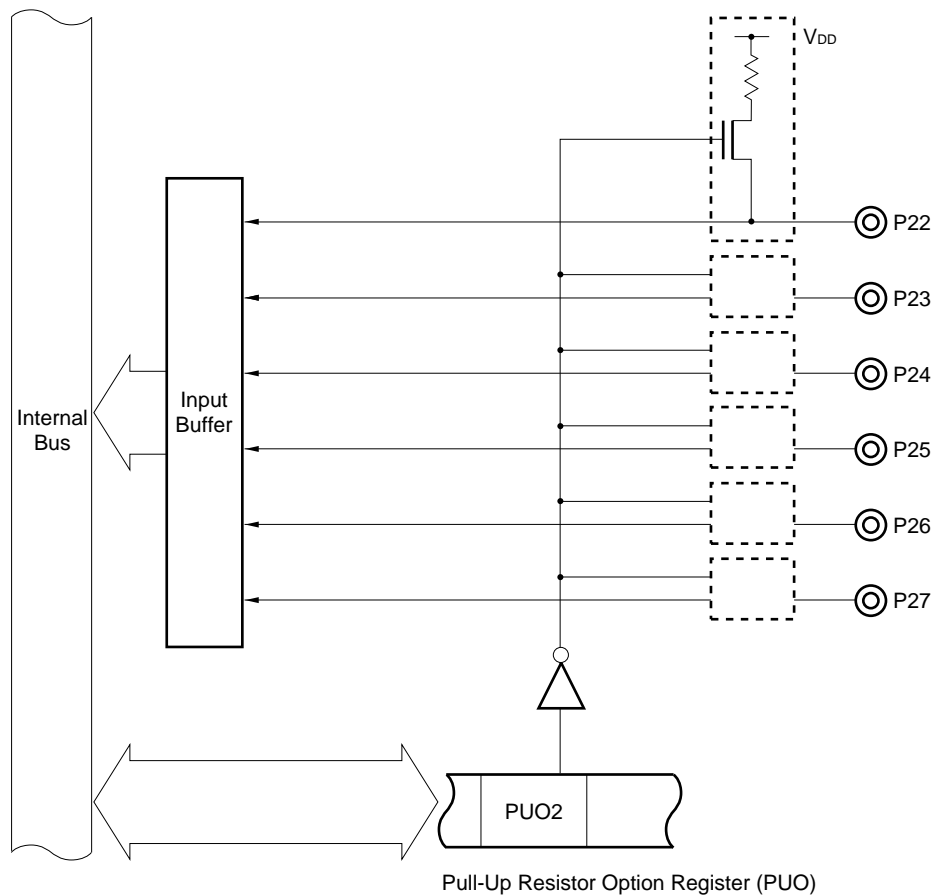
**Figure 5-25 Pull-Up Resistor Option Register (PUO) Format**

	7	6	5	4	3	2	1	0	Address	On Reset	R/W
PUO	0	PUO6	PUO5	PUO4	PUO3	PUO2	PUO1	PUO0	0FF4EH	00H	R/W

PUO2	Port 2 Pull-Up Resistor Specification
0	Not used in port 2
1	Used in pins P22 to P27

**Remark** When STOP mode is entered, assigning 00H to PUO is effective in reducing the current consumption.

**Figure 5-26 Pull-Up Specification (Port 2)**

**Caution** As P22 to P26 are not pulled up immediately after a reset, an interrupt request flag may be set depending on the function of the dual-function pins (INTP1 to INTP5). Therefore, the interrupt request flags should be cleared after specifying pull-up in the initialization routine.

**5.5 PORT 3**

Port 3 is an 8-bit input/output port with an output latch. Input/output can be specified bit-wise by means of the port 3 mode register (PM3). Each pin incorporates a software programmable pull-up resistor.

In addition to its function as an input/output port, port 3 also has various dual-function control signal pin functions.

The operating mode can be specified bit-wise by means of the port 3 mode control register (PMC3), as shown in Table 5-6. The pin level of all pins can always be read or tested regardless of the dual-function pin operation.

When RESET is input, port 3 is set as an input port (output high impedance state), and the output latch contents are undefined.

**Table 5-6 Port 3 Operating Modes**

(n = 0 to 7)

Mode	Port Mode	Control Signal Input/Output Mode
Setting Condition	PMC3n = 0	PMC3n = 1
P30	Input/output port	RxD input / S11 input
P31		TxD output / SO1 output
P32		$\overline{\text{SCK0}}$ input/output
P33		SO0 output / SB0 input/output
P34		TO0 output
P35		TO1 output
P36		TO2 output
P37		TO3 output

**(a) Port mode**

Each port specified as port mode by the port 3 mode control register (PMC3) can be specified as input/output bit-wise by means of the port 3 mode register (PM3).



**(b) Control signal input/output mode**

Pins can be set as control pins bit-wise by setting the port 3 mode control register (PMC3).

**(i) RxD (Receive Data)/SI1 (Serial Input 1)**

RxD is the asynchronous serial interface serial data input pin. SI1 is the serial data input pin (in 3-wire serial I/O 1 mode).

**(ii) TxD (Transmit Data)/SO1 (Serial Output 1)**

TxD is the asynchronous serial interface serial data output pin. SO1 is the serial data output pin (in 3-wire serial I/O 1 mode).

**(iii) SCK0 (Serial Clock 0)**

The clocked serial interface serial clock input/output pin.

**(iv) SO0 (Serial Output 0)/SB0 (Serial Bus)**

SO0 is the serial data output pin (in 3-wire serial I/O 0 mode), and SB0 is the serial bus input/output pin (in SBI mode).

**Remark** For bit 3 (P33) of port 3, the name “SB0” is a reserved word in the NEC assembler package. In the C compiler, it is defined as an sfr variable by the # pragma sfr directive.

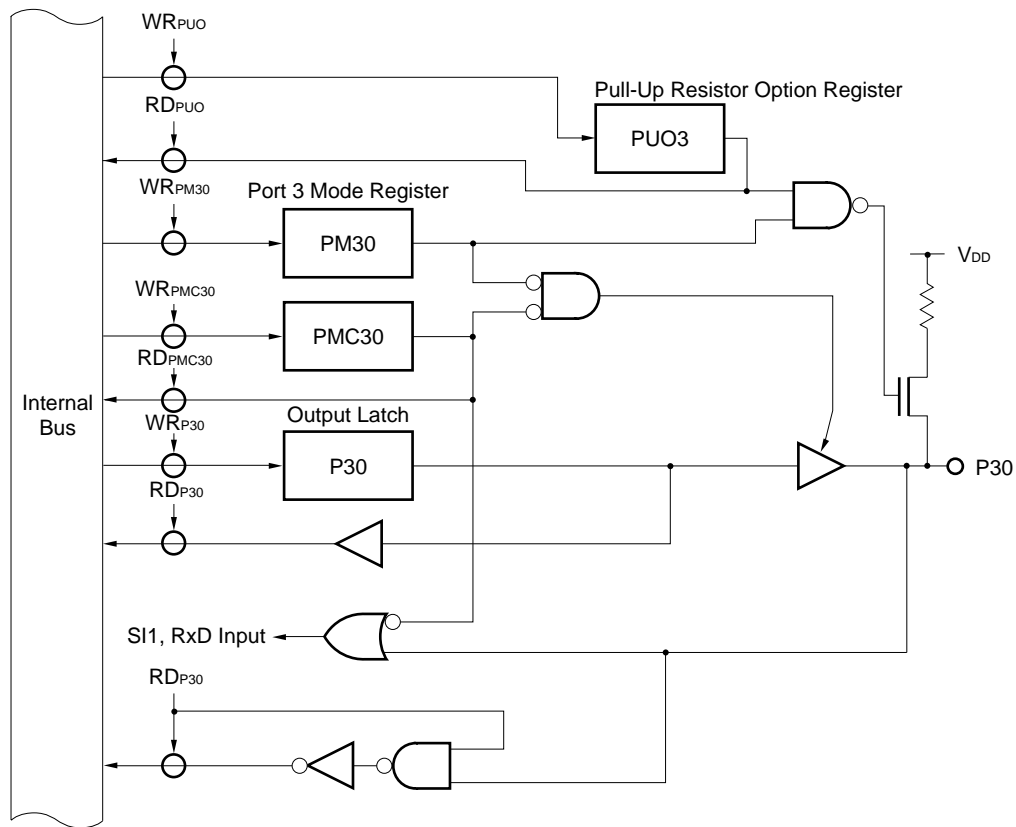
**(v) TO0 to TO3 (Timer Output)**

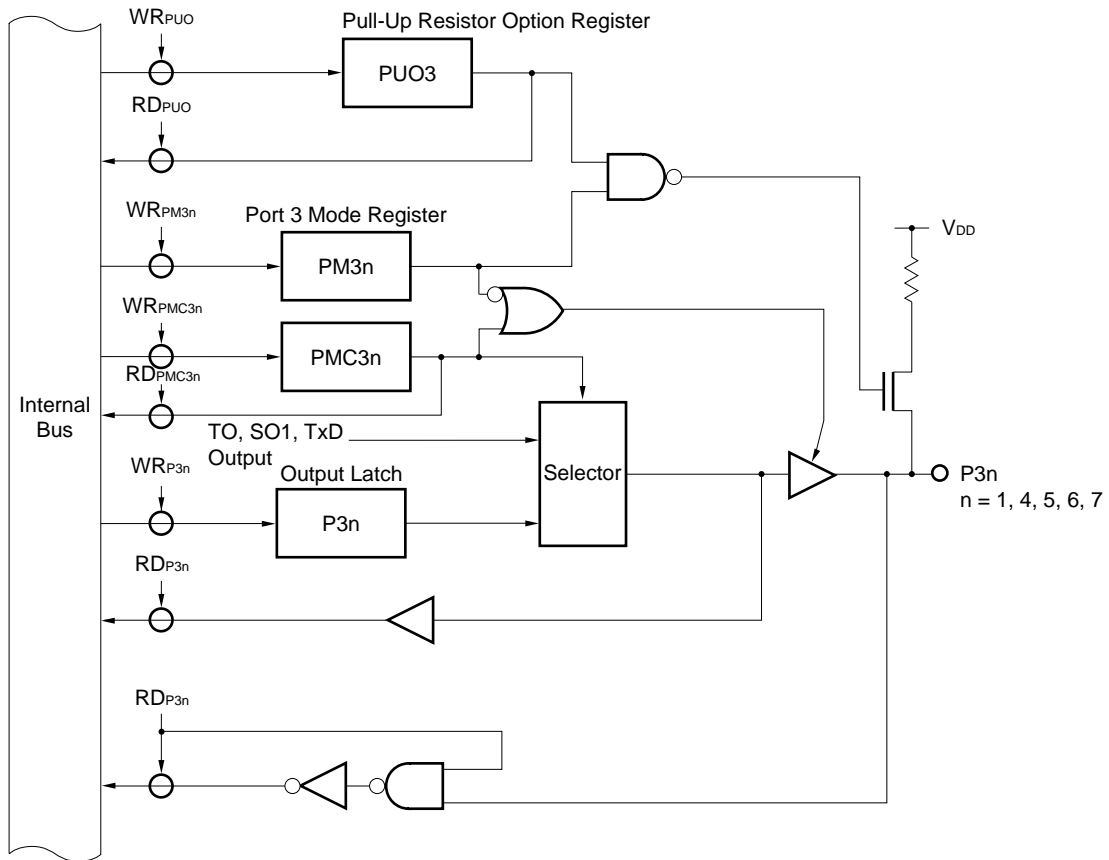
Timer output pins.

**5.5.1 Hardware Configuration**

The port 3 hardware configuration is shown in Figures 5-27 to 5-30.

**Figure 5-27 Block Diagram of P30 (Port 3)**



**Figure 5-28 Block Diagram of P31 and P34 to P37 (Port 3)**

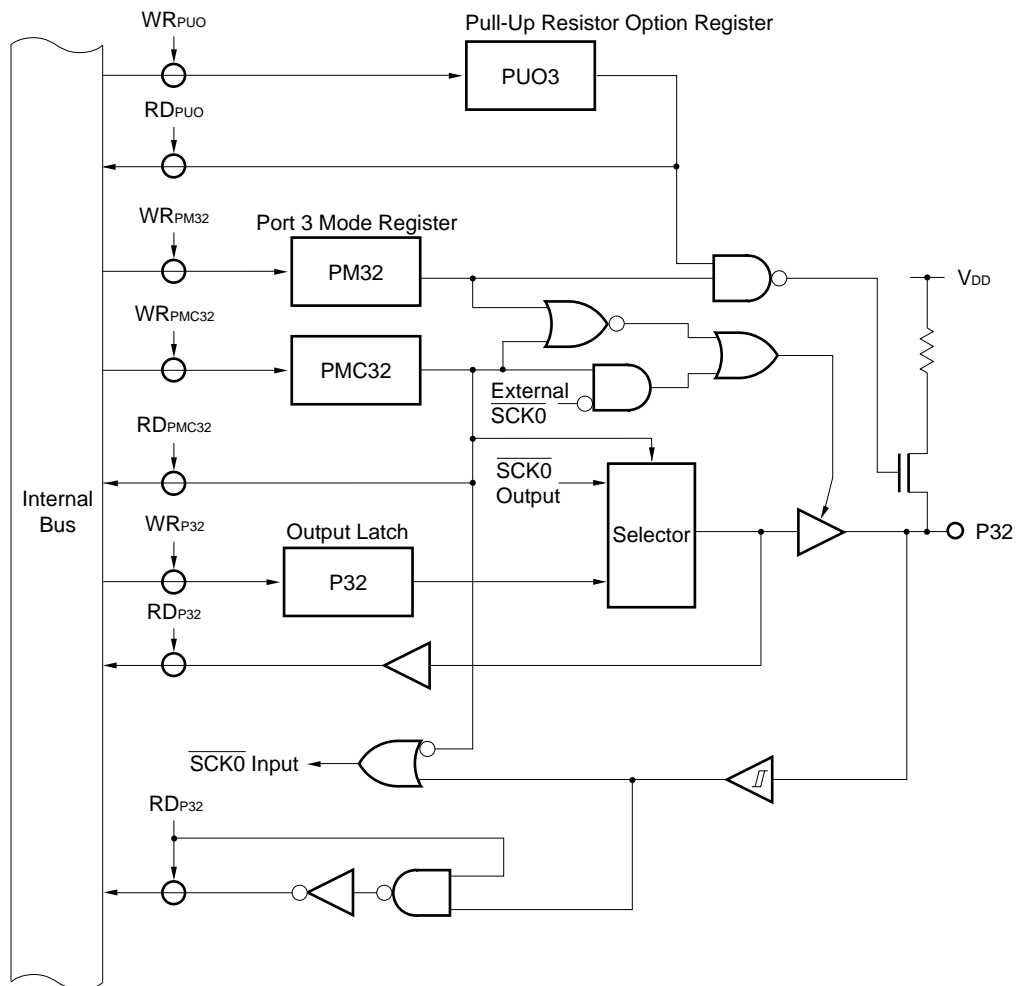
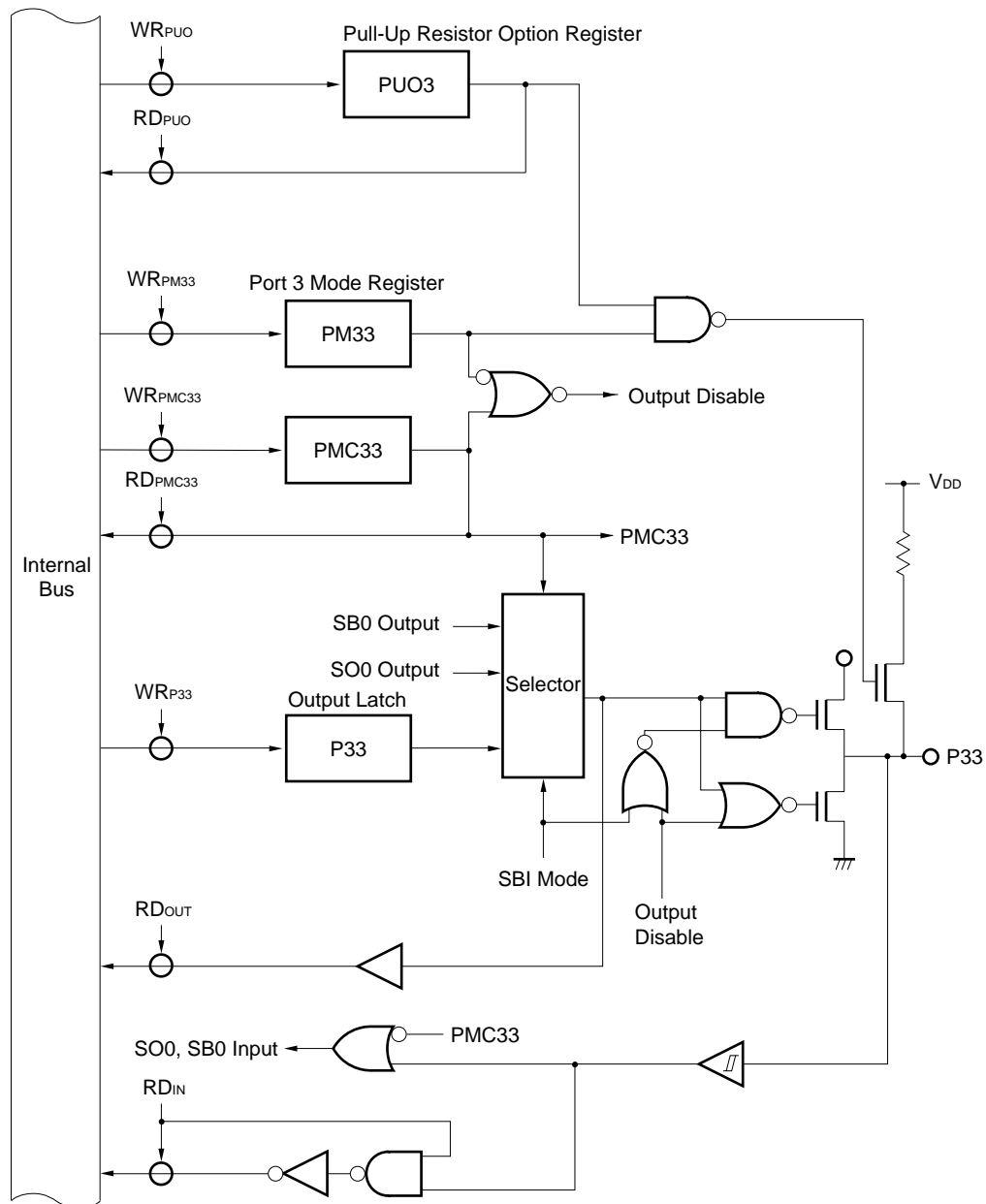
**Figure 5-29 Block Diagram of P32 (Port 3)**

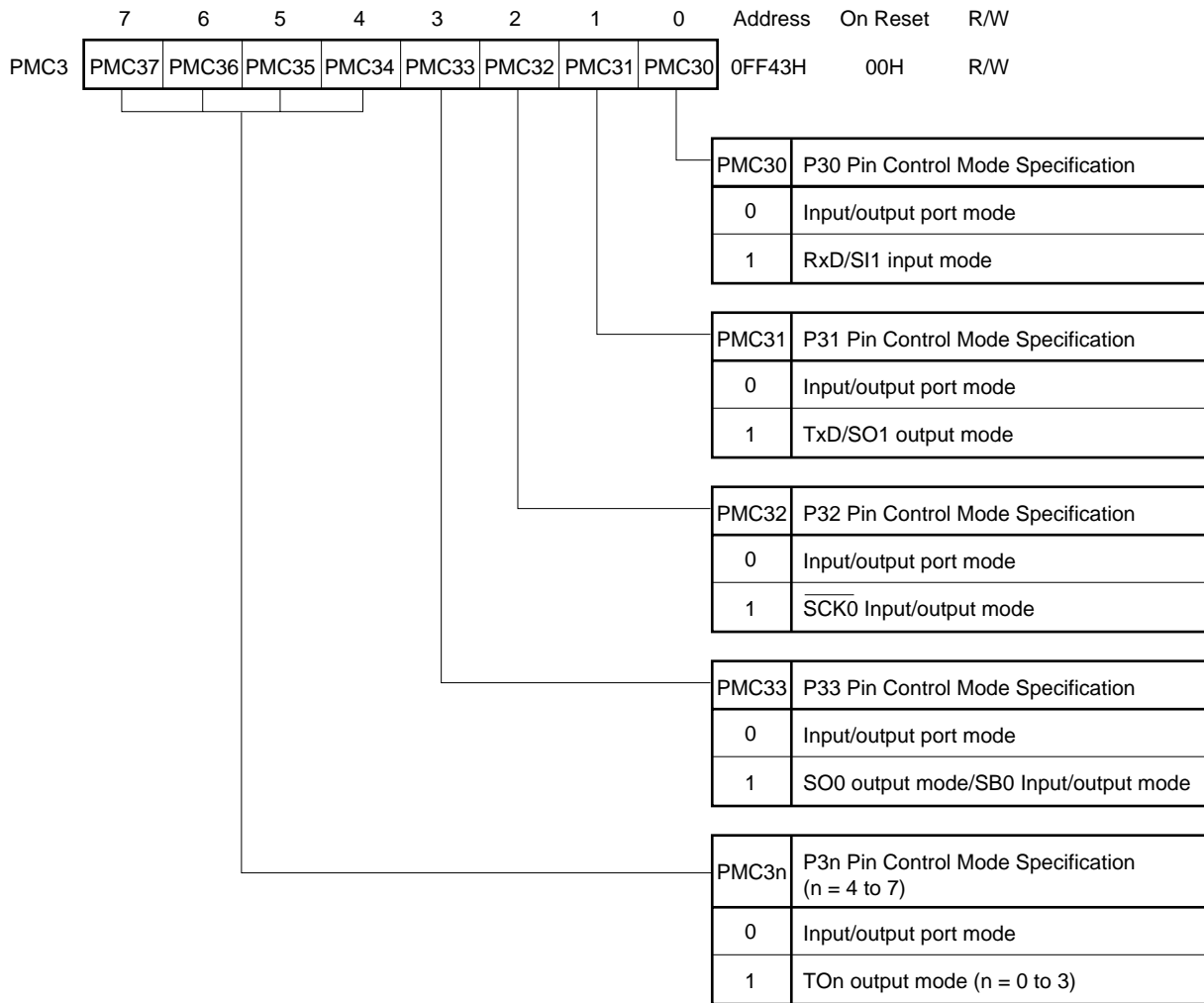
Figure 5-30 Block Diagram of P33 (Port 3)



In addition to their input/output port function, port 3 pins also have a dual function as various control signal pins, and the control mode is specified by means of the port 3 mode control register (PMC3) as shown in Figure 5-32.

	7	6	5	4	3	2	1	0	Address	On Reset	R/W
PM3	PM37	PM36	PM35	PM34	PM33	PM32	PM31	PM30	0FF23H	FFH	R/W

PM3n	P3n Pin Input/Output Mode Specification (n = 0 to 7)
0	Output mode (output buffer on)
1	Input mode (output buffer off)

**Figure 5-32 Port 3 Mode Control Register (PMC3) Format**

### 5.5.3 Operating Status

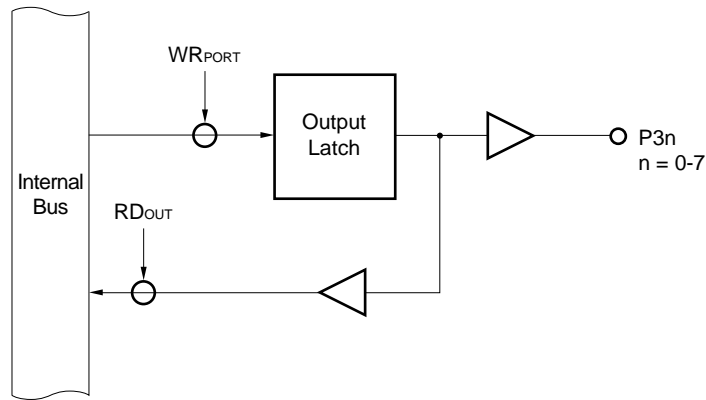
Port 3 is an input/output port, with a dual function as various control pins.

#### (1) When set as an output port

The output latch is enabled, and data transfers between the output latch and accumulator are performed by means of transfer instructions. The output latch contents can be freely set by means of logical operation instructions. Once data has been written to the output latch, it is retained until data is next written to the output latch\*.

\* Including the case where another bit of the same port is manipulated by a bit manipulation instruction.

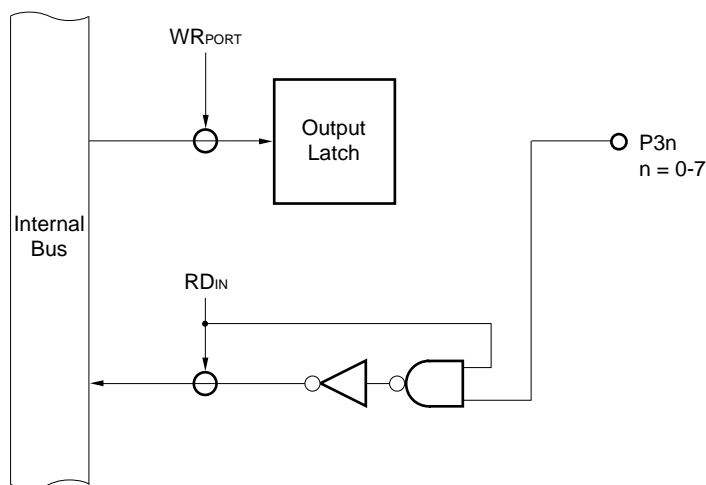
**Figure 5-33 Port Specified as Output Port**





**(2) When set as an input port**

The port pin level can be loaded into an accumulator by means of a transfer instruction. In this case, too, writes can be performed to the output latch, and data transferred from the accumulator by a transfer instruction, etc., is stored in all output latches irrespective of the port input/output specification. However, since the output buffer of a bit specified as an input port is high impedance, the data is not output to the port pin (when a bit specified as input is switched to an output port, the output latch contents are output to the port pin). Also, the contents of the output latch of a bit specified as an input port cannot be loaded into an accumulator.

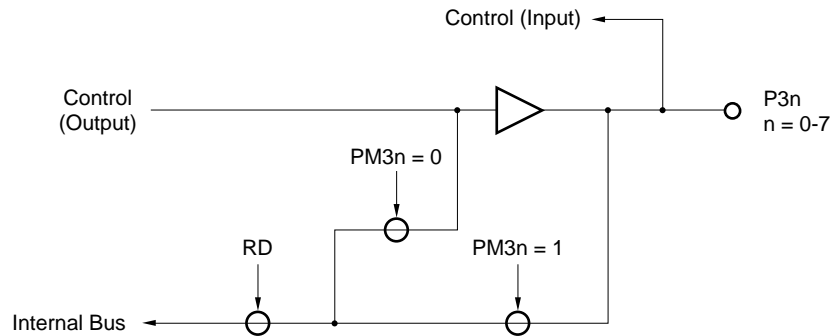
**Figure 5-34 Port Specified as Input Port**

**Caution** A bit manipulation instruction manipulates one bit as the result, but accesses the port in 8-bit units. Therefore, if a bit manipulation instruction is used on a port with a mixture of input and output pins or port mode and control mode, the contents of the output latch of pins specified as inputs and pins specified as control mode will be undefined (excluding bits manipulated with a SET1 or CLR1 instruction, etc.). Particular care is required when there are bits which are switched between input and output.

Caution is also required when manipulating the port with other 8-bit manipulation instructions.

**(3) When specified as control signal input/output**

By setting (1) bits of the port 3 mode control register (PMC3), port 3 can be used as control signal input or output bit-wise irrespective of the setting of the port 3 mode register (PM3). When a pin is used as a control signal, the control signal status can be seen by executing a port read instruction.

**Figure 5-35 Control Specification****(a) When port is control signal output**

When the port 3 mode register (PM3) is set (1), the control signal pin level can be read by executing a port read instruction.

When PM3 is reset (0), the  $\mu$ PD784026 internal control signal status can be read by executing a port read instruction.

**Remark** For bit 3 (P33) of port 3, the name "SB0" is a reserved word in the NEC assembler package. In the C compiler, it is defined as an sfr variable by the # pragma sfr directive.

**(b) When port is control signal input**

Only the port 3 mode register (PM3) is set (1), control signal pin levels can be read by executing a port read instruction.

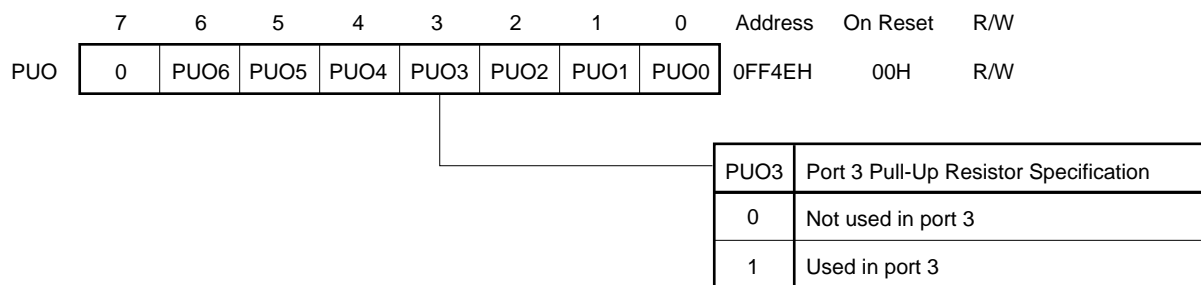
### 5.5.4 Internal Pull-Up Resistors

Port 3 incorporates pull-up resistors. Use of these internal resistors when pull-up is necessary enables the number of parts and the mounting area to be reduced.

Whether or not an internal pull-up resistor is to be used can be specified for each pin by means of the PU03 bit of the pull-up resistor option register (PUO) and the port 3 mode register (PM3). When PU03 is 1, the internal pull-up resistors of the pins for which input is specified by PM3 ( $PM3n = 1$ ,  $n = 0$  to 7) are enabled.

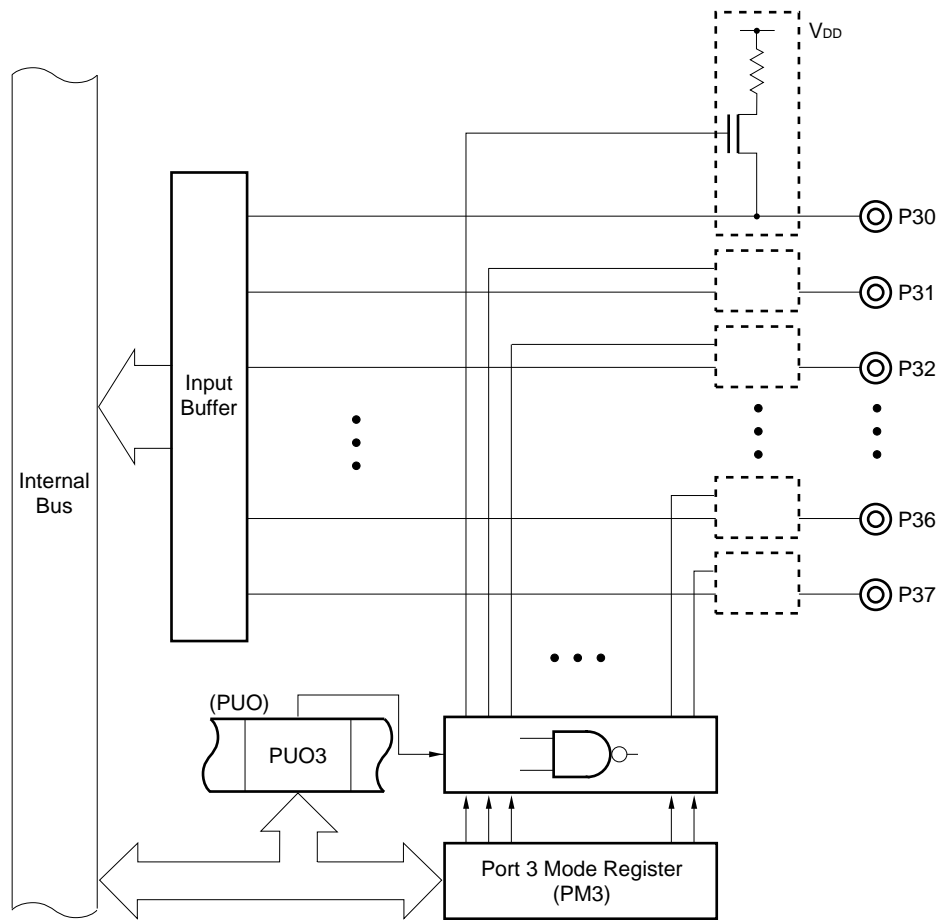
Also, the specification for use of the pull-up resistor is also valid for pins specified as control mode pins (pull-up resistors are also connected to pins that function as output pins in the control mode). Therefore, if you do not want to connect the pull-up resistors in the control mode, the contents of the corresponding bits of PM3 should be set to 0 (output mode).

**Figure 5-36 Pull-Up Resistor Option Register (PUO) Format**



**Remark** When STOP mode is entered, assigning 00H to PUO is effective in reducing the current consumption.

Figure 5-37 Pull-Up Specification (Port 3)



## 5.6 PORT 4

Port 4 is an 8-bit input/output port with an output latch. Input/output can be specified bit-wise by means of the port 4 mode register (PM4). Each pin incorporates a software programmable pull-up resistor. This port has direct LED drive capability.

Port 4 also functions as the time division address/data bus (AD0 to AD7) by the memory extension mode register (MM) when external memory or I/Os are extended.

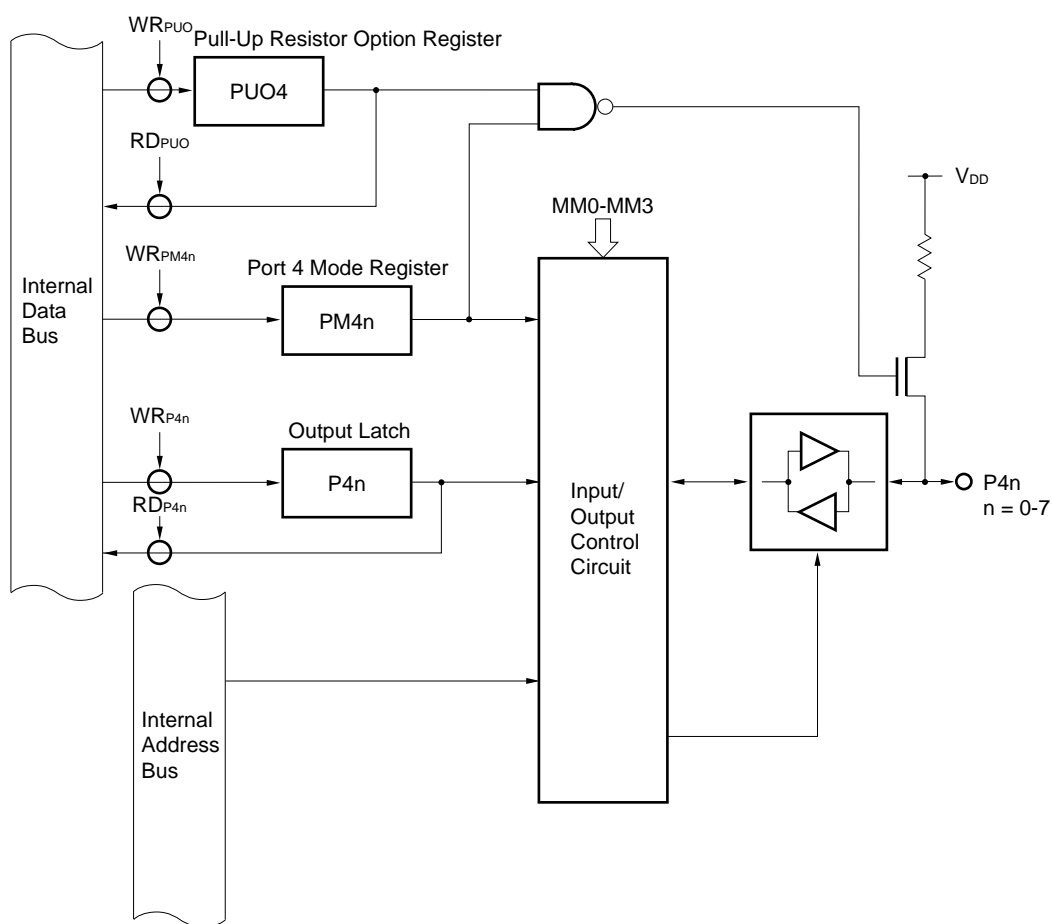
In the  $\mu$ PD784021, this port only functions as the time division address/data bus (AD0 to AD7).

When RESET is input, port 4 is set as an input port (output high-impedance state), and the output latch contents are undefined.

### 5.6.1 Hardware Configuration

The port 4 hardware configuration is shown in Figure 5-38.

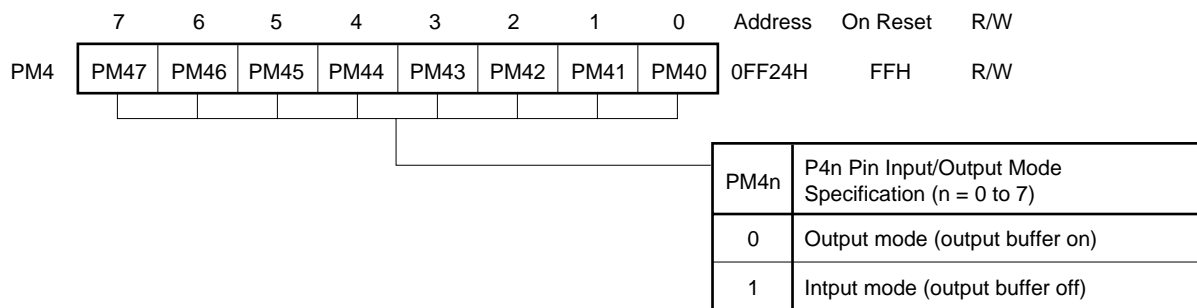
**Figure 5-38 Port 4 Block Diagram**



**5.6.2 Input/Output Mode / Control Mode Setting**

The port 4 input/output mode is set for each pin by means of the port 4 mode register (PM4) as shown in Figure 5-39.

When port 4 is used as the address/data bus, it is set by means of the memory extension mode register (MM: See **Figure 21-1**) as shown in Table 5-7.

**Figure 5-39 Port 4 Mode Register (PM4) Format****Table 5-7 Port 4 Operating Modes**

MM Bits				Operating Mode
MM3	MM2	MM1	MM0	
0	0	0	0	Port
0	0	1	1	Address/data bus (AD0 to AD7)
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	

In the  $\mu$ PD784021, port 4 only operates as the address/data bus (AD0 to AD7).

### 5.6.3 Operating Status

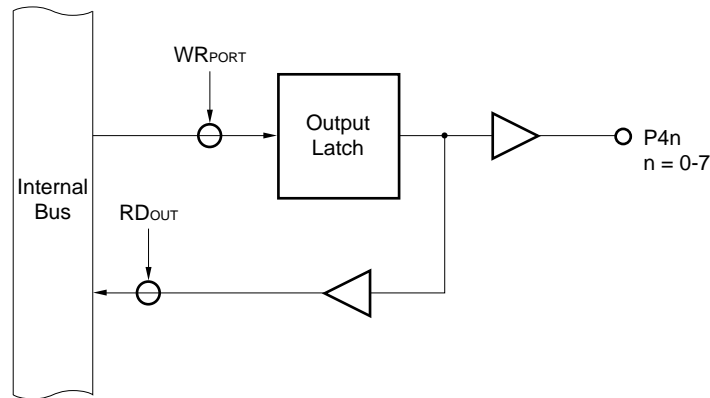
Port 4 is an input/output port, with a dual function as the address/data bus (AD0 to AD7).

#### (1) When set as an output port

The output latch is enabled, and data transfers between the output latch and accumulator are performed by means of transfer instructions. The output latch contents can be freely set by means of logical operation instructions. Once data has been written to the output latch, it is retained until data is next written to the output latch\*.

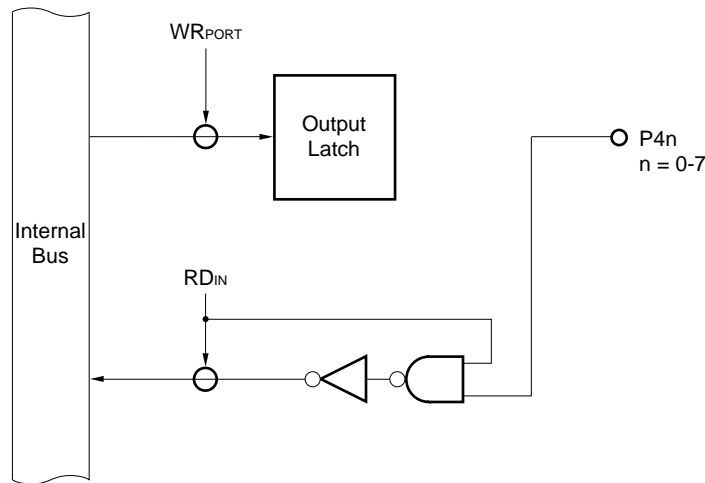
\* Including the case where another bit of the same port is manipulated by a bit manipulation instruction.

**Figure 5-40 Port Specified as Output Port**



**(2) When set as an input port**

The port pin level can be loaded into an accumulator by means of a transfer instruction. In this case, too, writes can be performed to the output latch, and data transferred from the accumulator by a transfer instruction, etc., is stored in all output latches irrespective of the port input/output specification. However, since the output buffer of a bit specified as an input port is high-impedance, the data is not output to the port pin (when a port specified as input is switched to an output port, the output latch contents are output to the port pin). Also, when specified as an input port, the output latch contents cannot be loaded into an accumulator.

**Figure 5-41 Port Specified as Input Port**

**Caution** A bit manipulation instruction manipulates one bit as the result, but accesses the port in 8-bit units. Therefore, if a bit manipulation instruction is used on a port with a mixture of input and output pins, the contents of the output latch of pins specified as inputs will be undefined (excluding bits manipulated with a SET1 or CLR1 instruction, etc.). Particular care is required when there are bits which are switched between input and output.

Caution is also required when manipulating the port with other 8-bit manipulation instructions.

**(3) When used as address/data bus (AD0 to AD7)**

Used automatically when an external access is performed.

Input/output instructions should not be executed on port 4.



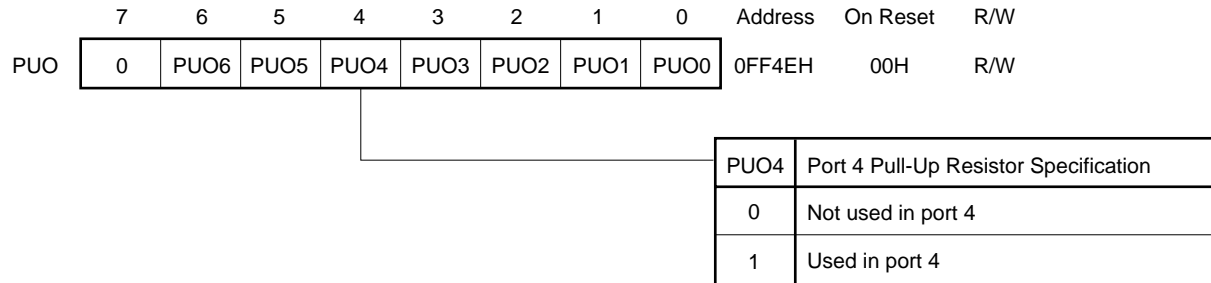
#### 5.6.4 Internal Pull-Up Resistors

Port 4 incorporates pull-up resistors. Use of these internal resistors when pull-up is necessary enables the number of parts and the mounting area to be reduced.

Whether or not an internal pull-up resistor is to be used can be specified for each pin by means of the PUO4 bit of the pull-up resistor option register (PUO) and the port 4 mode register (PM4).

When PUO4 is 1, the internal pull-up resistors of the pins for which input is specified by the PM4 for port 4 ( $PM4n = 1$ ,  $n = 0$  to 7) are enabled.

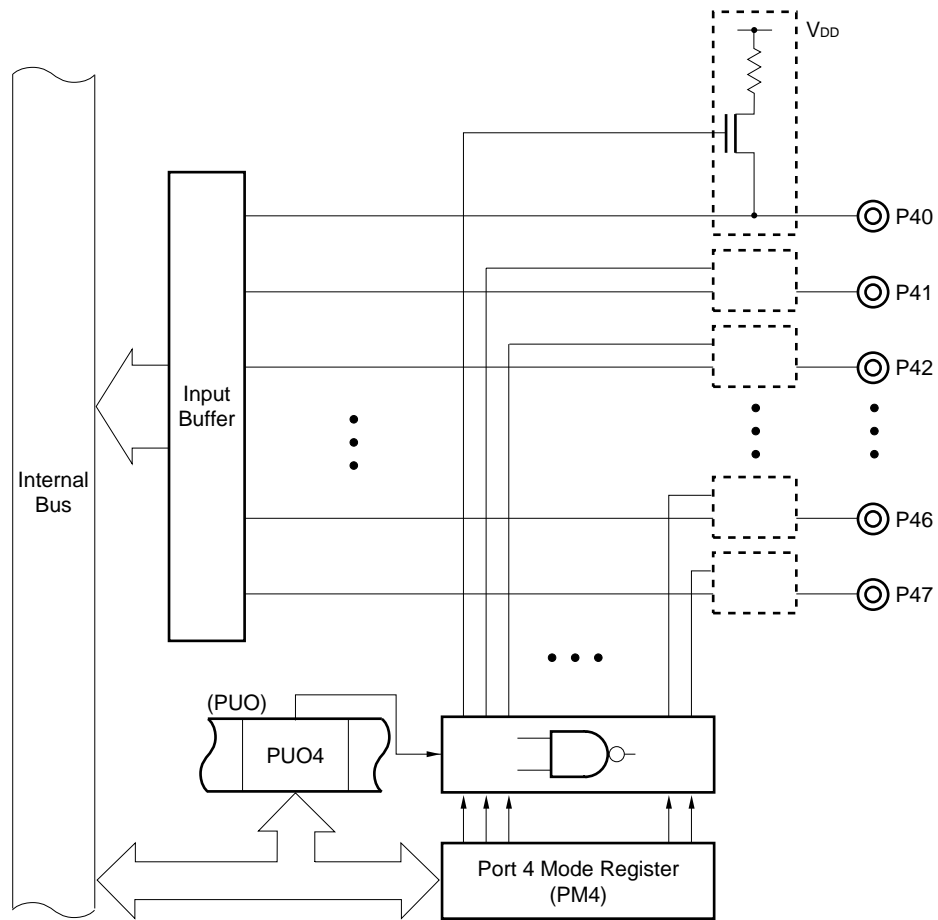
**Figure 5-42 Pull-Up Resistor Option Register (PUO) Format**



**Caution** In the  $\mu$ PD784021, port 4 is used as the address/data bus, and therefore “0” must be assigned to PUO4 so that internal pull-up resistor connection is not performed. The same also applies when port 4 is used as the address/data bus in the  $\mu$ PD784026.

**Remark** When STOP mode is entered, assigning 00H to PUO is effective in reducing the current consumption.

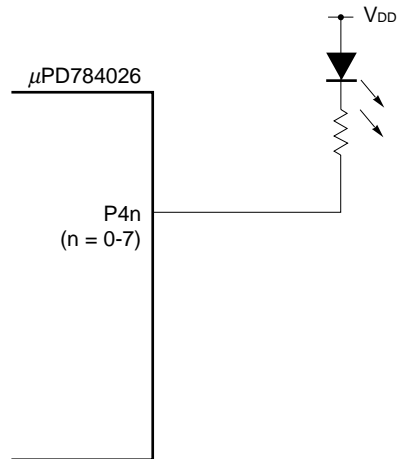
Figure 5-43 Pull-Up Specification (Port 4)



### 5.6.5 Direct LED Drive

In port 4, the output buffer low-level side drive capability has been reinforced, allowing active-low direct LED drive. An example of such use is shown in Figure 5-44.

**Figure 5-44 Example of Direct LED Drive**



## 5.7 PORT 5

Port 5 is an 8-bit input/output port with an output latch. Input/output can be specified bit-wise by means of the port 5 mode register (PM5). Each pin incorporates a software programmable pull-up resistor. This port has direct LED drive capability.

In addition, P50 to P57 function as the address bus (A8 to A15) when external memory or I/Os are extended.

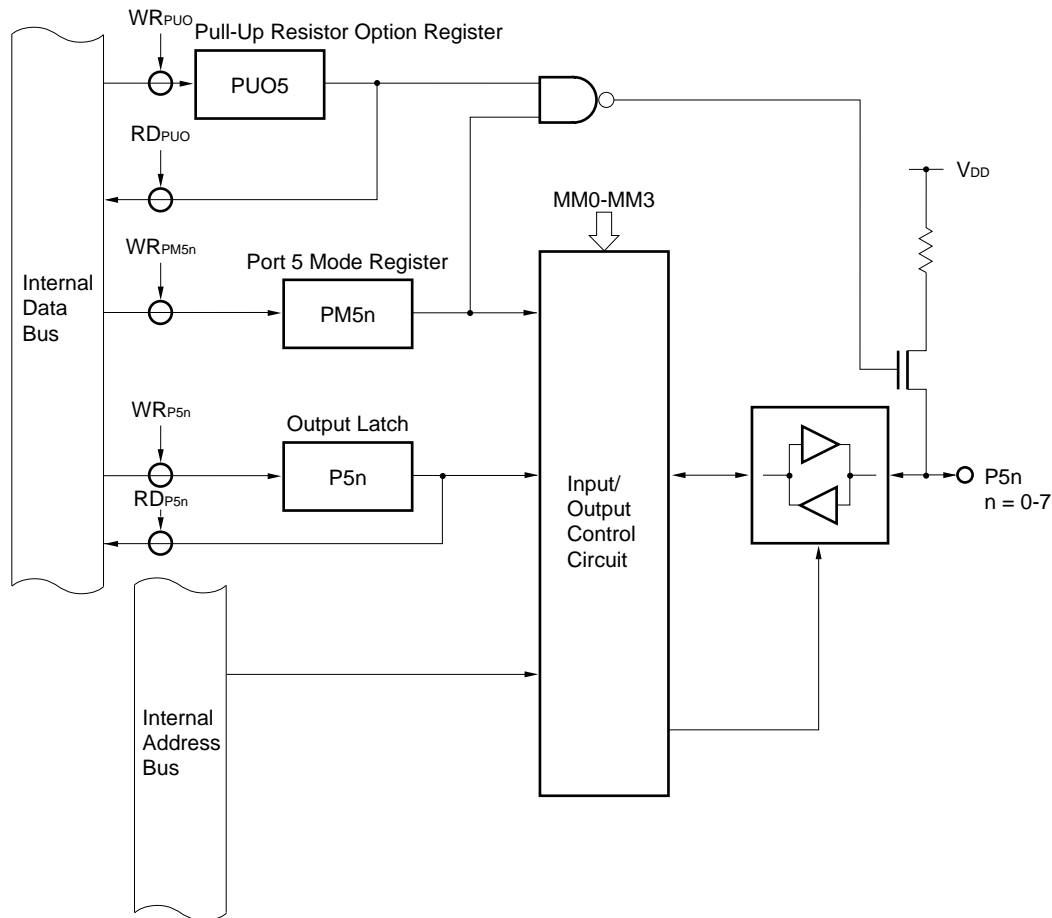
In the  $\mu$ PD784021, these pins only function as the address bus (A8 to A15).

When  $\overline{\text{RESET}}$  is input, port 5 is set as an input port (output high-impedance state), and the output latch contents are undefined.

### 5.7.1 Hardware Configuration

The port 5 hardware configuration is shown in Figure 5-45.

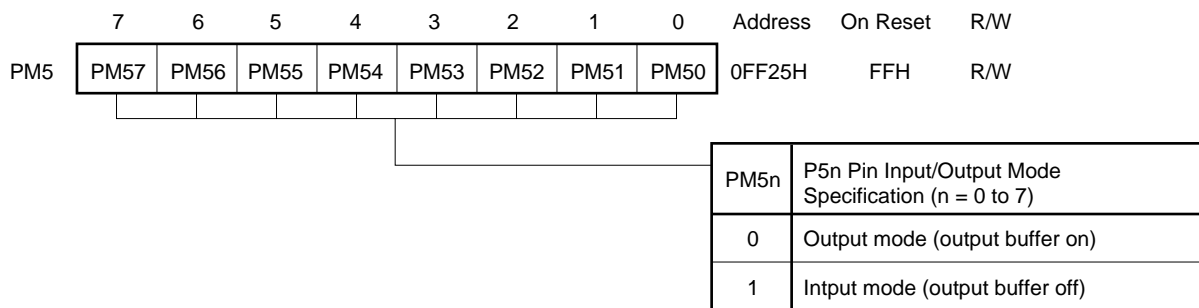
**Figure 5-45 Port 5 Block Diagram**



**5.7.2 Input/Output Mode/Control Mode Setting**

The port 5 input/output mode is set for each pin by means of the port 5 mode register (PM5) as shown in Figure 5-46.

When port 5 pins can be used as port or address pins in 2-bit units, the setting is performed by means of the memory extension mode register (MM: See **Figure 21-1**) as shown in Table 5-8.

**Figure 5-46 Port 5 Mode Register (PM5) Format****Table 5-8 Port 5 Operating Modes**

MM Bits				Operating Mode							
MM3	MM2	MM1	MM0	P50	P51	P52	P53	P54	P55	P56	P57
0	0	0	0	Port (P50 to P57)							
0	0	1	1								
0	1	0	0	A8	A9	Port					
0	1	0	1	A8	A9	A10	A11	Port			
0	1	1	0	A8	A9	A10	A11	A12	A13	Port	
0	1	1	1	A8	A9	A10	A11	A12	A13	A14	A15
1	0	0	0								
1	0	0	1								

In the  $\mu$ PD784021, port 5 only operates as the address bus (A8 to A15).

### 5.7.3 Operating Status

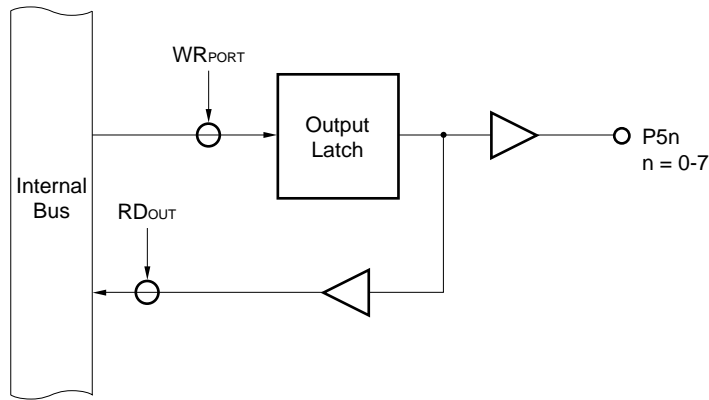
Port 5 is an input/output port, with a dual function as the address bus (A8 to A15).

#### (1) When set as an output port

The output latch is enabled, and data transfers between the output latch and accumulator are performed by means of transfer instructions. The output latch contents can be freely set by means of logical operation instructions. Once data has been written to the output latch, it is retained until data is next written to the output latch\*.

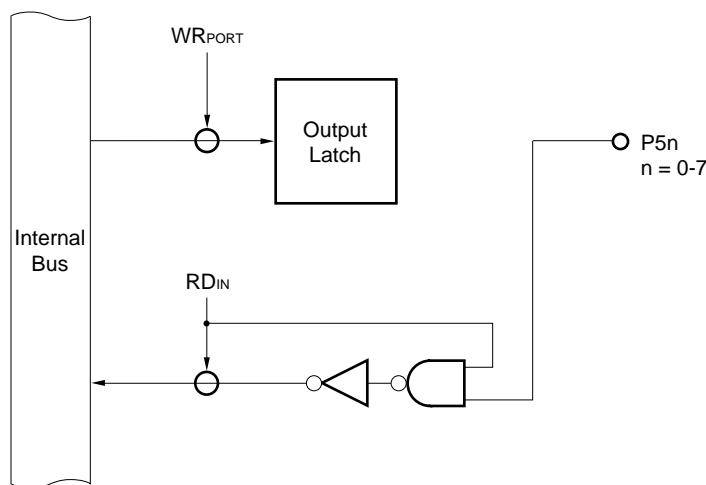
\* Including the case where another bit of the same port is manipulated by a bit manipulation instruction.

**Figure 5-47 Port Specified as Output Port**



**(2) When set as an input port**

The port pin level can be loaded into an accumulator by means of a transfer instruction. In this case, too, writes can be performed to the output latch, and data transferred from the accumulator by a transfer instruction, etc., is stored in all output latches irrespective of the port input/output specification. However, since the output buffer of a bit specified as an input port is high-impedance, the data is not output to the port pin (when a bit specified as input is switched to an output port, the output latch contents are output to the port pin). Also, the contents of the output latch of a bit specified as an input port cannot be loaded into an accumulator.

**Figure 5-48 Port Specified as Input Port**

**Caution** A bit manipulation instruction manipulates one bit as the result, but accesses the port in 8-bit units. Therefore, if a bit manipulation instruction is used on a port with a mixture of input and output pins, the contents of the output latch of pins specified as inputs will be undefined (excluding bits manipulated with a SET1 or CLR1 instruction, etc.). Particular care is required when there are bits which are switched between input and output.

Caution is also required when manipulating the port with other 8-bit operation instructions.

**(3) When used as address bus (A8 to A15)**

Used automatically when an external address is accessed.

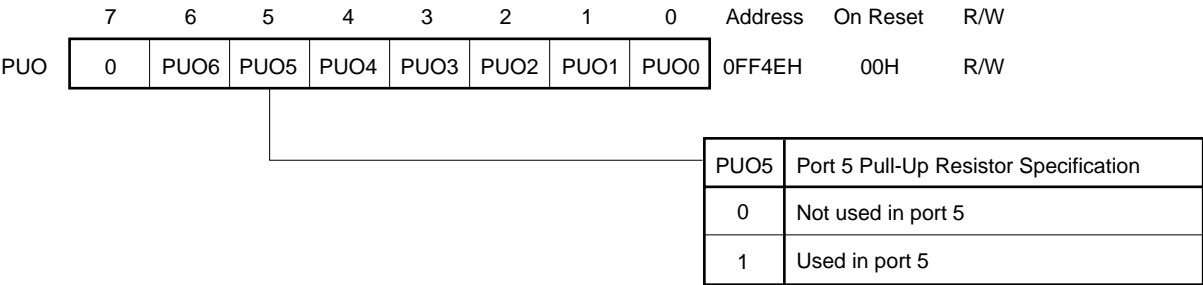
5.7.4 Internal Pull-Up Resistors

Port 5 incorporates pull-up resistors. Use of these internal resistors when pull-up is necessary enables the number of parts and the mounting area to be reduced.

Whether or not an internal pull-up resistor is to be used can be specified for each pin by means of the PU05 bit of the pull-up resistor option register (PUO) and the port 5 mode register (PM5).

When PU05 is 1, the internal pull-up resistors of the pins for which input is specified by the PM5 for port 5 (PM5n = 1, n = 0 to 7) are enabled .

Figure 5-49 Pull-Up Resistor Option Register (PUO) Format



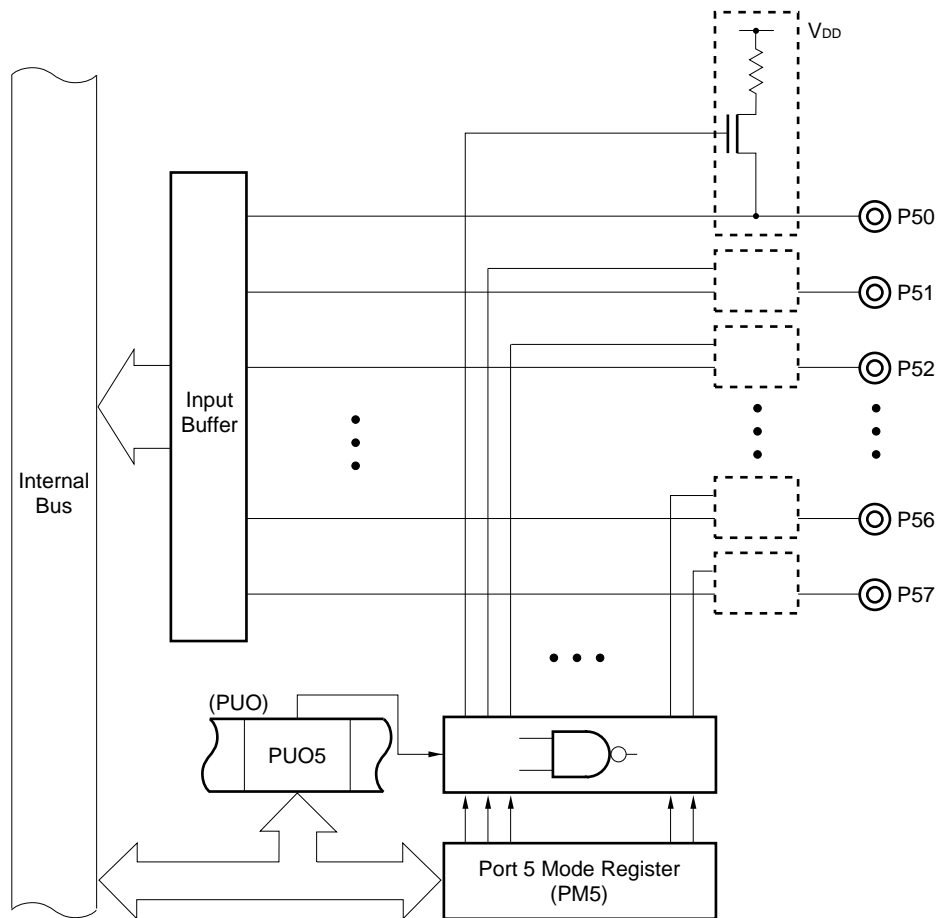
**Caution** In the  $\mu$ PD784021, port 5 is used as the address bus, and therefore “0” must be assigned to PU05 so that internal pull-up resistor connection is not performed.

The same also applies when port 5 is used as the address bus in the  $\mu$ PD784026.

**Remark** When STOP mode is entered, assigning 00H to PU0 is effective in reducing the current consumption.



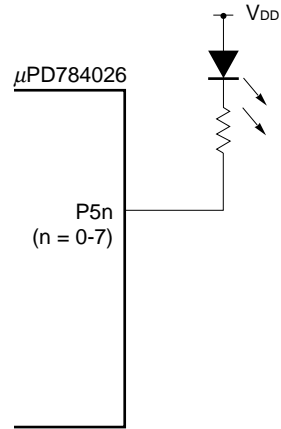
Figure 5-50 Pull-Up Specification (Port 5)



### 5.7.5 Direct LED Drive

In port 5, the output buffer low-level side drive capability has been reinforced, allowing active-low direct LED drive. An example of such use is shown in Figure 5-51.

**Figure 5-51 Example of Direct LED Drive**



## 5.8 PORT 6

Port 6 is an 8-bit input/output port with an output latch (in the  $\mu$ PD784021, P60 to P63 are output-only port pins). P60 to P67 (P64 to P67 in the  $\mu$ PD784021) incorporate a software programmable pull-up resistor.

In addition to its function as a port, port 6 also has various dual-function control signal pin functions as shown in Table 5-9. Operations as control pins are performed by the respective function operations.

In the  $\mu$ PD784021, P64 and P65 only function as the  $\overline{RD}$  output and  $\overline{WR}$  output, respectively.

When  $\overline{RESET}$  is input, P60 to P67 are set as input port pins (output high-impedance state), and the output latch contents are undefined.

In the  $\mu$ PD784021, P60 to P63 become high-impedance when  $\overline{RESET}$  is input, and are driven low after  $\overline{RESET}$  release. P64 to P67 are set as input port pins (output high-impedance state). In the  $\mu$ PD784021, the high-order 4 bits of the output latch contents are undefined, and the low-order 4 bits are 0H.

Table 5-9 Port 6 Operating Modes

Pin Name	Port Mode	Control Signal Input/ Output Mode	Operation to Operate as Control Pins
P60 to P63	Input/output ports*	A16 to A19 outputs	Specified by bits MM3 to MM0 of the MM in 2-bit units
P64	Input/output ports	$\overline{RD}$ output	In the $\mu$ PD784021, external memory extension mode specified by bits MM3 to MM0 of the MM
P65		$\overline{WR}$ output	
P66		$\overline{WAIT}$ input	Specified by bits PWN1 & PWN0 (n = 0 to 7) of the PWC1 & PWC2 or setting P66 in the input mode
P67		HLDRQ input	Bus hold enabled by the HLDE bit of the HLDM
		HLDAK output	
		$\overline{REFRQ}$ output	Set (1) the RFEN bit of the RFM

\* Output ports in the  $\mu$ PD784021

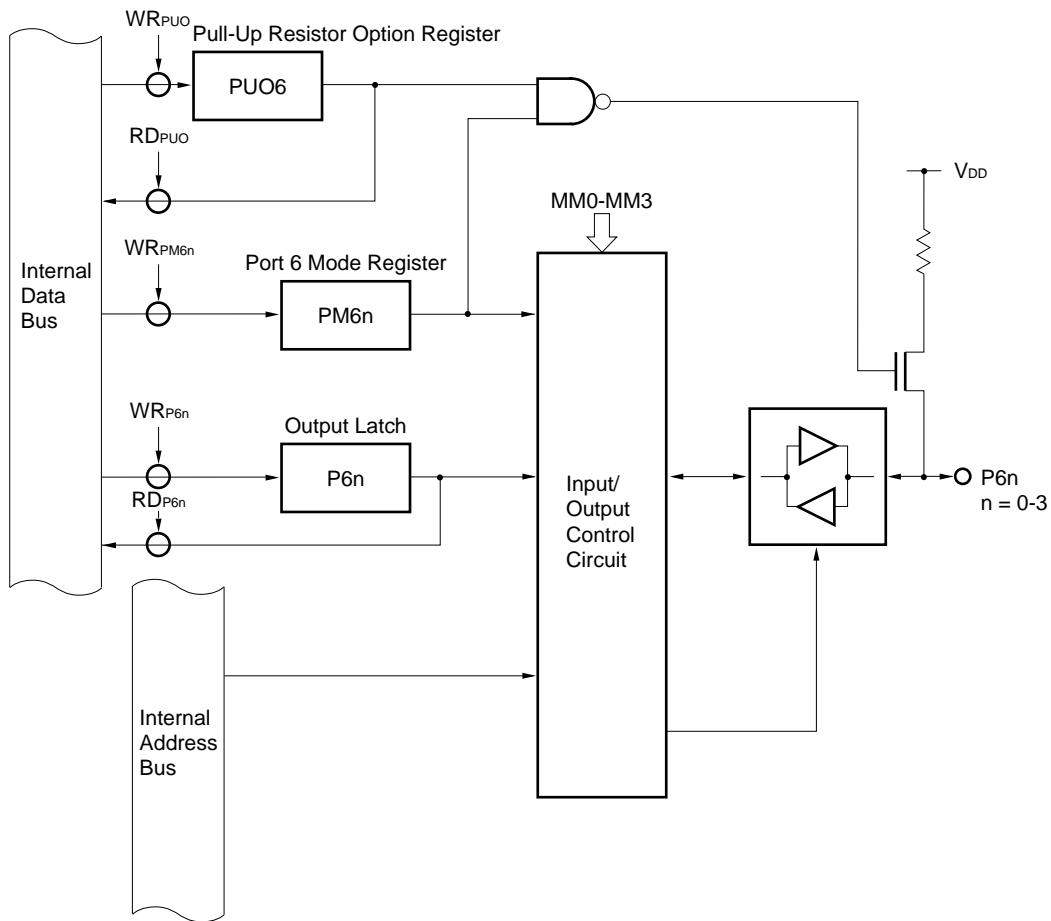
**Caution** In the  $\mu$ PD784021, P60 to P63 are in the output high-impedance state during  $\overline{RESET}$  input, but output a low level after  $\overline{RESET}$  release. Therefore, external circuitry should be designed so that output of a low-level signal as the initial state is no problem.

**Remark** See CHAPTER 21 LOCAL BUS INTERFACE FUNCTION for details.

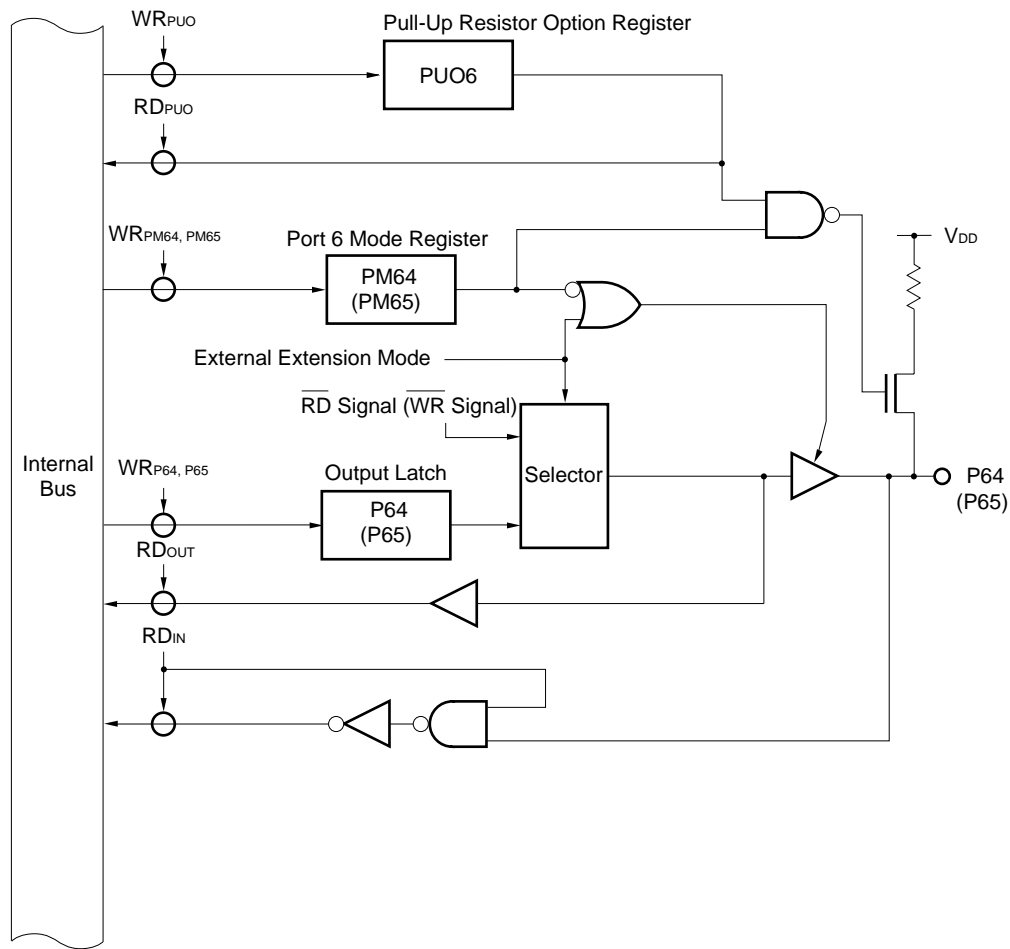
**5.8.1 Hardware Configuration**

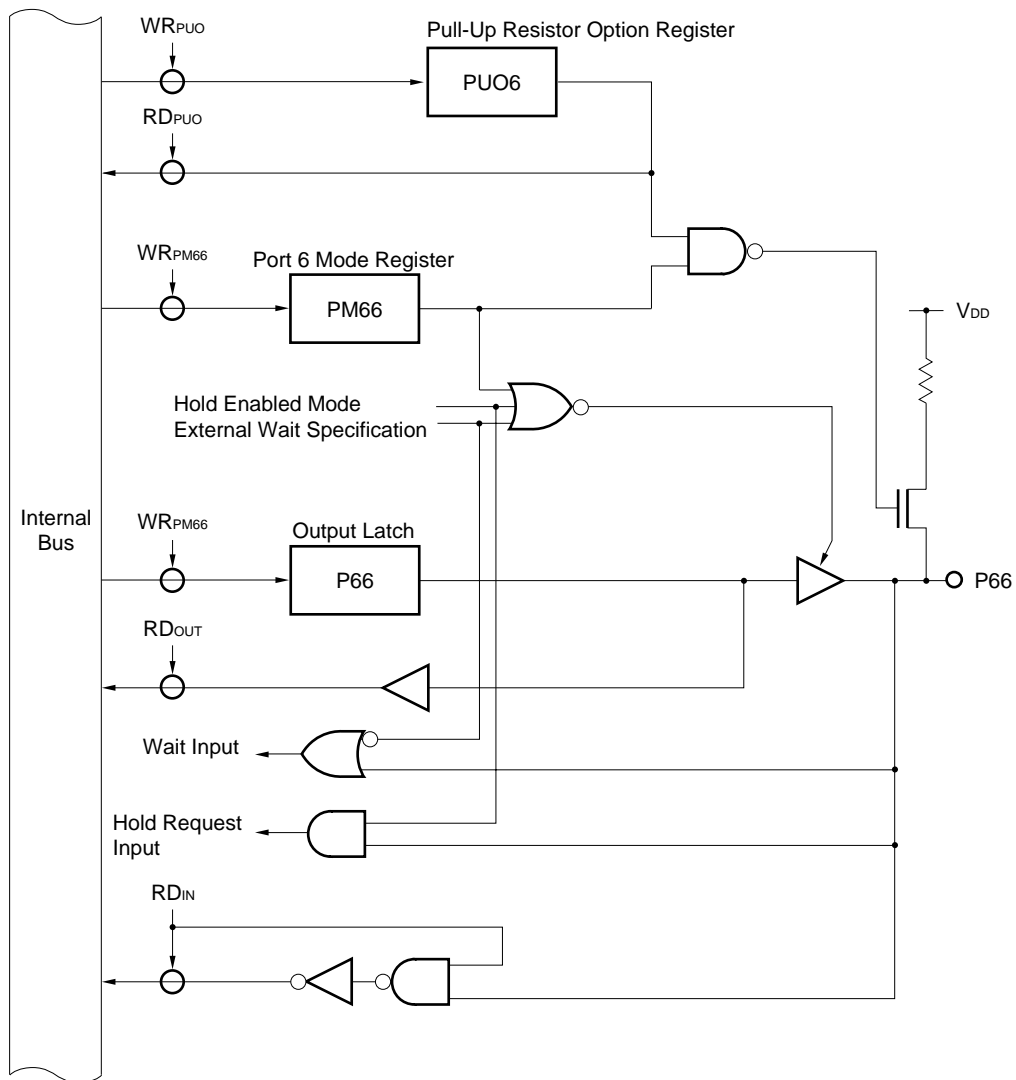
The port 6 hardware configuration is shown in Figures 5-52 to 5-55.

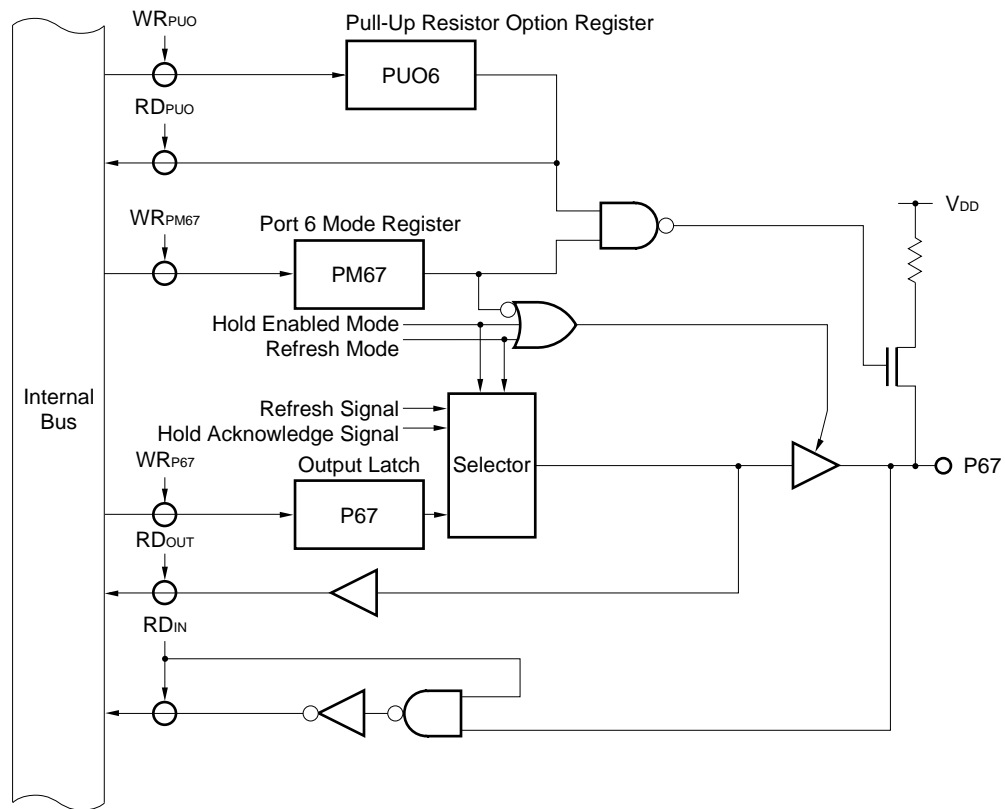
**Figure 5-52 Block Diagram of P60 to P63 (Port 6)**



**Remark** The  $\mu$ PD784021 has no function for input operations.

**Figure 5-53 Block Diagram of P64 and P65 (Port 6)**

**Figure 5-54 Block Diagram of P66 (Port 6)**

**Figure 5-55 Block Diagram of P67 (Port 6)**

### 5.8.2 Input/Output Mode/Control Mode Setting

The port 6 input/output mode is set by means of the port 6 mode register (PM6) as shown in Figure 5-56.

Operations for operating port 6 as control pins are shown in Table 5-10.

In the  $\mu$ PD784021, P64 and P65 operate only as the  $\overline{\text{RD}}$  and  $\overline{\text{WR}}$  signal output pins.

**Table 5-10 Port 6 Control Pin Functions and Operations**

Pin Name	Control Signal I/O Mode	Port Mode	Operation to Operate as Control Pins
P60	A16	Input/output port*	External memory extension mode specified by bits MM3 to MM0 of the MM (see <b>Table 5-11</b> )
P61	A17		
P62	A18		
P63	A19		
P64	$\overline{\text{RD}}$	Output port	In the $\mu$ PD784021, or external memory extension mode specified by bits MM3 to MM0 of the MM (see <b>Table 5-11</b> )
P65	$\overline{\text{WR}}$		
P66	$\overline{\text{WAIT}}$	Input/output port	External wait input specified by bits PWN1 & PWN0 (n = 0 to 7) of the PWC1 & PWC2
	HLDRQ		Bus hold enabled by the HLDE bit of the HLDM
P67	HLDKAK	Output port	Set (1) the RFEN bit of the RFM
	$\overline{\text{REFRQ}}$		

\* Output ports in the  $\mu$ PD784021

**Table 5-11 P60 to P65 Control Pin Specification**

MM Bits				Operating Mode					
MM3	MM2	MM1	MM0	P60	P61	P62	P63	P64	P65
0	0	0	0	Port (P60 to P65)					
0	0	1	1						
0	1	0	0	Port (P60 to P63)				$\overline{\text{RD}}$	$\overline{\text{WR}}$
0	1	0	1						
0	1	1	0						
0	1	1	1						
1	0	0	0						
1	0	0	1	A16	A17	Port			
				A16	A17	A18	A19		

#### (a) Port mode

##### (i) $\mu$ PD784026

Each port not specified as in control mode can be specified as input/output bit-wise by means of the port 6 mode register (PM6).

##### (ii) $\mu$ PD784021

Regarding each port not specified as in control mode, P60 to P63 are output-only port, and P64 to P67 can be specified bit-wise as input/output by means of the port 6 mode register (PM6).



**(b) Control signal input/output mode****(i) A16 to A19 (Address Bus)**

Upper address bus output pins when the external memory space is extended (10000H-FFFFFH). These pins operate in accordance with the memory extension mode register (MM).

**(ii)  $\overline{\text{RD}}$  (Read Strobe)**

The strobe signal for an external memory read operation. The operation of this pin is controlled by the memory extension mode register (MM).

This pin of the  $\mu\text{PD784021}$  usually operates as an  $\overline{\text{RD}}$  pin.

**(iii)  $\overline{\text{WR}}$  (Write Strobe)**

Pin that outputs the strobe signal for an external memory write operation. The operation of this pin is controlled by the memory extension mode register (MM).

This pin of the  $\mu\text{PD784021}$  usually operates as an  $\overline{\text{WR}}$  pin.

**(iv)  $\overline{\text{WAIT}}$  (Wait)**

Wait signal input pin. Operates in accordance with the programmable wait control registers (PWC1, PWC2).

**(v)  $\text{HLDRQ}$  (Hold Request)**

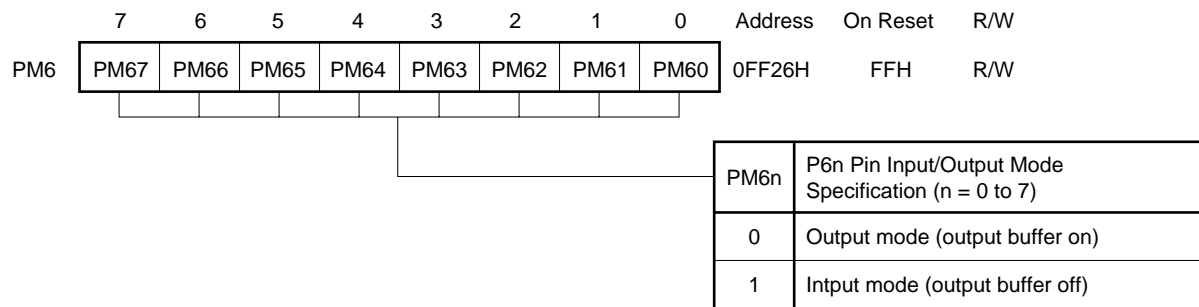
External bus hold request signal input pin. Operates in accordance with the hold mode register (HLDM).

**(vi)  $\text{HLDAK}$  (Hold Acknowledge)**

Bus hold acknowledge signal output pin. Operates in accordance with the hold mode register (HLDM).

**(vii)  $\overline{\text{REFRQ}}$  (Refresh Request)**

This pin outputs refresh pulses to pseudo-static memory when this memory is connected to it externally. Operates in accordance with the refresh mode register (RFM).

**Figure 5-56 Port 6 Mode Register (PM6) Format**

**Remark** In the  $\mu$ PD784021, the low-order 4 bits of port 6 (P60 to P63) are output-only ports.

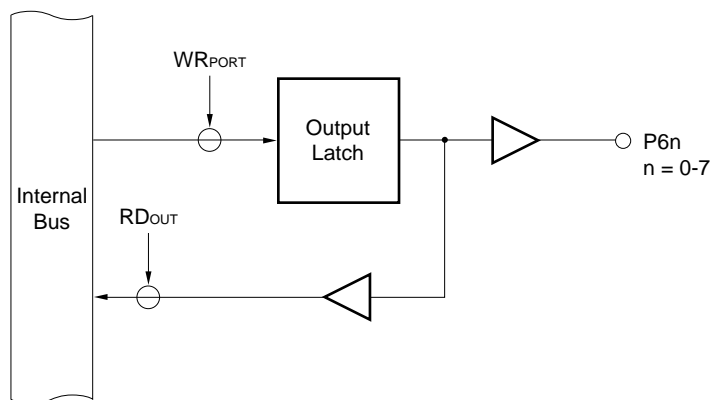
### 5.8.3 Operating Status

Port 6 is an input/output port, with a dual function as various control pins.

#### (1) When set as an output port

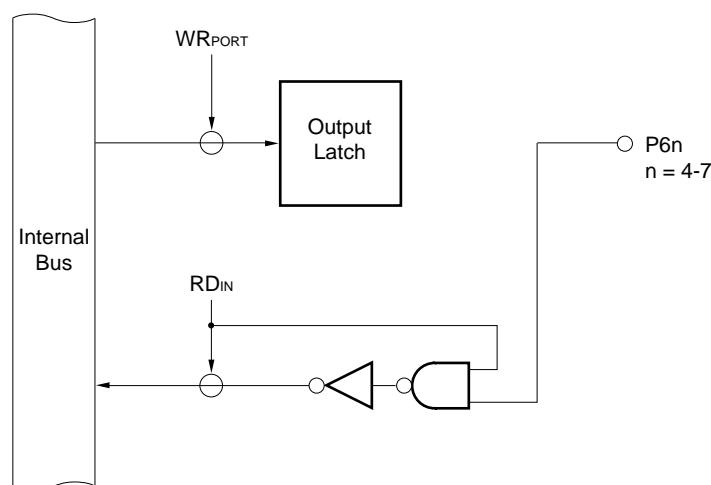
The output latch is enabled, and data transfers between the output latch and accumulator are performed by means of transfer instructions. The output latch contents can be freely set by means of logical operation instructions. Once data has been written to the output latch, it is retained until data is next written to the output latch\*.

\* Including the case where another bit of the same port is manipulated by a bit manipulation instruction.

**Figure 5-57 Port Specified as Output Port**

**(2) When set as an input port**

The port pin level can be loaded into an accumulator by means of a transfer instruction. In this case, too, writes can be performed to the output latch, and data transferred from the accumulator by a transfer instruction, etc., is stored in all output latches irrespective of the port input/output specification. However, since the output buffer of a bit specified as an input port is high-impedance, the data is not output to the port pin (when a bit specified as input is switched to an output port, the output latch contents are output to the port pin). Also, the contents of the output latch of a bit specified as an input port cannot be loaded into an accumulator.

**Figure 5-58 Port Specified as Input Port**

**Caution** A bit manipulation instruction manipulates one bit as the result, but accesses the port in 8-bit units. Therefore, if a bit manipulation instruction is used on a port with a mixture of input and output pins, or port mode and control mode, the contents of the output latch of pins specified as inputs or pins specified as in the control mode will be undefined (excluding bits manipulated with a SET1 or CLR1 instruction, etc.). Particular care is required when there are bits which are switched between input and output.

Caution is also required when manipulating the port with other 8-bit manipulation instructions.

**(3) When used as control pins**

Cannot be manipulated or tested by software.

5.8.4 Internal Pull-Up Resistors

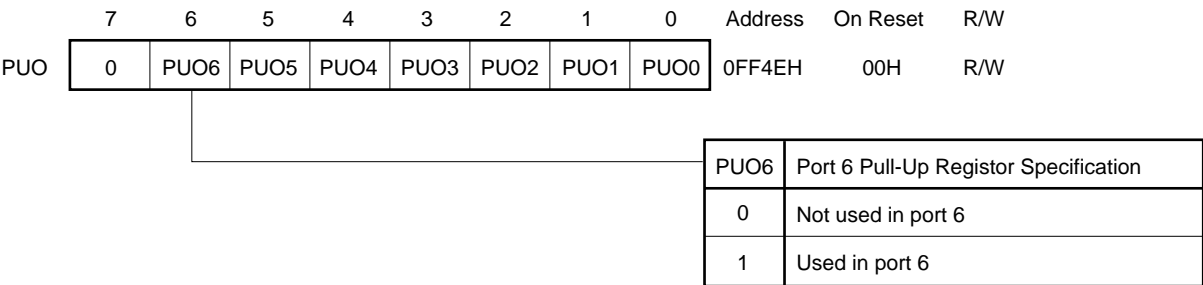
P60 to P67 (P64 to P67 in the  $\mu$ PD784021) incorporate pull-up resistors. Use of these internal resistors when pull-up is necessary enables the number of parts and the mounting area to be reduced.

Whether or not an internal pull-up resistor is to be used can be specified for each pin by means of the PUO6 bit of the pull-up resistor option register (PUO) and the port 6 mode register (PM6).

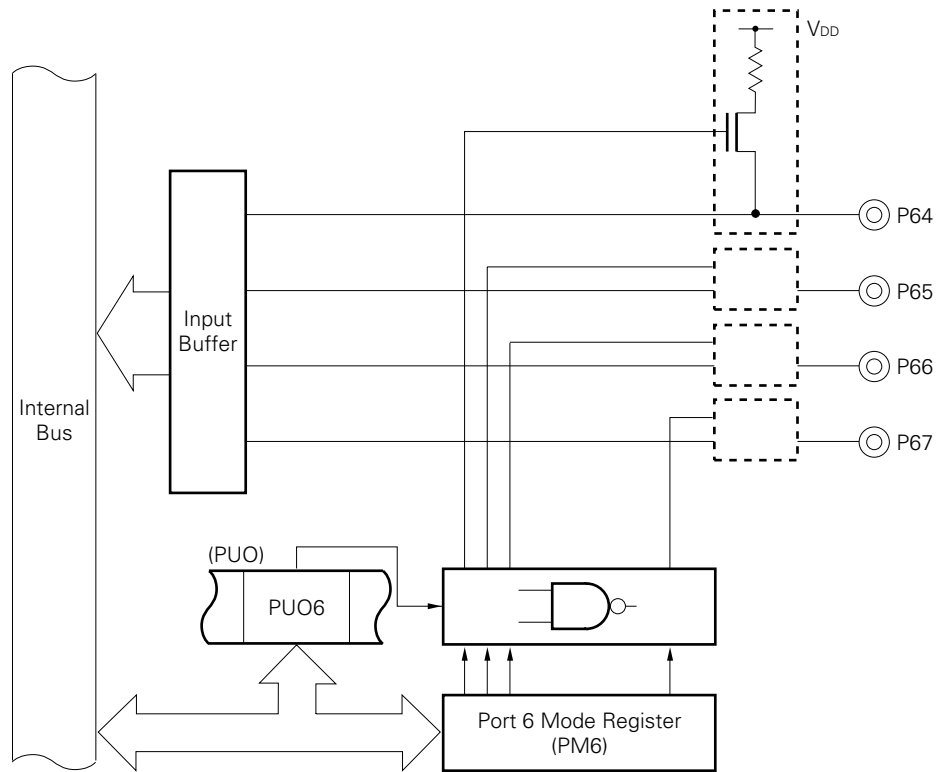
When PUO6 is 1, the internal pull-up resistors of the pins for which input is specified by the PM6 (PM6n = 1, n = 0 to 7) are enabled .

In the  $\mu$ PD784021, P60 to P63 do not incorporate a pull-up resistor.

Figure 5-59 Pull-Up Resistor Option Register (PUO) Format



**Remark** When STOP mode is entered, assigning 00H to PUO is effective in reducing the current consumption.

**Figure 5-60 Pull-Up Specification (Port 6)**

**5.9 PORT 7**

Port 7 is an 8-bit input/output port. In addition to operating as an input/output port, it also operates as the A/D converter analog input pins (ANI0 to ANI7).

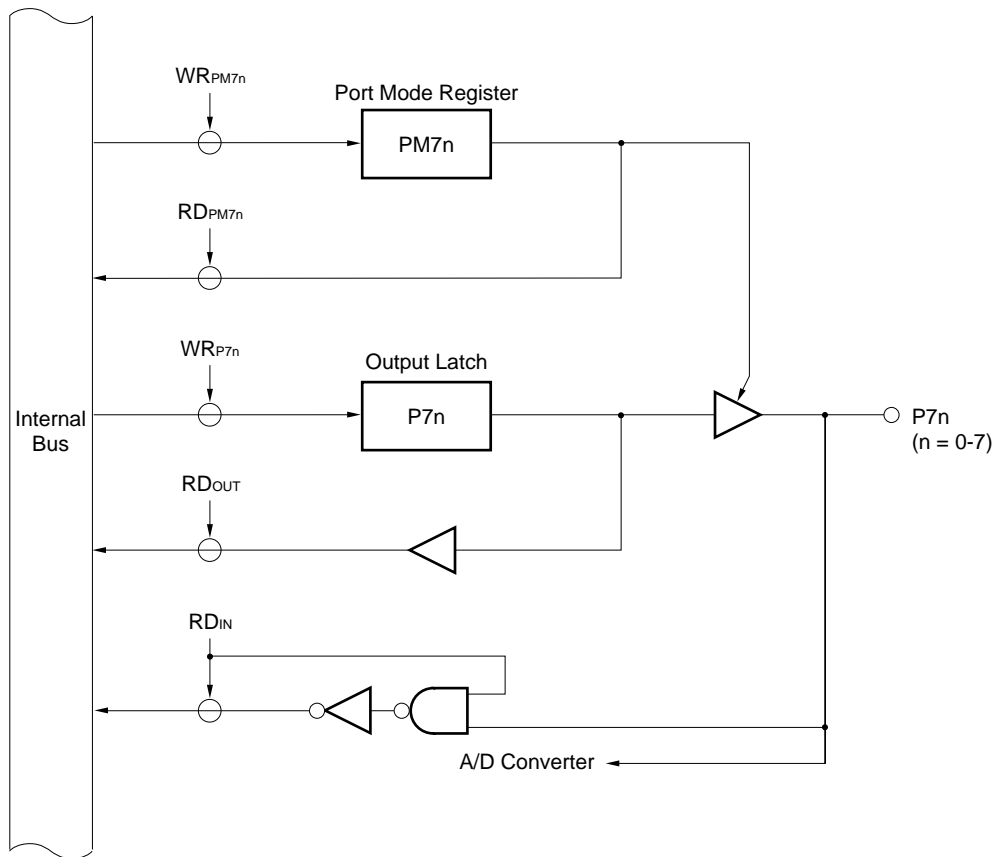
Input/output can be specified bit-wise by means of the port 7 mode register (PM7).

Pin levels can be read or tested at any time irrespective of dual-function pin operations.

When  $\overline{\text{RESET}}$  is input, port 7 is set as an input port (output high-impedance state), and the output latch contents are undefined.

**5.9.1 Hardware Configuration**

The port 7 hardware configuration is shown in Figure 5-61.

**Figure 5-61 Port 7 Block Diagram**

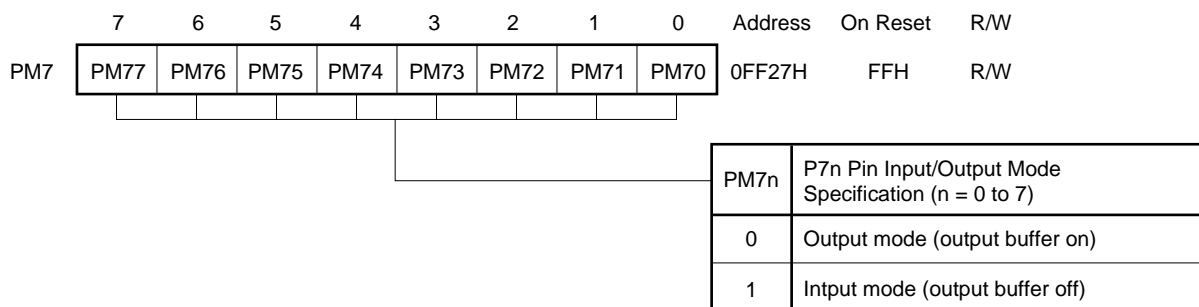
**5.9.2 Input/Output Mode/Control Mode Setting**

The port 7 input/output mode is set for each pin by means of the port 7 mode register (PM7) as shown in Figure 5-62.

In addition to the operation of port 7 as an input/output port, analog signal input can be performed at any time. Mode setting is not necessary.

Specification of the A/D conversion operation is performed by ADM of the A/D converter (see **CHAPTER 14 A/D CONVERTER** for details).

**Figure 5-62 Port 7 Mode Register (PM7) Format**



**5.9.3 Operating Status**

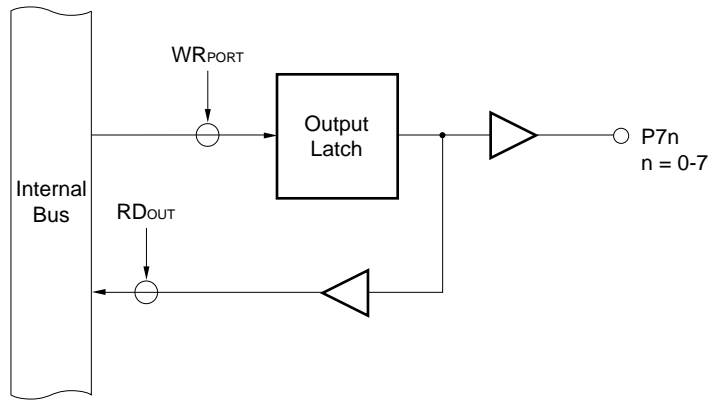
Port 7 is an input/output port, with a dual function as the A/D converter analog input pins (ANI0 to ANI7).

**(1) When set as an output port**

The output latch is enabled, and data transfers between the output latch and accumulator are performed by means of transfer instructions. The output latch contents can be freely set by means of logical operation instructions. Once data has been written to the output latch, it is retained until data is next written to the output latch\*.

\* Including the case where another bit of the same port is manipulated by a bit manipulation instruction.

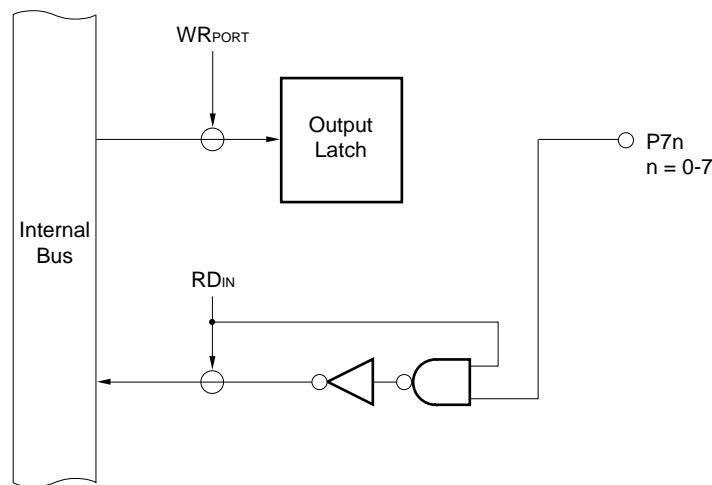
**Figure 5-63 Port Specified as Output Port**





**(2) When set as an input port**

The port pin level can be loaded into an accumulator by means of a transfer instruction. In this case, too, writes can be performed to the output latch, and data transferred from the accumulator by a transfer instruction, etc., is stored in all output latches-irrespective of the port input/output specification. However, since the output buffer of a bit specified as an input port is high-impedance, the data is not output to the port pin (when a bit specified as input is switched to an output port, the output latch contents are output to the port pin). Also, the contents of the output latch of a bit specified as an input port cannot be loaded into an accumulator.

**Figure 5-64 Port Specified as Input Port**

**Caution** A bit manipulation instruction manipulates one bit as the result, but accesses the port in 8-bit units. Therefore, if a bit manipulation instruction is used on a port with a mixture of input and output pins, the contents of the output latch of pins specified as inputs will be undefined (excluding bits manipulated with a SET1 or CLR1 instruction, etc.). Particular care is required when there are bits which are switched between input and output.

Caution is also required when manipulating the port with other 8-bit operation instructions.

**5.9.4 Internal Pull-Up Resistors**

Port 7 does not incorporate pull-up resistors.

**5.9.5 Caution**

A voltage outside the range  $AV_{SS}$  to  $AV_{REF}$  must not be applied to pins for which P70 to P77 are used as ANI0 to ANI7. See **14.5 CAUTIONS** in **CHAPTER 14 “A/D CONVERTER”** for details.

**5.10 PORT OUTPUT CHECK FUNCTION**

The  $\mu$ PD784026 has a function for reading and testing output port pin levels in order to improve the reliability of application systems. It is therefore possible to check the output data and the actual pin status as required. If there is a mismatch, appropriate action can be taken, such as replacement with another system.

Special instructions, CHKL and CHKLA, are provided to check the port status. These instructions perform a comparison by taking the exclusive OR of the pin status and the output latch contents (in port mode), or the pin status and the internal control output signal level (in control mode).

**Example** An example is shown below of a program that checks the pin status and output latch contents using the CHKL instruction and CHKLA instruction.

```

TEST :   SET1    P0.3           ; Set bit 3 of port 0
         CHKL    P0             ; Check port 0
         BNE     $ERR1          ; Branch to error processing (ERR1) in case of mismatch with output
                                   latch contents
         .
         .
         .
ERR1 :   CHKLA   P0             ; Faulty bit check
         BT      A.7, $BIT07     ; Bit 7?
         BT      A.6, $BIT06     ; Bit 6?
         .
         .
         .
         BT      A.1, $BIT01     ; Bit 1?
         BR      $BIT00          ; If none of the bits, bit 0 is faulty

```

- Cautions**
1. If each port is set to input mode, a comparison of the pin status with the output latch contents (or control output level) using the CHKL or CHKLA instruction will always show a match whether the individual pins of the port are port pins or control pins.  
Therefore, executing these instructions on a port set to input mode is actually ineffective.
  2. If the output levels of a port in which control outputs and port outputs are mixed in a single port are checked with the CHKL or CHKLA instruction, the input/output mode of control output pins should be set to input mode before executing these instructions (as the output levels of control outputs vary asynchronously, the output level cannot be checked with the CHKL or CHKLA instruction).
  3. As port 2 is an input-only port, a comparison of the pin status with the output latch contents using the CHKL or CHKLA instruction will always show a match. Therefore, executing these instructions on port 2 is actually ineffective.

**5.11 CAUTIONS**

- (1) All port pins become high-impedance after  $\overline{\text{RESET}}$  signal input (internal pull-up resistors are disconnected from the pins).  
If there is a problem with pins becoming high-impedance during  $\overline{\text{RESET}}$  input, this should be handled with external circuitry.
- (2) Bit 7 of the pull-up resistor option register (PUO) that sets the internal pull-up resistor connection is fixed at 0, but if "1" is written to bit 7 of the PUO in the in-circuit emulator, "1" will be read.
- (3) Output latch contents are not initialized by  $\overline{\text{RESET}}$  input. When a port is used as an output port, the output latch must be initialized without fail before turning on the output buffer. If the output latch is not initialized before turning on the output buffer, unexpected data will be output to the output port.  
Similarly, for pins used as control pins, internal peripheral hardware initialization must be performed before performing the control pin specification.
- (4) As P22 to P26 are not pulled up immediately after a reset, an interrupt request flag may be set depending on the function of the dual-function pins (INTP1 to INTP5). Therefore, the interrupt request flags should be cleared after specifying pull-up in the initialization routine.
- (5) When P40 to P47 and P50 to P57 are used as the address/data bus and address bus respectively in the  $\mu\text{PD784026}$ , bits PU04 and PU05 of the pull-up resistor option register (PUO) must be set to "0" so that internal pull-up resistor connection is not performed.  
Also, since P40 to P47 and P50 to P57 are used as the address/data bus and address bus respectively in the  $\mu\text{PD784021}$ , bits PU04 and PU05 must be set to "0" so that internal pull-up resistor connection is not performed.
- (6) In the  $\mu\text{PD784021}$ , P60 to P63 are in the high-impedance state during  $\overline{\text{RESET}}$  input/output, but output low level after  $\overline{\text{RESET}}$  release. Therefore, external circuitry should be designed so that output of a low-level signal as the initial state is no problem.
- (7) A voltage outside the range  $\text{AV}_{\text{SS}}$  to  $\text{AV}_{\text{REF}}$  must not be applied to pins for which P70 to P77 are used as ANI0 to ANI7. See **14.5 CAUTIONS IN CHAPTER 14 A/D CONVERTER** for details.
- (8) A bit manipulation instruction manipulates one bit as the result, but accesses the port in 8-bit units. Therefore, if a bit manipulation instruction is used on a port with a mixture of input and output pins or port mode and control mode, the contents of the output latch of pins specified as inputs or pins specified as in control mode will be undefined (excluding bits manipulated with a SET1 or CLR1 instruction, etc.). Particular care is required when there are bits which are switched between input and output.  
Caution is also required when manipulating the port with other 8-bit operation instructions.
- (9) If each port is set to input mode, a comparison of the pin status with the output latch contents (or control output level) using the CHKL or CHKLA instruction will always show a match whether the individual pins of the port are port pins or control pins. Therefore, executing these instructions on a port set to input mode is actually ineffective.
- (10) If the output levels of a port in which control outputs and port outputs are mixed in a single port are checked with the CHKL or CHKLA instruction, the input/output mode of control output pins should be set to input mode before executing these instructions (as the output levels of control outputs vary asynchronously, the output level cannot be checked with the CHKL or CHKLA instruction).

- (11) As port 2 is an input-only port, a comparison of the pin status with the output latch contents using the CHKL or CHKLA instruction will always show a match. Therefore, executing these instructions on port 2 is actually ineffective.

## CHAPTER 6 REAL-TIME OUTPUT FUNCTION

### 6.1 CONFIGURATION AND FUNCTION

The real-time output function is implemented by hardware, including primarily port 0 and the port 0 buffer registers (P0H, P0L), shown in Figure 6-1.

The real-time output function refers to the transfer to the output latch by hardware of data prepared in the P0H and P0L beforehand, simultaneously with the generation of an interrupt from timer/counter 1 or external interrupt, and its output off-chip. The pins that output the data off-chip are called real-time output ports.

The following two kinds of real-time output data are handled:

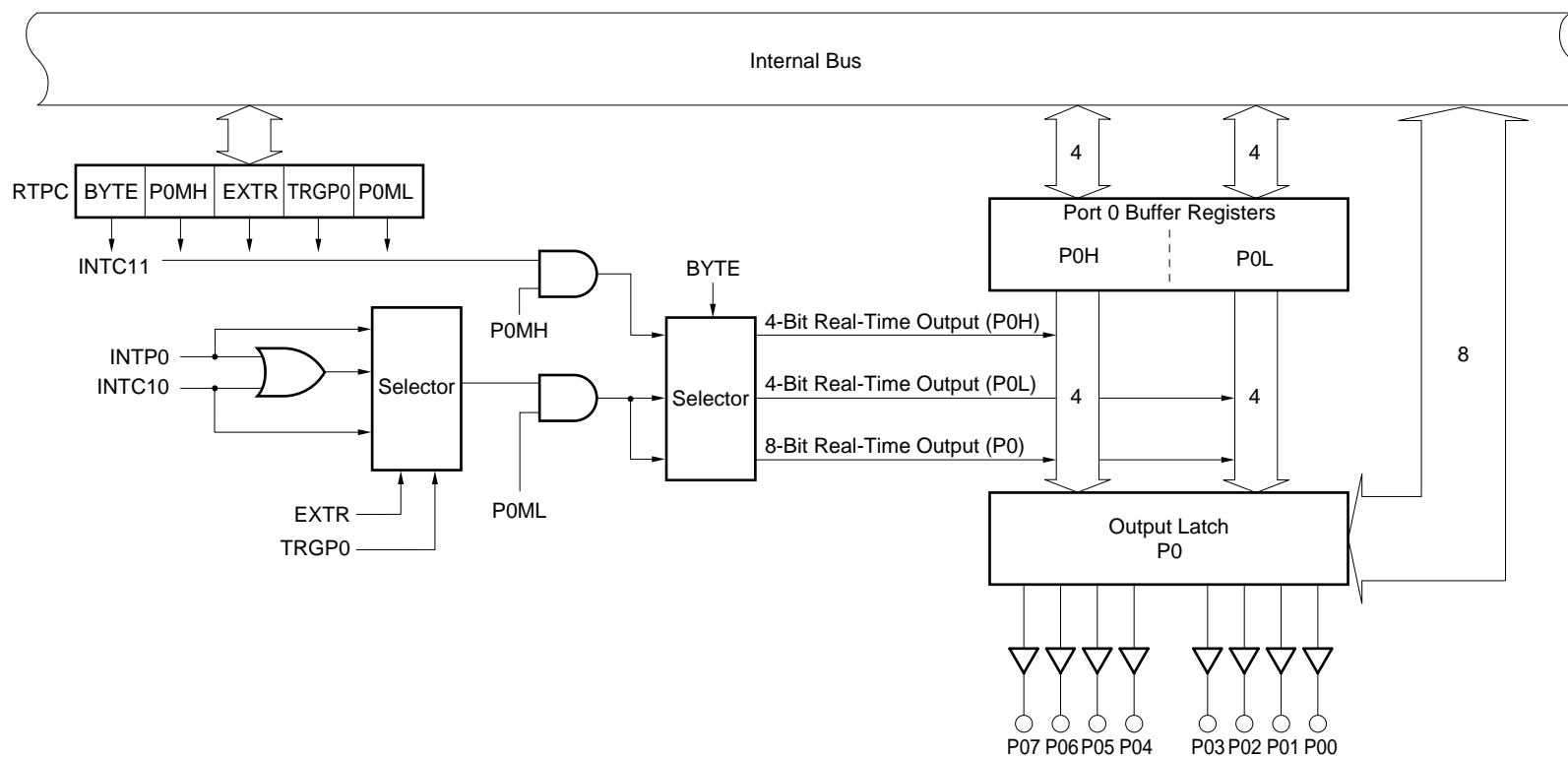
- 4 bits  $\times$  2 channels
- 8 bits  $\times$  1 channel

By combining the real-time output function with the macro service function described later, the functions of a pattern generator with programmable timing are implemented without software intermediation.

This is ideally suited to stepping motor control, for example.

Figure 6-1 shows the block diagram of the real-time output port.

Figure 6-1 Real-Time Output Port Block Diagram



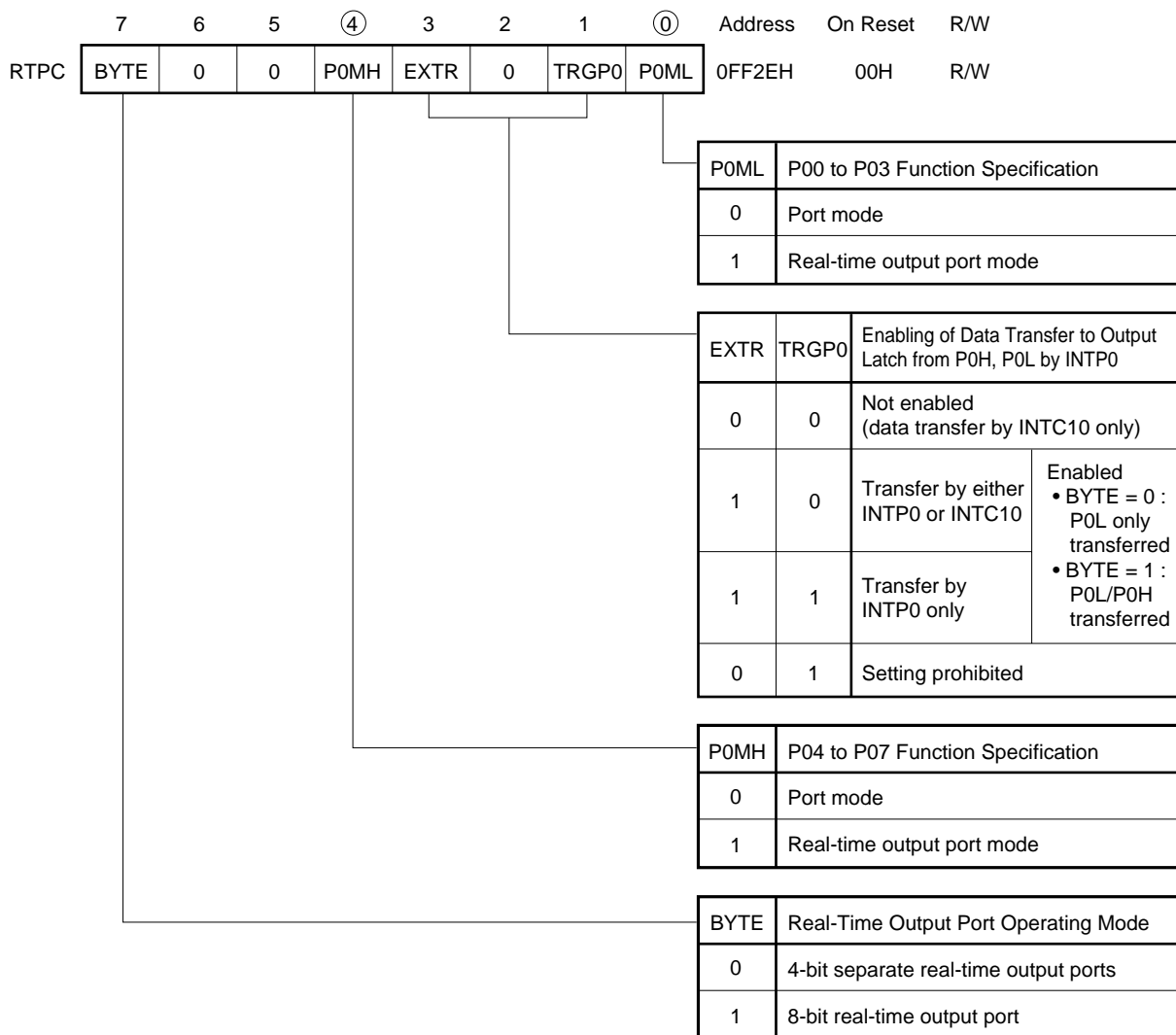
**6.2 REAL-TIME OUTPUT PORT CONTROL REGISTER (RTPC)**

The RTPC is an 8-bit register that specifies the function of port 0.

RTPC can be read or written to by an 8-bit manipulation instruction or bit-manipulation instruction. Figure 6-2 shows the format of RTPC.

$\overline{\text{RESET}}$  input clears the RTPC register to 00H.

**Figure 6-2 Real-Time Output Port Control Register (RTPC) Format**



**Caution** When P0ML and P0MH bits are set (1), the corresponding port output buffer is turned on and the port 0 output latch contents are output irrespective of the contents of the port 0 mode register (PM0). The output latch contents should therefore be initialized before making a real-time output port specification.

### 6.3 REAL-TIME OUTPUT PORT ACCESSES

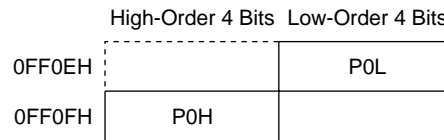
The port 0 buffer registers (P0H, P0L) are mapped onto mutually independent addresses in the SFR area as shown in Figure 6-3.

When the 4-bit  $\times$  2-channel real-time output function is specified, data can be set in the P0H, P0L independently of each other.

When the 8-bit  $\times$  1-channel real-time output function is specified, data can be set in P0H and P0L by writing 8-bit data to either one of the P0H or P0L.

Table 6-1 shows the operations when port 0, the P0H and P0L are manipulated.

**Figure 6-3 Port 0 Buffer Register (P0H, P0L) Configuration**



**Table 6-1 Operations When Port 0 and Port 0 Buffer Registers (P0H, P0L) are Manipulated**

Operating Mode	Register	Read Operation		Write Operation	
		High-Order 4 Bits	Low-Order 4 Bits	High-Order 4 Bits	Low-Order 4 Bits
8-bit port mode	P0	Output latch		Output latch	
	P0L	Buffer register*		—	Buffer register
	P0H	Buffer register*		Buffer register	—
8-bit real-time output port mode	P0	Output latch		—	
	P0L	Buffer register		Buffer register	
	P0H	Buffer register		Buffer register	
4-bit separate real-time output port mode	P0	Output latch		—	
	P0L	Buffer register*		—	Buffer register
	P0H	Buffer register*		Buffer register	—
P00 to P03: Ports P04 to P07: Real-time output port mode	P0	Output latch		—	Output latch
	P0L	Buffer register*		—	Buffer register
	P0H	Buffer register*		Buffer register	—
P00 to P03: Real-time output port mode P04 to P07: Ports	P0	Output latch		Output latch	—
	P0L	Buffer register*		—	Buffer register
	P0H	Buffer register*		Buffer register	—

\* The contents of P0H are read from the high-order 4 bits, and the contents of P0L from the low-order 4 bits.

**Remark** — : The output latch and port 0 buffer registers are not affected.



**<Examples of setting data in port 0 buffer registers>**

- 4-bit × 2-channel operation

MOV P0L, #05H ; Assigns 0101B to P0L

MOV P0H, #0C0H ; Assigns 1100B to P0H

- 8-bit × 1-channel operation

MOV P0L, #0C5H ; Assigns 0101B to P0L and 1100B to P0H

or

MOV P0H, #0C5H

The timing for transfer to the output latch can be determined by the following three sources:

- Interrupt from timer/counter 1 (INTC10 or INTC11)
- INTP0 external interrupt

## 6.4 OPERATION

When the port 0 function is specified as the real-time output port, the port 0 buffer register (P0H, P0L) contents are fetched into the output latch and output to the port 0 pins in synchronization with the generation of one of the trigger conditions shown in Table 6-2.

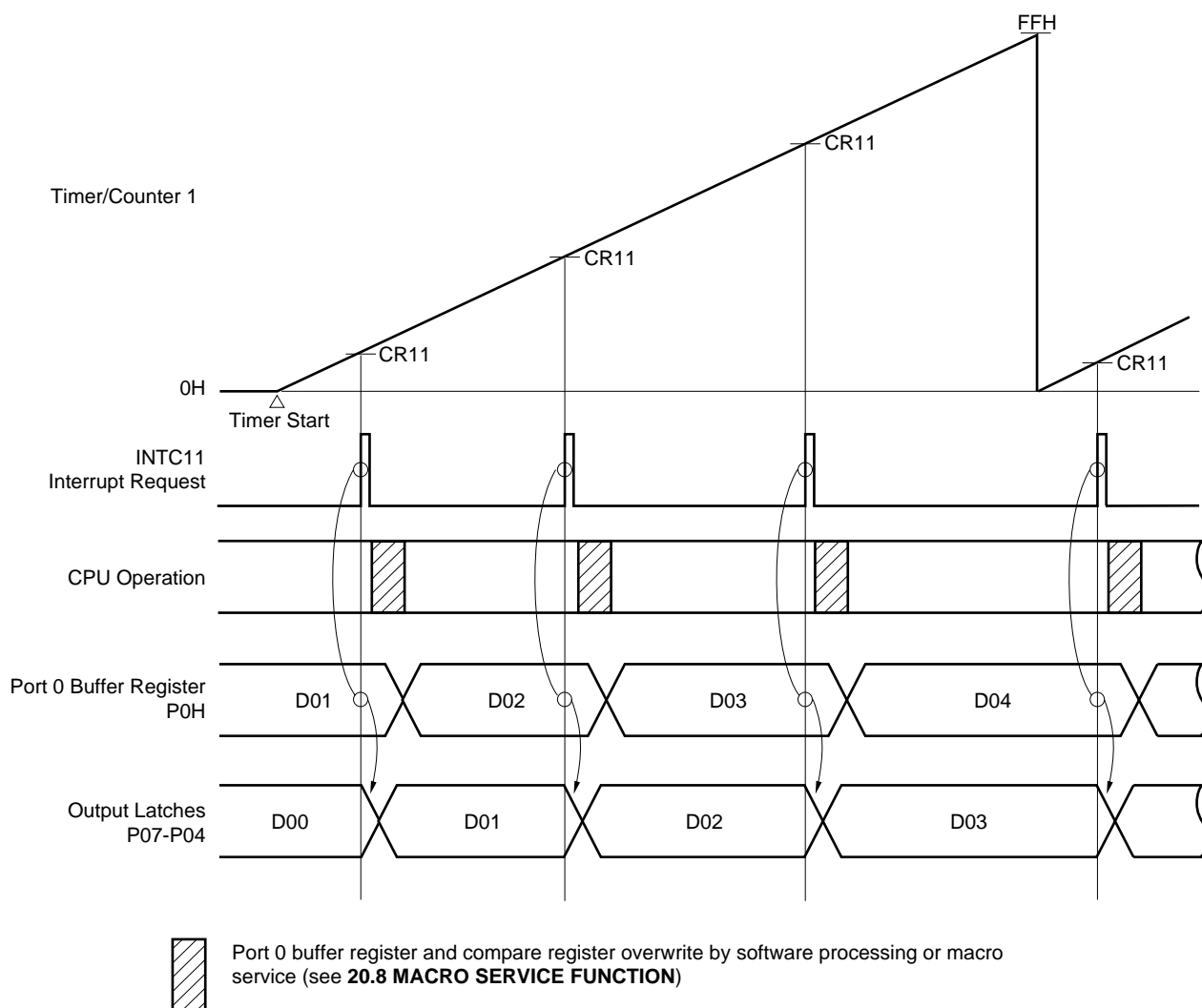
For example, the timer/counter 1 timer register 1 (TM1) and compare register (CR10, CR11) match signal (INTC10, INTC11) can be selected as the output trigger generation source. In this case, the port 0 pin output data can be changed to the P0H and P0L values using the value set in the CR10, CR11 beforehand as the timing interval. Combining this real-time output port function with the macro service function enables the port 0 output pin output data to be changed sequentially at any interval time (see **20.8 MACRO SERVICE FUNCTION**).

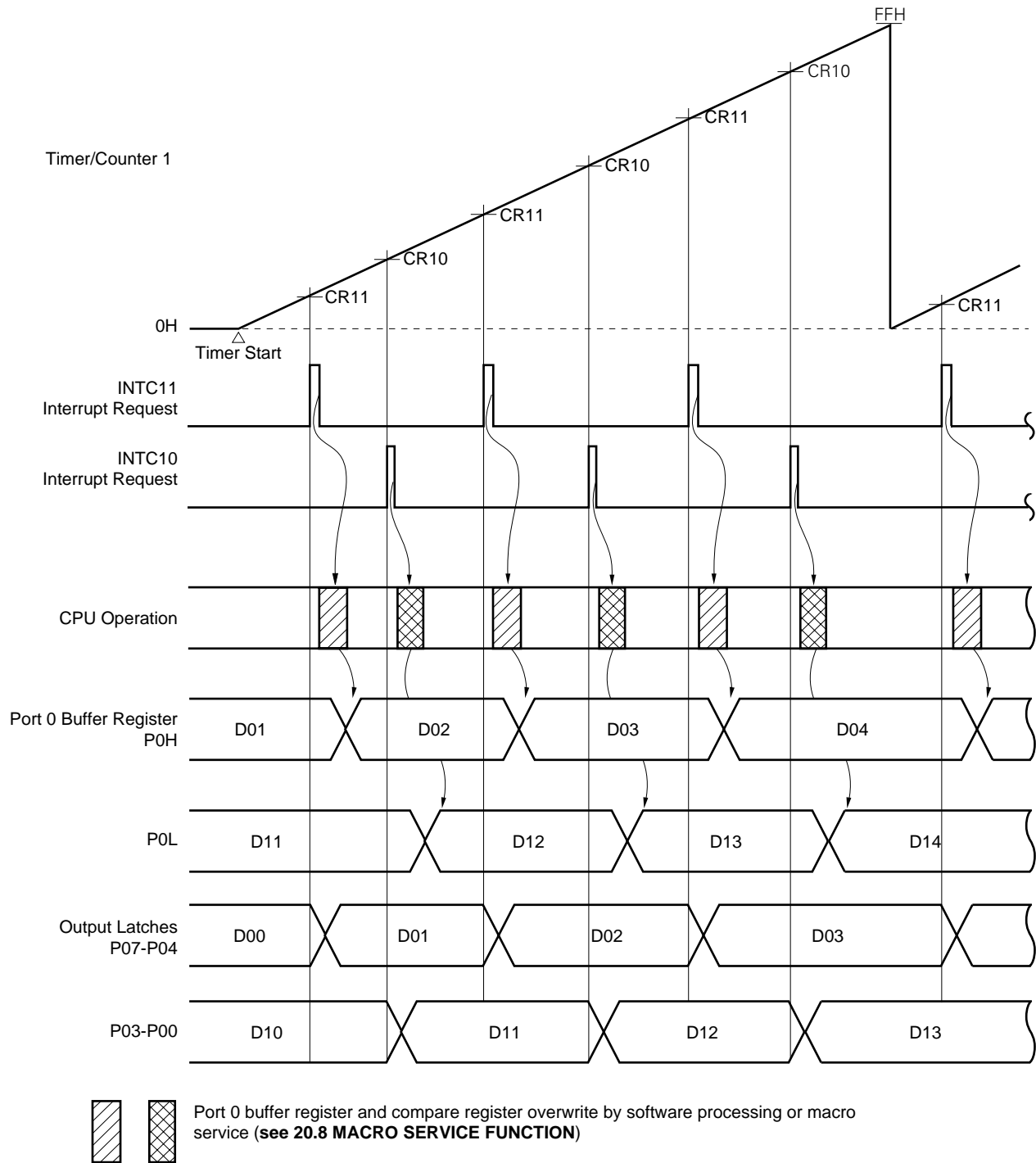
If the external interrupt pin (INTP0) is selected as the output trigger source, port 0 output can be obtained in synchronization with an external event.

**Table 6-2 Real-Time Output Port Output Triggers (When P0MH = P0ML = 1)**

RTPC			Output Mode	P0H	P0L
BYTE	EXTR	TRGP0			
0	0	0	4-bit real-time output	INTC11	INTC10
0	1	0		INTC11	INTC10 or INTP0
0	1	1		INTC11	INTP0
1	0	0	8-bit real-time output	INTC10	
1	1	0		INTC10 or INTP0	
1	1	1		INTP0	

Figure 6-4 Real-Time Output Port Operation Timing



**Figure 6-5 Real-Time Output Port Operation Timing (2 Channel Independent Control Example)**

## 6.5 EXAMPLE OF USE

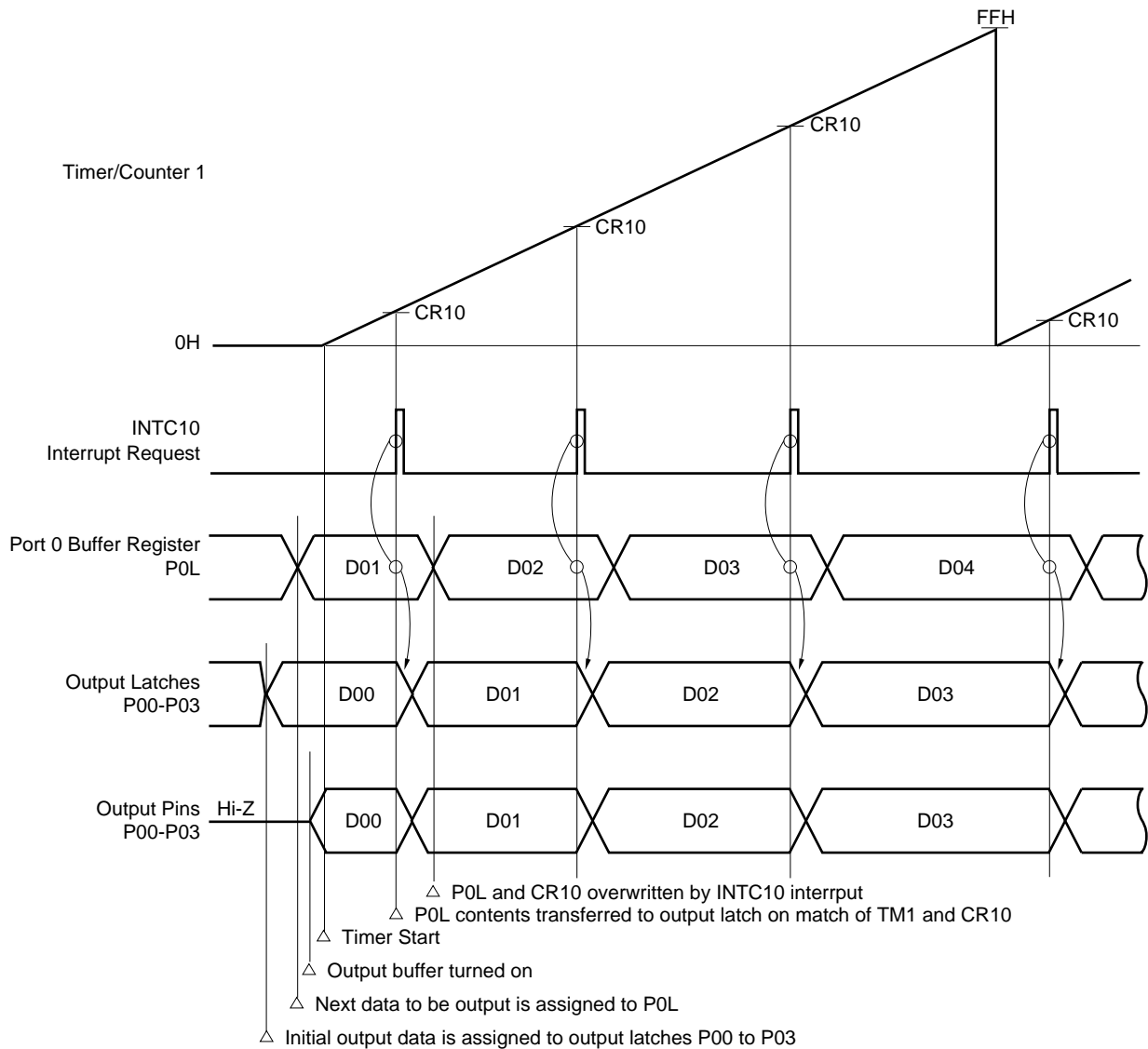
The case in which P00 to P03 are used as a 4-bit real-time output port is shown here.

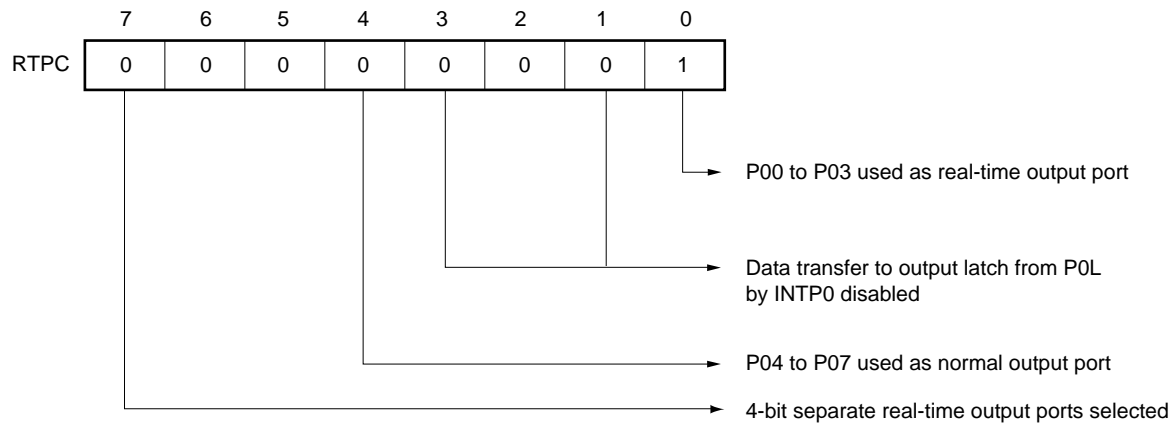
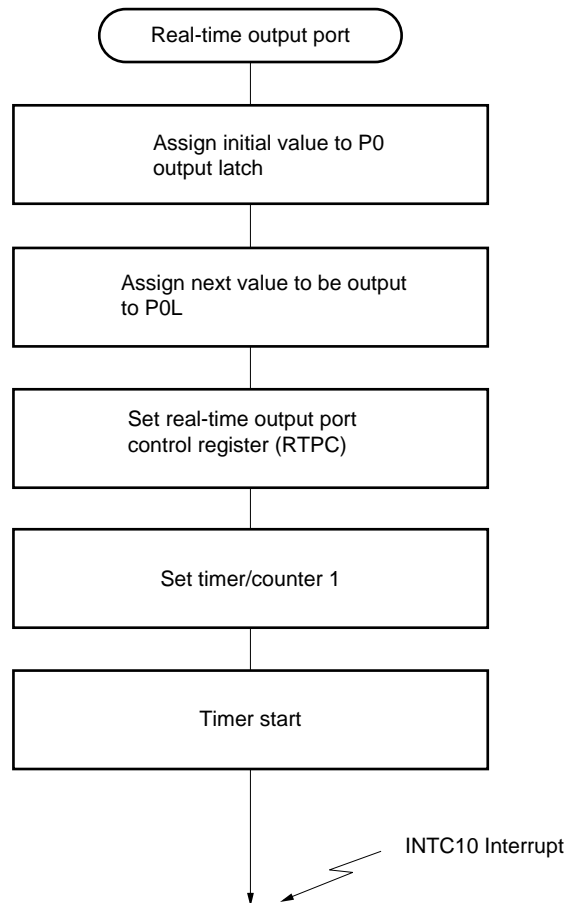
Each time the contents of timer/counter 1 timer register 1 (TM1) and compare register (CR10) match, the contents of port 0 buffer register (P0L) are output to P00 through P03. At this time, the next data to be output and the timing at which the output is to be changed next are set in the processing routine for the simultaneously generated interrupt (see **Figure 6-6**).

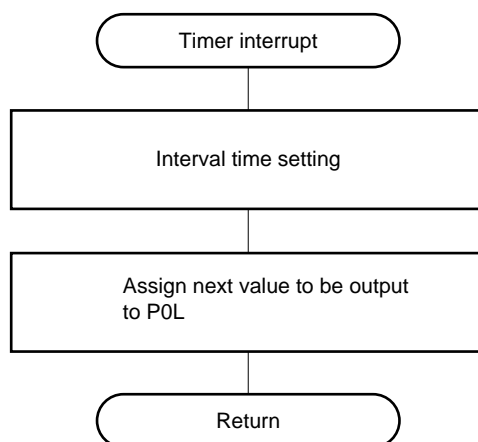
See **CHAPTER 9 TIMER/COUNTER 1** for the method of using timer/counter 1.

The control register settings are shown in Figure 6-7, the setting procedure in Figure 6-8, and the processing in the interrupt processing routine in Figure 6-9.

**Figure 6-6 Real-Time Output Port Operation Timing**



**Figure 6-7 Real-Time Output Function Control Register Settings****Figure 6-8 Real-Time Output Function Setting Procedure**

**Figure 6-9 Interrupt Request Processing when Real-Time Output Function is Used**

## 6.6 CAUTIONS

- (1) When P0ML and P0MH bits are set (1), the corresponding port output buffer is turned on and the port 0 output latch contents are output irrespective of the contents of the port 0 mode register (PM0). The output latch contents should therefore be initialized before making a real-time output port specification.
- (2) When the port is specified as a real-time output port, values cannot be directly written to the output latch by software. Therefore, the initial value of the output latch must be set in software before specifying use as a real-time output port. Also, if the need arises to forcibly set the output data to a fixed value while the port is being used as a real-time output port, you should change the port to a normal output port by manipulating the real-time output port control register (RTPC), then write the value to be output to the output latch.

[MEMO]



## CHAPTER 7 OUTLINE OF TIMER/COUNTER

The  $\mu$ PD784026 incorporates three timer/counter units and one time unit.

These timer/counter and timer units can be used as seven units of timer/counters because the  $\mu$ PD784026 supports seven interrupt requests.

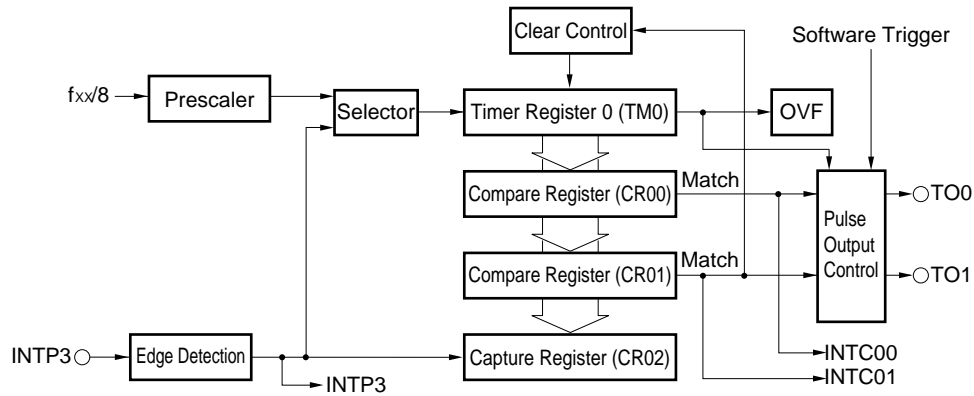
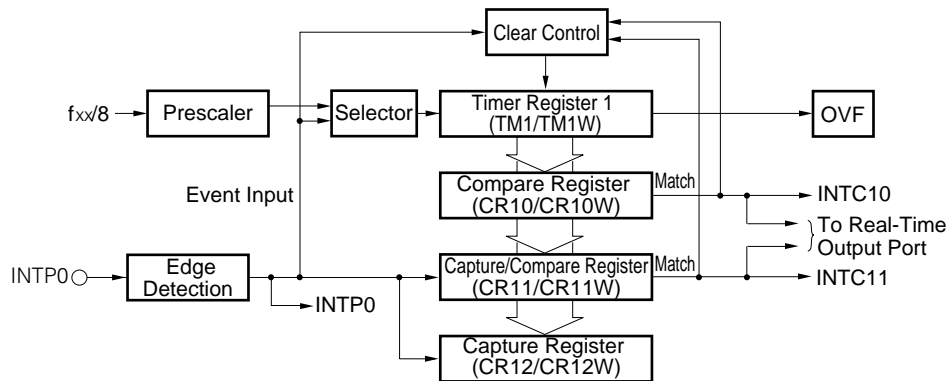
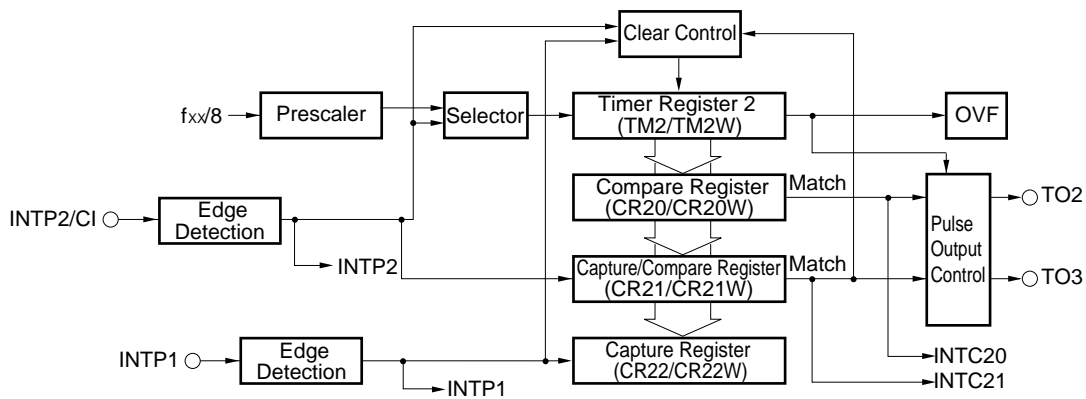
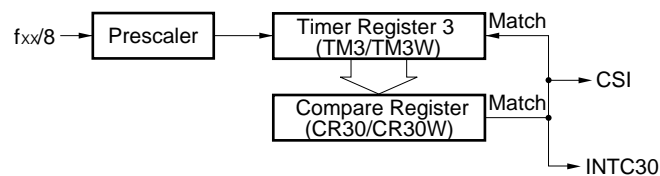
**Table 7-1 Operations of Timer/Counters**

Name		Timer/Counter 0	Timer/Counter 1	Timer/Counter 2	Timer 3
Item					
Count width	8 bit	—	○	○	○
	16 bit	○	○	○	○
Operation mode	Interval timer	2 ch	2 ch	2 ch	1 ch
	External event counter	○	○	○	—
	One-shot timer	—	—	○	—
Function	Timer output	2 ch	—	2 ch	—
	Toggle output	○	—	○	—
	PWM/PPG output	○	—	○	—
	One-shot pulse output*	○	—	—	—
	Real-time output	—	○	—	—
	Pulse width measurement	1 input	1 input	2 inputs	—
	Number of interrupt requests	2	2	2	1

\* In the one-shot pulse output function, the pulse output level activated by software and inactivated by hardware (an interrupt request signal).

This function is different in nature from the one-shot timer function of timer/counter 2.

Figure 7-1 Timer/Counter Block Diagram

**Timer/Counter 0****Timer/Counter 1****Timer/Counter 2****Timer 3**

**Remark** OVF: Overflow flag

## CHAPTER 8 TIMER/COUNTER 0

### 8.1 FUNCTIONS

Timer/counter 0 is a 16-bit timer/counter.

In addition to its basic functions of interval timer, programmable square-wave output, pulse width measurement and event counter, timer/counter 0 can be used for the following functions.

- PWM output
- Cycle measurement
- Software triggered one-shot pulse output

#### (1) Interval timer

Generates internal interrupts at preset intervals.

**Table 8-1 Timer/Counter 0 Interval Time**

Minimum Interval Time	Maximum Interval Time	Resolution
$8/f_{xx}$ (0.32 $\mu$ s)	$2^{16} \times 8/f_{xx}$ (21.0 ms)	$8/f_{xx}$ (0.32 $\mu$ s)
$16/f_{xx}$ (0.64 $\mu$ s)	$2^{16} \times 16/f_{xx}$ (41.9 ms)	$16/f_{xx}$ (0.64 $\mu$ s)
$32/f_{xx}$ (1.28 $\mu$ s)	$2^{16} \times 32/f_{xx}$ (83.9 ms)	$32/f_{xx}$ (1.28 $\mu$ s)
$64/f_{xx}$ (2.56 $\mu$ s)	$2^{16} \times 64/f_{xx}$ (168 ms)	$64/f_{xx}$ (2.56 $\mu$ s)
$128/f_{xx}$ (5.12 $\mu$ s)	$2^{16} \times 128/f_{xx}$ (336 ms)	$128/f_{xx}$ (5.12 $\mu$ s)
$256/f_{xx}$ (10.2 $\mu$ s)	$2^{16} \times 256/f_{xx}$ (671 ms)	$256/f_{xx}$ (10.2 $\mu$ s)
$512/f_{xx}$ (20.5 $\mu$ s)	$2^{16} \times 512/f_{xx}$ (1.34 s)	$512/f_{xx}$ (20.5 $\mu$ s)
$1024/f_{xx}$ (41.0 $\mu$ s)	$2^{16} \times 1024/f_{xx}$ (2.68 s)	$1024/f_{xx}$ (41.0 $\mu$ s)
$2048/f_{xx}$ (81.9 $\mu$ s)	$2^{16} \times 2048/f_{xx}$ (5.37 s)	$2048/f_{xx}$ (81.9 $\mu$ s)

( ): When  $f_{xx} = 25$  MHz

**(2) Programmable square-wave output**

Outputs square waves independently to the timer output pins (TO0, TO1).

**Table 8-2 Timer/Counter 0 Programmable Square-Wave Output Setting Range**

Minimum Pulse Width	Maximum Pulse Width
$8/f_{xx}$ (0.32 $\mu$ s)	$2^{16} \times 8/f_{xx}$ (21.0 ms)
$16/f_{xx}$ (0.64 $\mu$ s)	$2^{16} \times 16/f_{xx}$ (41.9 ms)
$32/f_{xx}$ (1.28 $\mu$ s)	$2^{16} \times 32/f_{xx}$ (83.9 ms)
$64/f_{xx}$ (2.56 $\mu$ s)	$2^{16} \times 64/f_{xx}$ (168 ms)
$128/f_{xx}$ (5.12 $\mu$ s)	$2^{16} \times 128/f_{xx}$ (336 ms)
$256/f_{xx}$ (10.2 $\mu$ s)	$2^{16} \times 256/f_{xx}$ (671 ms)
$512/f_{xx}$ (20.5 $\mu$ s)	$2^{16} \times 512/f_{xx}$ (1.34 s)
$1024/f_{xx}$ (41.0 $\mu$ s)	$2^{16} \times 1024/f_{xx}$ (2.68 s)
$2048/f_{xx}$ (81.9 $\mu$ s)	$2^{16} \times 2048/f_{xx}$ (5.37 s)

( ): When  $f_{xx} = 25$  MHz

**(3) Pulse width measurement**

Detects the pulse width of the signal input to the external interrupt request input pin (INTP3).

**Table 8-3 Timer/Counter 0 Pulse Width Measurement Range**

Measurable Pulse Width*		Resolution
$8/f_{xx}$ (0.32 $\mu$ s)	– $2^{16} \times 8/f_{xx}$ (21.0 ms)	$8/f_{xx}$ (0.32 $\mu$ s)
$16/f_{xx}$ (0.64 $\mu$ s)	– $2^{16} \times 16/f_{xx}$ (41.9 ms)	$16/f_{xx}$ (0.64 $\mu$ s)
$32/f_{xx}$ (1.28 $\mu$ s)	– $2^{16} \times 32/f_{xx}$ (83.9 ms)	$32/f_{xx}$ (1.28 $\mu$ s)
$64/f_{xx}$ (2.56 $\mu$ s)	– $2^{16} \times 64/f_{xx}$ (168 ms)	$64/f_{xx}$ (2.56 $\mu$ s)
$128/f_{xx}$ (5.12 $\mu$ s)	– $2^{16} \times 128/f_{xx}$ (336 ms)	$128/f_{xx}$ (5.12 $\mu$ s)
$256/f_{xx}$ (10.2 $\mu$ s)	– $2^{16} \times 256/f_{xx}$ (671 ms)	$256/f_{xx}$ (10.2 $\mu$ s)
$512/f_{xx}$ (20.5 $\mu$ s)	– $2^{16} \times 512/f_{xx}$ (1.34 s)	$512/f_{xx}$ (20.5 $\mu$ s)
$1024/f_{xx}$ (41.0 $\mu$ s)	– $2^{16} \times 1024/f_{xx}$ (2.68 s)	$1024/f_{xx}$ (41.0 $\mu$ s)
$2048/f_{xx}$ (81.9 $\mu$ s)	– $2^{16} \times 2048/f_{xx}$ (5.37 s)	$2048/f_{xx}$ (81.9 $\mu$ s)

( ): When  $f_{xx} = 25$  MHz

- \* The minimum pulse width that can be measured differs depending on the selected value of  $f_{CLK}$ .  
The minimum pulse width that can be measured is the value of  $4/f_{CLK}$  or the value in the above table, whichever is greater.

**(4) Software triggered one-shot pulse output**

This is a one-shot pulse output function in which the pulse output level is activated by software and inactivated by hardware (an interrupt request signal). Control can be performed for the timer output pins (TO0, TO1) independently.

**Caution** The software triggered one-shot pulse output function is different in nature from the one-shot timer function of timer/counter 2.

**(5) External event counter**

Counts the clock pulses input from the external interrupt request input pin (INTP3).

The clocks that can be input to timer/counter 0 are shown in Table 8-4.

★

**Table 8-4 Timer/Counter 0 Pulse Width Measurement Time**

	When Counting One Edge	When Counting Both Edges
Maximum frequency	$f_{CLK}/8$ (1.56 MHz)	$f_{CLK}/8$ (1.56 MHz)
Minimum pulse width (High and low levels)	$4/f_{CLK}$ (0.32 $\mu$ s)	$4/f_{CLK}$ (0.32 $\mu$ s)

( ): When  $f_{CLK} = 12.5$  MHz

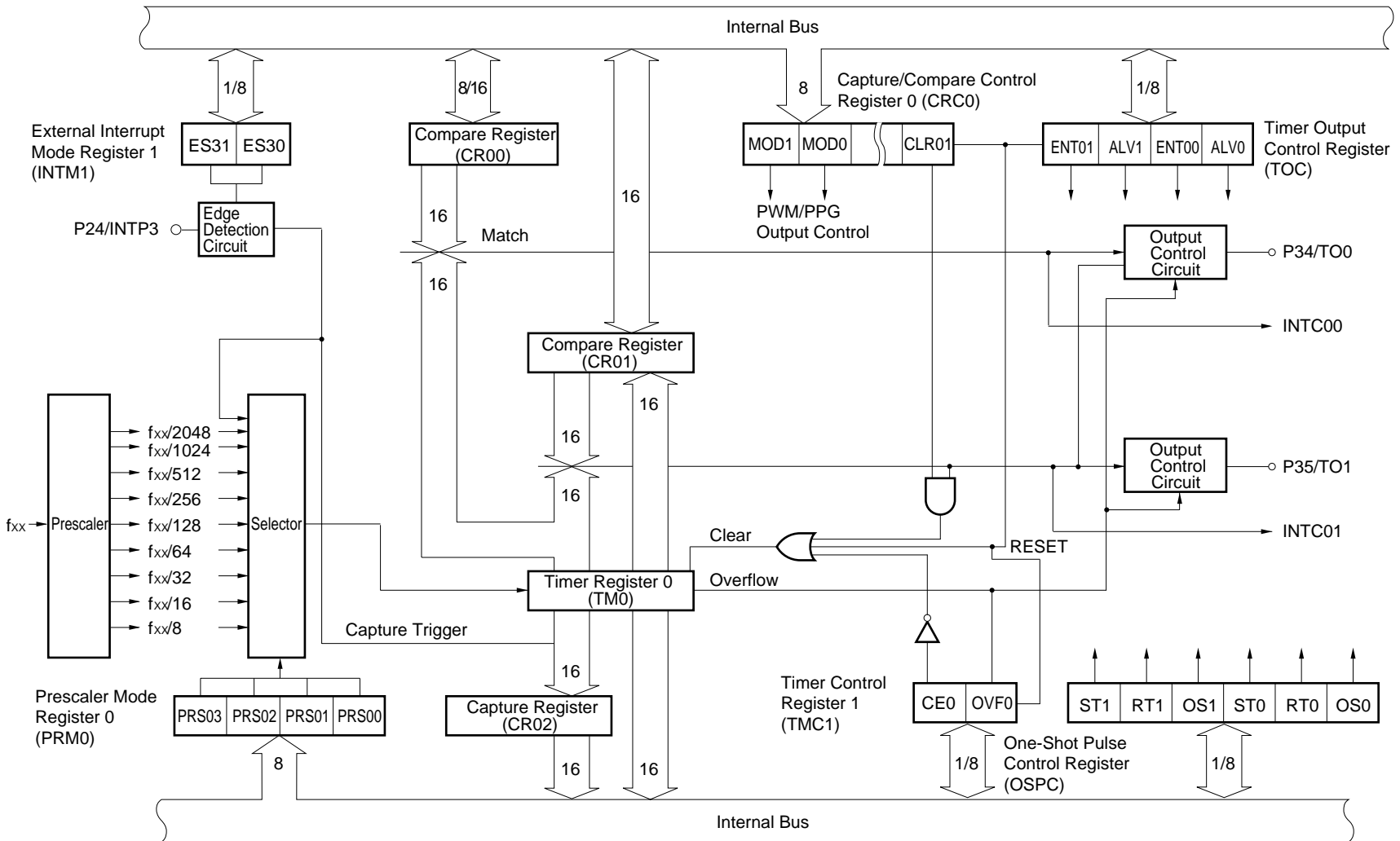
**8.2 CONFIGURATION**

Timer/counter 0 consists of the following registers:

- Timer register (TM0)  $\times 1$
- Compare register (CR00, CR01)  $\times 2$
- Capture register (CR02)  $\times 1$

The block diagram of timer/counter 0 is shown in Figure 8-1.

**Figure 8-1 Timer/Counter 0 Block Diagram**



**(1) Timer register 0 (TM0)**

TM0 is a timer register that counts up using the count clock specified by the low-order 4 bits of prescaler mode register 0 (PRM0).

The count operation is stopped or enabled by means of timer control register 0 (TMC0).

TM0 can be read only with a 16-bit manipulation instruction. When  $\overline{\text{RESET}}$  is input, TM0 is cleared to 0000H and the count is stopped.

- ★ **Caution** Although the timer register can be read, if counting up the timer and reading the timer register conflict, the read value of the timer register may be illegal. Therefore, do not read the timer register if at all possible.

If it is necessary to read the timer register, read it two or more times in a row in software, and assume the value of the timer register is correct only when all the read values match.

**(2) Compare registers (CR00/CR01)**

CR00 and CR01 are 16-bit registers that hold the values that determine the interval timer frequency.

If the CR00/CR01 contents match the contents of TM0, an interrupt request (INTC00/INTC01) and timer output control signal are generated. Also, the count value can be cleared by a content match (CR01).

CR00 and CR01 can be read or written with a 16-bit manipulation instruction. The contents of these registers are undefined after  $\overline{\text{RESET}}$  input.

**(3) Capture register (CR02)**

CR02 is a 16-bit register that captures the contents of TM0.

The capture operation is synchronized with the input of a valid edge (capture trigger) on the external interrupt request input pin (INTP3). The contents of the CR02 are retained until the next capture trigger is generated.

CR02 can be read only with a 16-bit manipulation instruction. The contents of this register are undefined after  $\overline{\text{RESET}}$  input.

**(4) Edge detection circuit**

The edge detection circuit detects an external input valid edge.

When the valid edge set by external interrupt mode register 1 (INTM1) is detected in the INTP3 pin input, the external interrupt request (INTP3), a capture trigger, and a external event count clock are generated (see **Figure 19-2** for details of the INTM1).



**(5) Output control circuit**

It is possible to invert the timer output when the compare register (CR00, CR01) register contents and the contents of the timer register (TM0) match. A square wave can be output from the timer output pins (TO0/TO1) in accordance with the setting of the low-order 4 bits of the timer output control register (TOC). At this time, PWM output or PPG output can be performed according to the specification of capture/compare control register 0 (CRC0).

In addition, one-shot pulse output can also be performed by means of a software trigger.

Timer output can be disabled/enabled by means of the TOC. When timer output is disabled, a fixed level is output to the TO0 and TO1 pins (the output level is set by the TOC).

**(6) Prescaler**

The prescaler generates the count clock from the internal system clock. The clock generated by this prescaler is selected by the selector, and is used as the count clock by the timer register 0 (TM0) to perform count operations.

**(7) Selector**

The selector selects a signal resulting from dividing the internal clock or the edge detected by the edge detection circuit as the count clock of timer register 0 (TM0).

8.3 TIMER/COUNTER 0 CONTROL REGISTERS

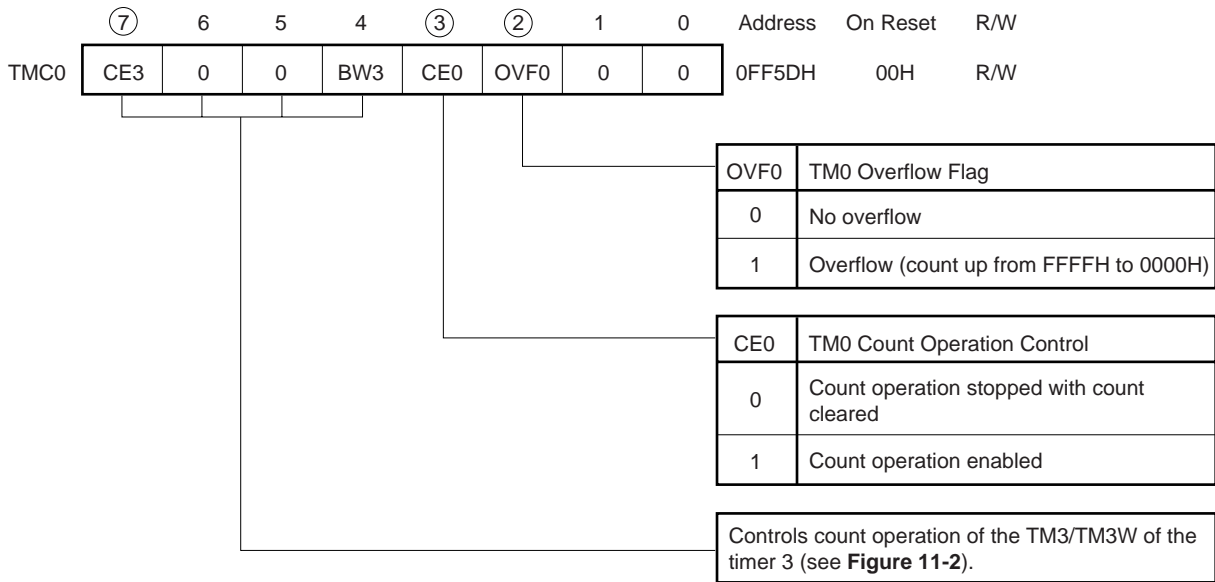
(1) Timer control register 0 (TMC0)

The timer/counter 0 TM0 count operation is controlled by the low-order 4 bits in the TMC0 (the high-order 4 bits control the count operation of the TM3/TM3W of the timer 3).

TMC0 can be read or written to with an 8-bit manipulation instruction or bit manipulation instruction. The format of the TMC0 is shown in Figure 8-2.

RESET input clears TMC0 to 00H.

Figure 8-2 Timer Control Register 0 (TMC0) Format



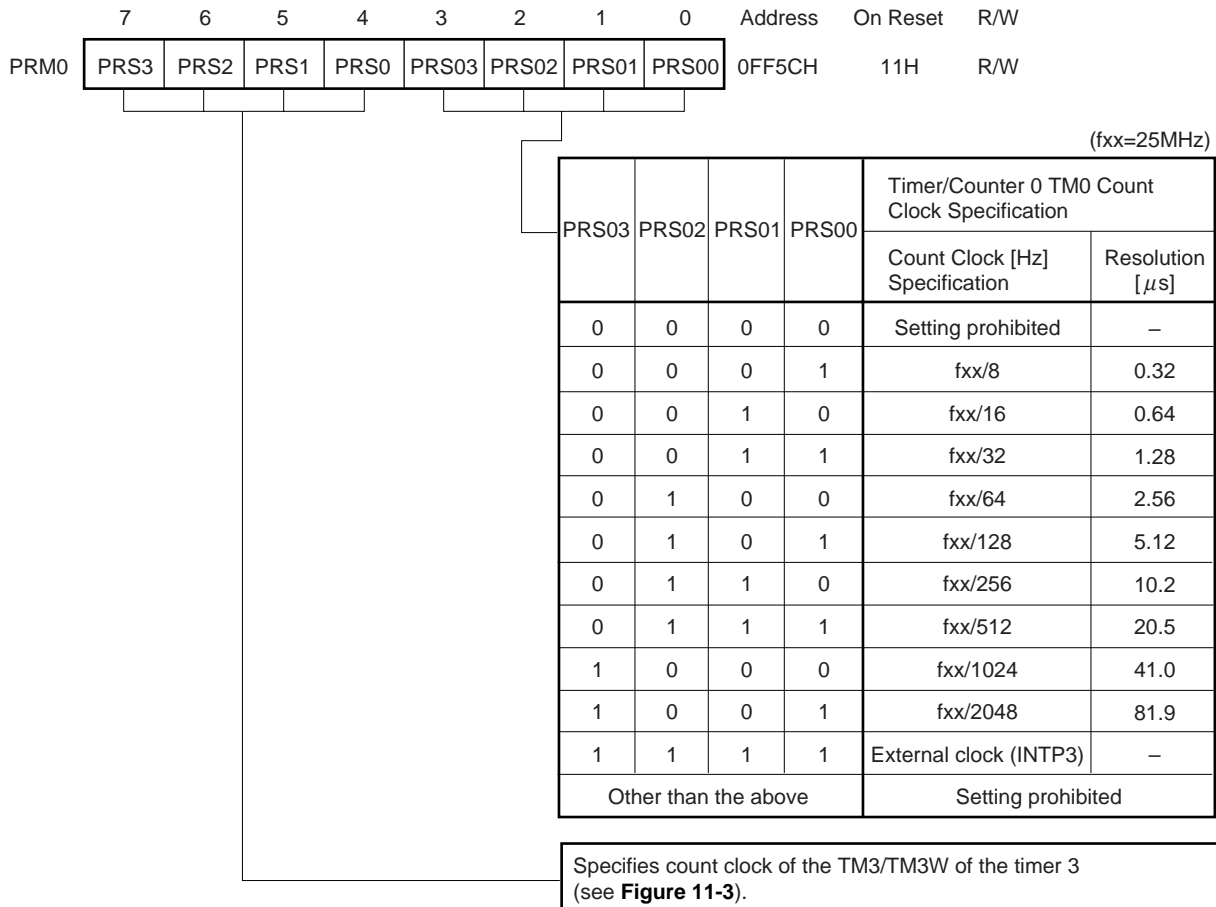
**Remark** The OVF0 bit is reset by software only.

**(2) Prescaler mode register 0 (PRM0)**

The count clock of the timer/counter 0, TM0, is specified by the low-order 4 bits of the PRM0 (the high-order 4 bits specify the count clock of the timer 3, TM3/TM3W).

PRM0 can be read/written with an 8-bit manipulation instruction. The format of the PRM0 is shown in Figure 8-3.

$\overline{\text{RESET}}$  input sets PRM0 to 11H.

**Figure 8-3 Prescaler Mode Register 0 (PRM0) Format**

**Remark** fxx: X1 input frequency or oscillation frequency

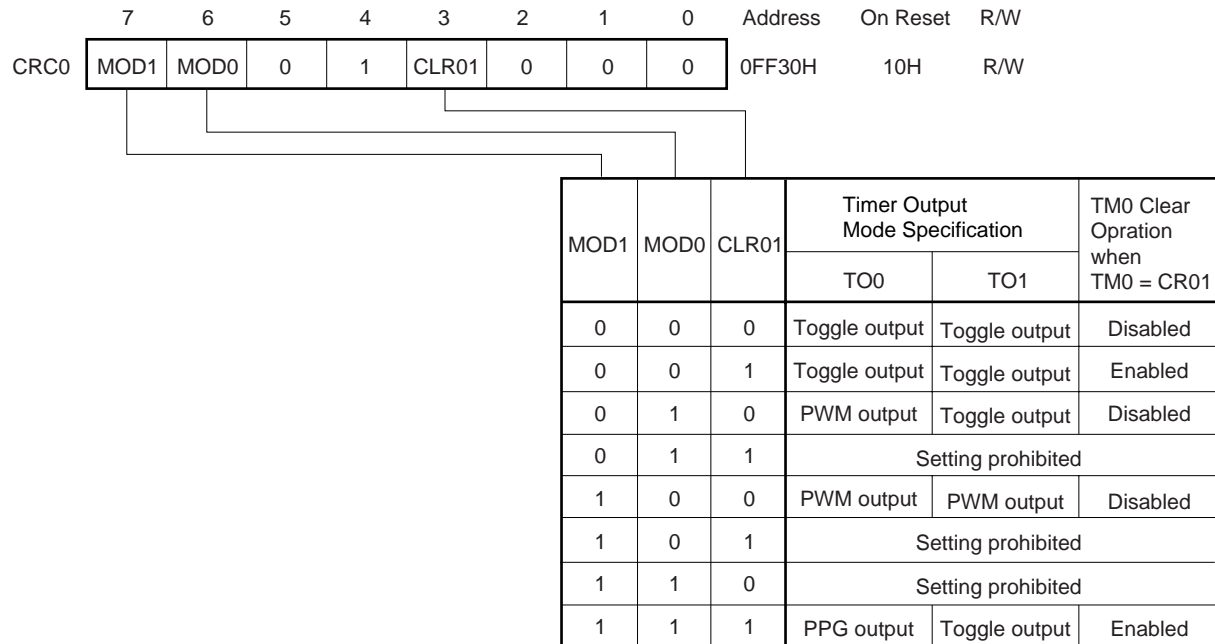
**(3) Capture/compare control register 0 (CRC0)**

The CRC0 specifies the enabling conditions for the TM0 clear operation by a match signal between the contents of the compare register (CR01) and the timer register 0 (TM0) counter value, and the timer outputs (TO0/TO1) mode.

CRC0 can be read/written with an 8-bit manipulation instruction. The format of the CRC0 is shown in Figure 8-4.

$\overline{\text{RESET}}$  input sets CRC0 to 10H.

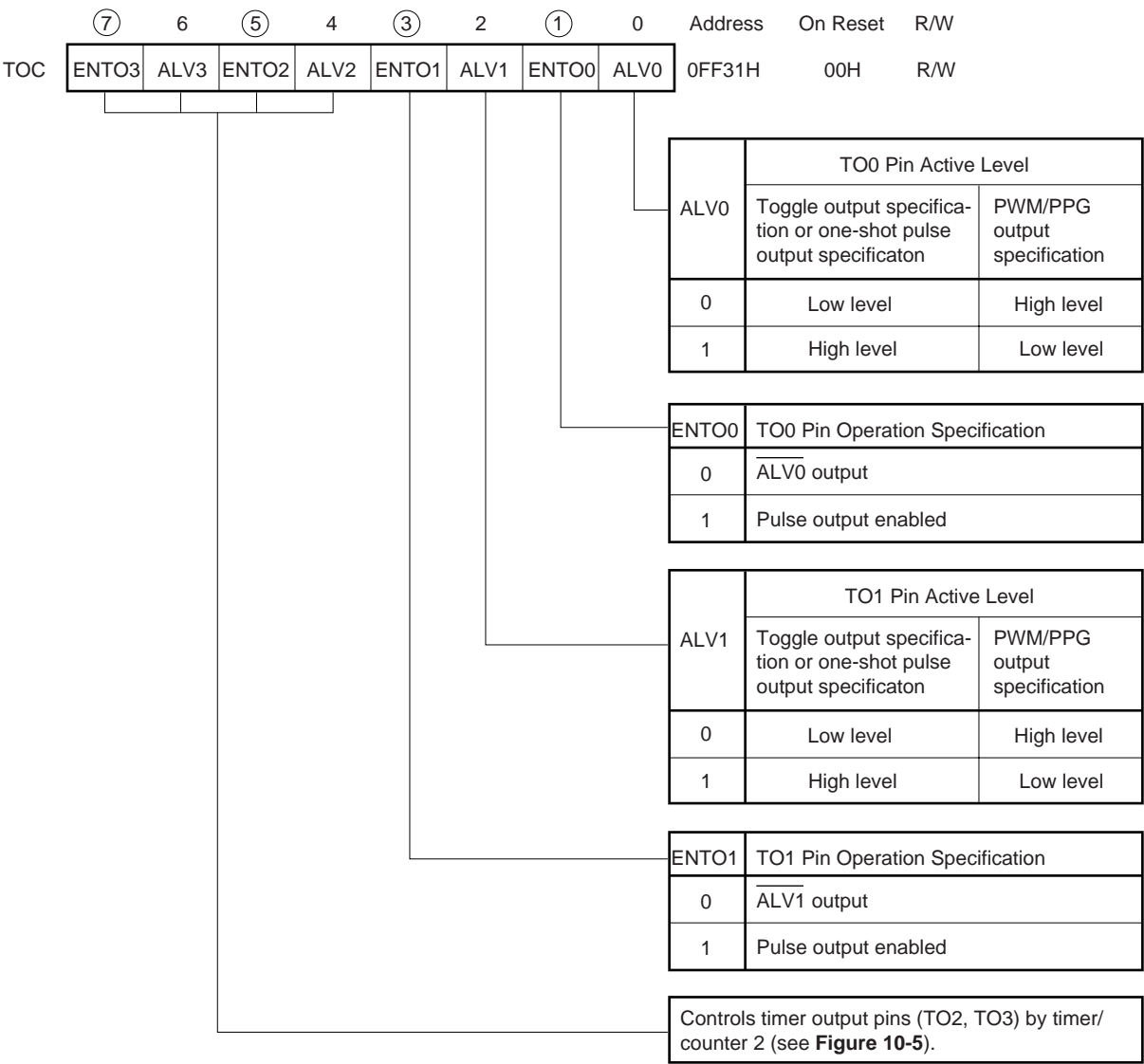
**Figure 8-4 Capture/Compare Control Register 0 (CRC0) Format**



(4) Timer output control register (TOC)

TOC is an 8-bit register that controls the active level of timer output and output enabling/disabling. The operation of the timer output pins (TO0 and TO1) by the timer/counter 0 is controlled by the low-order 4 bits (the high-order 4 bits control the operation of the timer output pins (TO2 and TO3 by the timer/counter 2). TOC can be written to or read with an 8-bit manipulation instruction or bit manipulation instruction. The format of the TOC is shown in Figure 8-5. RESET input clears TOC to 00H.

Figure 8-5 Timer Output Control Register (TOC) Format



**(5) One-shot pulse output control register (OSPC)**

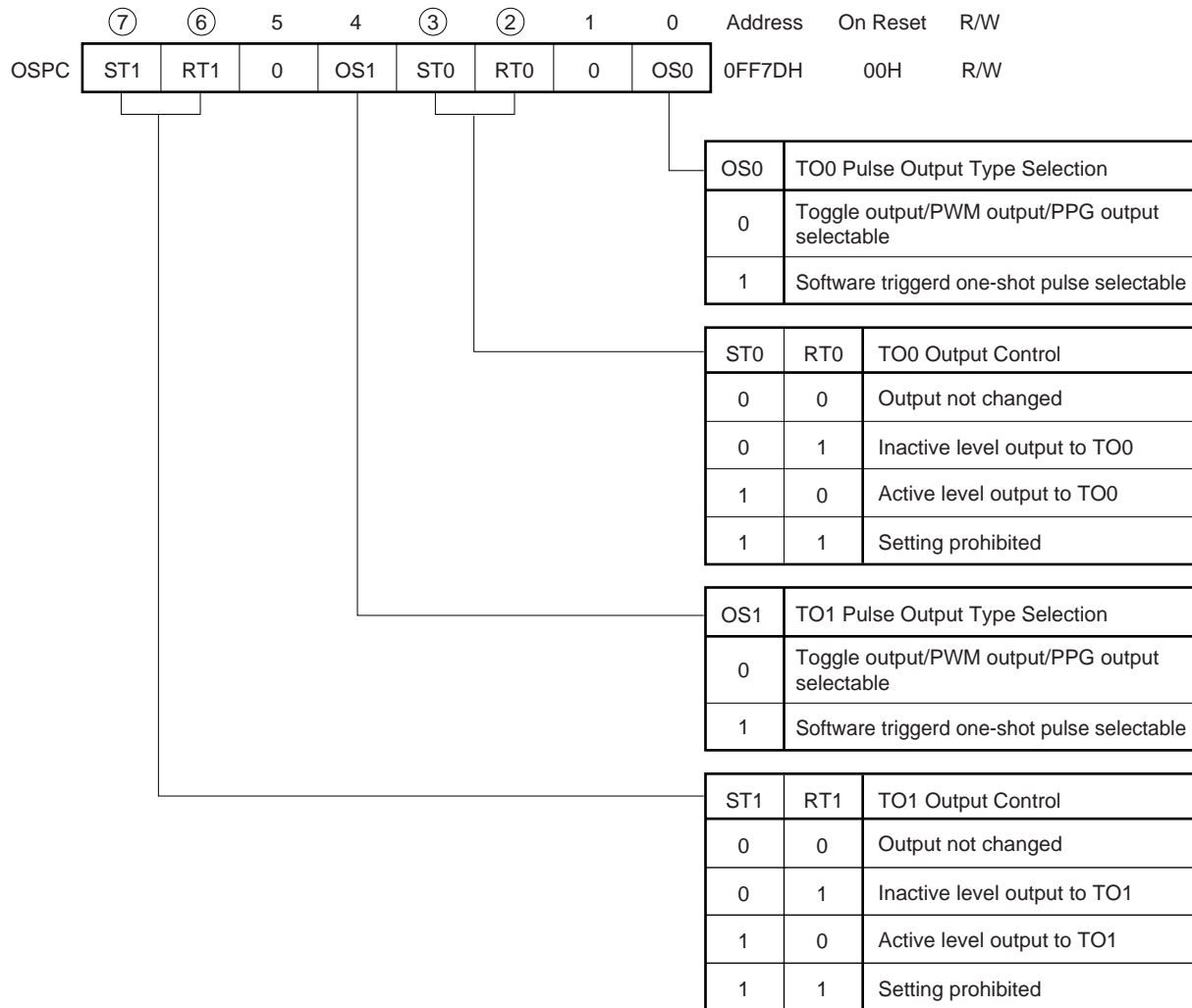
The OSPC is an 8-bit register that specifies enabling/disabling of one-shot pulse output by a software trigger and the output level, etc.

OSPC can be read or written to with an 8-bit manipulation instruction or bit manipulation instruction.

The format of the OSPC is shown in Figure 8-6.

RESET input clears OSPC to 00H.

**Figure 8-6 One-Shot Pulse Output Control Register (OSPC) Format**



- Remarks**
1. The RT0, ST0, RT1 and ST1 bits are write-only, and show a value of "0" if read.
  2. Pin pulse output disabling/enabling and active level setting are performed by means of the timer output control register (TOC).

## 8.4 16-BIT TIMER REGISTER 0 (TM0) OPERATION

### 8.4.1 Basic Operation

In the timer/counter 0 count operation, an up-count is performed using the count clock specified by the low-order 4 bits of prescaler mode register 0 (PRM0).

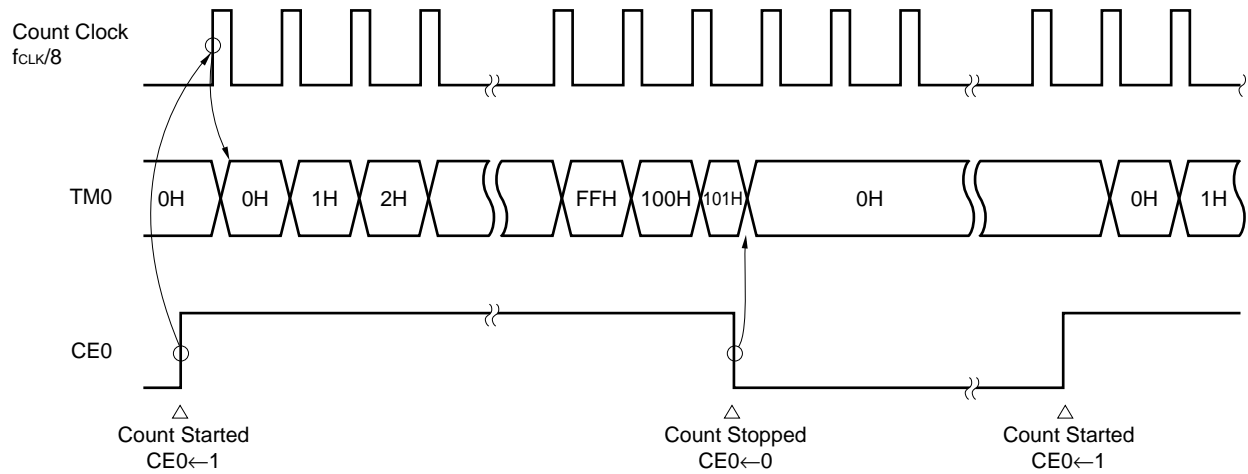
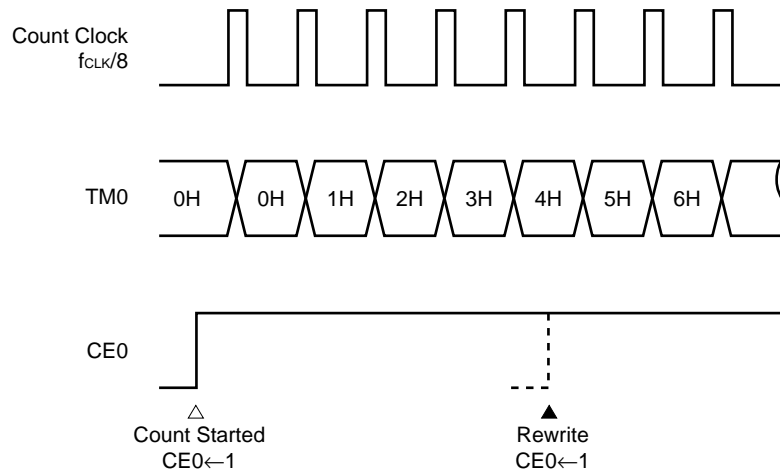
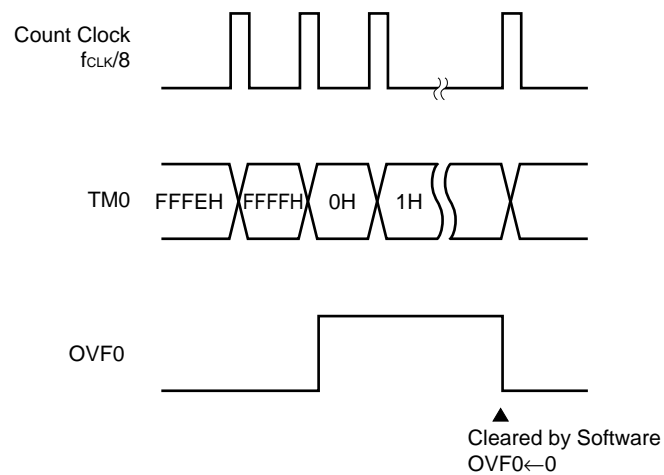
Count operation enabling/disabling is controlled by bit 3 (CE0) of timer control register 0 (TMC0). When the CE0 bit is set (1) by software, the contents of TM0 are cleared to 0000H on the first count clock, and then the up-count operation is performed.

When the CE0 bit is cleared (0), TM0 becomes 0000H immediately, and capture operations and match signal generation are stopped.

If the CE0 bit is set (1) again when it is already set (1), TM0 continues the count operation without being cleared.

If the count clock is input when TM0 is FFFFH, TM0 becomes 0000H. In this case, OVF0 bit is set (1) and an overflow signal is sent to the output control circuit. OVF0 bit is cleared by software only. The count operation is continued.

When  $\overline{\text{RESET}}$  is input, TM0 is cleared to 0000H, and the count operation is stopped.

**Figure 8-7 Basic Operation of Timer Register 0 (TM0)****(a) Count started → count stopped → count started****(b) When "1" is written to the CE0 bit again after the count starts****(c) Operation when TM0 = FFFFH**

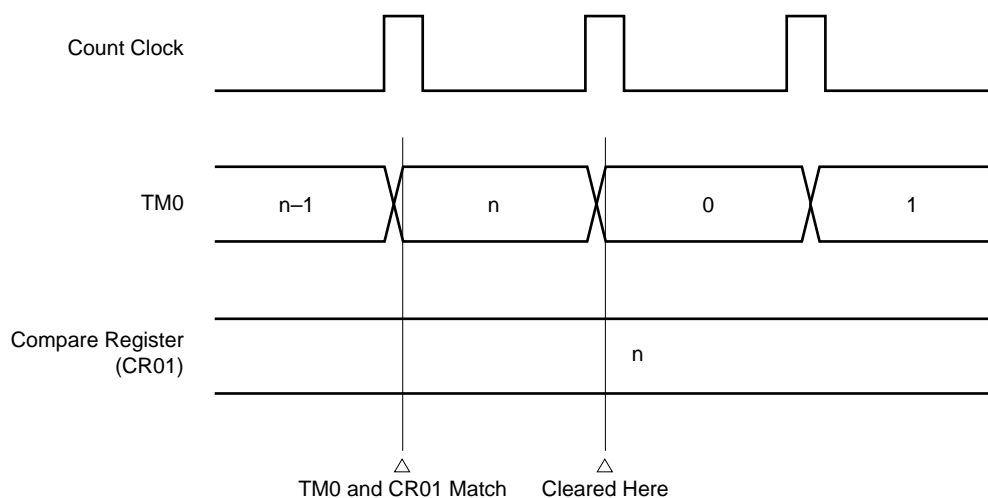


### 8.4.2 Clear Operation

#### (1) Clear operation after a match with the compare register

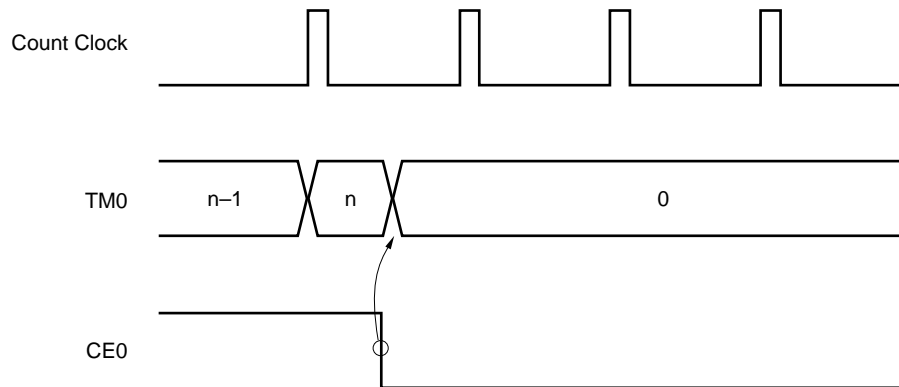
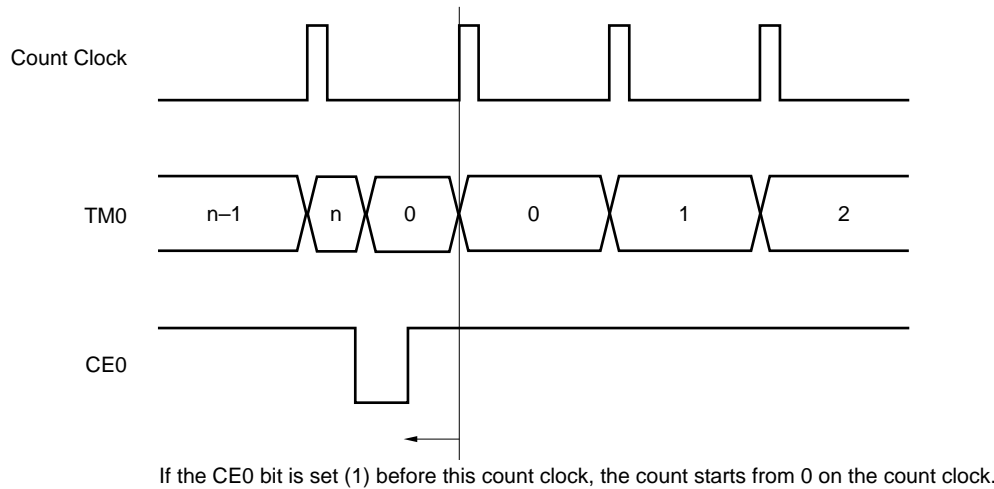
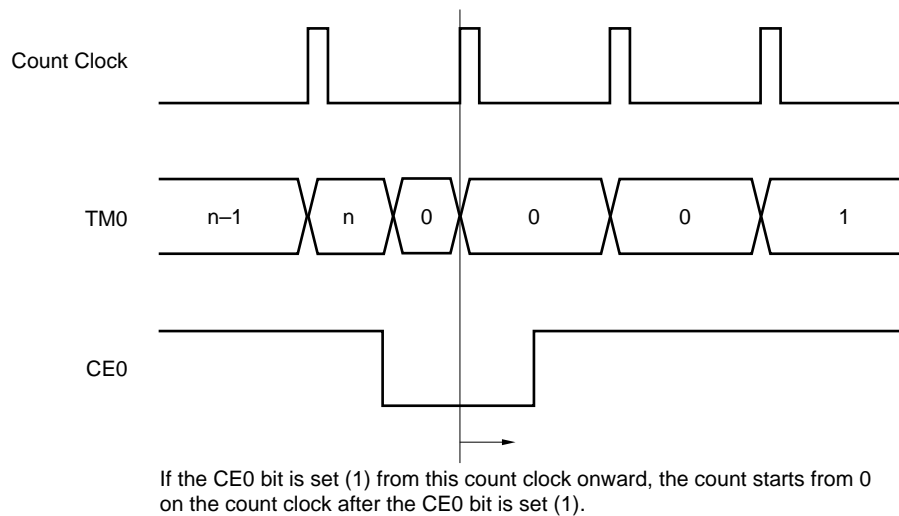
The timer register 0 (TM0) can be cleared automatically after a match with the compare register (CR01). When a clearance source arises, TM0 is cleared to 0000H on the next count clock. Therefore, even if a clearance source arises, the value at the point at which the clearance source arose is retained until the next count clock arrives.

**Figure 8-8 TM0 Clearance by Match with Compare Register (CR01)**



#### (2) Clear operation by the CE0 bit of the timer control register 0 (TMC0)

The timer register 0 (TM0) is also cleared when the CE0 bit of TMC0 is cleared (0) by software. The clear operation is performed immediately after clearance (0) of the CE0 bit.

**Figure 8-9 Clear Operation When CE0 Bit is Cleared (0)****(a) Basic operation****(b) Restart before count clock input after clearance****(c) Restart after count clock input after clearance**

## 8.5 EXTERNAL EVENT COUNTER FUNCTION

The timer/counter 0 can count clock pulses input from the external interrupt request input pin (INTP3).

No special selection method is needed for the external event counter operating mode. When the timer register 0 (TM0) count clock is specified as external clock input by the setting of the low-order 4 bits of prescaler mode register 0 (PRM0), TM0 operates as an external event counter.

The maximum frequency of external clock pulses that can be counted by TM0 as the external event counter is 1.56 MHz ( $f_{CLK} = 12.5$  MHz) irrespective of whether only one edge or both edges are counted on INTP3 input.

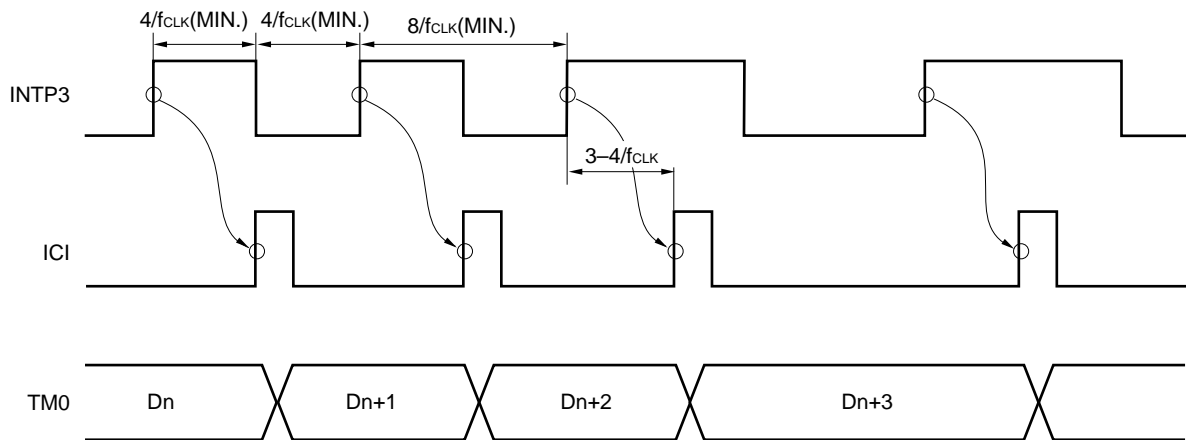
The pulse width of the INTP3 input must be at least 4 system clocks ( $0.32 \mu\text{s}$ ;  $f_{CLK} = 12.5$  MHz) for both the high level and low level. If the pulse width is shorter than this, the pulse may not be counted.

The timer/counter 0 external event counter timing is shown in Figure 8-10.

★

**Figure 8-10 Timer/Counter 0 External Event Count Timing (1/2)**

(1) Counting one edge (maximum frequency =  $f_{CLK}/8$ )

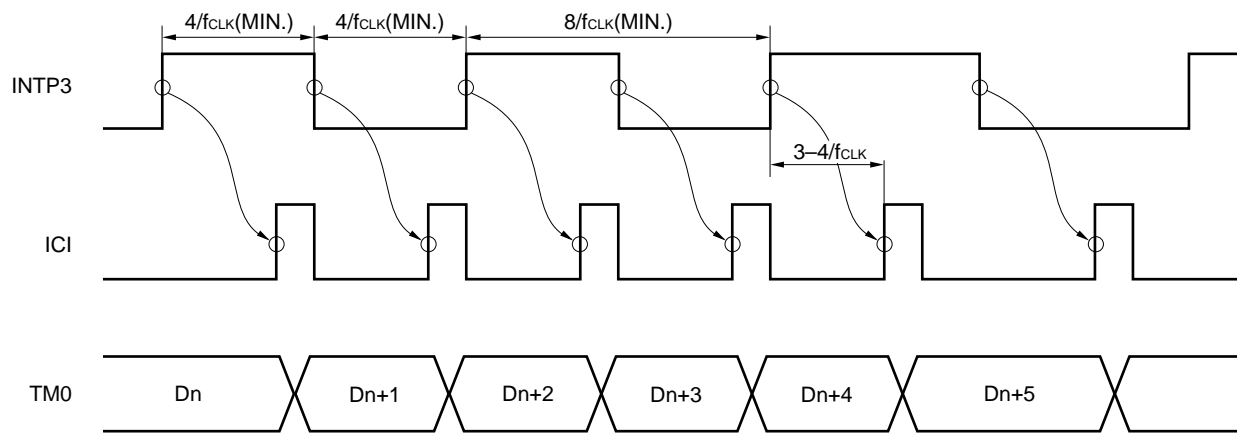


**Remark** ICI: INTP3 input signal after passing through edge detection circuit

★

Figure 8-10 Timer/Counter 0 External Event Count Timing (2/2)

(2) Counting both edges (maximum frequency =  $f_{CLK}/8$ )



**Remark** ICI: INTP3 input signal after passing through edge detection circuit

The TM0 count operation is controlled by the CE0 bit of the timer control register 0 (TMC0) in the same way as with basic operation.

When the CE0 bit is set (1) by software, the contents of TM0 are set to 0000H and the up-count is started on the initial count clock.

When the CE0 bit is cleared (0) by software during a TM0 count operation, the contents of TM0 are set to 0000H immediately and the stopped state is entered. The TM0 count operation is not affected if the CE0 bit is set (1) by software again when it is already set (1).

**Caution** When timer/counter 0 is used as an external event counter, it is not possible to distinguish between the case where there is no valid edge input at all and the case where there is a single valid edge input, using the timer register 0 (TM0) alone (see Figure 8-11), since the contents of TM0 are 0 in both cases. If it is necessary to make this distinction, the INTP3 interrupt request flag should be used. An example is shown in Figure 8-12.

**Figure 8-11** Example of the Case Where the External Event Counter Does Not Distinguish Between One Valid Edge Input and No Valid Edge Input

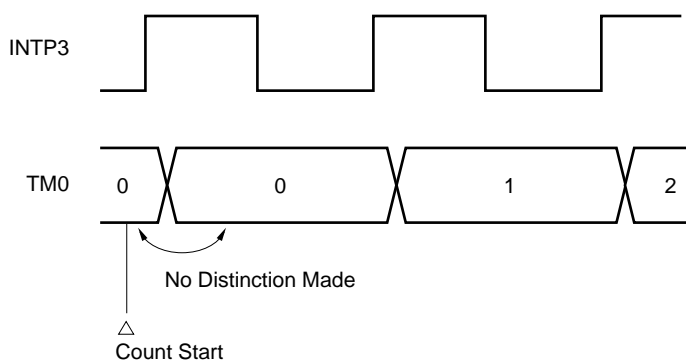
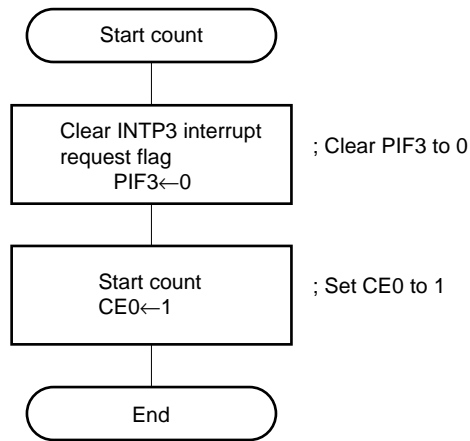
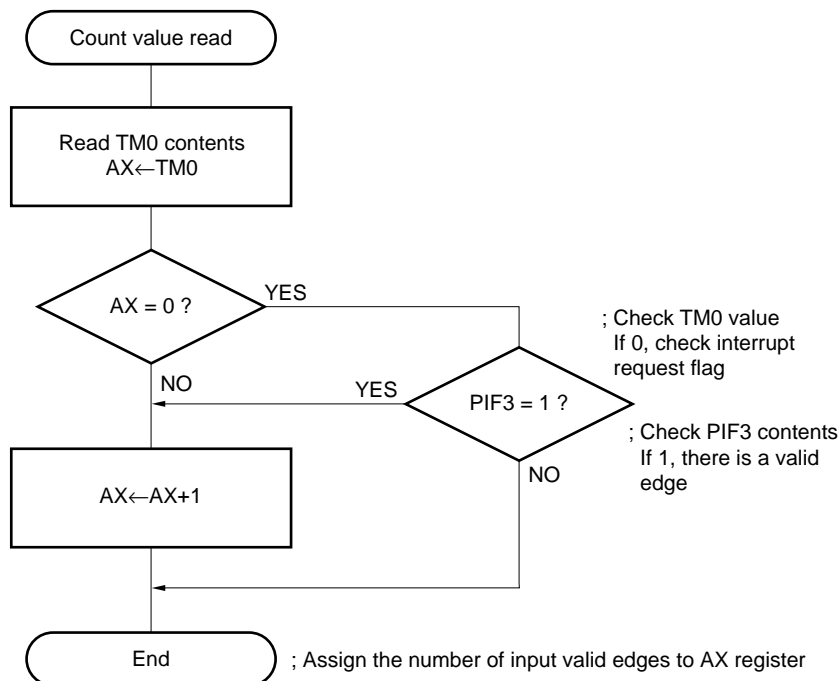


Figure 8-12 To Distinguish Whether One or No Valid Edge Has Been Input with External Event Counter

## (a) Processing when count is started



## (b) Processing when count value is read

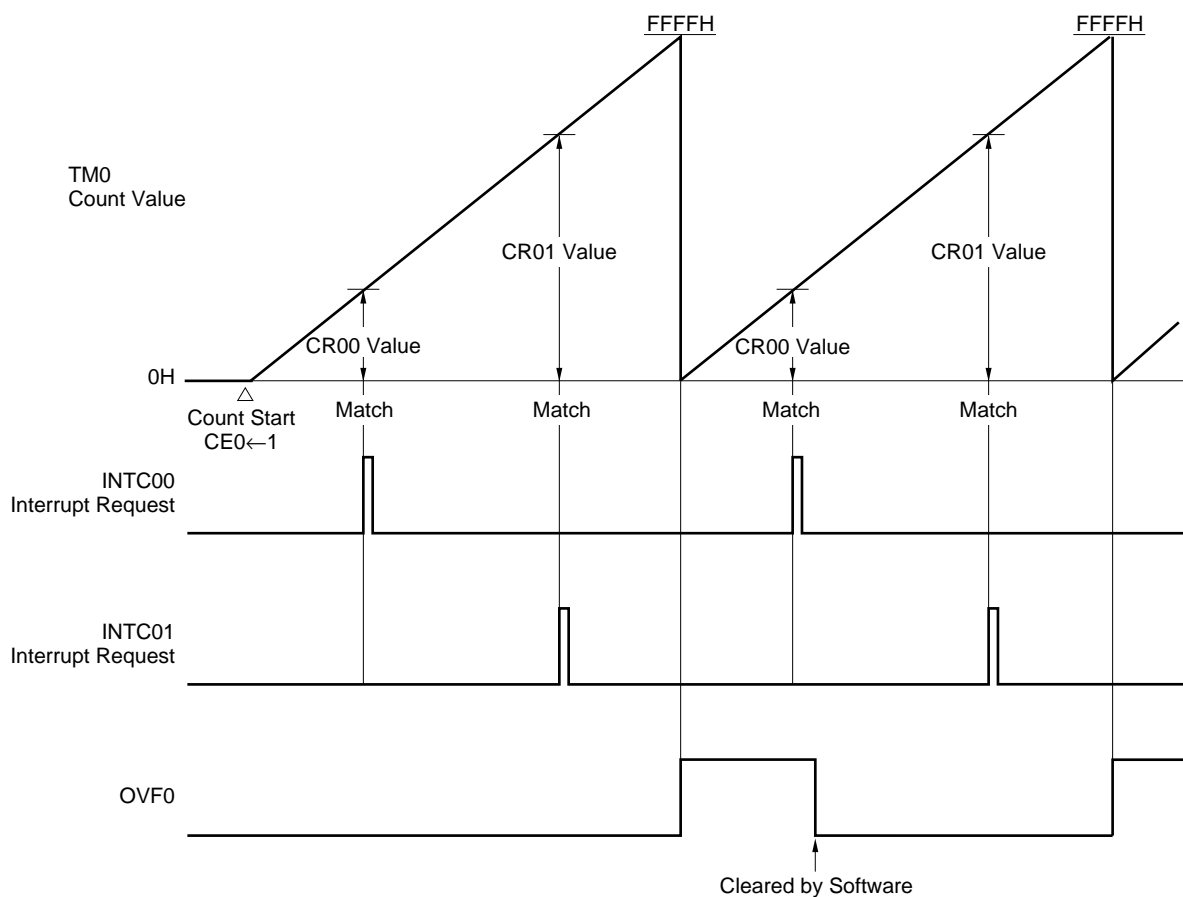


**8.6 COMPARE REGISTER AND CAPTURE REGISTER OPERATION****8.6.1 Compare Operations**

Timer/counter 0 performs compare operations in which the value set in compare registers (CR00, CR01) are compared with the timer register 0 (TM0) count value.

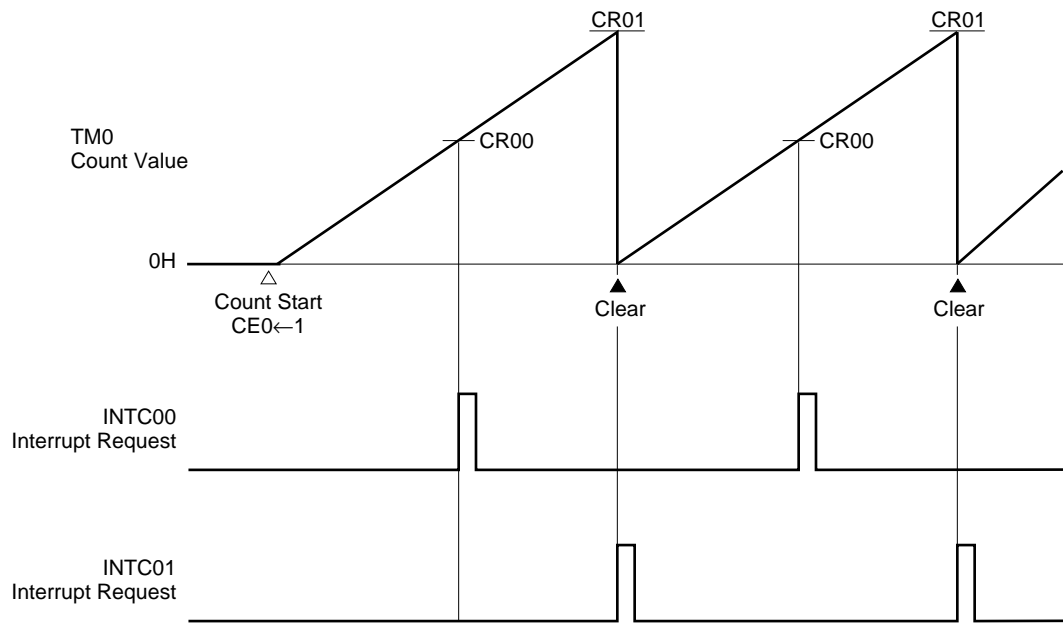
If the count value of TM0 matches the preset CR0n ( $n = 0, 1$ ) value as the result of the count operation, a match signal is sent to the output control circuit, and at the same time an interrupt request (INTC00/INTC01) is generated.

After a match with the CR01 value, the TM0 count value can be cleared, and the timer functions as an interval timer that repeatedly counts up to the value set in the CR01.

**Figure 8-13 Compare Operation**

**Remark** CLR01 = 0

Figure 8-14 TM0 Clearance After Match Detection



**Remark** CLR01 = 0



### 8.6.2 Capture Operations

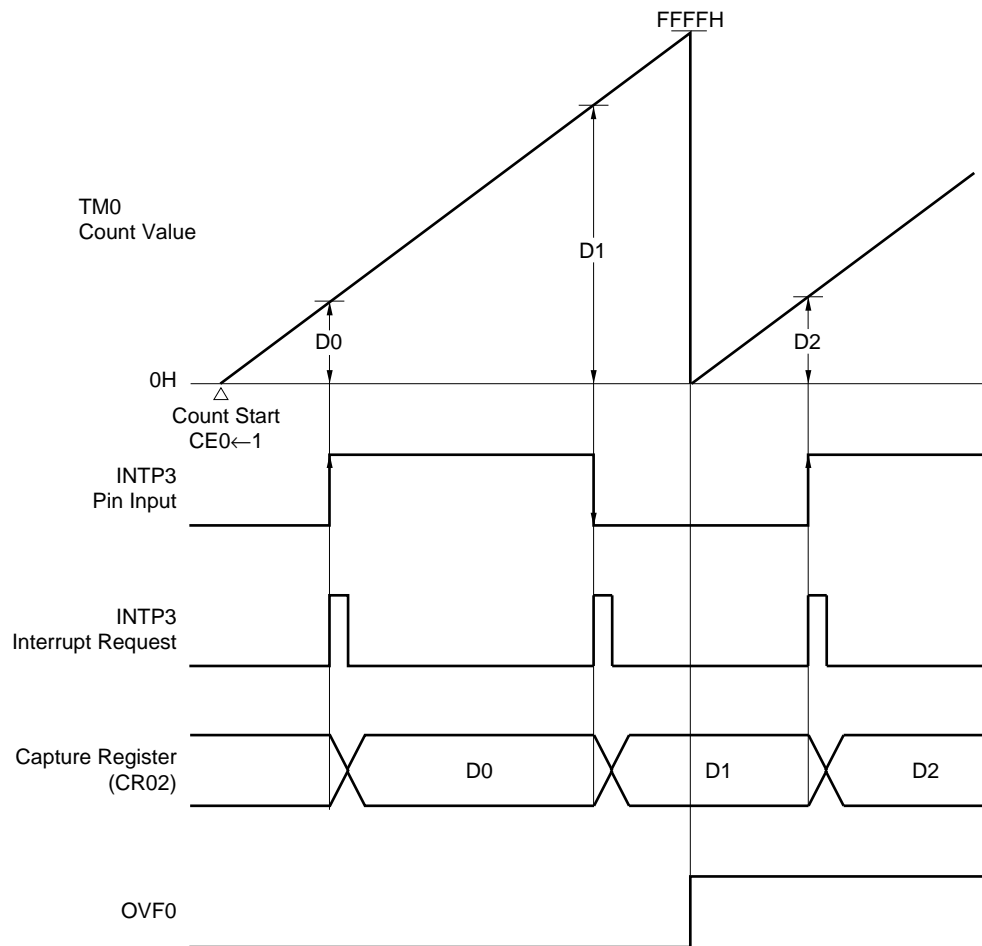
Timer/counter 0 performs capture operations in which the timer register 0 (TM0) count value is fetched into the capture register in synchronization with an external trigger, and retained there.

A valid edge detected from the input of the external interrupt request input pin (INTP3) is used as the external trigger (capture trigger). The count value of TM0 in the process of being counted is fetched into the capture register (CR02) in synchronization with the capture trigger, and is retained there. The contents of the CR02 are retained until the next capture trigger is generated.

The capture trigger valid edge is set by means of external interrupt mode register 1 (INTM1). If both rising and falling edges are set as capture triggers, the width of pulses input from off-chip can be measured. Also, if a capture trigger is generated by a single edge, the input pulse cycle can be measured.

See **Figure 19-2** in **CHAPTER 19 EDGE DETECTION FUNCTION** for details of the INTM1.

**Figure 8-15 Capture Operation**



**Remark** Dn: TM0 count value (n = 0, 1, 2, ...)  
CLR01 = 0

## 8.7 BASIC OPERATION OF OUTPUT CONTROL CIRCUIT

The output control circuit controls the timer output pin (TO0/TO1) levels by means of overflow signals or match signals from the compare registers (CR00, CR01). The operation of the output control circuit is determined by the timer output control register (TOC), capture/compare control register 0 (CRC0), and the one-shot pulse output control register (OSPC) (see **Table 8-5**).

When TO0, TO1 signals are output to a pin, the relevant pin must be in control mode in the port 3 mode register (PMC3).

Table 8-5 Timer Output (TO0/TO1) Operations

TOC				OSPC		CRC0			TO1	TO0
ENTO1	ALV1	ENTO0	ALV0	OS1	OS0	MOD1	MOD0	CLR01		
0	0/1	0	0/1	×	×	×	×	×	High/low level fixed	High/low level fixed
0	0/1	1	0/1	×	0	0	0	×	High/low level fixed	Toggle output (active-low/high)
0	0/1	1	0/1	×	0	0	1	0	High/low level fixed	PWM output (active-high/low)
0	0/1	1	0/1	×	0	1	0	0	High/low level fixed	PWM output (active-high/low)
0	0/1	1	0/1	×	0	1	1	1	High/low level fixed	PPG output (active-high/low)
0	0/1	1	0/1	×	1	×	×	×	High/low level fixed	One-shot pulse output (active-low/high)
1	0/1	0	0/1	0	×	0	×	×	Toggle output (active-low/high)	High/low level fixed
1	0/1	0	0/1	0	×	1	0	0	PWM output (active-high/low)	High/low level fixed
1	0/1	0	0/1	0	×	1	1	×	Toggle output (active-low/high)	High/low level fixed
1	0/1	0	0/1	1	×	×	×	×	One-shot pulse output (active-low/high)	High/low level fixed
1	0/1	1	0/1	0	0	0	0	×	Toggle output (active-low/high)	Toggle output (active-low/high)
1	0/1	1	0/1	0	0	0	1	0	Toggle output (active-low/high)	PWM output (active-high/low)
1	0/1	1	0/1	0	0	1	0	0	PWM output (active-high/low)	PWM output (active-high/low)
1	0/1	1	0/1	0	0	1	1	1	Toggle output (active-low/high)	PPG output (active-high/low)
1	0/1	1	0/1	0	1	0	×	×	Toggle output (active-low/high)	One-shot pulse output (active-low/high)
1	0/1	1	0/1	0	1	1	0	0	PWM output (active-high/low)	One-shot pulse output (active-low/high)
1	0/1	1	0/1	0	1	1	1	1	Toggle output (active-low/high)	One-shot pulse output (active-low/high)
1	0/1	1	0/1	1	0	0	0	×	One-shot pulse output (active-low/high)	Toggle output (active-low/high)
1	0/1	1	0/1	1	0	0	1	0	One-shot pulse output (active-low/high)	PWM output (active-high/low)
1	0/1	1	0/1	1	0	1	0	0	One-shot pulse output (active-low/high)	PWM output (active-high/low)
1	0/1	1	0/1	1	0	1	1	1	One-shot pulse output (active-low/high)	PPG output (active-high/low)
1	0/1	1	0/1	1	1	×	×	×	One-shot pulse output (active-low/high)	One-shot pulse output (active-low/high)

- Remarks 1.** In the ALVn (n = 0, 1) columns, the figures on the left and right of the slash ("/") correspond to the items on the left and right of the slash in the TOn (n = 0, 1) columns.
- 2.** The "×" mark indicates that the operation is the same for either 0 or 1, but some prohibited combinations are included (see **Figure 8-4**).
- 3.** Use with combinations not shown in this table is prohibited.

### 8.7.1 Basic Operation

Setting (1) the ENTOn ( $n = 0, 1$ ) bit of the timer output control register (TOC) enables timer output (TON:  $n = 0, 1$ ) to be varied at a timing in accordance with the settings of MOD0, MOD1 and CLR01 bits of capture/compare control register 0 (CRC0) and the one-shot pulse output control register (OSPC).

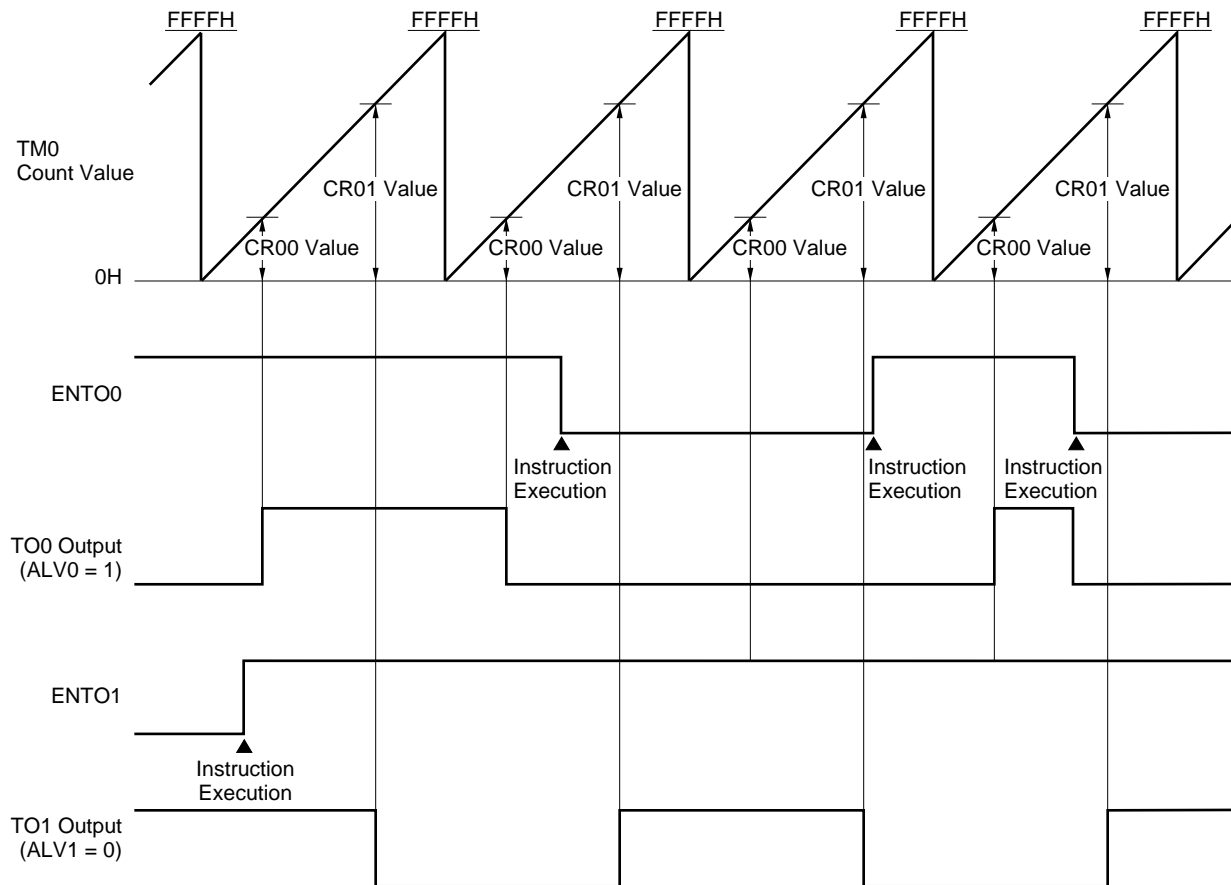
Clearing (0) ENTOn sets the TON to a fixed level. The fixed level is determined by the ALVn ( $n = 0, 1$ ) bit of the TOC. The level is high when ALVn is 0, and low when 1.

### 8.7.2 Toggle Output

Toggle output is an operating mode in which the output level is inverted each time the compare register (CR00/CR01) value coincides with the timer register 0 (TM0) value. The output level of timer output (TO0) is inverted by a match between CR00 and TM0, and the output level of TO1 is inverted by a match between CR01 and TM0.

- ★ When timer/counter 0 is stopped by clearing (0) the CE0 bit of the timer control register 0 (TMC0), the inactive level ( $\overline{ALVn}$ :  $n = 0, 1$ ) is output.

**Figure 8-16 Toggle Output Operation**



**Table 8-6 TO0, TO1 Toggle Output ( $f_{xx} = 25 \text{ MHz}$ )**

Count Clock	Minimum Pulse Width	Maximum Interval Time
$8/f_{xx}$	$0.32 \mu\text{s}$	21.0 ms
$16/f_{xx}$	$0.64 \mu\text{s}$	41.9 ms
$32/f_{xx}$	$1.28 \mu\text{s}$	83.9 ms
$64/f_{xx}$	$2.56 \mu\text{s}$	168 ms
$128/f_{xx}$	$5.12 \mu\text{s}$	336 ms
$256/f_{xx}$	$10.2 \mu\text{s}$	671 ms
$512/f_{xx}$	$20.5 \mu\text{s}$	1.34 s
$1024/f_{xx}$	$41.0 \mu\text{s}$	2.68 s
$2048/f_{xx}$	$81.9 \mu\text{s}$	5.37 s

### 8.7.3 PWM Output

#### (1) Basic operation of PWM output

In this mode, a PWM signal with the period in which timer register 0 (TM0) reaches a full count used as one cycle is output. The timer output (TO0) pulse width is determined by the value of compare register (CR00), and the timer output (TO1) pulse width is determined by the value of compare register (CR01). When this function is used, the CLR01 bit of capture/compare control register 0 (CRC0) must be set to 0.

The pulse cycle and pulse width are as shown below.

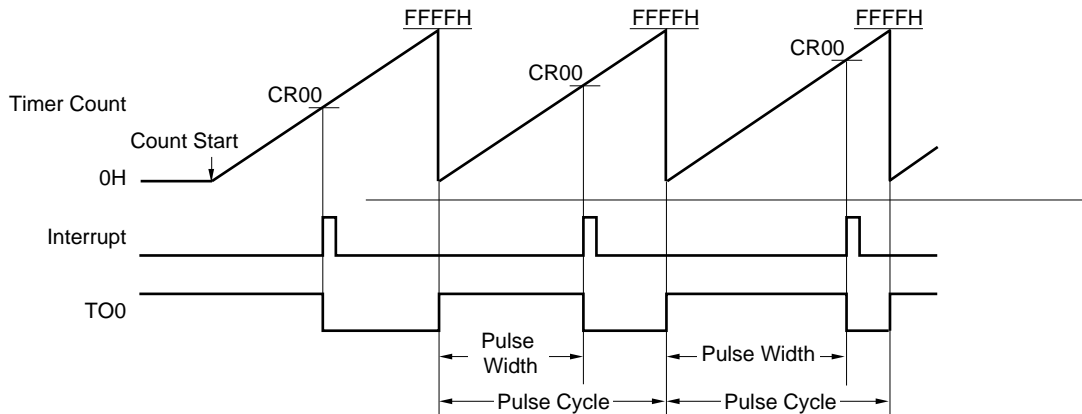
- PWM cycle =  $65536 \times x / f_{xx}$
- PWM pulse width =  $CR0n \times x / f_{xx}^*$ ;  $x = 8, 16, 32, 64, 128, 256, 512, 1024, 2048$

\* 0 cannot be assigned to the CR0n.

$$\bullet \text{ Duty} = \frac{\text{PWM pulse width}}{\text{PWM cycle}} = \frac{CR0n}{65536}$$

**Remark**  $n = 0, 1$

**Figure 8-17 PWM Pulse Output**

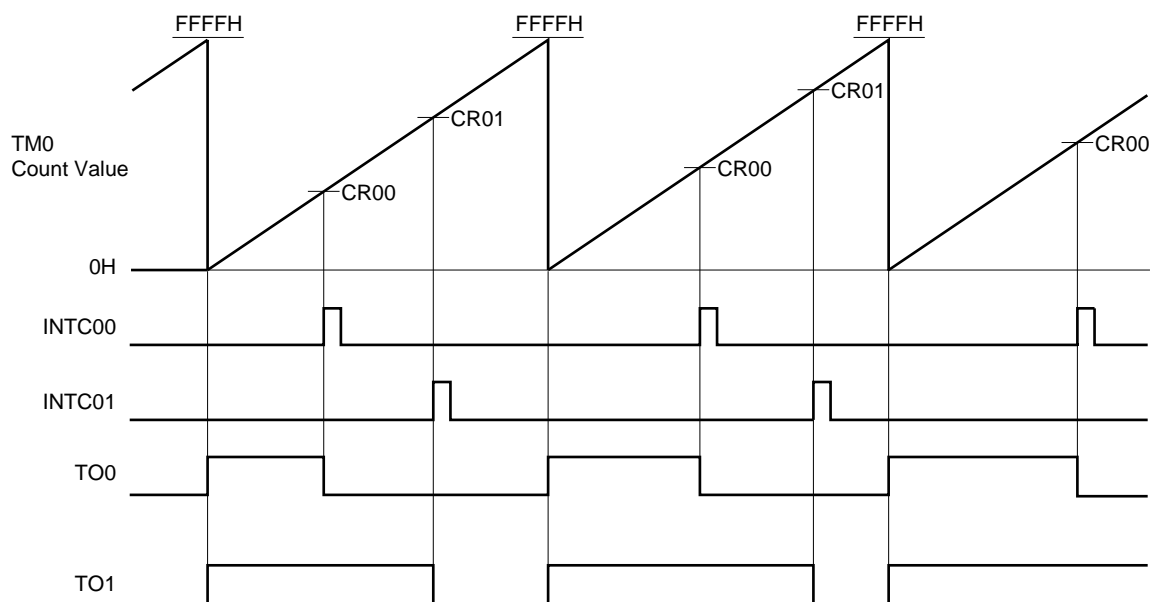


**Remark**  $ALV0 = 0$

**Table 8-7 TO0, TO1 PWM Cycle ( $f_{xx} = 25 \text{ MHz}$ )**

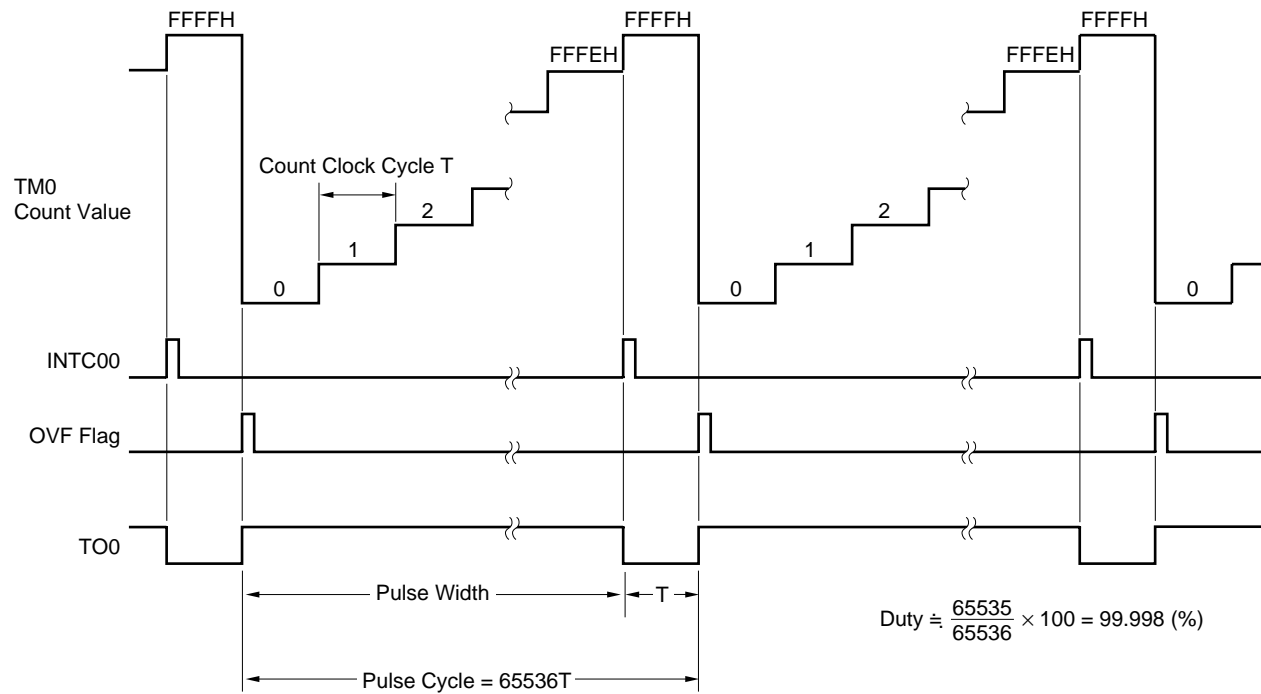
Count Clock	Minimum Pulse Width [ $\mu\text{s}$ ]	PWM Cycle [s]	PWM Frequency [Hz]
$f_{xx}/8$	0.32	0.02	47.7
$f_{xx}/16$	0.64	0.04	23.8
$f_{xx}/32$	1.28	0.08	11.9
$f_{xx}/64$	2.56	0.17	6.0
$f_{xx}/128$	5.12	0.34	3.0
$f_{xx}/256$	10.2	0.67	1.5
$f_{xx}/512$	20.5	1.34	0.7
$f_{xx}/1024$	41.0	2.68	0.4
$f_{xx}/2048$	81.9	5.37	0.2

Figure 8-18 shows an example of 2-channel PWM output, and Figure 8-19 shows the operation of the case where FFFFH is assigned to the CR00.

**Figure 8-18 Example of PWM Output Using TM0**

**Remark** ALV0 = 0, ALV1 = 0

Figure 8-19 Example of PWM Output When CR00 = FFFFH



**Remarks 1.** ALV0 = 0

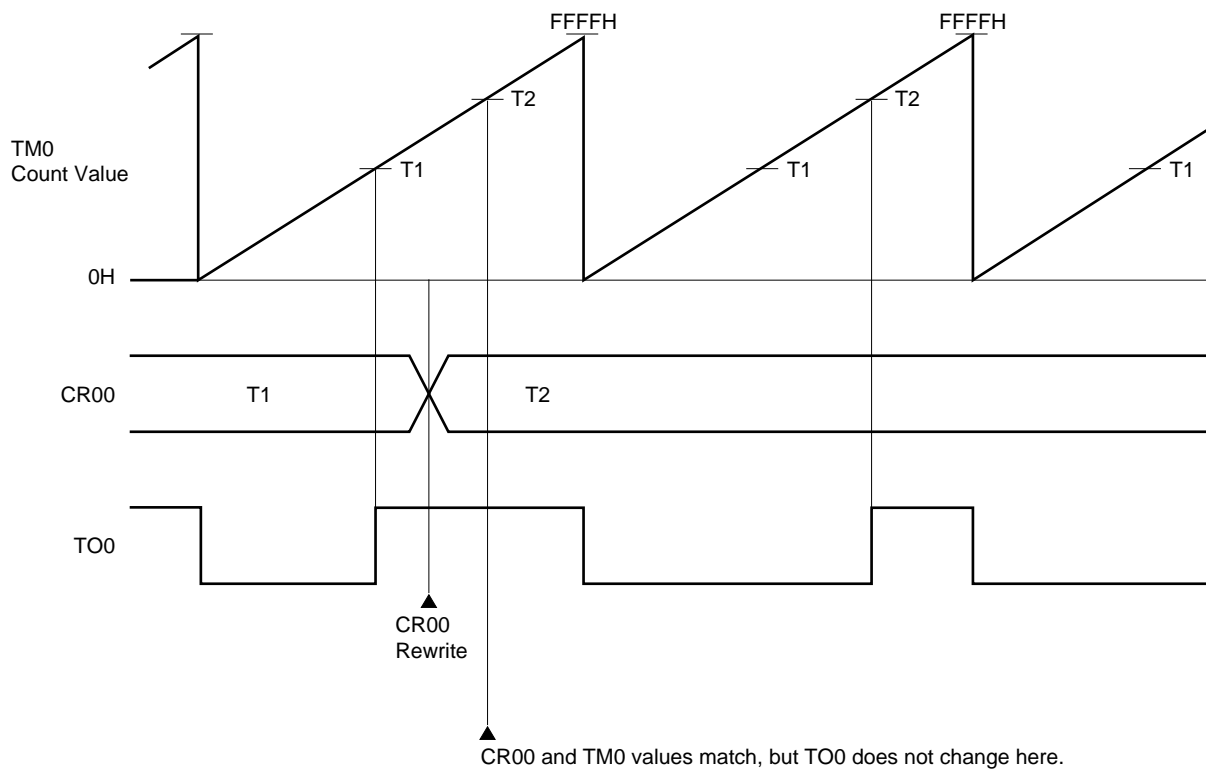
**2.**  $T = x/f_{xx}$  ( $x = 8, 16, 32, 64, 128, 256, 512, 1024, 2048$ )



**(2) Rewriting compare registers (CR00, CR01)**

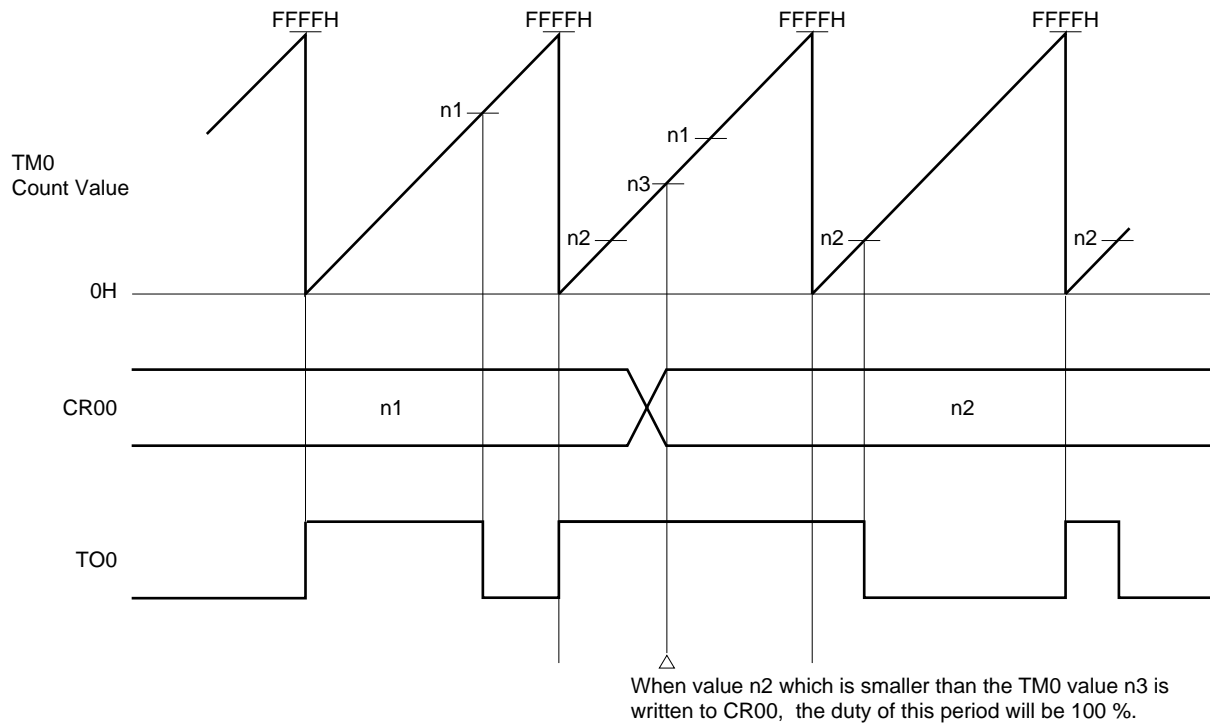
The output level of the timer output (TOn:  $n = 0, 1$ ) does not change even if the CR0n ( $n = 0, 1$ ) value matches the timer register 0 (TM0) value more than once during one PWM output cycle.

**Figure 8-20 Example of Compare Register (CR00) Rewrite**



If a value smaller than that of the TM0 is set as the CR0n value, a 100 % duty PWM signal will be output. CR0n rewriting should be performed by the interrupt due to a match between TM0 and the CR0n on which the rewrite is performed.

**Figure 8-21 Example of 100 % Duty With PWM Output**

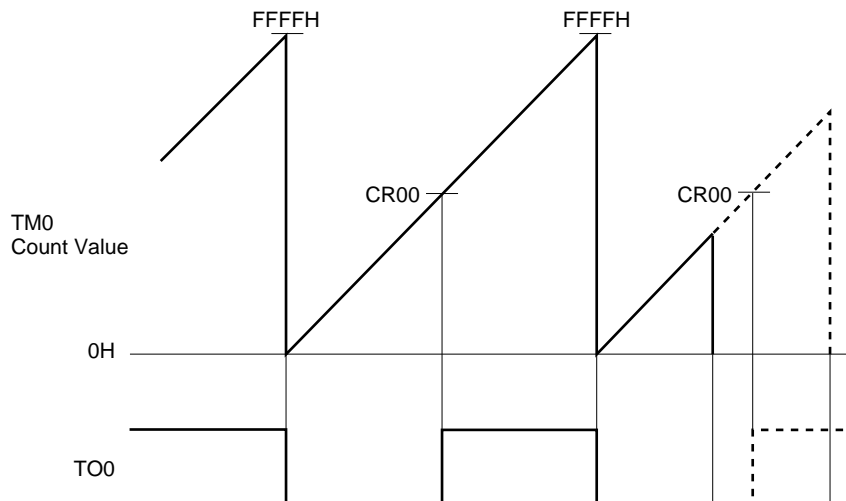


**Remark** ALV0 = 0

**(3) Stopping PWM output**

If timer/counter 0 is stopped by clearing (0) the CE0 bit of the timer control register 0 (TMC0) during PWM signal output, the active level is output.

★

**Figure 8-22 When Timer/Counter 0 is Stopped During PWM Signal Output**

**Remark** ALV0 = 1

**Caution** The output level of the TOn (n = 0, 1) pin when timer output is disabled (ENTOn = 0, n = 0, 1) is the inverse of the value set in ALVn (n = 0, 1) bit. Caution is therefore required as the active level is output when timer output is disabled when the PWM output function has been selected.

## 8.7.4 PPG Output

## (1) Basic Operation of PPG Output

This function outputs a square-wave with the time determined by compare register CR01 value as one cycle, and the time determined by compare register CR00 value as the pulse width. The PWM cycle output by the PWM is made variable. This signal can only be output from the timer output (TO0).

When this function is used, the CLR01 bit of capture/compare control register 0 (CRC0) must be set to 1.

The pulse cycle and pulse width are as shown below.

- PPG cycle =  $(CR01 + 1) \times x / f_{xx}$ ;  $x = 8, 16, 32, 64, 128, 256, 512, 1024, 2048$

- PPG pulse width =  $CR00 \times x / f_{xx}$

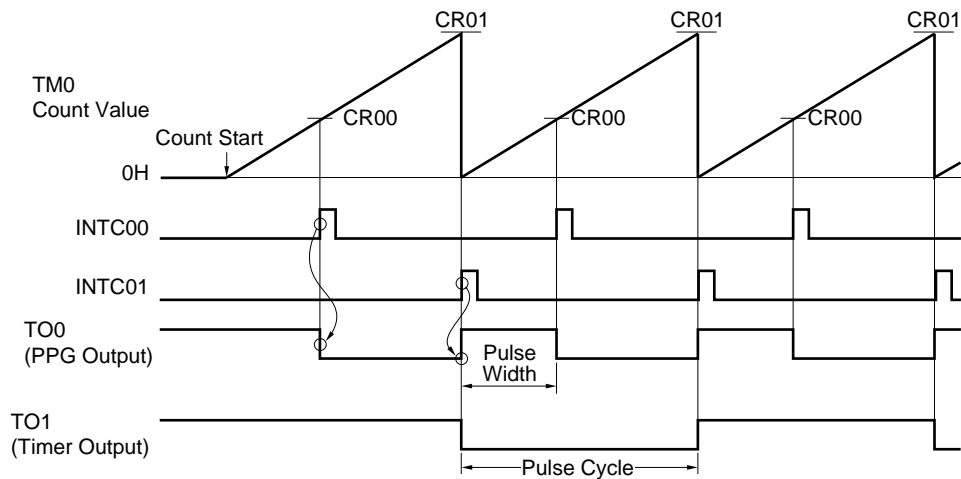
$$\text{where } 1 \leq CR00 \leq CR01 *$$

- Duty =  $\frac{\text{PPG pulse width}}{\text{PPG cycle}} = \frac{CR00}{CR01 + 1} *$

\* 0 cannot be assigned to either CR00 or CR01.

Figure 8-23 shows an example of PPG output using timer register 0 (TM0), Figure 8-24 shows an example of the case where  $CR00 = CR01$ .

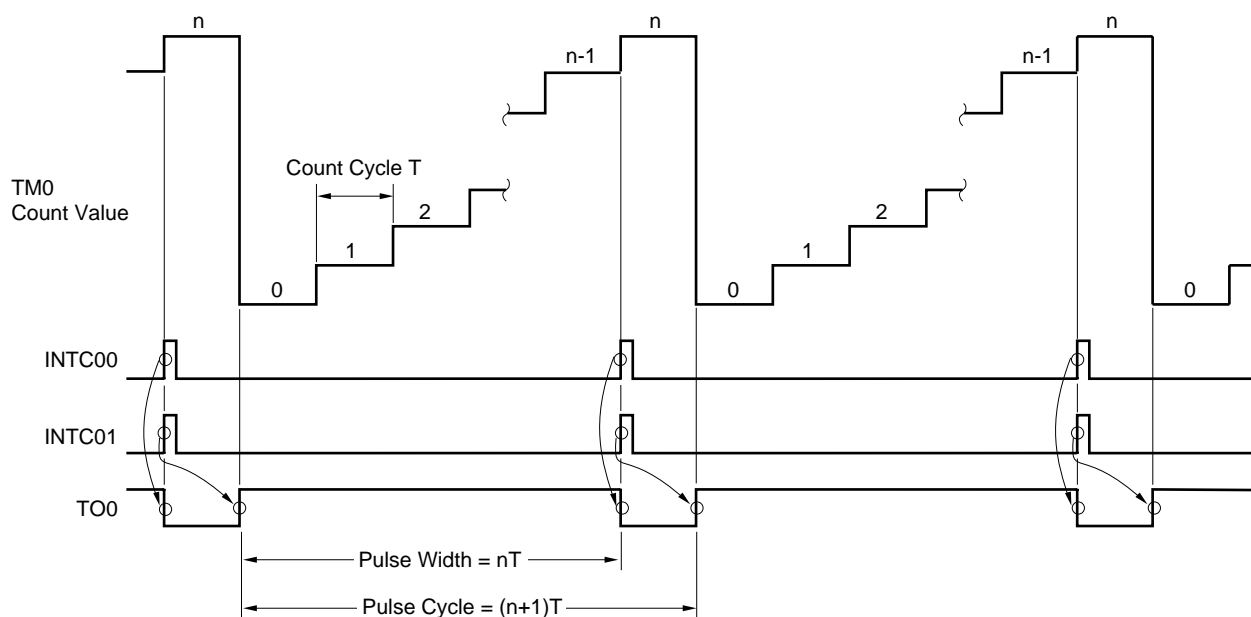
Figure 8-23 Example of PPG Output Using TM0



**Remark** ALV0 = 0, ALV1 = 0

**Table 8-8 TO0 PPG Output ( $f_{xx} = 25$  MHz)**

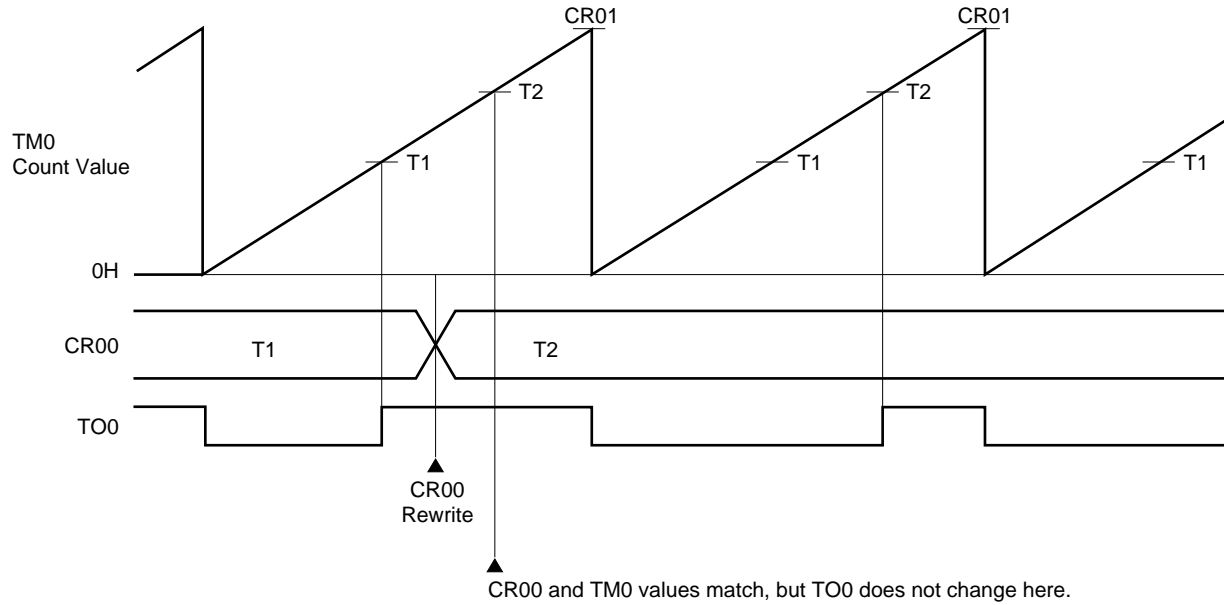
Count Clock	Minimum Pulse Width	PPG Cycle	PPG Frequency
$f_{xx}/8$	$0.32\ \mu\text{s}$	$0.64\ \mu\text{s}$ to $21.0\ \text{ms}$	$1563\ \text{kHz}$ to $47.7\ \text{Hz}$
$f_{xx}/16$	$0.64\ \mu\text{s}$	$1.28\ \mu\text{s}$ to $41.9\ \text{ms}$	$781\ \text{kHz}$ to $23.8\ \text{Hz}$
$f_{xx}/32$	$1.28\ \mu\text{s}$	$2.56\ \mu\text{s}$ to $83.9\ \text{ms}$	$391\ \text{kHz}$ to $11.9\ \text{Hz}$
$f_{xx}/64$	$2.56\ \mu\text{s}$	$5.12\ \mu\text{s}$ to $0.17\ \text{s}$	$195\ \text{kHz}$ to $5.96\ \text{Hz}$
$f_{xx}/128$	$5.12\ \mu\text{s}$	$10.2\ \mu\text{s}$ to $0.34\ \text{s}$	$97.7\ \text{kHz}$ to $2.98\ \text{Hz}$
$f_{xx}/256$	$10.2\ \mu\text{s}$	$20.5\ \mu\text{s}$ to $0.67\ \text{s}$	$48.8\ \text{kHz}$ to $1.49\ \text{Hz}$
$f_{xx}/512$	$20.5\ \mu\text{s}$	$41.0\ \mu\text{s}$ to $1.34\ \text{s}$	$24.4\ \text{kHz}$ to $0.75\ \text{Hz}$
$f_{xx}/1024$	$41.0\ \mu\text{s}$	$81.9\ \mu\text{s}$ to $2.68\ \text{s}$	$12.2\ \text{kHz}$ to $0.37\ \text{Hz}$
$f_{xx}/2048$	$81.9\ \mu\text{s}$	$164\ \mu\text{s}$ to $5.37\ \text{s}$	$6.10\ \text{kHz}$ to $0.19\ \text{Hz}$

**Figure 8-24 Example of PPG Output When  $CR00 = CR01$** **Remark**  $ALV0 = 0$  $T = x/f_{xx}$  ( $x = 8, 16, 32, 64, 128, 256, 512, 1024, 2048$ )

**(2) Rewriting compare register (CR00)**

The output level of the timer output (TO0) does not change even if the CR00 value matches the timer register 0 (TM0) value more than once during one PPG output cycle.

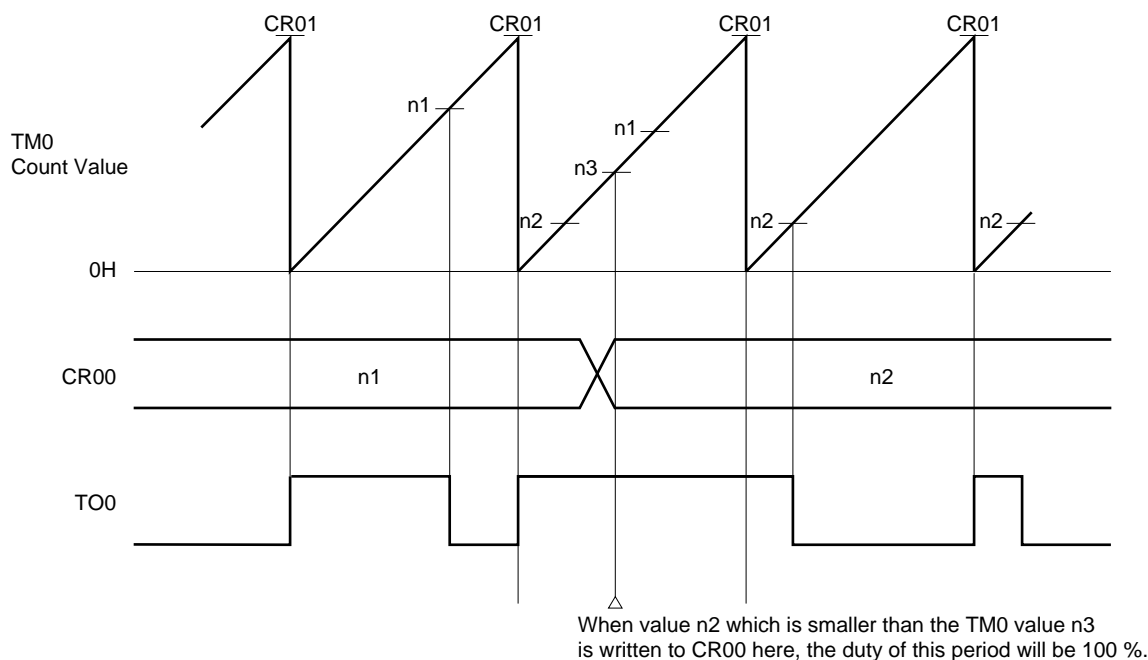
**Figure 8-25 Example of Compare Register (CR00) Rewrite**



**Remark** ALV0 = 1

If a value equal to or less than the TM0 value is written to CR00 before the CR00 and TM0 match, the duty of the PPG cycle will be 100 %. CR00 rewriting should be performed by the interrupt due to a match between TM0 and CR00.

**Figure 8-26 Example of 100 % Duty With PPG Output**



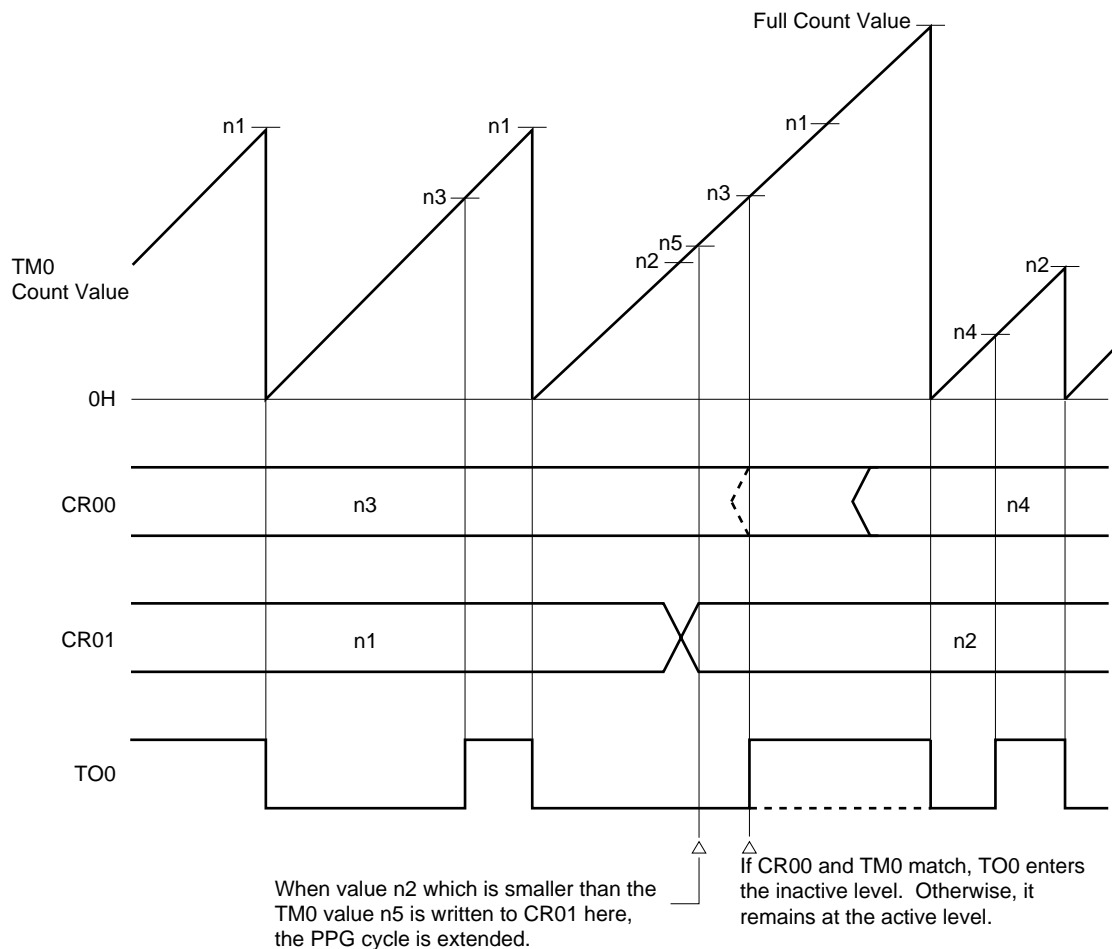
**Remark** ALV0 = 0

**Caution** If the PPG cycle is extremely short as compared with the time required to acknowledge an interrupt, the value of CR00 cannot be rewritten by interrupt processing that is performed due to a match between TM0 and CR00. Use another method (for example, to poll the interrupt request flags in software with all the interrupts masked).

**(3) Rewriting compare register (CR01)**

If the current value of the CR01 is changed to a smaller value, and the CR01 value is made smaller than the timer register 0 (TM0) value, the PPG cycle at that time will be extended to the time equivalent to a full-count by TM0. If CR01 is rewritten after the compare register (CR00) and TM0 match, the output level at this time will be the inactive level until TM0 overflows and becomes 0, and will then return to normal PPG output.

If CR01 is rewritten before CR00 and TM0 match, the active level will be output until CR00 and TM0 match. If CR00 and TM0 match before TM0 overflows and becomes 0, the inactive level is output at that point. When TM0 overflows and becomes 0, the active level will be output, and normal PPG output will be restored. CR01 rewriting should be performed by the interrupt due to a match between TM0 and CR01, etc.

**Figure 8-27 Example of Extended PPG Output Cycle**

**Remark** ALV0 = 1

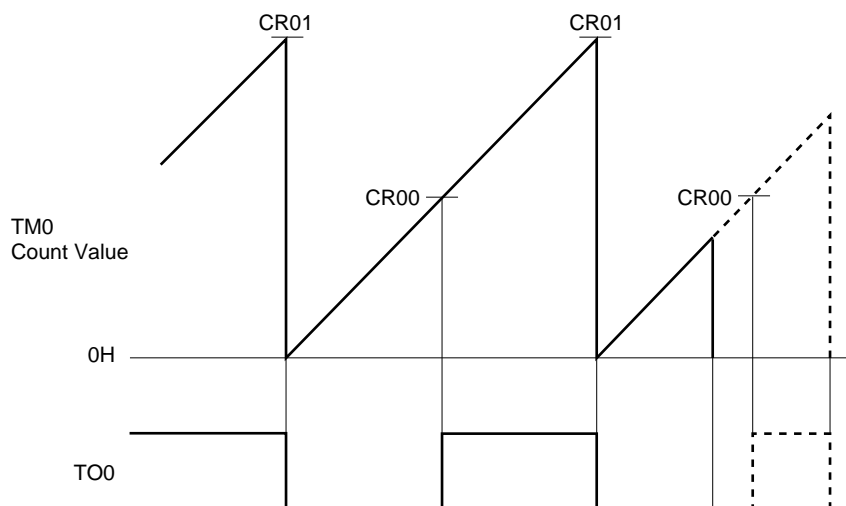


**Caution** If the PPG cycle is extremely short as compared with the time required to acknowledge an interrupt, the value of CR01 cannot be rewritten by interrupt processing that is performed due to a match between the timer register (TM0) and compare register (CR01). Use another method (for example, to poll the interrupt request flags in software with all the interrupts masked).

#### (4) Stopping PPG output

If timer/counter 0 is stopped by clearing (0) the CE0 bit of the timer control register 0 (TMC0) during PPG signal output, the active level is output irrespective of the output level at the time it was stopped.

**Figure 8-28 When Timer/Counter 0 is Stopped During PPG Signal Output**



**Caution** The output level of the TOn ( $n = 0, 1$ ) pin when timer output is disabled ( $ENTOn = 0$ :  $n = 0, 1$ ) is the inverse of the value set in ALVn ( $n = 0, 1$ ) bit. Caution is therefore required as the active level is output when timer output is disabled when the PPG output function has been selected.

**8.7.5 Software Triggered One-Shot Pulse Output**

In the software triggered one-shot pulse output mode, a one-shot pulse is output by software.

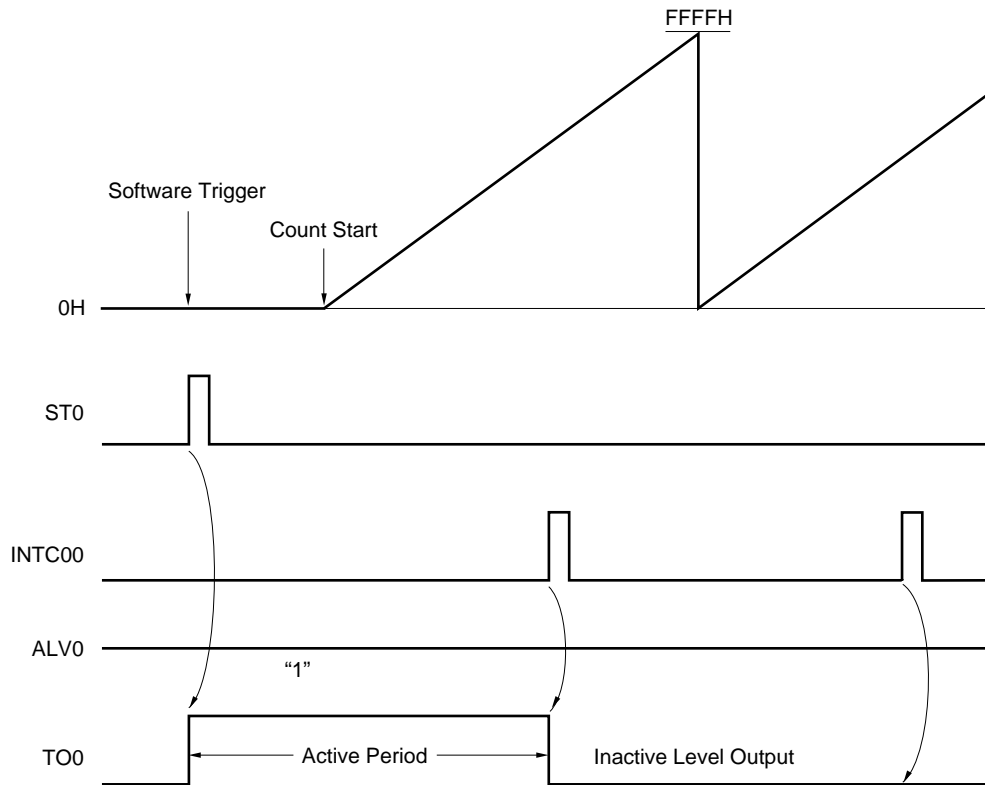
When the STn (n = 0/1) bit of the one-shot pulse output control register (OSPC) is set (1), timer output pin (TOn: n = 0, 1) is set to the active level. TOn then remains at the active level until the timer register 0 (TM0) value and the compare register (CR0n: n = 0, 1) value match, at which point TOn changes to the inactive level. TOn then retains at the inactive level until the STn bit is set again. TOn can also be set to the inactive level by setting (1) the RTn bit (n = 0/1), and in the same way, TOn retains at the inactive level until the STn bit is set again.

TO0 and TO1 can be controlled independently.

An example of software triggered one-shot pulse output is shown in Figure 8-29.

When timer/counter 0 is stopped by clearing (0) the CE0 bit of the TMC0, the level at the time was stopped is retained.

**Figure 8-29 Example of Software Triggered One-Shot Pulse Output**



**Caution** "1" should not be written to STn and RTn simultaneously.

## 8.8 EXAMPLES OF USE

### 8.8.1 Operation as Interval Timer (1)

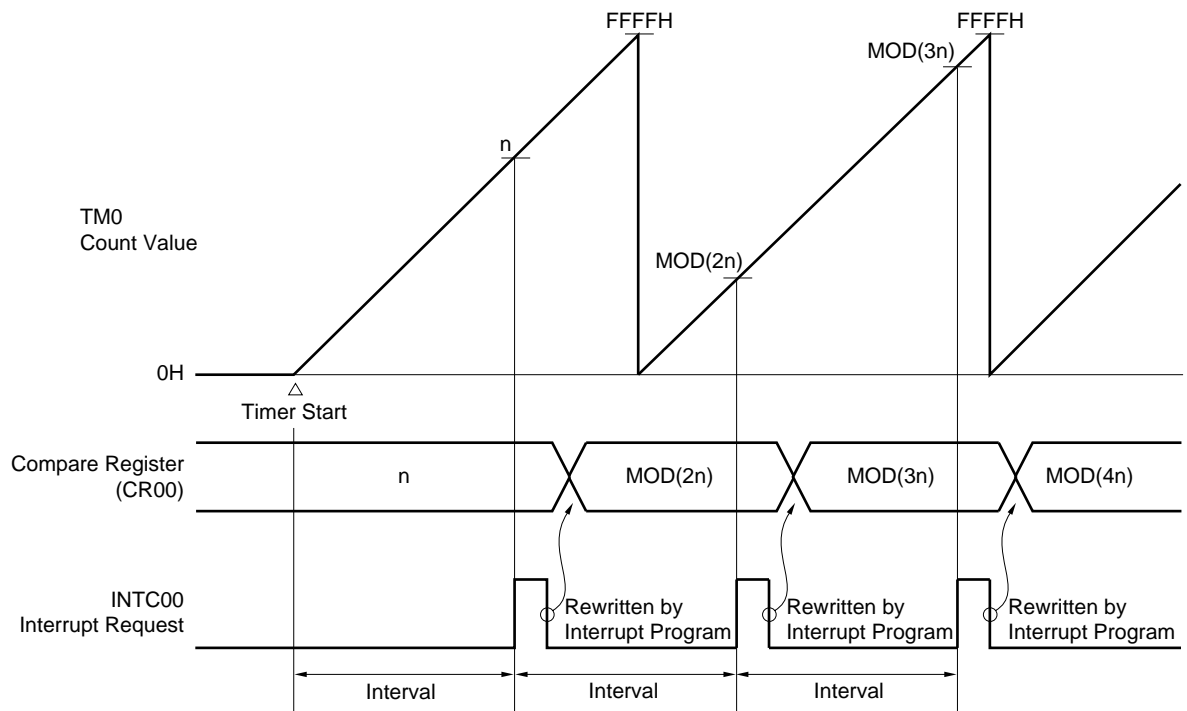
When timer register 0 (TM0) is made free-running and a fixed value is added to the compare register (CR0n: n = 0, 1) in the interrupt processing routine, TM0 operates as an interval timer with the added fixed value as the cycle (see **Figure 8-30**).

This interval timer can count within the range shown in Table 8-1 (internal system clock  $f_{xx} = 25$  MHz).

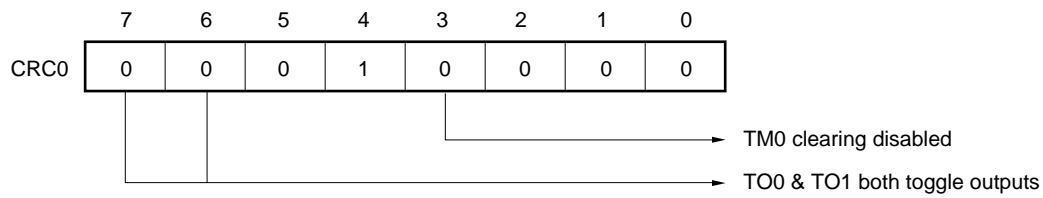
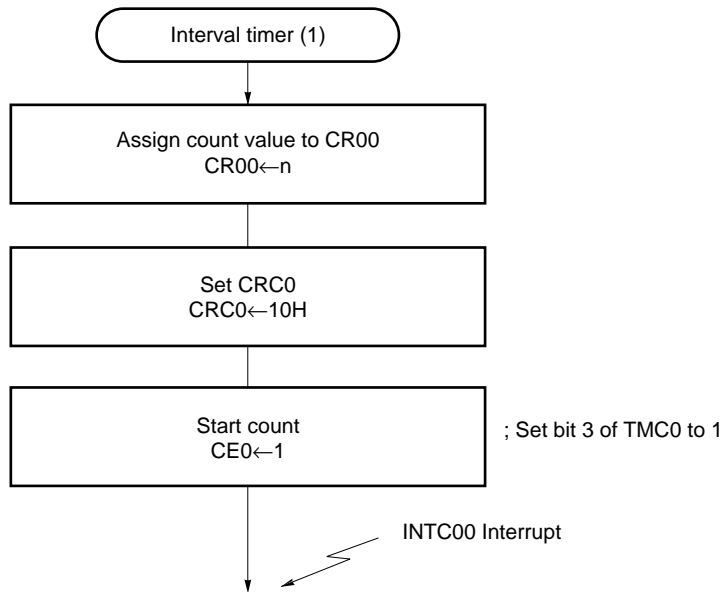
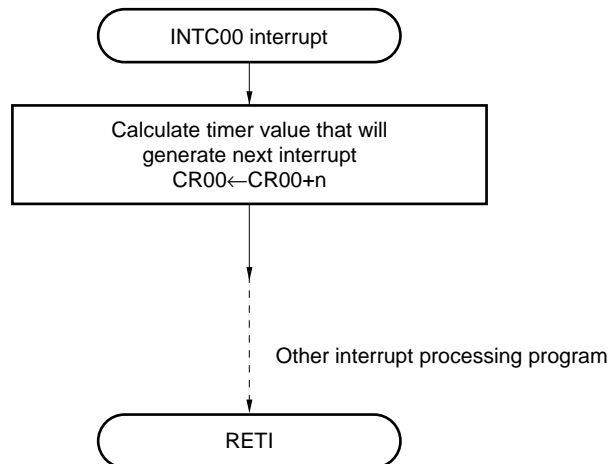
Since TM0 has two compare registers, two interval timers with different cycles can be constructed.

The control register settings are shown in Figure 8-31, the setting procedure in Figure 8-32, and the processing in the interrupt processing routine in Figure 8-33.

**Figure 8-30 Interval Timer Operation (1) Timing**



**Remark** Interval =  $n \times 8/f_{xx}$ ,  $1 \leq n \leq \text{FFFFH}$

**Figure 8-31 Control Register Settings for Interval Timer Operation (1)****Capture/compare control register 0 (CRC0)****Figure 8-32 Interval Timer Operation (1) Setting Procedure****Figure 8-33 Interval Timer Operation (1) Interrupt Request Processing**

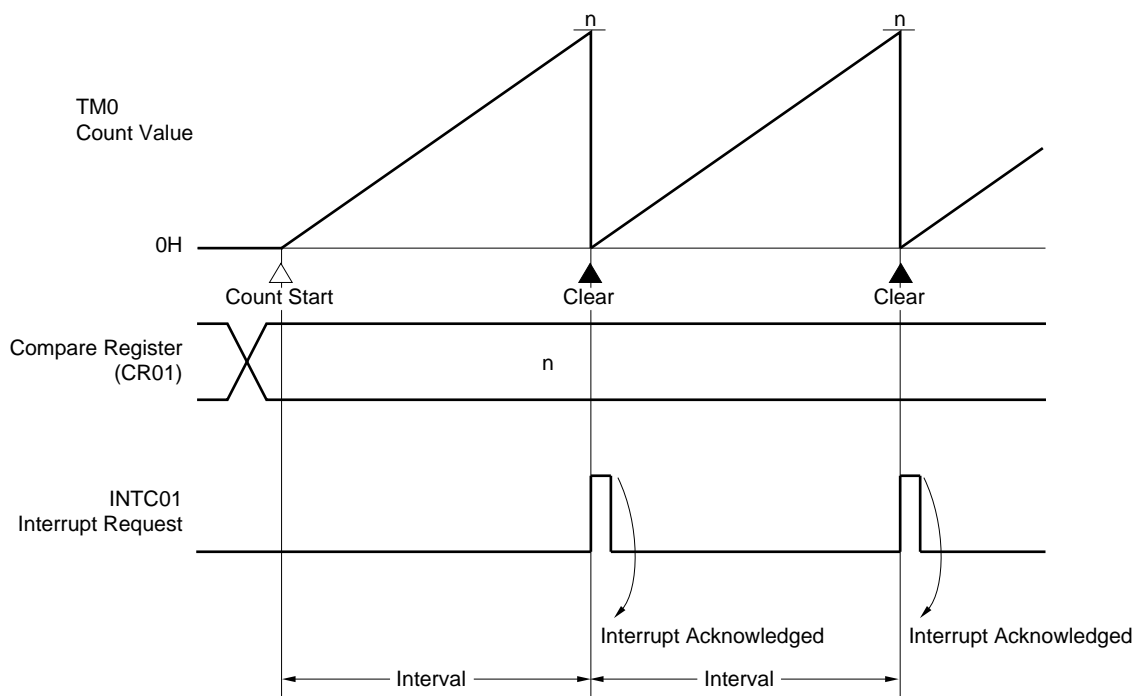
**8.8.2 Operation as Interval Timer (2)**

TM0 operates as an interval timer that generates interrupts repeatedly with the preset count time as the interval (see **Figure 8-34**).

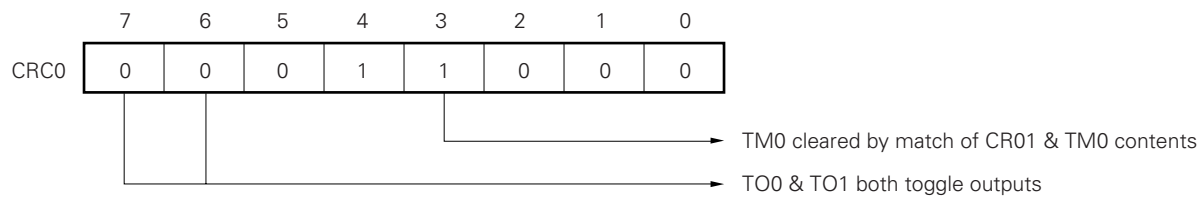
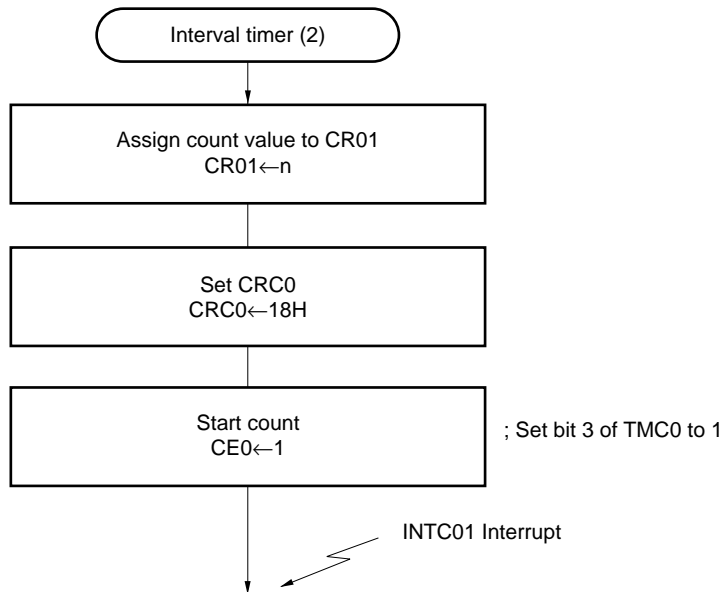
This interval timer can count within the range shown in Table 8-1 (internal system clock  $f_{xx} = 25 \text{ MHz}$ ).

The control register settings are shown in Figure 8-35, and the setting procedure in Figure 8-36.

**Figure 8-34 Interval Timer Operation (2) Timing**



**Remark**  $\text{Interval} = (n + 1) \times 8/f_{xx}$ ,  $0 \leq n \leq \text{FFFFH}$

**Figure 8-35 Control Register Settings for Interval Timer Operation (2)****Capture/compare control register 0 (CRC0)****Figure 8-36 Interval Timer Operation (2) Setting Procedure**

### 8.8.3 Pulse Width Measurement Operation

In pulse width measurement, the high-level or low-level width of external pulses input to the external interrupt request input pin (INTP3) is measured.

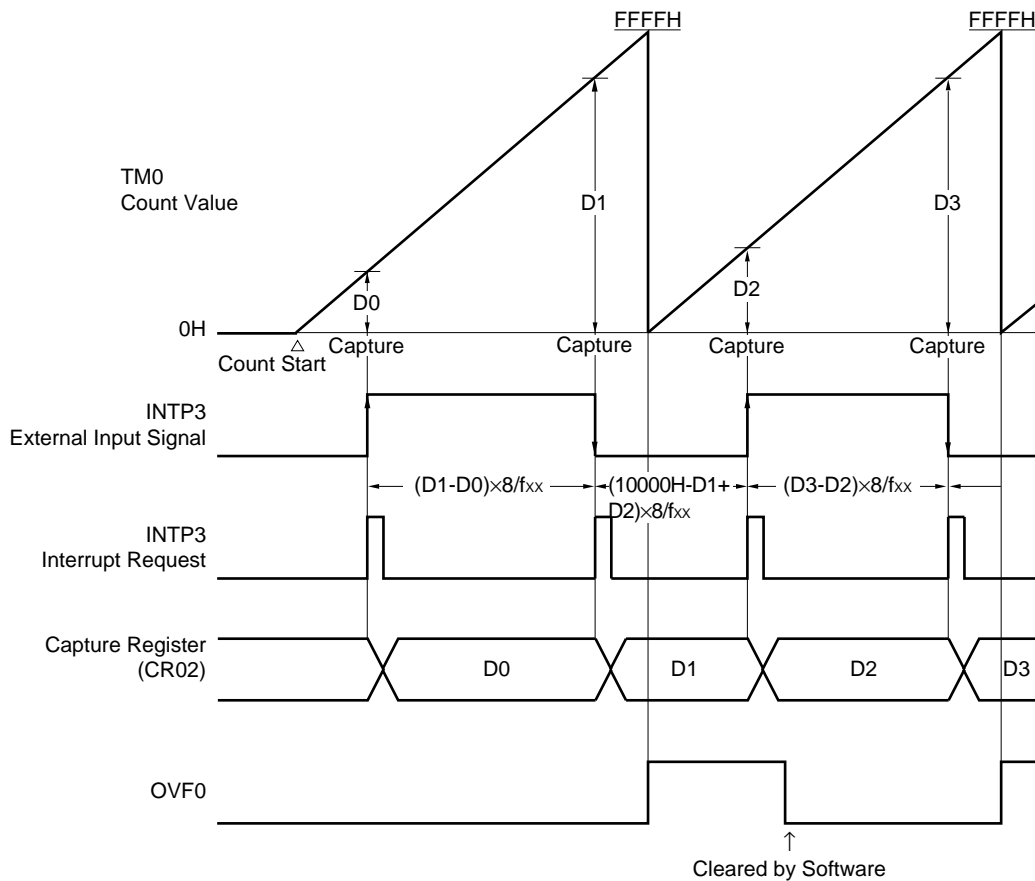
Both the high-level and low-level widths of pulses input to the INTP3 pin must be at least 3 system clocks ( $0.24 \mu\text{s}$ :  $f_{\text{CLK}} = 12.5 \text{ MHz}$ ); if shorter than this, the valid edge will not be detected and a capture operation will not be performed.

This pulse width measurement can be performed within the range shown in Table 8-3 ( $f_{\text{CLK}} = 12.5 \text{ MHz}$ ).

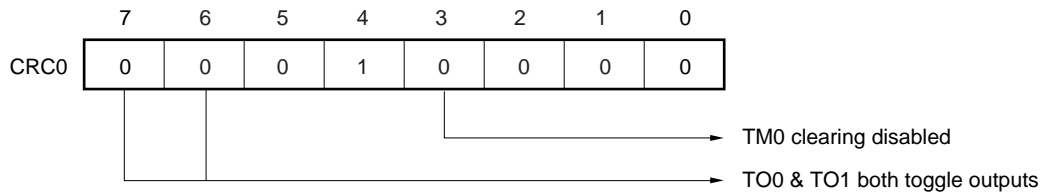
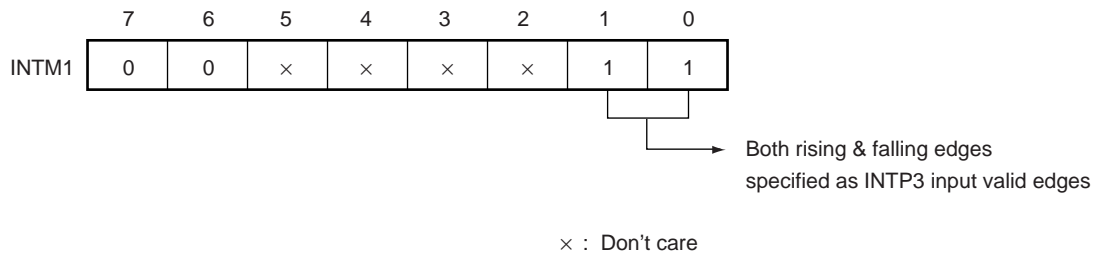
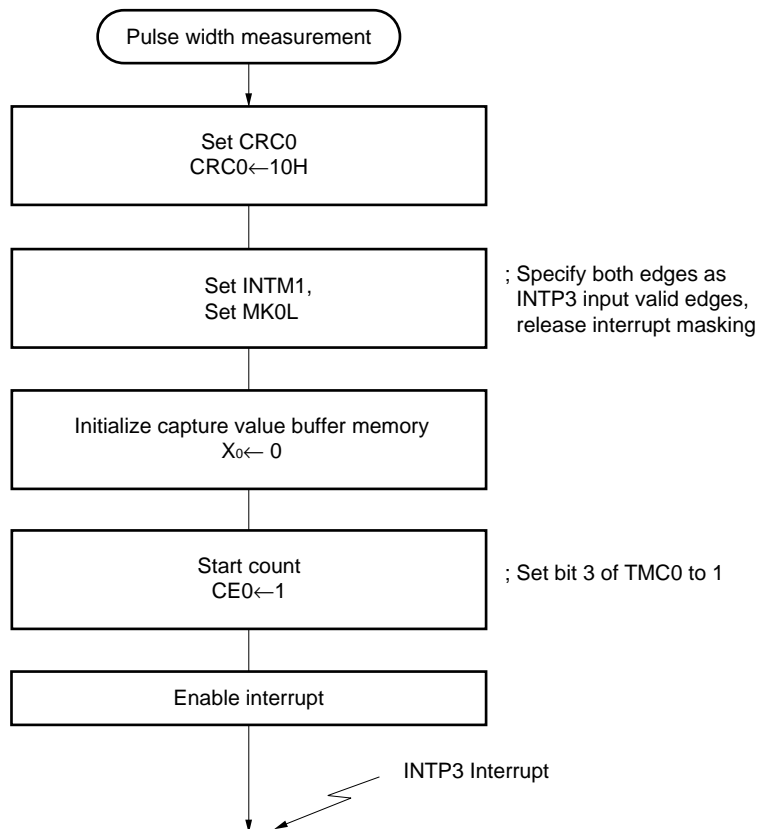
As shown in Figure 8-37, the timer register 0 (TM0) value being counted is fetched into the capture register (CR02) in synchronization with a valid edge (specified as both rising and falling edges) in the INTP3 pin input, and retained there. The pulse width is obtained from the product of the difference between the TM0 count value ( $D_n$ ) fetched into and retained in the CR02 on detection of the  $n$ th valid edge and the count value ( $D_{n-1}$ ) fetched and retained on detection of valid edge  $n-1$ , and the number of count clocks ( $x/f_{xx}$ ;  $x = 8, 16, 32, 64, 128, 256, 512, 1024, 2048$ ).

The control register settings are shown in Figure 8-38, and the setting procedure in Figure 8-39.

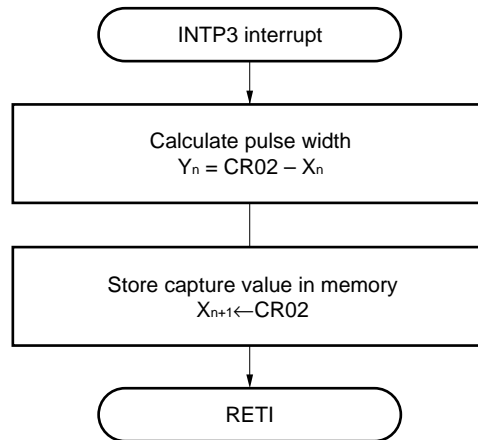
Figure 8-37 Pulse Width Measurement Timing



**Remark**  $D_n$ : TM0 count value ( $n = 0, 1, 2, \dots$ )  
 $x = 8, 16, 32, 64, 128, 256, 512, 1024, 2048$

**Figure 8-38 Control Register Settings for Pulse Width Measurement****(a) Capture/compare control register 0 (CRC0)****(b) External interrupt mode register 1 (INTM1)****Figure 8-39 Pulse Width Measurement Setting Procedure**



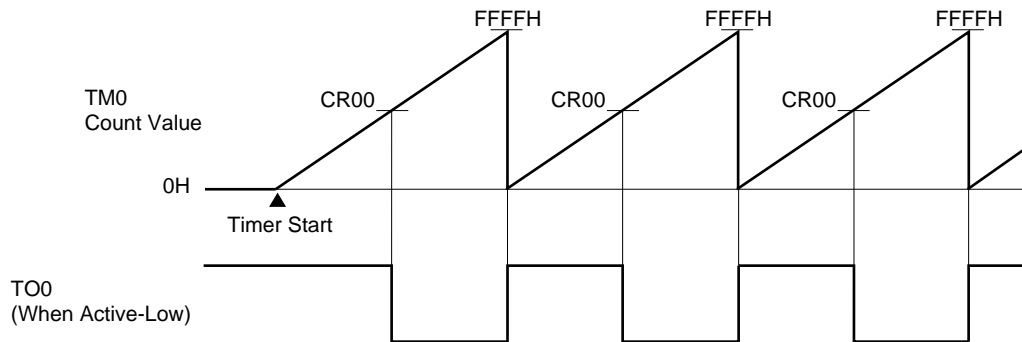
**Figure 8-40** Interrupt Request Processing that Calculates Pulse Width**8.8.4 Operation as PWM Output**

In PWM output, pulses with the duty ratio determined by the value assigned to the compare register (CR0n: n = 0, 1) are output (see **Figure 8-41**).

This PWM output duty ratio can be varied in the range 1/65536 to 65535/65536 in 1/65536 units.

Since timer register 0 (TM0) has two compare registers, two different PWM signals can be output.

The control register settings are shown in Figure 8-42, the setting procedure in Figure 8-43, and the procedure for varying the duty in Figure 8-44.

**Figure 8-41** Example of Timer/Counter 0 PWM Signal Output

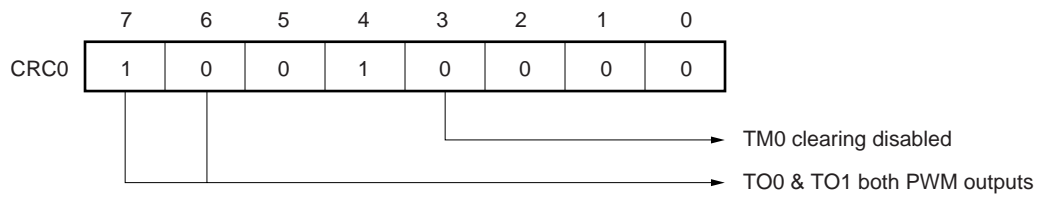
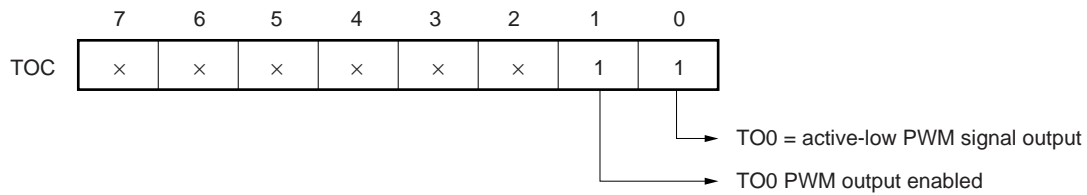
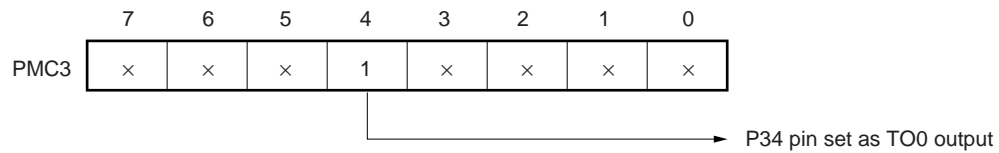
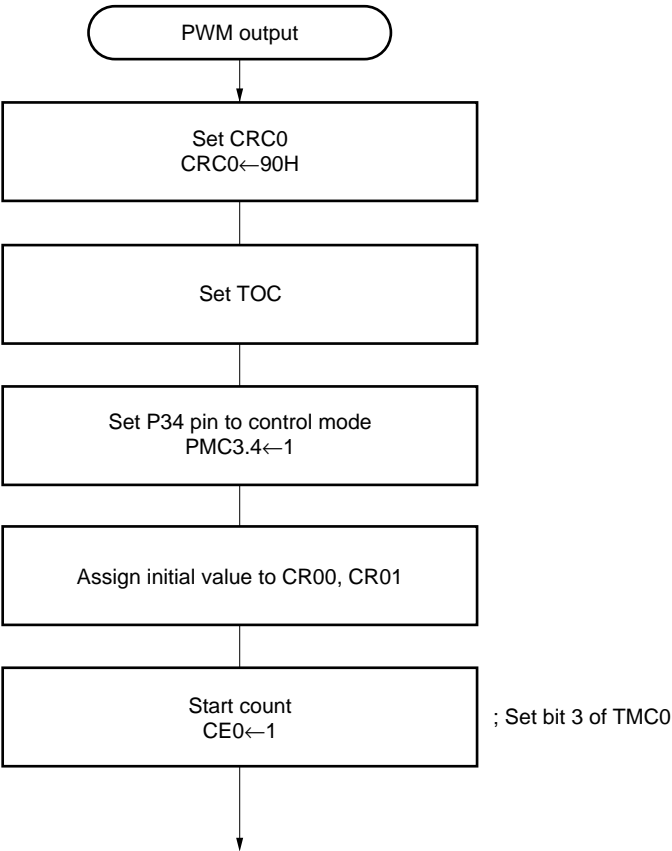
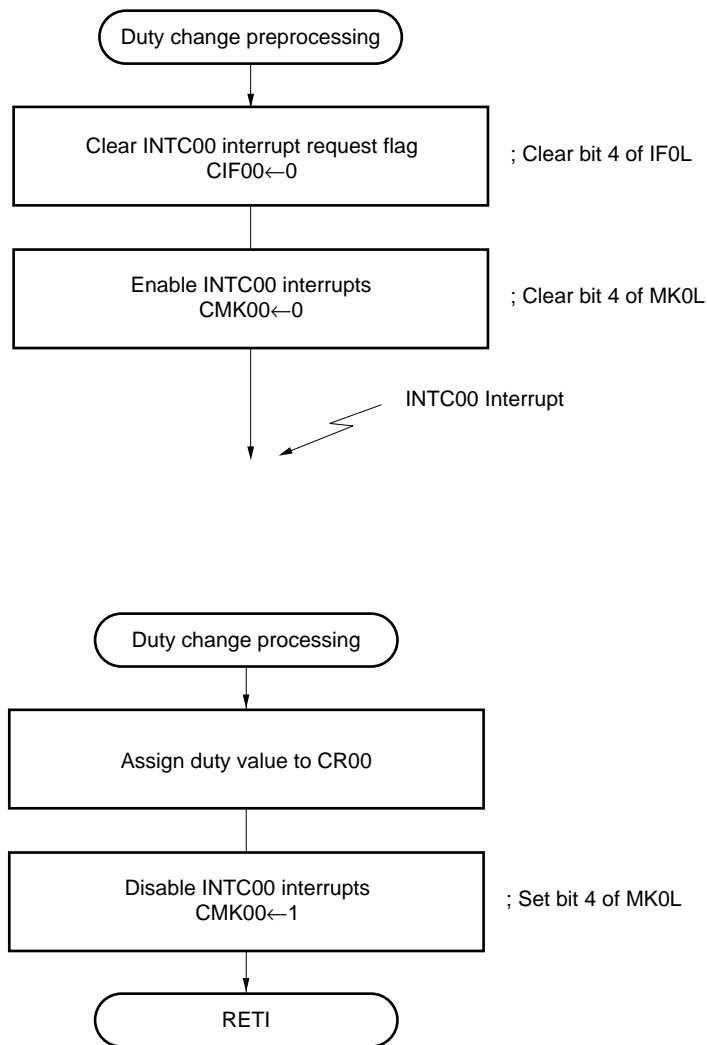
**Figure 8-42 Control Register Settings for PWM Output Operation****(a) Capture/compare control register 0 (CRC0)****(b) Timer output control register (TOC)****(c) Port 3 mode control register (PMC3)**

Figure 8-43 PWM Output Setting Procedure



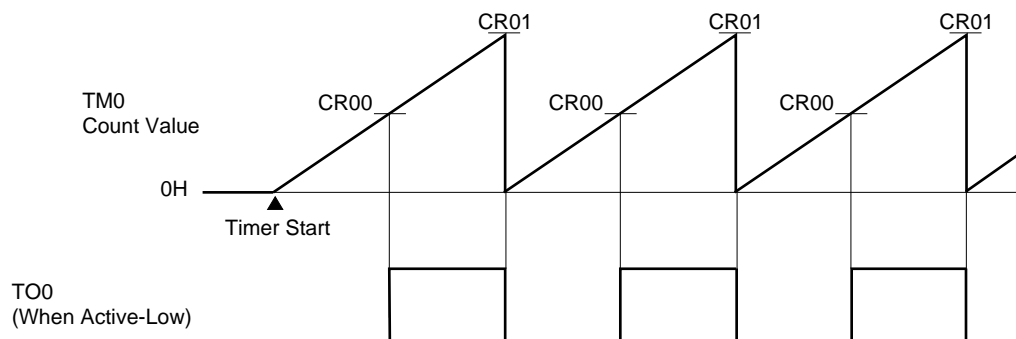
**Figure 8-44 Changing PWM Output Duty**

### 8.8.5 Operation as PPG Output

In PPG output, pulses with the cycle and duty ratio determined by the values assigned to the compare registers (CR0n: n = 0, 1) are output (see **Figure 8-45**).

The control register settings are shown in Figure 8-46, the setting procedure in Figure 8-47, and the procedure for varying the duty in Figure 8-48.

**Figure 8-45 Example of Timer/Counter 0 PPG Signal Output**



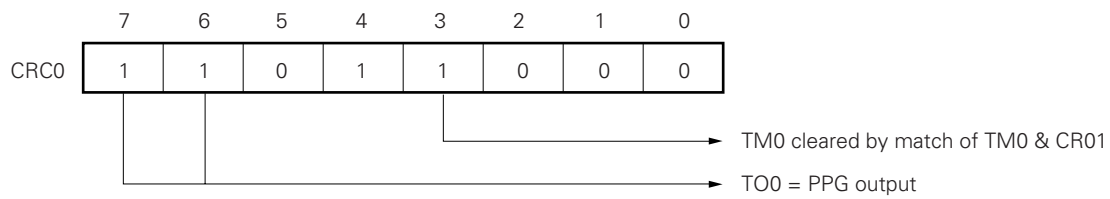
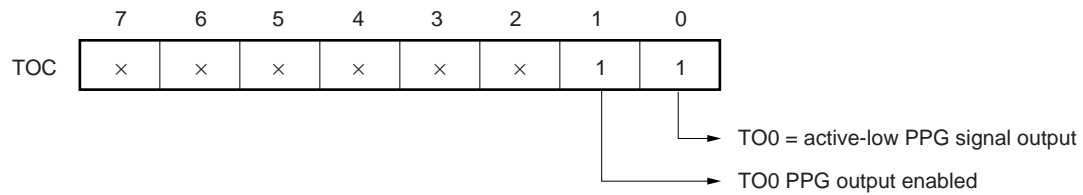
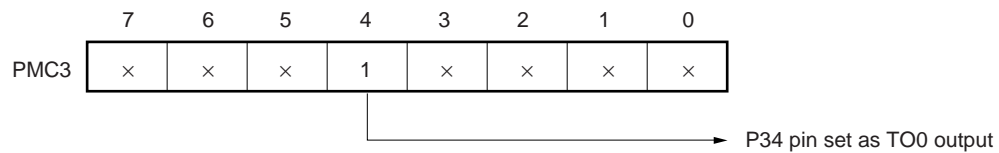
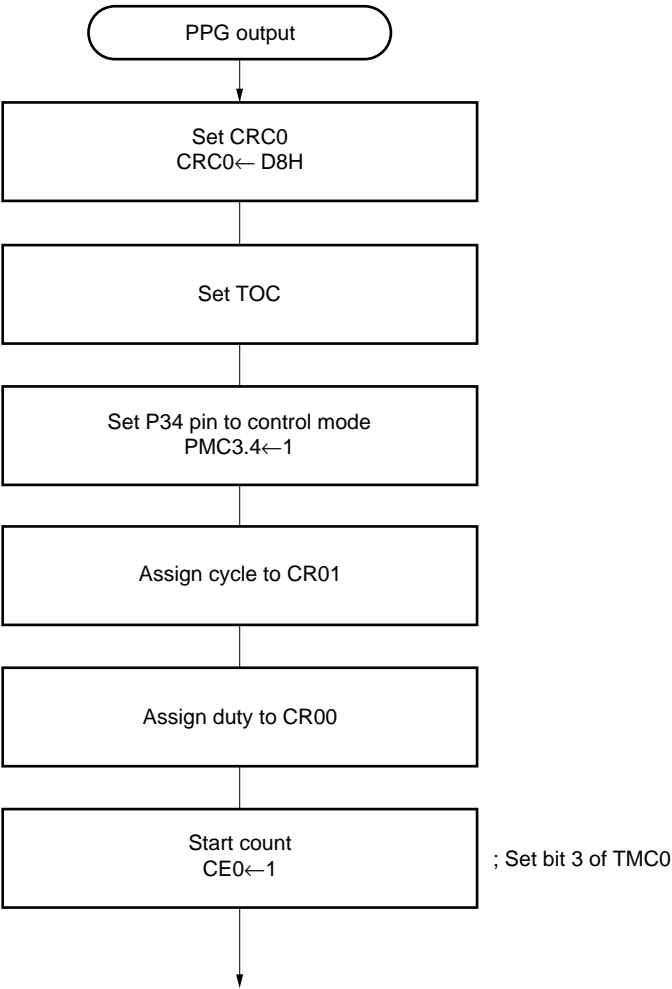
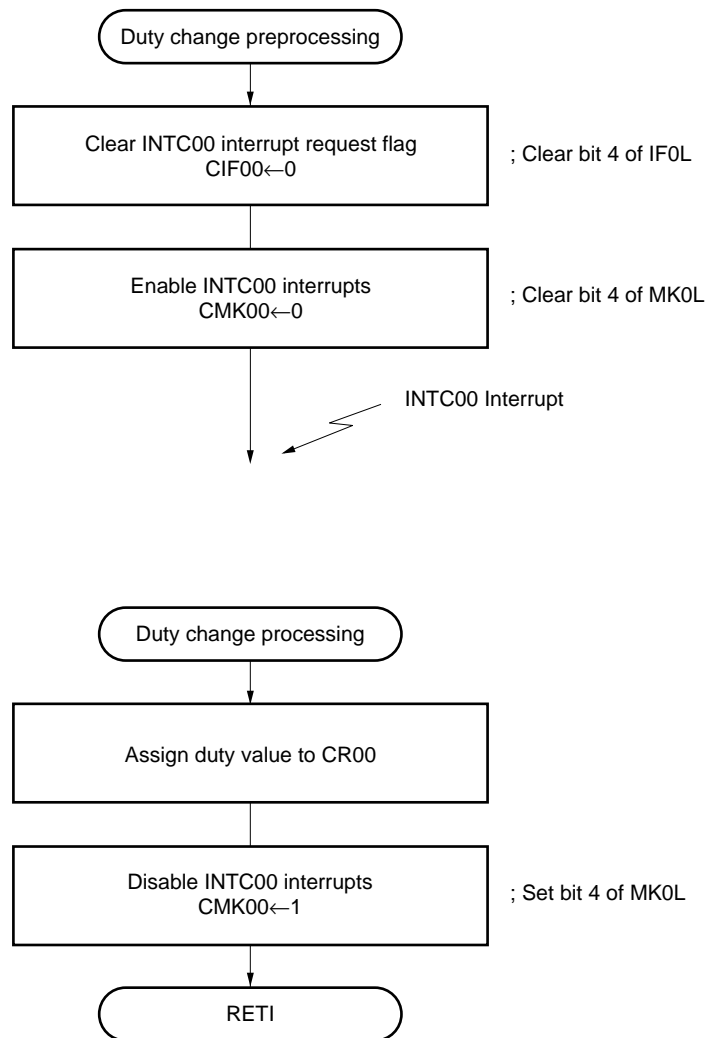
**Figure 8-46 Control Register Settings for PPG Output Operation****(a) Capture/compare control register 0 (CRC0)****(b) Timer output control register (TOC)****(c) Port 3 mode control register (PMC3)**

Figure 8-47 PPG Output Setting Procedure



**Figure 8-48 Changing PPG Output Duty**

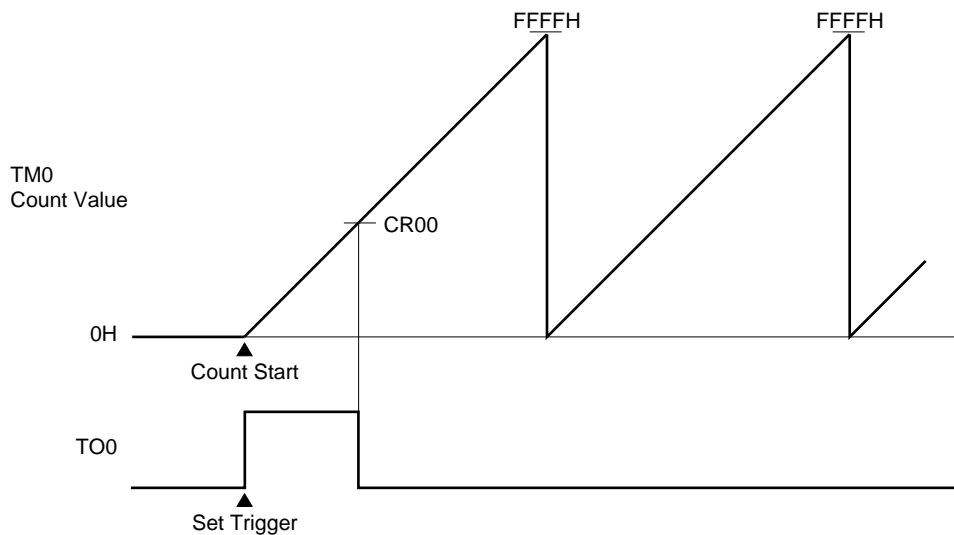


**8.8.6 Example of Software Triggered One-Shot Pulse Output**

In the software triggered one-shot pulse output mode, a one-shot pulse is output in response to a trigger activated by software (see **Figure 8-49**).

The control register settings are shown in Figure 8-50, and the setting procedure in Figure 8-51.

**Figure 8-49 Example of Timer/Counter 0 One-Shot Pulse Output**



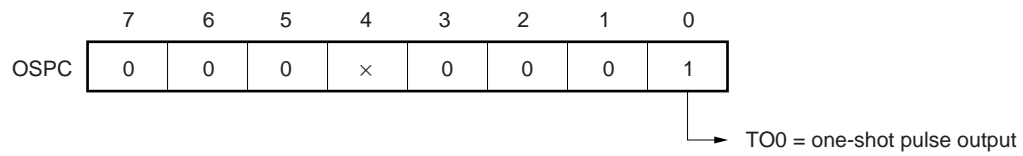
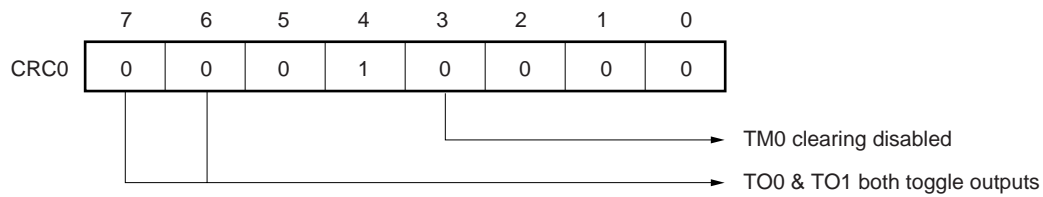
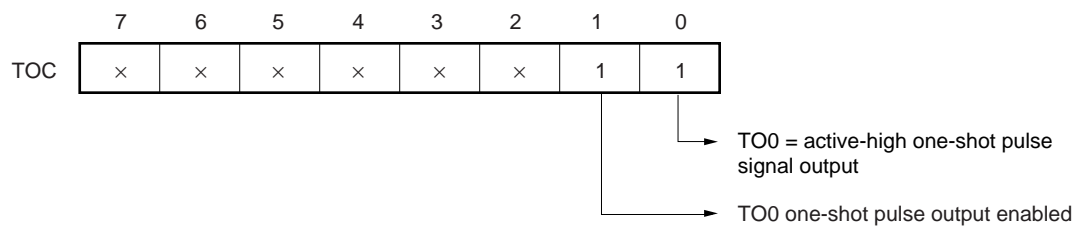
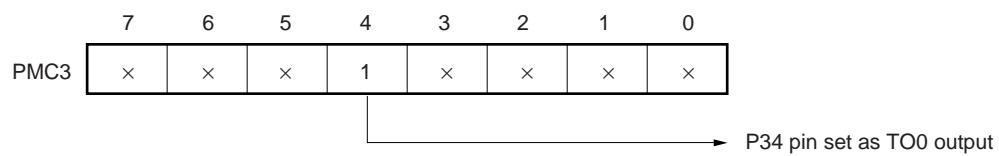
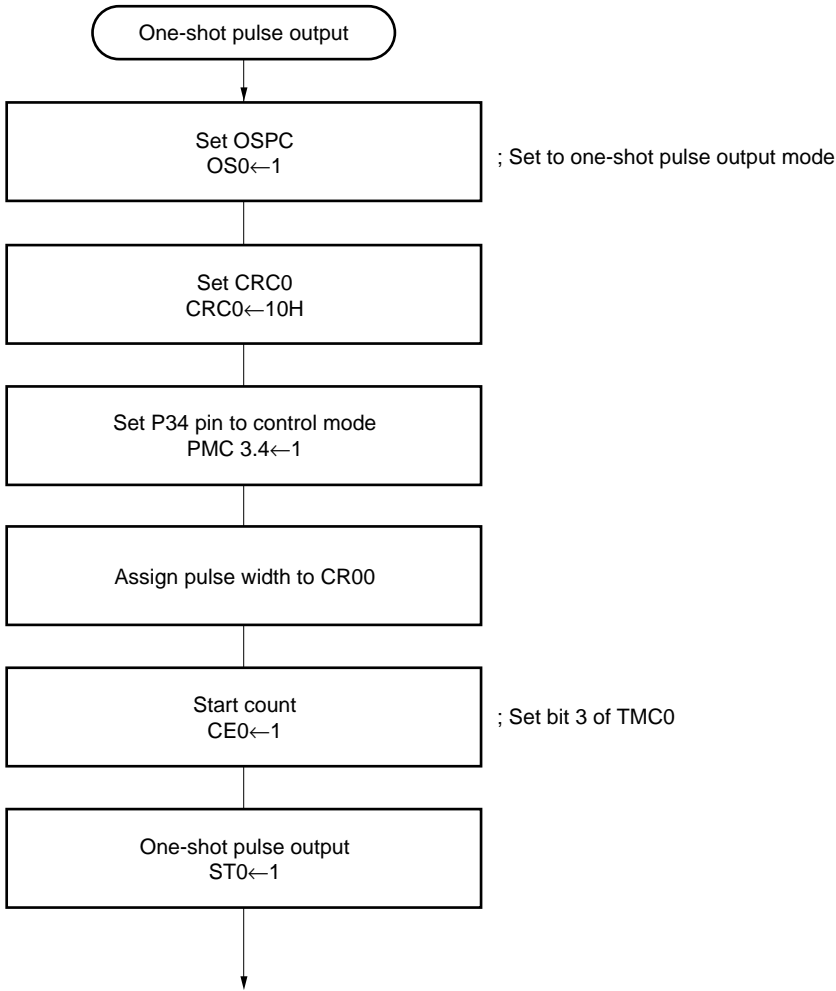
**Figure 8-50 Control Register Settings for One-Shot Pulse Output****(a) One-shot pulse output control register (OSPC)****(b) Capture/compare control register 0 (CRC0)****(c) Timer output control register (TOC)****(d) Port 3 mode control register (PMC3)**

Figure 8-51 One-Shot Pulse Output Setting Procedure



**8.9 CAUTIONS**

- (1) While timer/counter 0 is operating (while the CE0 bit of the timer control register 0 (TMC0) is set), malfunctioning may occur if the contents of the following registers are rewritten. This is because it is undefined which takes precedence in a contention the change in the hardware functions due to rewriting the register, or the change in the status because of the function before rewriting.

Therefore, be sure to stop the counter operation for the sake of safety before rewriting the contents of the following registers.

- Prescaler mode register 0 (PRM0)
- Capture/compare control register 0 (CRC0)
- Timer output control register (TOC)

- (2) If the contents of the compare register (CR0n: n = 0 or 1) match those of TM0 operation when an instruction that stops timer register 0 (TM0) operation operation is executed, the counting operation of TM0 stops, but an interrupt request is generated. In order not to generate the interrupt when stopping the operation of TM0, mask the interrupt in advance by using the interrupt mask register before stopping TM0.

**Example**

Program that may generate interrupt request

```

      :
CLR1  CE0
OR     MK0L, #30H      ← Interrupt request
      :                from timer/counter 0
                        occurs between
                        these instructions

```

Program that does not generate interrupt request

```

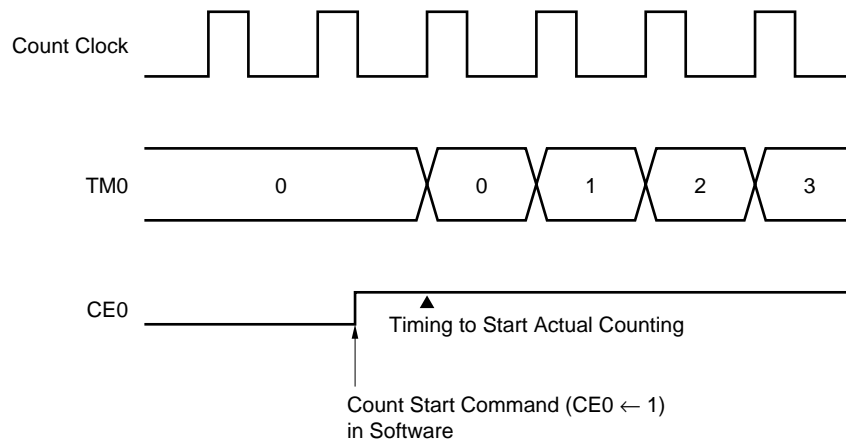
      :
OR     MK0L, #30H      ← Disables interrupt
CLR1   CE0             from timer/counter 0
CLR1   CIF00           ← Clears interrupt request
CLR1   CIF01           flag for timer/counter 0
      :

```

- (3) Up to 1 count clock is required after an operation to start timer/counter 0 ( $CE0 \leftarrow 1$ ) has been performed before timer/counter 0 actually starts (see **Figure 8-52**).

For example, when using timer/counter 0 as an interval timer, the first interval time is delayed by up to 1 clock. The second and those that follow are at the specified interval.

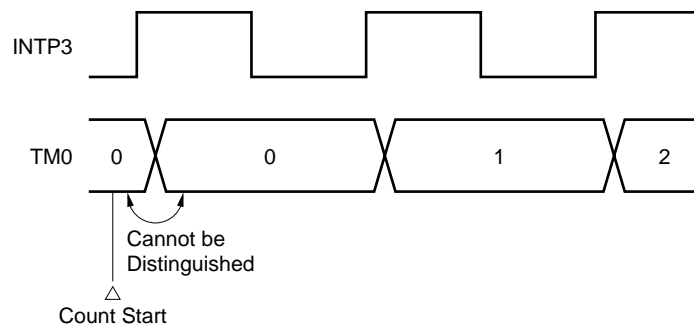
**Figure 8-52 Operation When Counting Is Started**

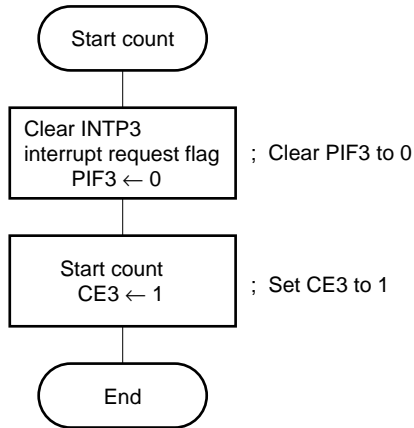
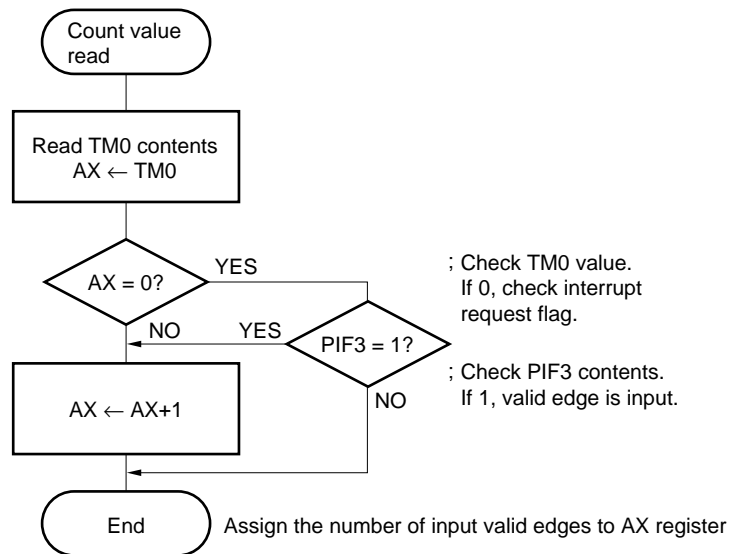


- (4) While an instruction that writes data to the compare register ( $CR0n$ :  $n = 0, 1$ ) is executed, a match between  $CR0n$ , to which the data is to be written, and timer register 0 (TM0) is not detected. For example, if the contents of  $CR0n$  do not change before and after the writing, the interrupt request is not generated even if the value of TM0 matches the value of  $CR0n$ , nor does the timer output (TON:  $n = 0, 1$ ) change.
- Write data to  $CR0n$  when timer/counter 0 is executing counting operation, in the timing that the contents of TM0 do not coincide with the value of  $CR0n$  before and after writing (e.g., immediately after an interrupt request has been generated because TM0 and  $CR0n$  have matched).
- (5) A match between timer register 0 (TM0) and compare register ( $CR0n$ :  $n = 0, 1$ ) is detected only when TM0 is incremented. Therefore, the interrupt request is not generated even if the same value as TM0 is written to  $CR0n$ , and the timer output (TON:  $n = 0, 1$ ) does not change.
- (6) If the PPG cycle is extremely short as compared with the time required to acknowledge an interrupt, the value of the  $CR0n$  cannot be rewritten by interrupt processing that is performed when the values in the timer register 0 (TM0) and the compare register ( $CR0n$ :  $n = 0, 1$ ) match. Use another method (for example, to poll the interrupt request flags in software with all the interrupts masked).

- (7) The output level of the TOn ( $n = 0, 1$ ) when the timer output is disabled ( $ENTOn = 0$ :  $n = 0, 1$ ) is the reverse value of the value set to the ALVn ( $n = 0, 1$ ) bit. Note, therefore, that an active level is output when the timer output is disabled with the PWM output function or PPG output function selected.
- (8) When using timer/counter 0 as an external event counter, the status where no valid edge is input cannot be distinguished from the status where only one valid edge has been input, by using timer register 0 (TM0) alone (see **Figure 8-53**), because the contents of TM0 are 0 in both the cases. To make a distinction, use the interrupt request flag of INTP3, as shown in Figure 8-54.

**Figure 8-53 Example Where Whether One or No Valid Edge Has Been Input Cannot Be Distinguished with External Event Counter**



**Figure 8-54 To Distinguish Whether One or No Valid Edge Has Been Input with External Event Counter****(a) Processing on starting counting****(b) Processing on reading count value**

- ★ (9) Although the timer register can be read, if counting up the timer and reading the timer register conflict, the read value of the timer register may be illegal. Therefore, do not read the timer register if at all possible. If it is necessary to read the timer register, read it two or more times in a row in software, and assume the value of the timer register is correct only when all the read values match.

[MEMO]



## CHAPTER 9 TIMER/COUNTER 1

### 9.1 FUNCTIONS

Timer/counter 1 is 16-bit or 8-bit timer/counter.

In addition to its basic functions of interval timer, pulse width measurement and event counter, timer/counter 1 can be used as a real-time output port output trigger generation timer.

#### (1) Interval timer

Generates internal interrupts at preset intervals.

**Table 9-1 Timer/Counter 1 Intervals**

Minimum Interval	Maximum Interval	Resolution
$8/f_{xx}$ (0.32 $\mu$ s)	$2^{16} \times 8/f_{xx}$ (21.0 ms)	$8/f_{xx}$ (0.32 $\mu$ s)
$16/f_{xx}$ (0.64 $\mu$ s)	$2^{16} \times 16/f_{xx}$ (41.9 ms)	$16/f_{xx}$ (0.64 $\mu$ s)
$32/f_{xx}$ (1.28 $\mu$ s)	$2^{16} \times 32/f_{xx}$ (83.9 ms)	$32/f_{xx}$ (1.28 $\mu$ s)
$64/f_{xx}$ (2.56 $\mu$ s)	$2^{16} \times 64/f_{xx}$ (168 ms)	$64/f_{xx}$ (2.56 $\mu$ s)
$128/f_{xx}$ (5.12 $\mu$ s)	$2^{16} \times 128/f_{xx}$ (336 ms)	$128/f_{xx}$ (5.12 $\mu$ s)
$256/f_{xx}$ (10.2 $\mu$ s)	$2^{16} \times 256/f_{xx}$ (671 ms)	$256/f_{xx}$ (10.2 $\mu$ s)
$512/f_{xx}$ (20.5 $\mu$ s)	$2^{16} \times 512/f_{xx}$ (1.34 s)	$512/f_{xx}$ (20.5 $\mu$ s)
$1024/f_{xx}$ (41.0 $\mu$ s)	$2^{16} \times 1024/f_{xx}$ (2.68 s)	$1024/f_{xx}$ (41.0 $\mu$ s)
$2048/f_{xx}$ (81.9 $\mu$ s)	$2^{16} \times 2048/f_{xx}$ (5.37 s)	$2048/f_{xx}$ (81.9 $\mu$ s)

( ): When  $f_{xx} = 25$  MHz

**(2) Pulse width measurement**

Detects the pulse width of the signal input to the external interrupt request input pin INTPO.

**Table 9-2 Timer/Counter 1 Pulse Width Measurement Range**

Measurable Pulse Width*	Resolution
$8/f_{xx} - 2^{16} \times 8/f_{xx}$ (0.32 $\mu$ s) (21.0 ms)	$8/f_{xx}$ (0.32 $\mu$ s)
$16/f_{xx} - 2^{16} \times 16/f_{xx}$ (0.64 $\mu$ s) (41.9 ms)	$16/f_{xx}$ (0.64 $\mu$ s)
$32/f_{xx} - 2^{16} \times 32/f_{xx}$ (1.28 $\mu$ s) (83.9 ms)	$32/f_{xx}$ (1.28 $\mu$ s)
$64/f_{xx} - 2^{16} \times 64/f_{xx}$ (2.56 $\mu$ s) (168 ms)	$64/f_{xx}$ (2.56 $\mu$ s)
$128/f_{xx} - 2^{16} \times 128/f_{xx}$ (5.12 $\mu$ s) (336 ms)	$128/f_{xx}$ (5.12 $\mu$ s)
$256/f_{xx} - 2^{16} \times 256/f_{xx}$ (10.2 $\mu$ s) (671 ms)	$256/f_{xx}$ (10.2 $\mu$ s)
$512/f_{xx} - 2^{16} \times 512/f_{xx}$ (20.5 $\mu$ s) (1.34 s)	$512/f_{xx}$ (20.5 $\mu$ s)
$1024/f_{xx} - 2^{16} \times 1024/f_{xx}$ (41.0 $\mu$ s) (2.68 s)	$1024/f_{xx}$ (41.0 $\mu$ s)
$2048/f_{xx} - 2^{16} \times 2048/f_{xx}$ (81.9 $\mu$ s) (5.37 s)	$2048/f_{xx}$ (81.9 $\mu$ s)

( ): When  $f_{xx} = 25$  MHz

- ★ \* The minimum pulse width that can be measured changes depending on the sampling clock selected by the sampling clock select register (SCS0). The minimum pulse width that can be measured is the value in the table below or above, whichever is greater.

Sampling Clock		Minimum Pulse Width
$f_{CLK}$	$f_{CLK} = f_{xx}/2$	$4/f_{CLK} = 8/f_{xx}$ (0.32 $\mu$ s)
	$f_{CLK} = f_{xx}/4$	$4/f_{CLK} = 16/f_{xx}$ (0.64 $\mu$ s)
	$f_{CLK} = f_{xx}/8$	$4/f_{CLK} = 32/f_{xx}$ (1.28 $\mu$ s)
	$f_{CLK} = f_{xx}/16$	$4/f_{CLK} = 64/f_{xx}$ (2.56 $\mu$ s)
$f_{xx}/64$		$256/f_{xx}$ (10.2 $\mu$ s)
$f_{xx}/128$		$512/f_{xx}$ (20.5 $\mu$ s)
$f_{xx}/256$		$1024/f_{xx}$ (41.0 $\mu$ s)

**(3) External event counter**

Counts the clock pulses input from the external interrupt request input pin (INTP0).

The clocks that can be input to timer/counter 1 are shown in Table 9-3.

**Table 9-3 Timer/Counter 1 Pulse Width Measurement Time**

( ): When  $f_{CLK} = 12.5 \text{ MHz}$  and  $f_{xx} = 25 \text{ MHz}$

Sampling Clock*		When Counting One Edge	When Counting Both Edges
$f_{CLK}$	Maximum frequency	$f_{CLK}/8$ (1.56 MHz)	$f_{CLK}/8$ (1.56 MHz)
	Minimum pulse width (High and low levels)	$4/f_{CLK}$ (0.32 $\mu\text{s}$ )	$4/f_{CLK}$ (0.32 $\mu\text{s}$ )
$f_{xx}/64$	Maximum frequency	$f_{xx}/512$ (48.8 kHz)	$f_{xx}/512$ (48.8 kHz)
	Minimum pulse width (High and low levels)	$256/f_{xx}$ (10.2 $\mu\text{s}$ )	$256/f_{xx}$ (10.2 $\mu\text{s}$ )
$f_{xx}/128$	Maximum frequency	$f_{xx}/1024$ (24.4 kHz)	$f_{xx}/1024$ (24.4 kHz)
	Minimum pulse width (High and low levels)	$512/f_{xx}$ (20.5 $\mu\text{s}$ )	$512/f_{xx}$ (20.5 $\mu\text{s}$ )
$f_{xx}/256$	Maximum frequency	$f_{xx}/2048$ (12.2 kHz)	$f_{xx}/2048$ (12.2 kHz)
	Minimum pulse width (High and low levels)	$1024/f_{xx}$ (41.0 $\mu\text{s}$ )	$1024/f_{xx}$ (41.0 $\mu\text{s}$ )

\* Selected by means of the sampling clock selection register (SCS0)

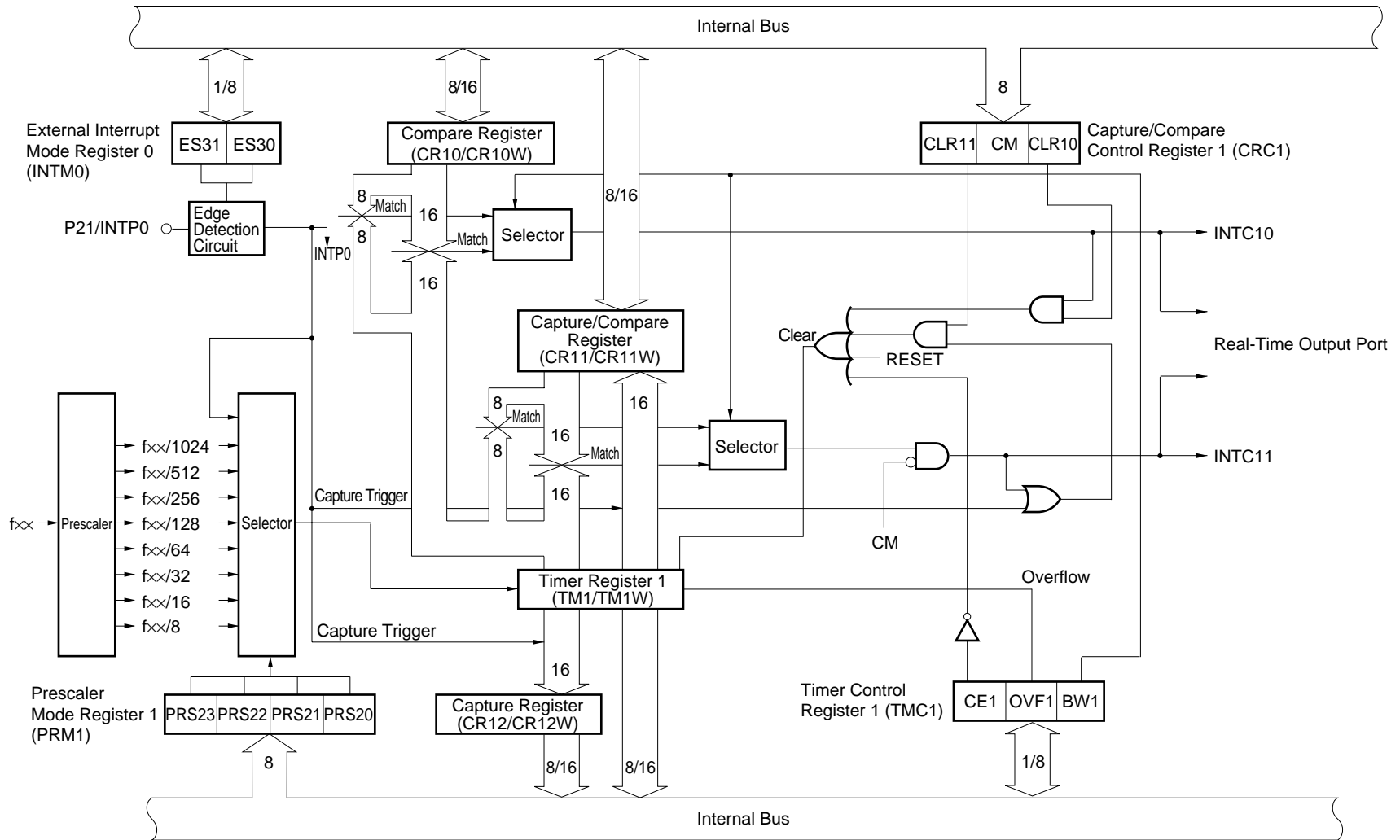
## 9.2 CONFIGURATION

Timer/counter 1 consists of the following registers:

- Timer register (TM1/TM1W)  $\times 1$
- Compare register (CR10/CR10W)  $\times 1$
- Capture/compare register (CR11/CR11W)  $\times 1$
- Capture register (CR12/CR12W)  $\times 1$

The block diagram of timer/counter 1 is shown in Figure 9-1.

Figure 9-1 Timer/Counter 1 Block Diagram



**(1) Timer register 1 (TM1/TM1W)**

TM1/TM1W is a timer register that counts up using the count clock specified by the low-order 4 bits of prescaler mode register 1 (PRM1).

The count operation can be specified to stop or enable, and an 8-bit operation mode (TM1)/16-bit operation mode (TM1W) can be selected, by means of timer control register 1 (TMC1).

TM1/TM1W can be read only with an 8/16-bit manipulation instruction. When  $\overline{\text{RESET}}$  is input, TM1/TM1W is cleared to 00H and the count is stopped.

★ **Caution** Although the timer register can be read, if counting up the timer and reading the timer register conflict, the read value of the timer register may be illegal. Therefore, do not read the timer register if at all possible.

If it is necessary to read the timer register, read it two or more times in a row in software, and assume the value of the timer register is correct only when all the read values match.

**(2) Compare register (CR10/CR10W)**

CR10/CR10W is an 8/16-bit register that holds the value that determines the interval timer operation cycle.

If the contents of the CR10/CR10W match the values of TM1/TM1W, an interrupt request (INTC10) is generated. This match signal is also a real-time output port trigger signal. Also, the count value can be cleared by a match.

This compare register operates as CR10 in the 8-bit operating mode, and CR10W in the 16-bit operating mode.

CR10/CR10W can be read or written to with an 8/16-bit manipulation instruction. The contents of this register are undefined after  $\overline{\text{RESET}}$  input.

**(3) Capture/compare register (CR11/CR11W)**

CR11/CR11W is an 8/16-bit register that can be specified as a compare register for detecting a match with the TM1/TM1W count value or a capture register for capturing the TM1/TM1W count value according to the setting of capture/compare control register 1 (CRC1).

This capture/compare register operates as CR11 in the 8-bit operating mode, and CR11W in the 16-bit operating mode.

CR11/CR11W can be read or written to with an 8/16-bit manipulation instruction. The contents of this register are undefined after  $\overline{\text{RESET}}$  input.

**(a) When specified as compare register**

CR11/CR11W functions as an 8/16-bit register that holds the value that determines the interval timer operation cycle.

An interrupt request (INTC11) is generated by a match between the contents of the CR11/CR11W register and the contents of TM1/TM1W.

Also, the count value can be cleared by a match. This match signal is also a real-time output port trigger signal.

**(b) When specified as capture register**

CR11/CR11W functions as an 8/16-bit register that captures the contents of TM1/TM1W in synchronization with the input of a valid edge (capture trigger) on the external interrupt request input pin (INTP0).

The contents of the CR11/CR11W are retained until the next capture trigger is generated. TM1/TM1W can be cleared after a capture operation.

**(4) Capture register (CR12/CR12W)**

CR12/CR12W is an 8/16-bit register that captures the contents of TM1/TM1W.

The capture operation is synchronized with the input of a valid edge (capture trigger) on the external interrupt request input pin (INTP0). The contents of the CR12/CR12W are retained until the next capture trigger is generated.

This capture/compare register operates as CR12 in the 8-bit operating mode, and CR12W in the 16-bit operating mode. CR12/CR12W can be read only with an 8/16-bit manipulation instruction. The contents of this register are undefined after RESET input.

**(5) Edge detection circuit**

The edge detection circuit detects an external input valid edge.

When the valid edge set by external interrupt mode register 0 (INTM0) is detected in the INTP0 pin input, the external interrupt request (INTP0), a capture trigger and a count clock of the external event are generated (see **Figure 19-1** for details of the INTM0).

**(6) Prescaler**

The prescaler generates the count clock from the internal system clock. The clock generated by this prescaler is selected by the selector, and is used as the count clock by the timer register 1 (TM1/TM1W) to perform count operations.

**(7) Selector**

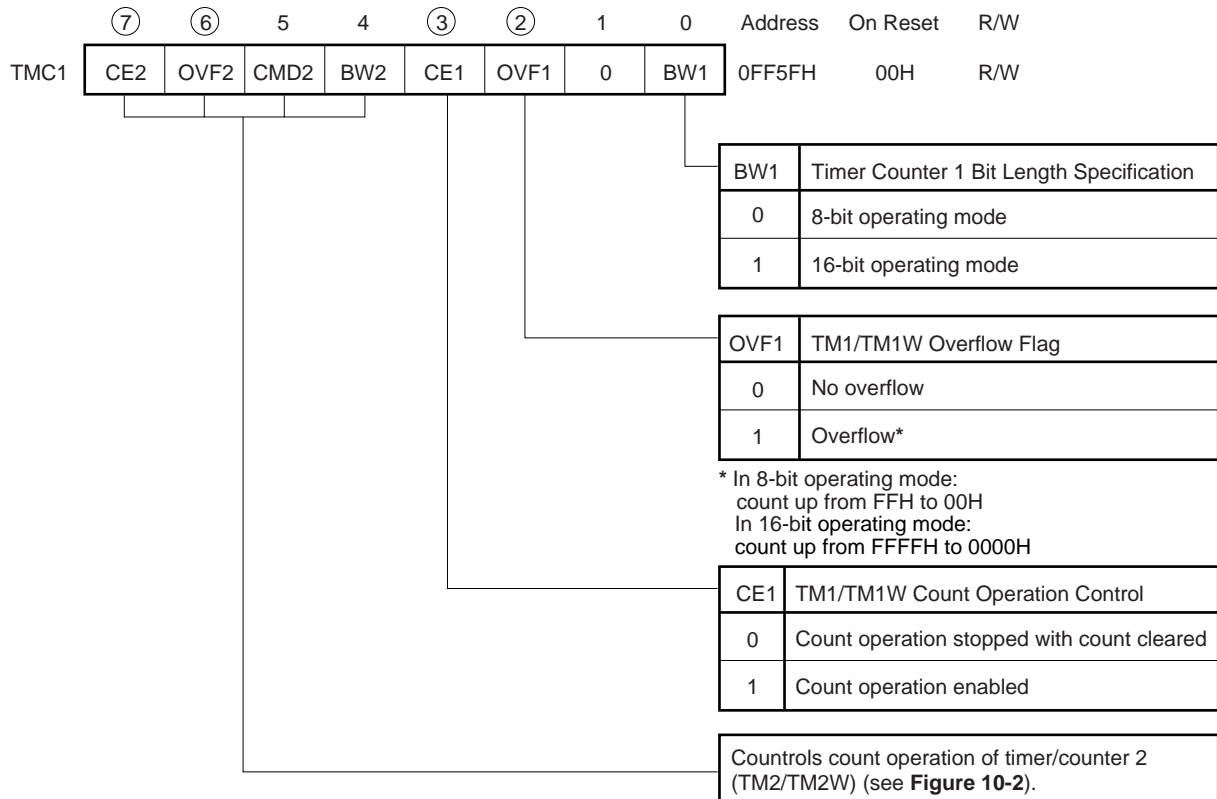
The selector selects a signal resulting from dividing the internal clock or the edge detected by the edge detection circuit as the count clock of timer register 1 (TM1/TM1W).

**9.3 TIMER/COUNTER 1 CONTROL REGISTERS****(1) Timer control register 1 (TMC1)**

TMC1 controls the timer/counter 1, TM1/TM1W, count operation by the low-order 4 bits (the high-order 4 bits control the count operation of timer/counter 2, TM2/TM2W).

TMC1 can be read or written to with an 8-bit manipulation instruction or bit manipulation instruction. The format of the TMC1 is shown in Figure 9-2.

RESET input clears TMC1 to 00H.

**Figure 9-2 Timer Control Register 1 (TMC1) Format**

**Remark** The OVF1 bit is reset by software only.

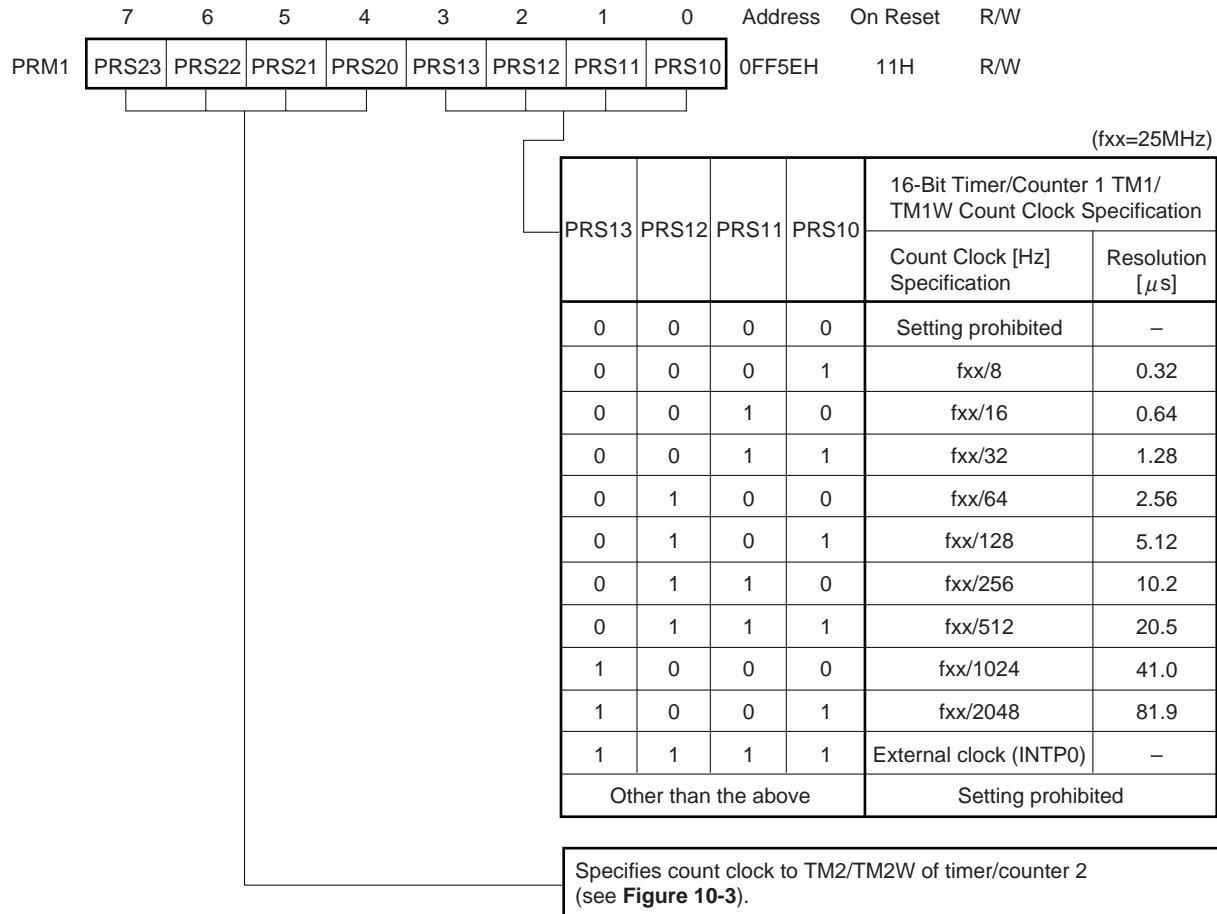
**(2) Prescaler mode register 1 (PRM1)**

The count clock of PRM1 to timer/counter 1, TM1/TM1W, is specified by the low-order 4 bits (the high-order 4 bits specify the count clock to timer/counter 2, TM2/TM2W).

PRM1 can be read or written to with an 8-bit manipulation instruction. The format of the PRM1 is shown in Figure 9-3.

RESET input sets PRM1 to 11H.

**Figure 9-3 Prescaler Mode Register 1 (PRM1) Format**



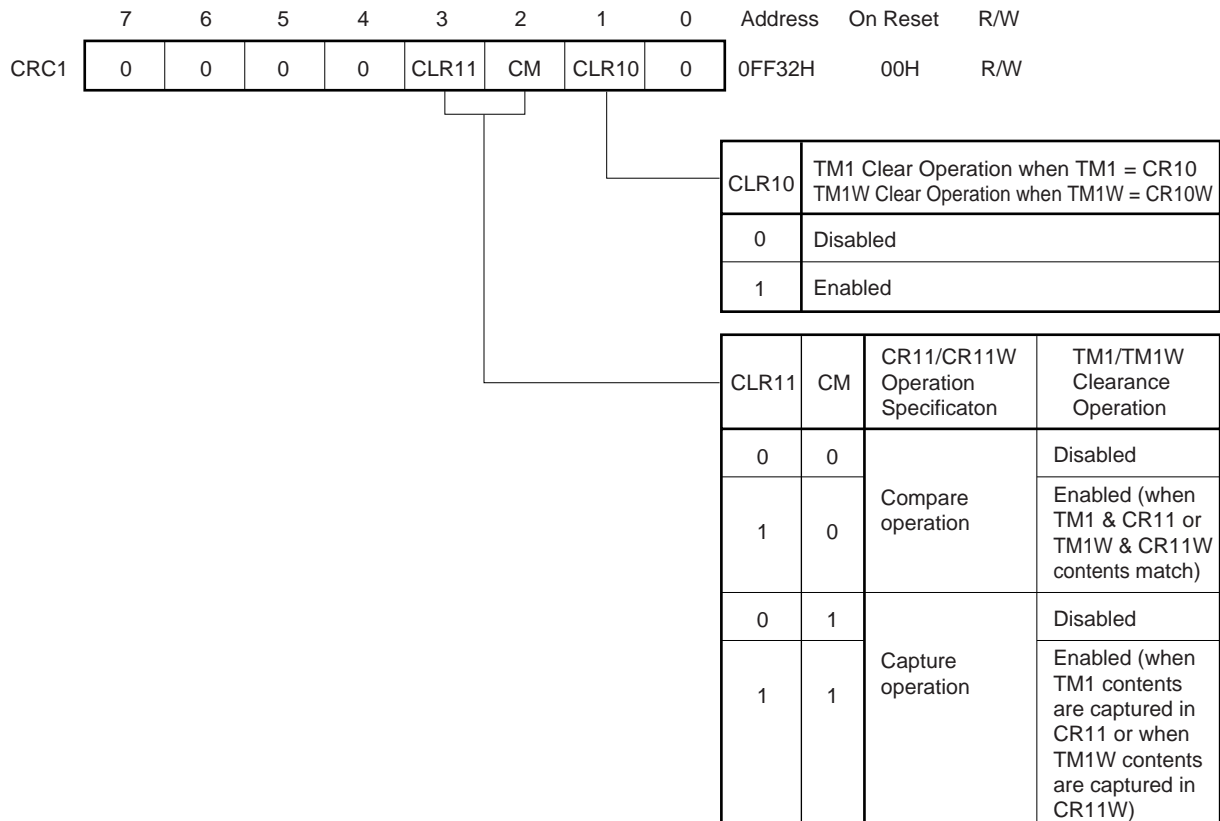
**Remark** fxx: X1 input frequency or oscillation frequency



**(3) Capture/compare control register 1 (CRC1)**

The CRC1 specifies the operation of the capture/compare register (CR11/CR11W) and the enabling condition for a timer register 1 (TM1/TM1W) clear operation.

CRC1 can be read or written to with an 8-bit manipulation instruction. The format of the CRC1 is shown in Figure 9-4. RESET input clears CRC1 to 00H.

**Figure 9-4 Capture/Compare Control Register 1 (CRC1) Format**

## 9.4 TIMER REGISTER 1 (TM1) OPERATION

### 9.4.1 Basic Operation

8-bit operating mode/16-bit operating mode control can be performed for timer/counter 1 by means of bit 0 (BW1) of timer control register 1 (TMC1).\*

In the timer/counter 1 count operation, an up-count is performed using the count clock specified by the low-order 4 bits of prescaler mode register 1 (PRM1).

Count operation enabling/disabling is controlled by bit 3 (CE1) of TMC1 (timer/counter 1 operation control is performed by the low-order 4 bits of the TMC1). When the CE1 bit is set (1) by software, the contents of TM1 are cleared to 0H on the first count clock, and then the up-count operation is performed.

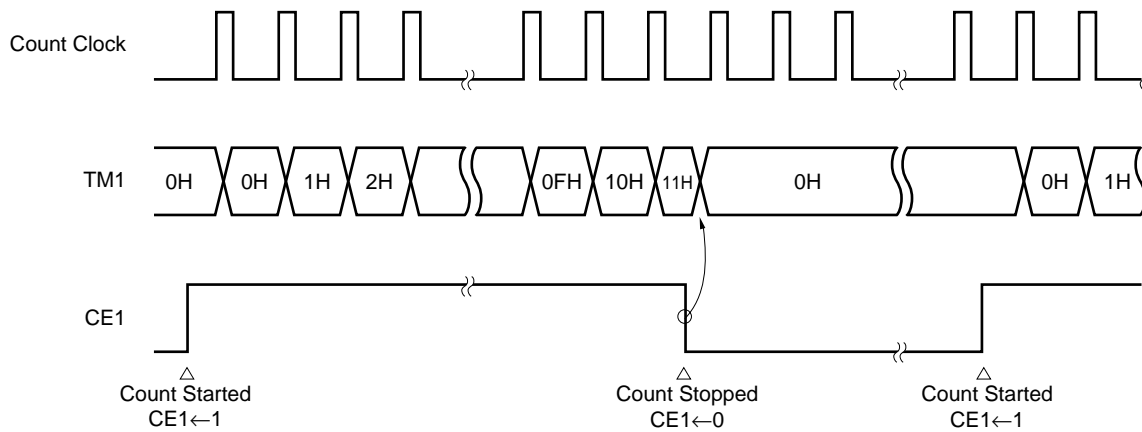
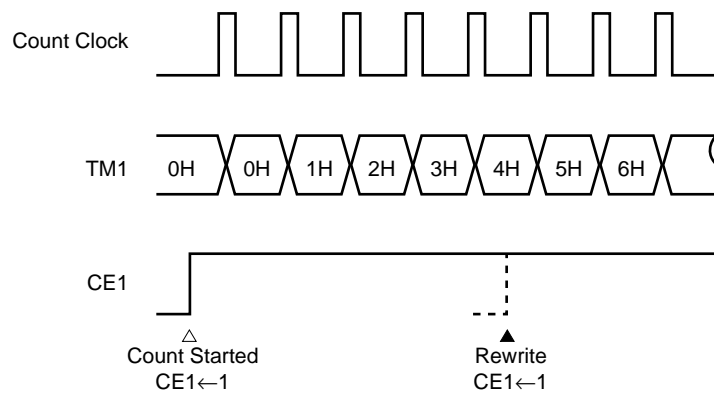
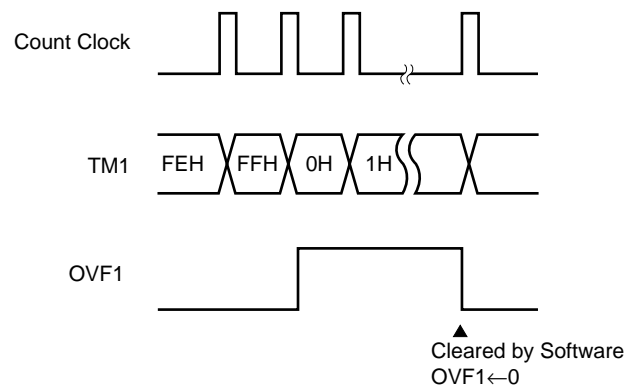
When the CE1 bit is cleared (0), TM1 becomes 0H immediately, and capture operations and match signal generation are stopped.

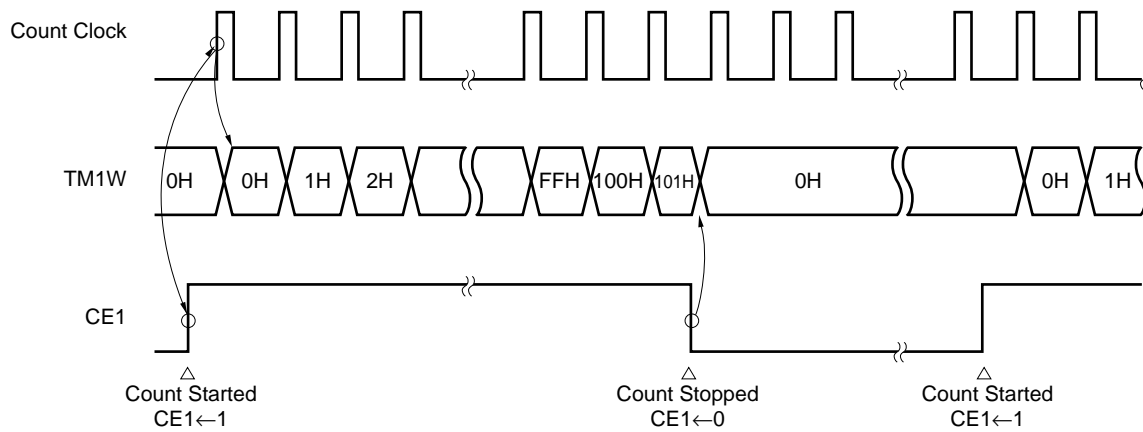
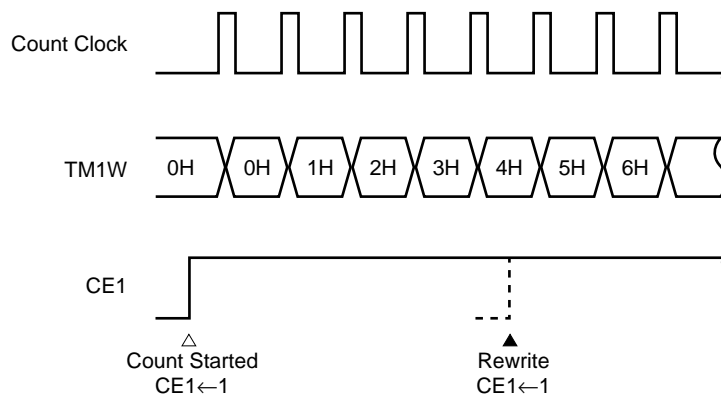
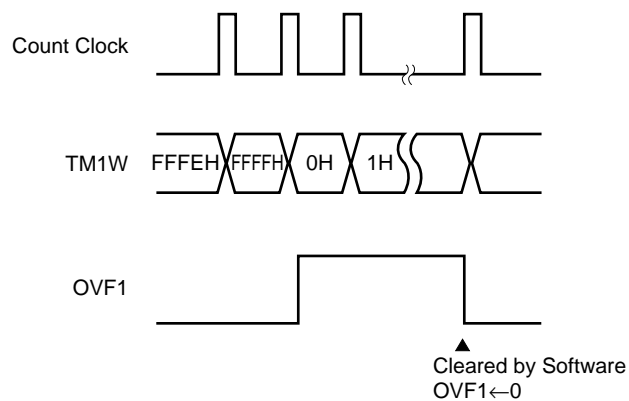
If the CE1 bit is set (1) again when it is already set (1), TM1 continues the count operation without being cleared.

If the count clock is input when TM1 is FFH in 8-bit operating mode and when TM1W is FFFFH in 16-bit operating mode, TM1/TM1W becomes 0H. In this case, OVF1 bit is set. OVF1 bit is cleared by software only. The count operation is continued.

When  $\overline{\text{RESET}}$  is input, TM1 is cleared to 0H, and the count operation is stopped.

- \* Unless otherwise specified, the functions of timer register 1 in the 8-bit operating mode are described hereafter. In the 16-bit operating mode, TM1, CR10, CR11, and CR12 operate as TM1W, CR10W, CR11W, and CR12W respectively.

**Figure 9-5 Basic Operation in 8-Bit Operating Mode (BW1 = 0)****(a) Count started → count disabled → count started****(b) When "1" is written to the CE1 bit again after the count starts****(c) Operation when TM1 = FFH**

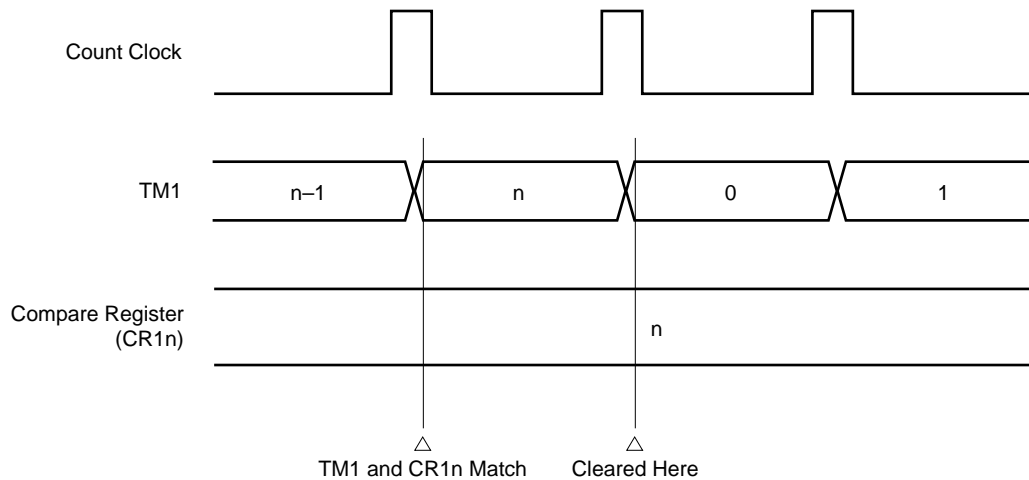
**Figure 9-6 Basic Operation in 16-Bit Operating Mode (BW1 = 1)****(a) Count started → count disabled → count started****(b) When "1" is written to the CE1 bit again after the count starts****(c) Operation when TM1W = FFFFH**

### 9.4.2 Clear Operation

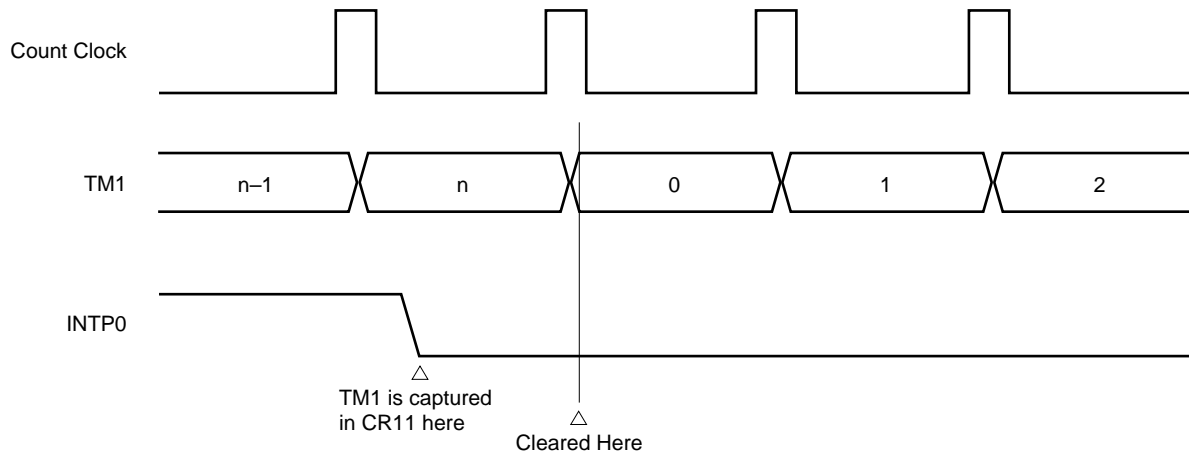
#### (1) Clear operation after match with compare register and after capture operation

Timer register 1 (TM1) can be cleared automatically after a match with the compare register (CR1n:  $n = 0, 1$ ) and a capture operation. When a clearance source arises, TM1 is cleared to 0H on the next count clock. Therefore, even if a clearance source arises, the value at the point at which the clearance source arose is retained until the next count clock arrives.

**Figure 9-7 TM1 Clearance by Match With Compare Register (CR10, CR11)**

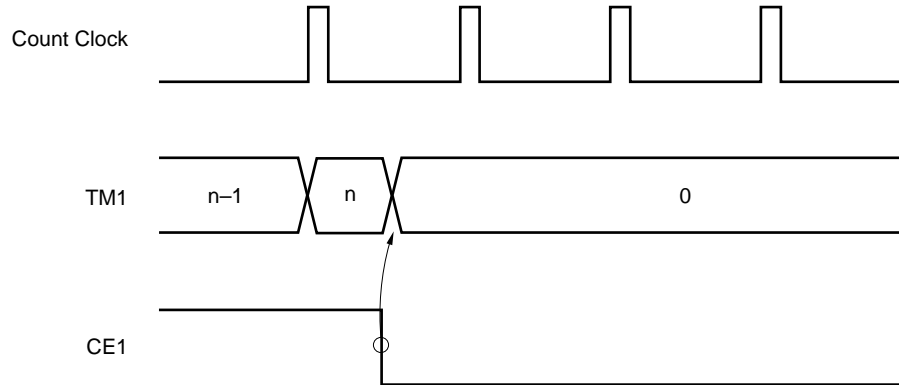
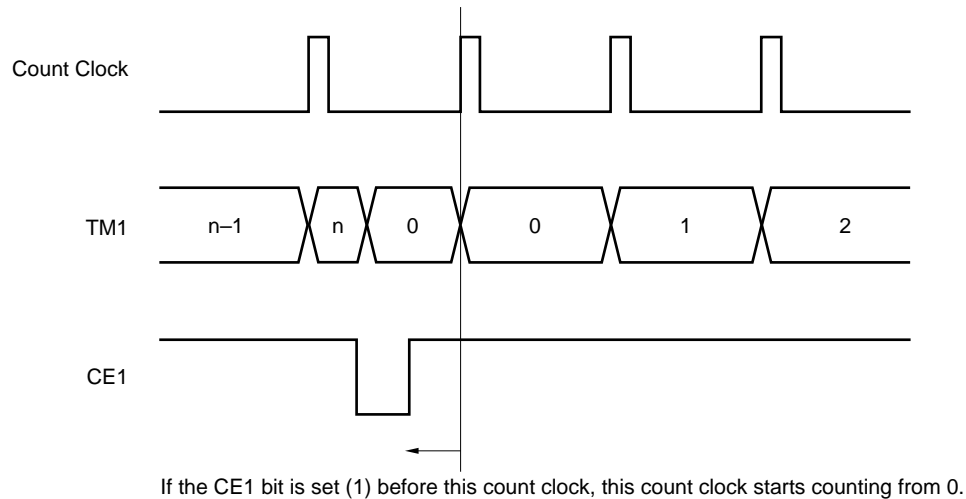
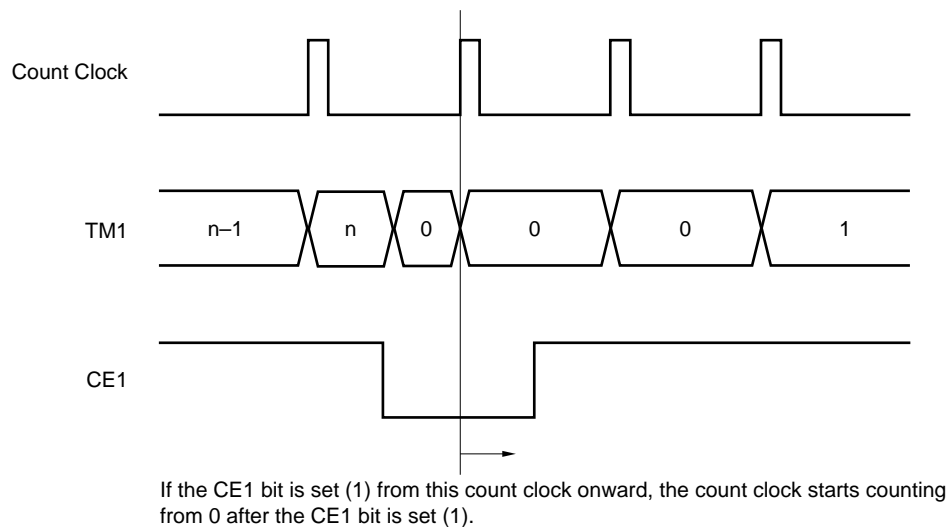


**Figure 9-8 TM1 Clearance after Capture Operation**



#### (2) Clear operation by CE1 bit of timer control register 1 (TMC1)

Timer register 1 (TM1) is also cleared when the CE1 bit of TMC1 is cleared (0) by software. The clear operation is performed immediately after the clearance (0) of the CE1 bit.

**Figure 9-9 Clear Operation When CE1 Bit is Cleared (0)****(a) Basic operation****(b) Restart before count clock is input after clearance****(c) Restart after count clock is input after clearance**

## 9.5 EXTERNAL EVENT COUNTER FUNCTION

Timer/counter 1 can count clock pulses input from the external interrupt request input pin (INTP0) pin.

No special selection method is needed for the external event counter operating mode. When the timer register 1 (TM1) count clock is specified as external clock input by the setting of the low-order 4 bits of prescaler mode register 1 (PRM1), TM1 operates as an external event counter.

The maximum frequency of the external clock pulse that can be counted by the external event counter is determined by the sampling clock select register (SCS0) as shown in Table 9-4.

The maximum frequency is the same when both the edges of the INTP0 input are counted and when only one edge is counted.

The pulse width of the INTP0 input must be three or more sampling clocks selected by SCS0, regardless of whether the level is high or low. If the width is shorter than this, the pulse may not be counted.

Figure 9-10 shows the timing of the external event count by timer/counter 1.

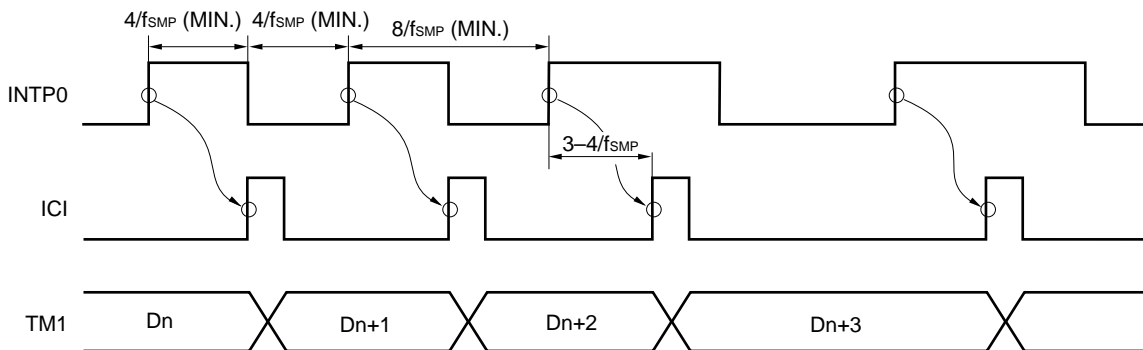
★ **Table 9-4 Maximum Input Frequency and Minimum Input Pulse Width That Can be Counted as Events**

( ):  $f_{xx} = 25 \text{ MHz}$ ,  $f_{CLK} = 12.5 \text{ MHz}$

Sampling Clock Selected by SCS0	Maximum Input Frequency	Minimum Pulse Width
$f_{CLK}$	$f_{CLK}/8$ (1.56 MHz)	$4/f_{CLK}$ (0.32 $\mu\text{s}$ )
$f_{CLK}/64$	$f_{CLK}/512$ (24.4 kHz)	$256/f_{xx}$ (10.2 $\mu\text{s}$ )
$f_{CLK}/128$	$f_{CLK}/1024$ (12.2 kHz)	$512/f_{xx}$ (20.5 $\mu\text{s}$ )
$f_{CLK}/256$	$f_{CLK}/2048$ (6.10 kHz)	$1024/f_{xx}$ (41.0 $\mu\text{s}$ )

★ **Figure 9-10 Timing of External Event Count by Timer/Counter 1 (1/2)**

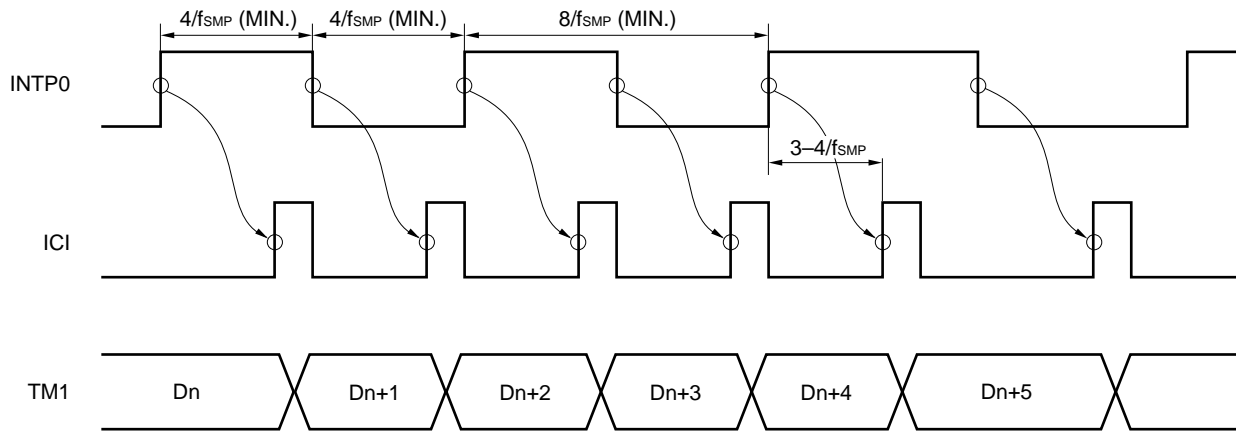
### (1) Counting one edge (maximum frequency = $f_{CLK}/8$ )



- Remarks**
1. ICI: INTP0 input signal after passing through edge detection circuit
  2.  $f_{SMP}$  is selected by the sampling clock selection register (SCS0).

★

Figure 9-10 Timing of External Event Count by Timer/Counter 1 (2/2)

(2) Counting both edges (maximum frequency =  $f_{CLK}/8$ )

- Remarks**
1. ICI: INTPO input signal after passing through edge detection circuit
  2.  $f_{SMP}$  is selected by the sampling clock selection register (SCS0).

The TM1 count operation is controlled by the CE1 bit of the timer control register 1 (TMC1) in the same way as with the basic operation.

When the CE1 bit is set (1) by software, the contents of TM1 are set to 0H and the up-count operation is started on the initial count clock.

When the CE1 bit is cleared (0) by software during a TM1 count operation, the contents of TM1 are set to 0H immediately and the stopped state is entered. The TM1 count operation is not affected if the CE1 bit is set (1) by software again when it is already set (1).



**Caution** When timer/counter 1 is used as an external event counter, it is not possible to distinguish between the case where there is no valid edge input at all and the case where there is a single valid edge input using the timer register 1 (TM1) alone (see Figure 9-11), since the contents of TM1 are 0 in both cases. If it is necessary to make this distinction, the INTP0 interrupt request flag should be used. An example is shown in Figure 9-12.

**Figure 9-11** Example of the Case Where the External Event Counter Does Not Distinguish Between One Valid Edge Input and No Valid Edge Input

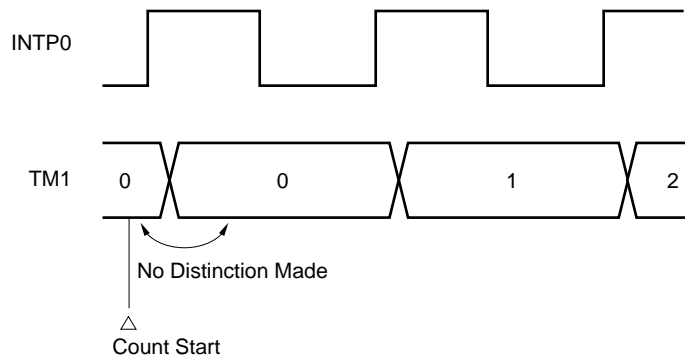
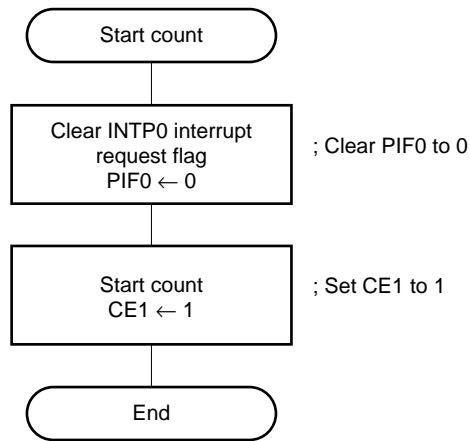
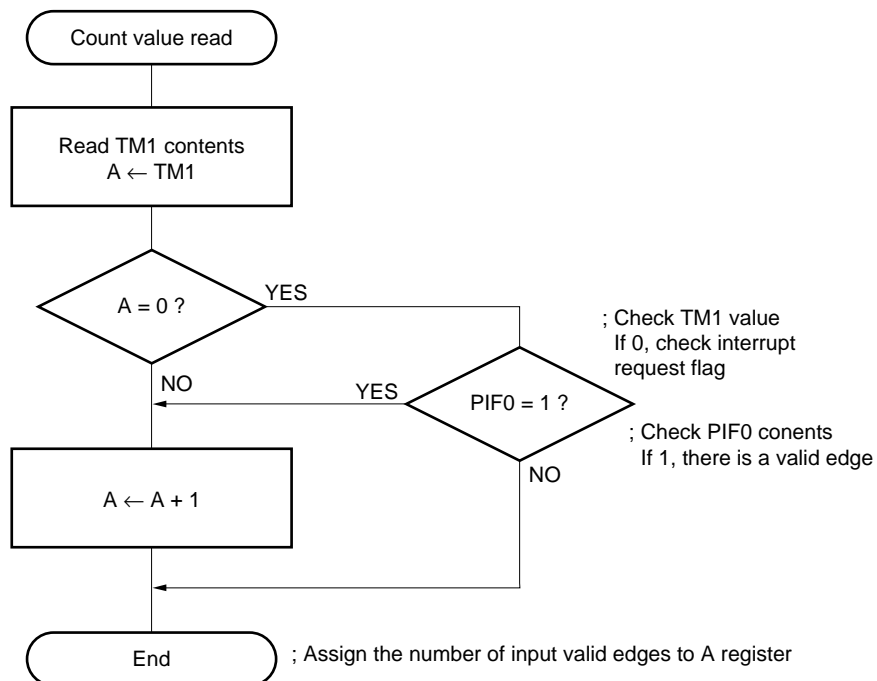


Figure 9-12 To Distinguish Whether One or No Valid Edge Has Been Input with External Event Counter

## (a) Processing when count is started



## (b) Processing when count value is read



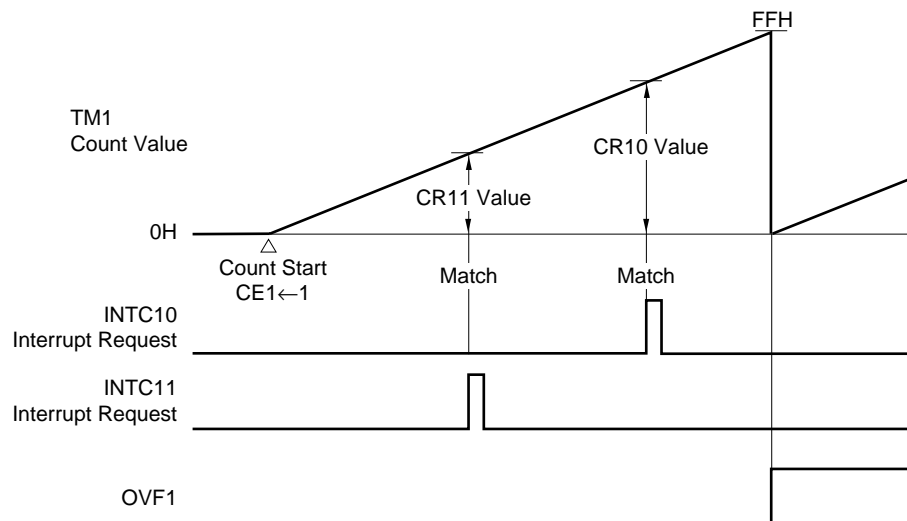
**9.6 COMPARE REGISTER, CAPTURE/COMPARE REGISTER, AND CAPTURE REGISTER OPERATION****9.6.1 Compare Operations**

Timer/counter 1 performs compare operations in which the value assigned to a compare register (CR10), capture/compare register (CR11), specified for compare operation is compared with the timer register 1 (TM1) count value.

If the count value of TM1 matches the preset value of the CR10, or the CR11 as the result of the count operation, an interrupt request signal (INTC10 or INTC11) is generated.

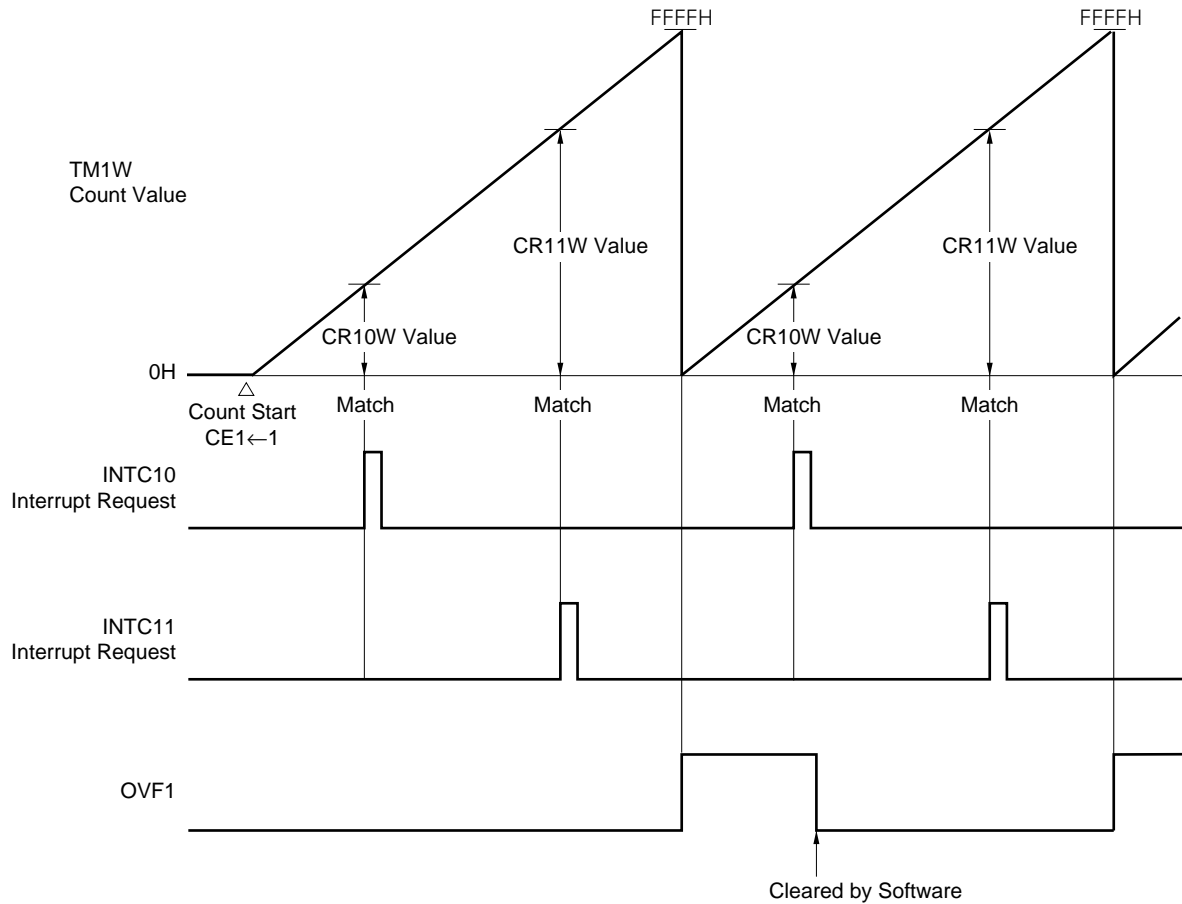
After a match with the CR10 or CR11 value, the TM1 contents can be cleared, and the timer functions as an interval timer that repeatedly counts up to the value set in the CR10 or CR11.

**Figure 9-13 Compare Operation in 8-Bit Operating Mode**



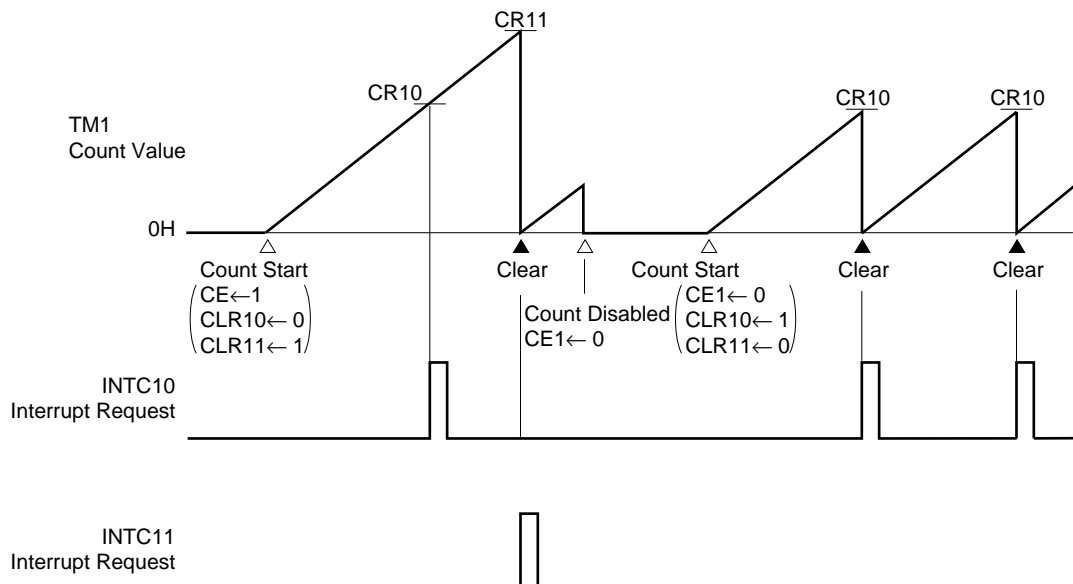
**Remark** CLR10 = 0, CLR11 = 0, CM = 0, BW1 = 0

Figure 9-14 Compare Operation in 16-Bit Operating Mode



**Remark** CLR10 = 0, CLR11 = 0, BW1 = 1

Figure 9-15 TM1 Clearance after Match Detection



### 9.6.2 Capture Operations

Timer/counter 1 performs capture operations in which the timer register 1 (TM1) count value is fetched into the capture register in synchronization with an external trigger, and retained there.

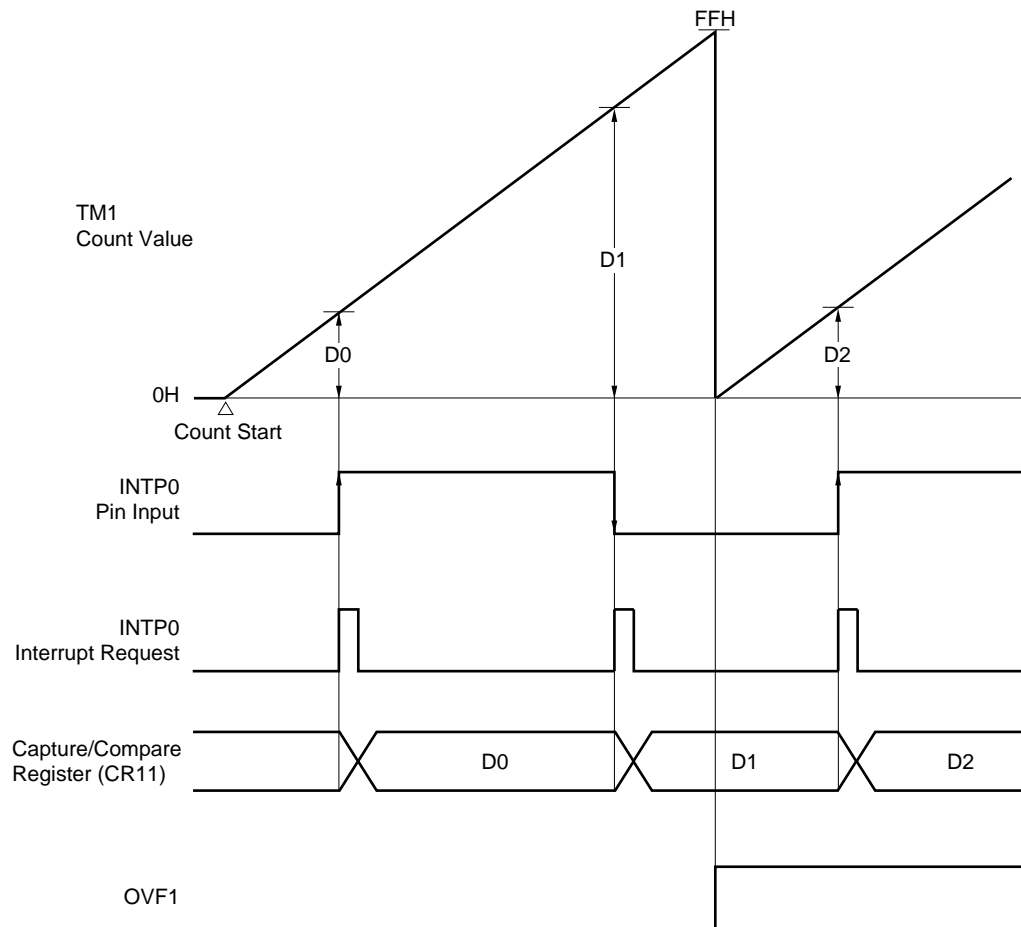
A valid edge detected from the input of the external interrupt request input pin (INTP0) is used as the external trigger (capture trigger). The count value of TM1 in the process of being counted is fetched into the capture register (CR12), or the capture/compare register (CR11) when a capture operation is specified, in synchronization with the capture trigger, and is retained there. The contents of the CR11 and CR12 are retained until the next capture trigger is generated.

The capture trigger valid edge is set by means of external interrupt mode register 0 (INTM0). If both rising and falling edges are set as capture triggers, the width of pulses input from off-chip can be measured, and if a capture trigger is generated by a single edge, the input pulse cycle can be measured.

See **Figure 19-1** in **CHAPTER 19 EDGE DETECTION FUNCTION** for details of the INTM0 format.

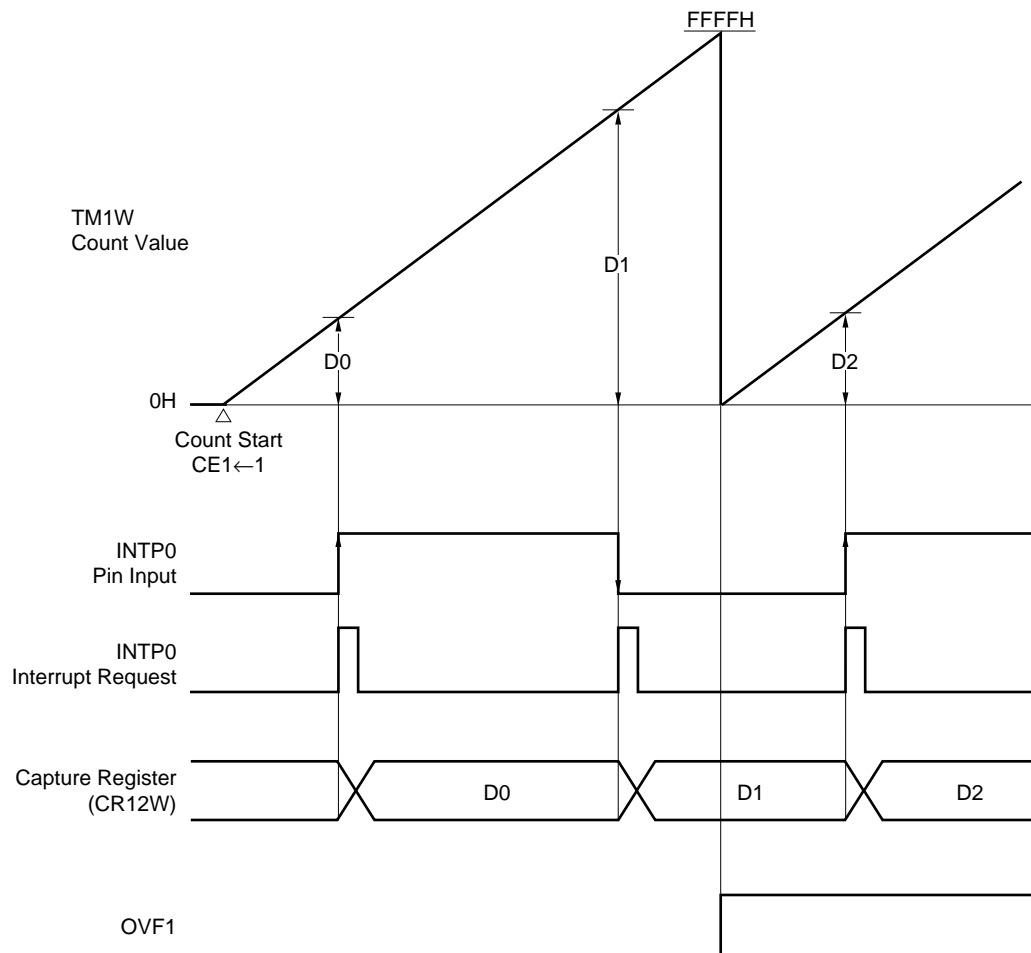
When CR11 is used as a capture register, TM1 can be cleared as soon as the contents of TM1 have been captured to CR11 by capture trigger.

**Figure 9-16 Capture Operation in 8-Bit Operating Mode**

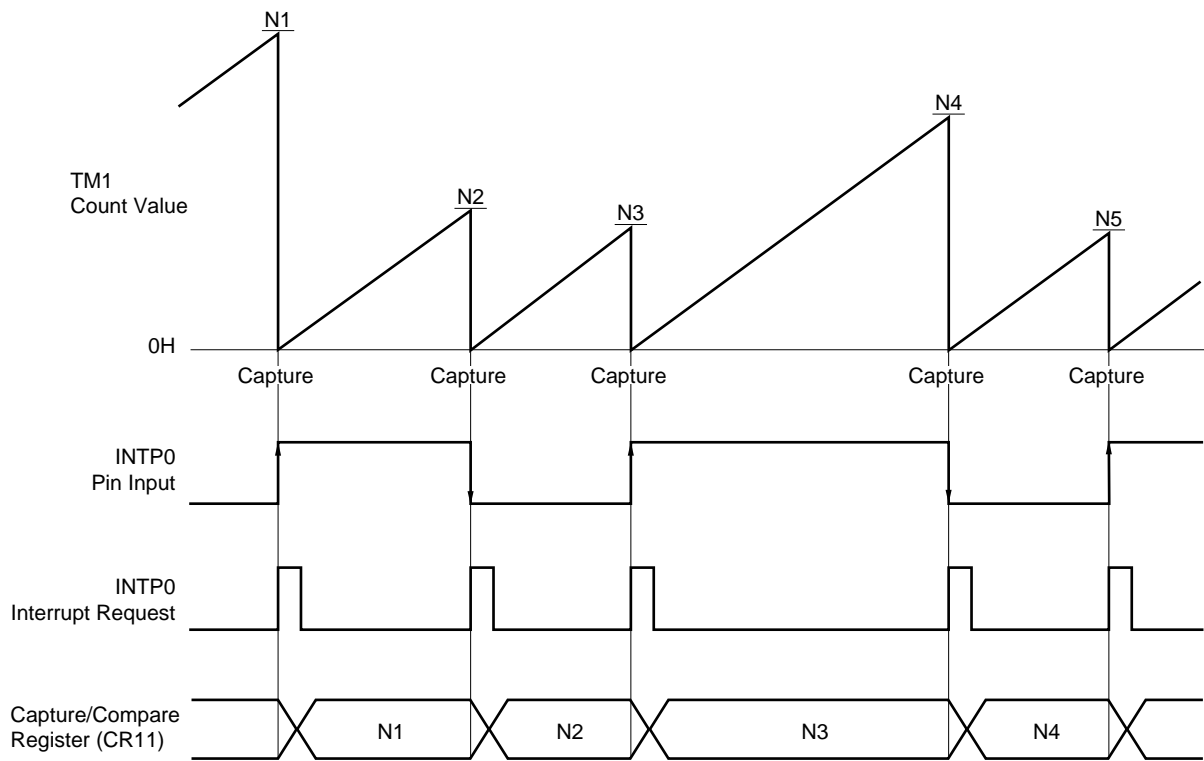


**Remark** Dn: TM1 count value (n = 0, 1, 2, ...)

CLR10 = 0, CLR11 = 0, CM = 1, BW1 = 0

**Figure 9-17 Capture Operation in 16-Bit Operating Mode**

**Remark** Dn: TM1W count value (n = 0, 1, 2, ...)  
 CLR10 = 0, CLR11 = 0, CM = 1, BW1 = 1

**Figure 9-18 TM1 Clearance after Capture Operation**

**Remark** NI: TM1 count value ( $n = 0, 1, 2, \dots$ )  
 CLR10 = 0, CLR11 = 1, CM = 1

## 9.7 EXAMPLES OF USE

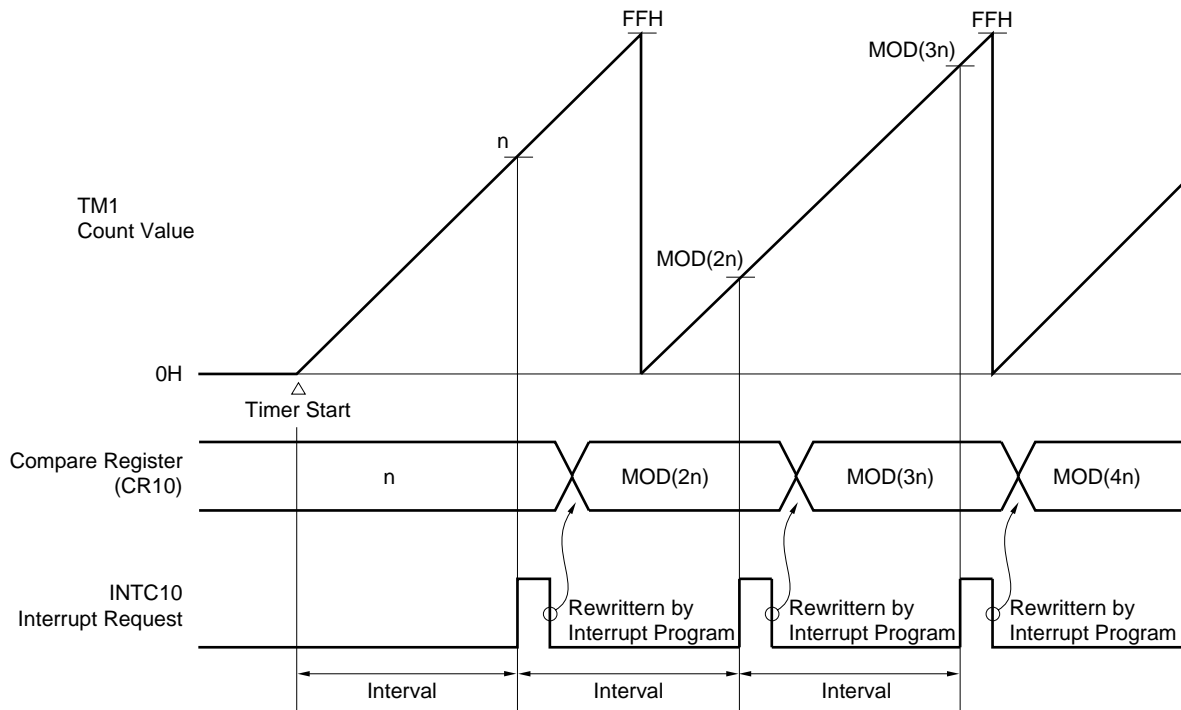
### 9.7.1 Operation as Interval Timer (1)

When timer register 1 (TM1) is made free-running and a fixed value is added to the compare register (CR1n:  $n = 0, 1$ ) in the interrupt processing routine, TM1 operates as an interval timer with the added fixed value as the cycle (see **Figure 9-19**).

Since TM1 has two compare registers, two interval timers with different intervals can be constructed.

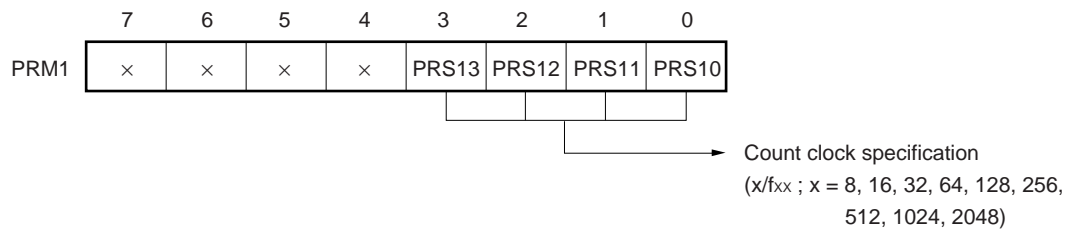
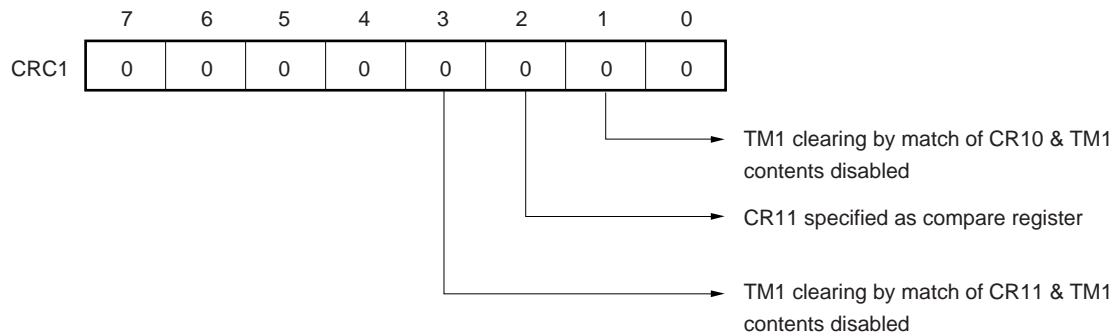
The control register settings are shown in Figure 9-20, the setting procedure in Figure 9-21, and the processing in the interrupt processing routine in Figure 9-23.

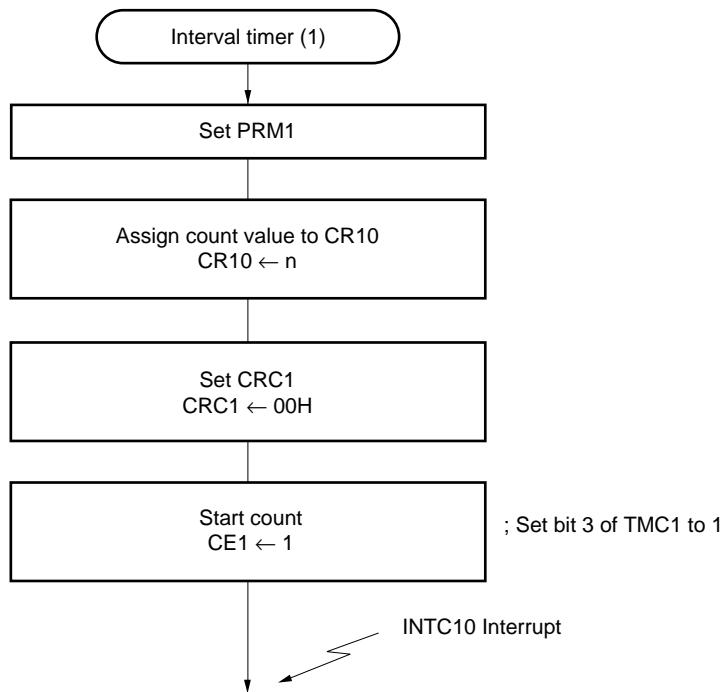
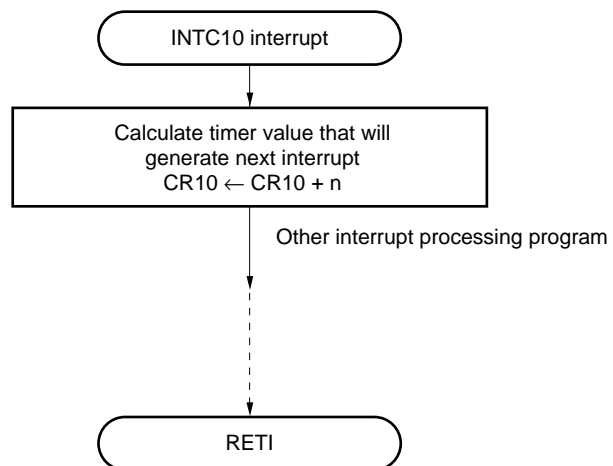
**Figure 9-19 Interval Timer Operation (1) Timing**



**Remark** Interval =  $n \times x / f_{xx}$ ,  $1 \leq n \leq FFH$   
 $x = 8, 16, 32, 64, 128, 256, 512, 1024, 2048$



**Figure 9-20 Control Register Settings for Interval Timer Operation (1)****(a) Prescaler mode register 1 (PRM1)****(b) Capture/compare control register 1 (CRC1)**

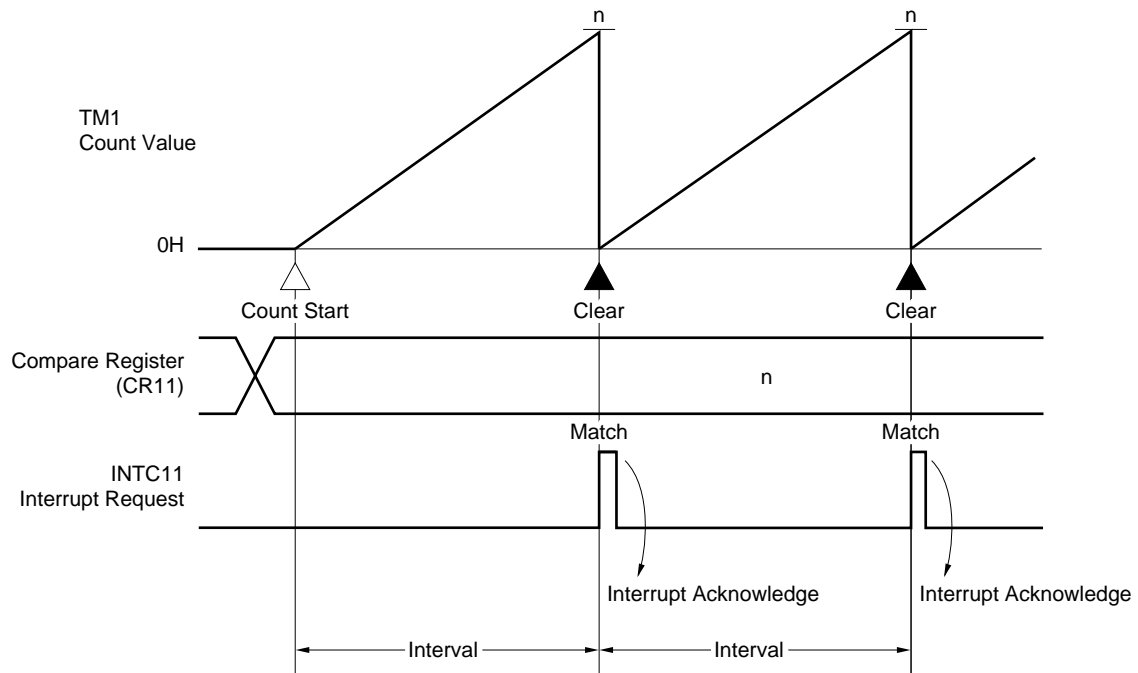
**Figure 9-21 Interval Timer Operation (1) Setting Procedure****Figure 9-22 Interval Timer Operation (1) Interrupt Request Processing**

**9.7.2 Operation as Interval Timer (2)**

TM1 operates as an interval timer that generates interrupts repeatedly with the preset count time as the interval (see **Figure 9-23**).

The control register settings are shown in Figure 9-24, and the setting procedure in Figure 9-25.

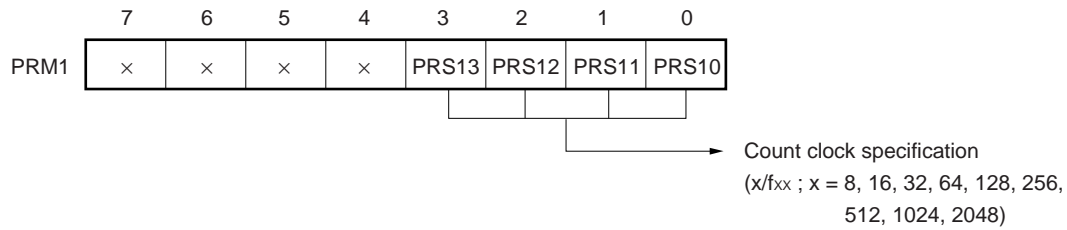
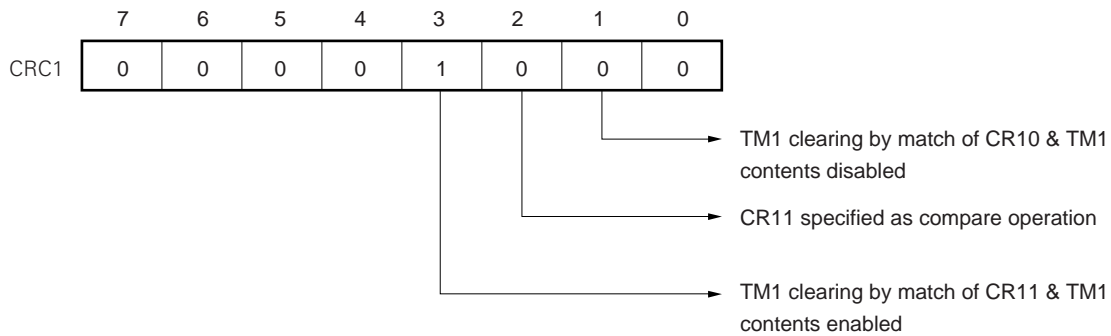
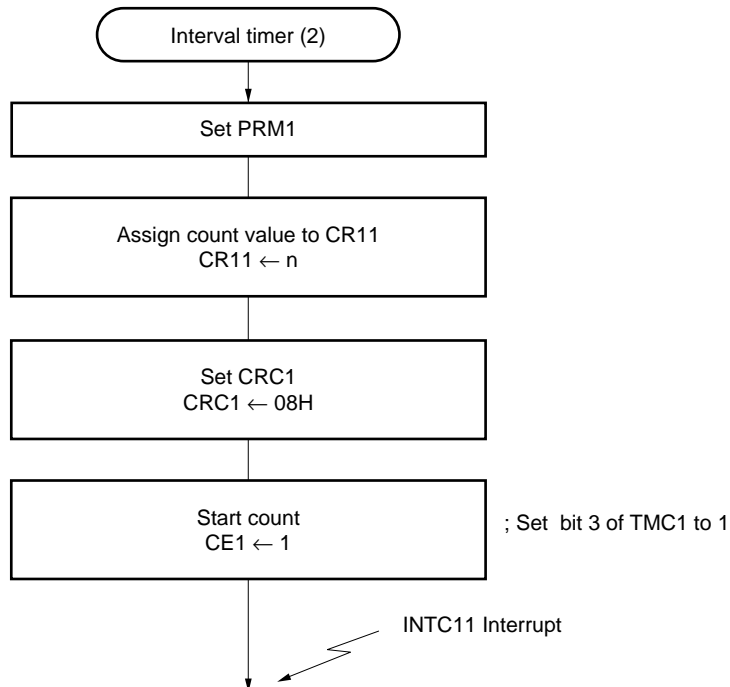
**Figure 9-23 Interval Timer Operation (2) Timing (When CR11 is Used as Compare Register)**



**Remark**  $\text{Interval} = (n+1) \times x/f_{xx}$

$0 \leq n \leq \text{FFH}$

$x = 8, 16, 32, 64, 128, 256, 512, 1024, 2048$

**Figure 9-24 Control Register Settings for Interval Timer Operation (2)****(a) Prescaler mode register 1 (PRM1)****(b) Capture/compare control register 1 (CRC1)****Figure 9-25 Interval Timer Operation (2) Setting Procedure**

### 9.7.3 Pulse Width Measurement Operation

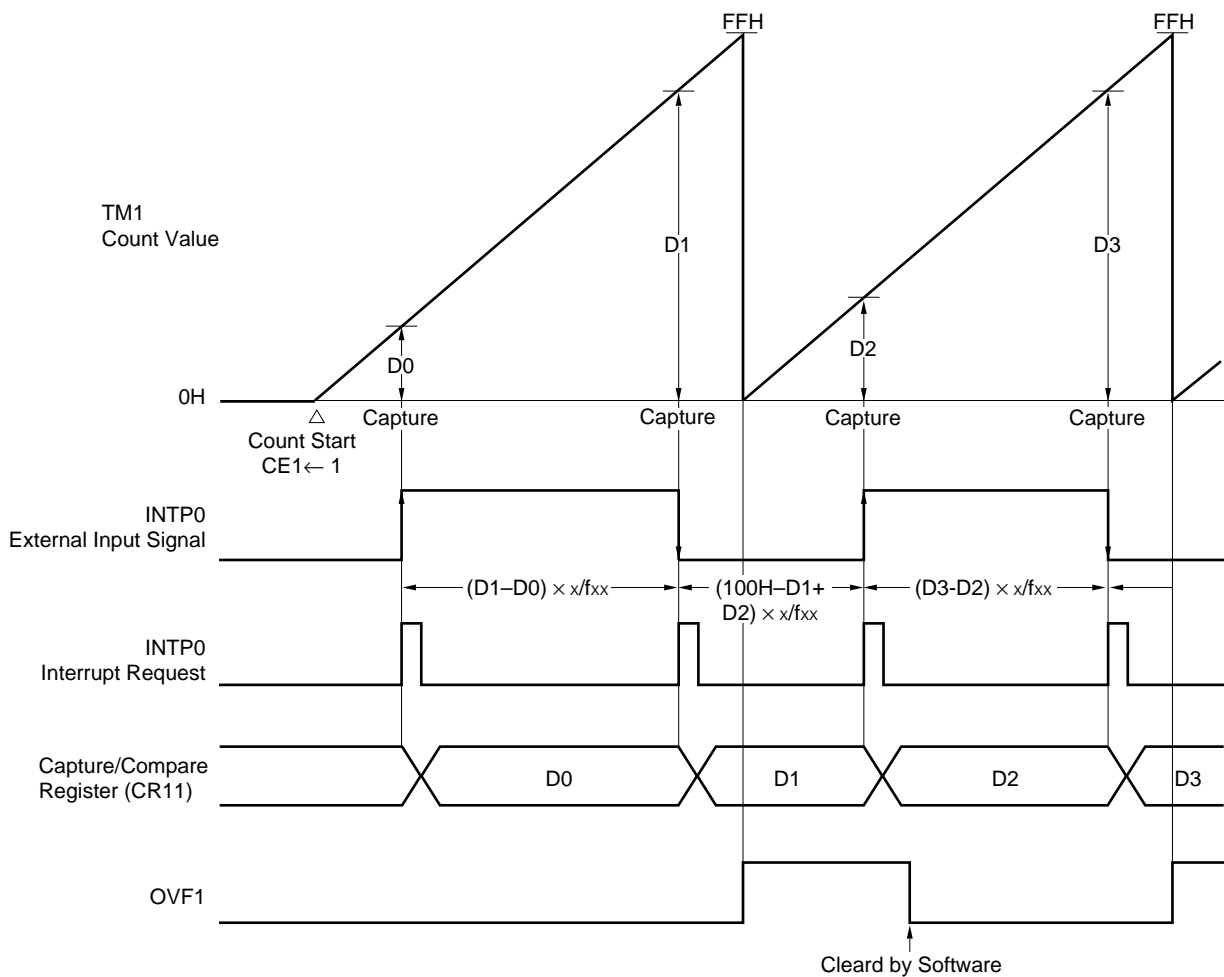
In pulse width measurement, the high-level or low-level width of external pulses input to the external interrupt request input pin (INTP0) is measured.

Both the high-level and low-level widths of pulses input to the INTP0 pin must be at least 3 sampling clocks selected by SCS0; if shorter than this, the valid edge will not be detected and a capture operation will not be performed.

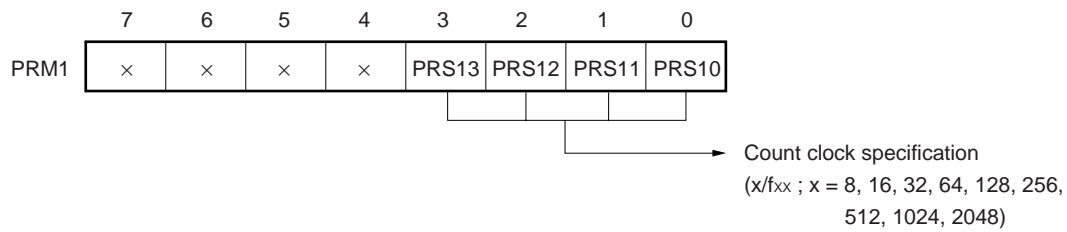
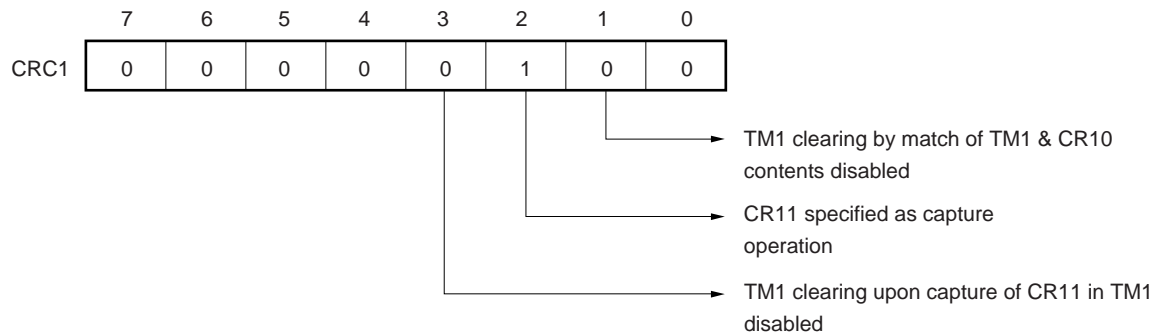
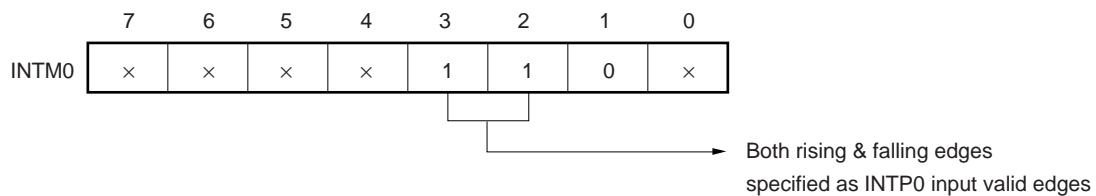
As shown in Figure 9-26, the timer register 1 (TM1) value being counted is fetched into the capture/compare register (CR11) set as a capture register in synchronization with a valid edge (set as both rising and falling edges) in the INTP0 pin input, and retained there. The pulse width is obtained from the product of the difference between the TM1 count value ( $D_n$ ) fetched into and retained in the CR11 on detection of the  $n$ th valid edge and the count value ( $D_{n-1}$ ) fetched and retained on detection of valid edge  $n-1$ , and the number of count clocks ( $x/f_{xx}$ ;  $x = 8, 16, 32, 64, 128, 256, 512, 1024, 2048$ ).

The control register settings are shown in Figure 9-27, and the setting procedure in Figure 9-28.

**Figure 9-26 Pulse Width Measurement Timing (When CR11 is Used as Capture Register)**



**Remark**  $D_n$ : TM1 count value ( $n = 0, 1, 2, \dots$ )  
 $x = 8, 16, 32, 64, 128, 256, 512, 1024, 2048$

**Figure 9-27 Control Register Settings for Pulse Width Measurement****(a) Prescaler mode register 1 (PRM1)****(b) Capture/compare control register 1 (CRC1)****(c) External interrupt mode register 0 (INTM0)**

× : Don't care

Figure 9-28 Pulse Width Measurement Setting Procedure

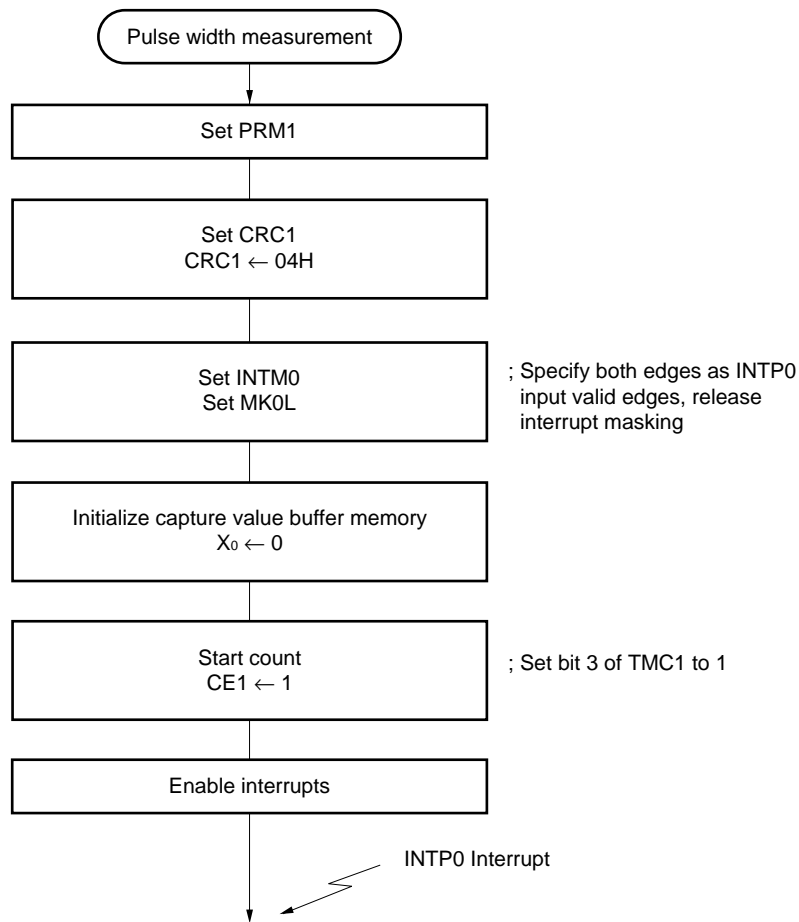
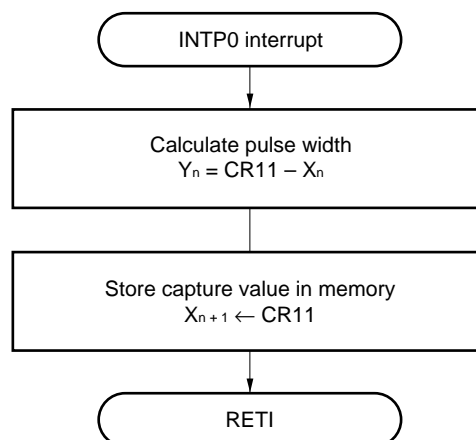


Figure 9-29 Interrupt Request Processing that Calculates Pulse Width



**9.8 CAUTIONS**

- (1) While timer/counter 1 is operating (while the CE1 bit of the timer control register 1 (TMC1) is set), malfunctioning may occur if the contents of the following registers are rewritten. This is because it is undefined which takes precedence in a contention, the change in the hardware functions due to rewriting the register, or the change in the status because of the function before rewriting.

Therefore, be sure to stop the counter operation for the sake of safety before rewriting the contents of the following registers.

- Prescaler mode register 1 (PRM1)
- Capture/compare control register 1 (CRC1)
- CMD2 bit of timer control register 1 (TMC1)

- (2) If the contents of the compare register (CR1n: n = 0 or 1) match those of TM1 when an instruction that stops timer register 1 (TM1) operation is executed, the counting operation of TM1 stops, but an interrupt request is generated.

In order not to generate the interrupt when stopping the operation of TM1, mask the interrupt in advance by using the interrupt mask register before stopping TM1.

**Example**

Program that may generate interrupt request

```

      ⋮
CLR1  CE1
OR     MK0L, #C0H
      ⋮

```

← Interrupt request  
from timer/counter 1  
occurs between  
these instructions

Program that does not generate interrupt request

```

      ⋮
OR     MK0L, #C0H
CLR1   CE1
CLR1   CIF10
CLR1   CIF11
      ⋮

```

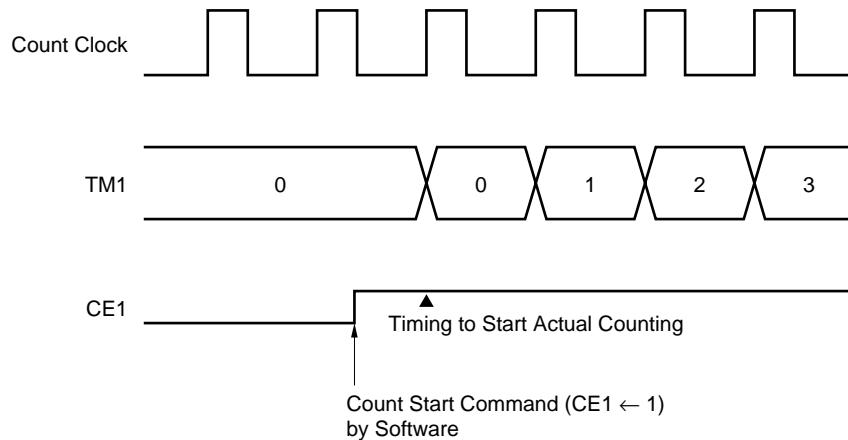
← Disables interrupt  
from timer/counter 1  
← Clears interrupt request  
flag from timer/counter 1



- (3) Up to 1 count clock is required after an operation to start timer/counter 1 ( $CE1 \leftarrow 1$ ) has been performed before timer/counter 1 actually starts (refer to **Figure 9-30**).

For example, when using timer/counter 1 as an interval timer, the first interval time is delayed by up to 1 clock. The second and those that follow are at the specified interval.

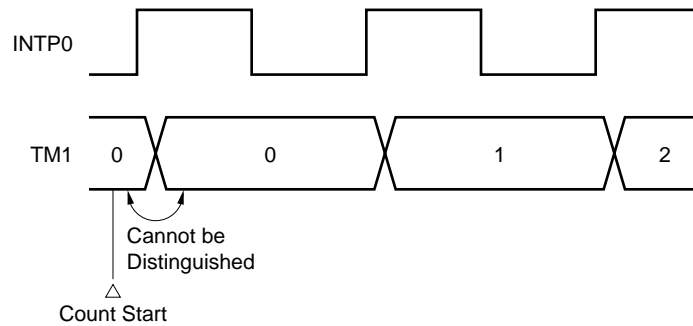
**Figure 9-30 Operation When Counting Is Started**

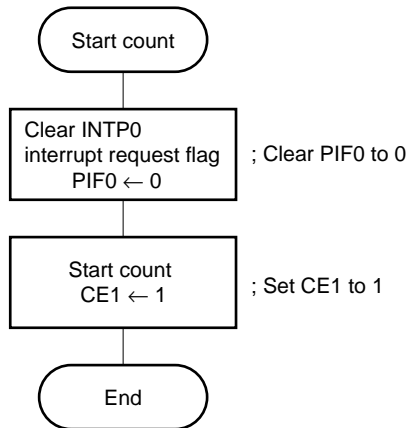
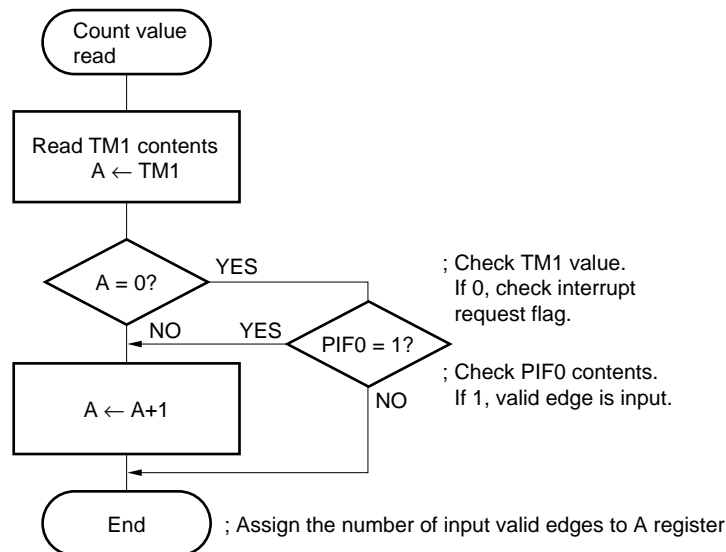


- (4) While an instruction that writes data to the compare register ( $CR1n$ :  $n = 0, 1$ ) is executed, a match between  $CR1n$ , to which the data is to be written, and timer register 1 (TM1) is not detected.  
Write data to  $CR1n$  when timer/counter 1 is executing counting operation in the timing that the contents of TM1 do not match the value of  $CR1n$  before and after writing (e.g., immediately after an interrupt request has been generated because TM1 and  $CR1n$  have matched).
- (5) A match between timer register 1 (TM1) and compare register ( $CR1n$ :  $n = 0, 1$ ) is detected only when TM1 is incremented. Therefore, the interrupt request is not generated even if the same value as TM1 is written to  $CR1n$ .

- (6) When using timer/counter 1 as an external event counter, the status where no valid edge is input cannot be distinguished from the status where only one valid edge has been input, by using timer register 1 (TM1) alone (refer to **Figure 9-31**), because the contents of TM1 are 0 in both the cases. To make a distinction, use the interrupt request flag of INTP0, as shown in Figure 9-32.

**Figure 9-31 Example Where Whether One or No Valid Edge Has been Input Cannot Be Distinguished with External Event Counter**



**Figure 9-32 To Distinguish Whether One or No Valid Edge Has Been Input with External Event Counter****(a) Processing on starting counting****(b) Processing on reading count value**

- ★ (7) Although the timer register can be read, if counting up the timer and reading the timer register conflict, the read value of the timer register may be illegal. Therefore, do not read the timer register if at all possible. If it is necessary to read the timer register, read it two or more times in a row in software, and assume the value of the timer register is correct only when all the read values match.

[MEMO]

## CHAPTER 10 TIMER/COUNTER 2

### 10.1 FUNCTIONS

Timer/counter 2 is 16-bit or 8-bit timer/counter, and has the following function which the other three timer/counters do not have:

- One-shot timer\*

\* The one-shot timer function is a count operation of timer/counter 2 (TM2/TM2W), and is thus different in nature from the one-shot pulse output function of timer/counter 0.

In this section, the following four basic functions are described in order:

- Interval timer
- Programmable square-wave output
- Pulse width measurement
- External event counter

**(1) Interval timer**

Generates internal interrupts at preset intervals.

**Table 10-1 Timer/Counter 2 Intervals**

Minimum Interval	Maximum Interval	Resolution
$8/f_{xx}$ (0.32 $\mu$ s)	$2^{16} \times 8/f_{xx}$ (21.0 ms)	$8/f_{xx}$ (0.32 $\mu$ s)
$16/f_{xx}$ (0.64 $\mu$ s)	$2^{16} \times 16/f_{xx}$ (41.9 ms)	$16/f_{xx}$ (0.64 $\mu$ s)
$32/f_{xx}$ (1.28 $\mu$ s)	$2^{16} \times 32/f_{xx}$ (83.9 ms)	$32/f_{xx}$ (1.28 $\mu$ s)
$64/f_{xx}$ (2.56 $\mu$ s)	$2^{16} \times 64/f_{xx}$ (168 ms)	$64/f_{xx}$ (2.56 $\mu$ s)
$128/f_{xx}$ (5.12 $\mu$ s)	$2^{16} \times 128/f_{xx}$ (336 ms)	$128/f_{xx}$ (5.12 $\mu$ s)
$256/f_{xx}$ (10.2 $\mu$ s)	$2^{16} \times 256/f_{xx}$ (671 ms)	$256/f_{xx}$ (10.2 $\mu$ s)
$512/f_{xx}$ (20.5 $\mu$ s)	$2^{16} \times 512/f_{xx}$ (1.34 s)	$512/f_{xx}$ (20.5 $\mu$ s)
$1024/f_{xx}$ (41.0 $\mu$ s)	$2^{16} \times 1024/f_{xx}$ (2.68 s)	$1024/f_{xx}$ (41.0 $\mu$ s)
$2048/f_{xx}$ (81.9 $\mu$ s)	$2^{16} \times 2048/f_{xx}$ (5.37 s)	$2048/f_{xx}$ (81.9 $\mu$ s)

( ): When  $f_{xx} = 25$  MHz

**(2) Programmable square-wave output**

Outputs square waves independently to the timer output pins (TO2 and TO3).

**Table 10-2 Timer/Counter 2 Programmable Square-Wave Output Setting Range**

Minimum Pulse Width	Maximum Pulse Width
$8/f_{xx}$ (0.32 $\mu$ s)	$2^{16} \times 8/f_{xx}$ (21.0 ms)
$16/f_{xx}$ (0.64 $\mu$ s)	$2^{16} \times 16/f_{xx}$ (41.9 ms)
$32/f_{xx}$ (1.28 $\mu$ s)	$2^{16} \times 32/f_{xx}$ (83.9 ms)
$64/f_{xx}$ (2.56 $\mu$ s)	$2^{16} \times 64/f_{xx}$ (168 ms)
$128/f_{xx}$ (5.12 $\mu$ s)	$2^{16} \times 128/f_{xx}$ (336 ms)
$256/f_{xx}$ (10.2 $\mu$ s)	$2^{16} \times 256/f_{xx}$ (671 ms)
$512/f_{xx}$ (20.5 $\mu$ s)	$2^{16} \times 512/f_{xx}$ (1.34 s)
$1024/f_{xx}$ (41.0 $\mu$ s)	$2^{16} \times 1024/f_{xx}$ (2.68 s)
$2048/f_{xx}$ (81.9 $\mu$ s)	$2^{16} \times 2048/f_{xx}$ (5.37 s)

( ): When  $f_{xx} = 25$  MHz

**Caution** The above table is applicable to use of an internal clock.

**(3) Pulse width measurement**

Detects the pulse width of the signal input to an external interrupt request input pins (INTP1/INTP2).

**Table 10-3 Timer/Counter 2 Pulse Width Measurement Range**

Measurable Pulse Width*			Resolution
$8/f_{xx}$ (0.32 $\mu$ s)	–	$2^{16} \times 8/f_{xx}$ (21.0 ms)	$8/f_{xx}$ (0.32 $\mu$ s)
$16/f_{xx}$ (0.64 $\mu$ s)	–	$2^{16} \times 16/f_{xx}$ (41.9 ms)	$16/f_{xx}$ (0.64 $\mu$ s)
$32/f_{xx}$ (1.28 $\mu$ s)	–	$2^{16} \times 32/f_{xx}$ (83.9 ms)	$32/f_{xx}$ (1.28 $\mu$ s)
$64/f_{xx}$ (2.56 $\mu$ s)	–	$2^{16} \times 64/f_{xx}$ (168 ms)	$64/f_{xx}$ (2.56 $\mu$ s)
$128/f_{xx}$ (5.12 $\mu$ s)	–	$2^{16} \times 128/f_{xx}$ (336 ms)	$128/f_{xx}$ (5.12 $\mu$ s)
$256/f_{xx}$ (10.2 $\mu$ s)	–	$2^{16} \times 256/f_{xx}$ (671 ms)	$256/f_{xx}$ (10.2 $\mu$ s)
$512/f_{xx}$ (20.5 $\mu$ s)	–	$2^{16} \times 512/f_{xx}$ (1.34 s)	$512/f_{xx}$ (20.5 $\mu$ s)
$1024/f_{xx}$ (41.0 $\mu$ s)	–	$2^{16} \times 1024/f_{xx}$ (2.68 s)	$1024/f_{xx}$ (41.0 $\mu$ s)
$2048/f_{xx}$ (81.9 $\mu$ s)	–	$2^{16} \times 2048/f_{xx}$ (5.37 s)	$2048/f_{xx}$ (81.9 $\mu$ s)

( ): When  $f_{xx} = 25$  MHz

- \* The minimum pulse width that can be measured differs depending on the selected value of  $f_{CLK}$ .  
The minimum pulse width that can be measured is the value of  $4/f_{CLK}$  or the value in the above table, whichever greater.

**(4) External event counter**

Counts the clock pulses input from the external interrupt request input pin (INTP2) (CI pin input pulses). The clocks that can be input to timer/counter 2 are shown in Table 10-4.

★

**Table 10-4 Clocks Enabled to be Input to Timer/Counter 2**

	When Counting One Edge	When Counting Both Edges
Maximum frequency	$f_{CLK}/8$ (1.56 MHz)	$f_{CLK}/8$ (1.56 MHz)
Minimum pulse width (High and low levels)	$4/f_{CLK}$ (0.32 $\mu$ s)	$4/f_{CLK}$ (0.32 $\mu$ s)

( ): When  $f_{CLK} = 12.5$  MHz and  $f_{xx} = 25$  MHz



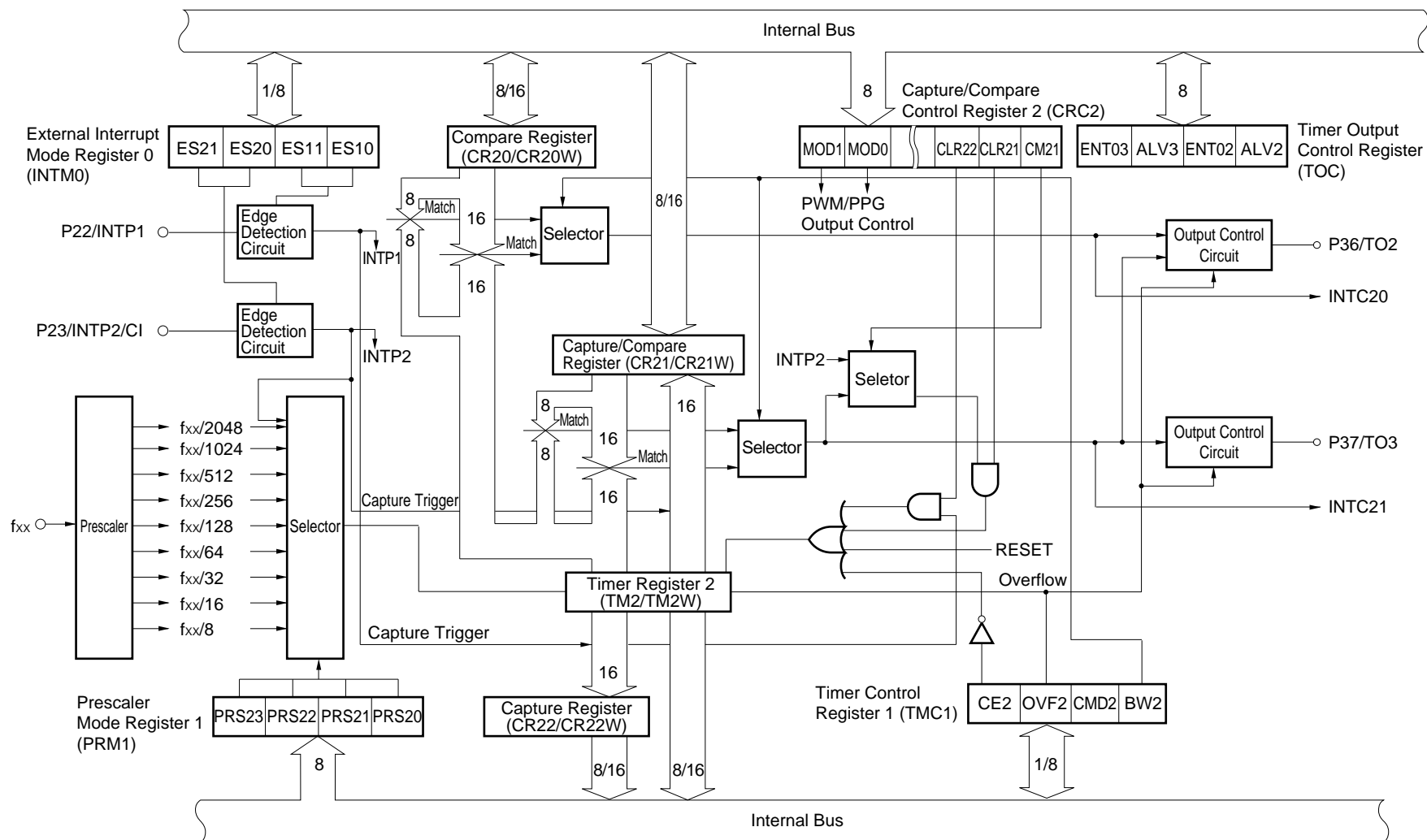
## 10.2 CONFIGURATION

Timer/counter 2 consists of the following registers.

- Timer register (TM2/TM2W)  $\times$  1
- Compare register (CR20/CR20W)  $\times$  1
- Capture/compare register (CR21/CR21W)  $\times$  1
- Capture register (CR22/CR22W)  $\times$  1

The block diagram of timer/counter 2 is shown in Figure 10-1.

Figure 10-1 Timer/Counter 2 Block Diagram



**(1) Timer register 2 (TM2/TM2W)**

TM2/TM2W is a timer register that counts up the count clock specified by the high-order 4 bits of prescaler mode register 1 (PRM1). An internal clock or external clock can be selected as the count clock.

The count operation can be stopped or enabled by means of timer control register 1 (TMC1). The timer register can select to operate in an 8-bit (TM1) or 16-bit (TM1W) mode.

TM2/TM2W can be read only with an 8/16-bit manipulation instruction.

When  $\overline{\text{RESET}}$  is input, TM2/TM2W is cleared to 00H and the count is stopped.

★ **Caution** Although the timer register can be read, if counting up the timer and reading the timer register conflict, the read value of the timer register may be illegal. Therefore, do not read the timer register if at all possible.

If it is necessary to read the timer register, read it two or more times in a row in software, and assume the value of the timer register is correct only when all the read values match.

**(2) Compare register (CR20/CR20W)**

CR20/CR20W is an 8/16-bit register that holds the value that determines the interval timer operation cycle.

If the contents of the CR20/CR20W register match the contents of TM2/TM2W, an interrupt request (INTC20) and a timer output control signal are generated. This compare register operates as CR20 in the 8-bit mode, and CR20W in the 16-bit mode.

CR20/CR20W can be read or written to with an 8/16-bit manipulation instruction. The contents of this register are undefined after  $\overline{\text{RESET}}$  input.

**(3) Capture/compare register (CR21/CR21W)**

CR21/CR21W is an 8/16-bit register that can be specified as a compare register for detecting a match with the TM2/TM2W count value or a capture register for capturing the TM2/TM2W count value according to the setting of the capture/compare control register 2 (CRC2).

This capture/compare register operates as CR21 in the 8-bit mode, and CR21W in the 16-bit mode.

CR21/CR21W can be read or written to with an 8/16-bit manipulation instruction.

The contents of this register are undefined after  $\overline{\text{RESET}}$  input.

**(a) When specified as compare register**

CR21/CR21W functions as an 8/16-bit register that holds the value that determines the interval timer operation cycle.

An interrupt request (INTC21) and a timer output control signal are generated by a match between the contents of the CR21/CR21W register and the contents of TM2/TM2W.

Also, the count value can be cleared by a match of the contents.

**(b) When specified as capture register**

CR21/CR21W functions as an 8/16-bit register that captures the contents of TM2/TM2W in synchronization with the input of a valid edge on the external interrupt input pin (INTP2) (capture trigger).

The contents of the CR21/CR21W register are retained until the next capture trigger is generated.

★ Also, TM2/TM2W can be cleared after a capture operation.

**(4) Capture register (CR22/CR22W)**

CR22/CR22W is an 8/16-bit register that captures the contents of TM2/TM2W.

The capture operation is synchronized with the input of a valid edge to the external interrupt request input pin (INTP1)(capture trigger). The contents of the CR22/CR22W register are retained until the next capture trigger is generated. Also, TM2/TM2W can be cleared after a capture operation.

This capture register operates as CR22 in the 8-bit mode, and CR22W in the 16-bit mode.

CR22/CR22W can be read only with an 8/16-bit manipulation instruction. The contents of this register are undefined after RESET input.

**(5) Edge detection circuit**

The edge detection circuit detects an external input valid edge.

This circuit generates an external interrupt request (INTP1) and capture trigger by detecting the valid edge of the INTP1 pin input specified by the external interrupt mode register 0 (INTM0). It also generates a capture trigger, the count clock of an external event, and external interrupt request (INTP2) by detecting the valid edge from an external interrupt request input pin (INTP2).

**(6) Output control circuit**

It is possible to invert the timer output when the CR20/CR21 register contents and the contents of TM2 match or the CR20W/CR21W contents and the contents of TM2W match.

A square wave can be output from the timer output pins (TO2/TO3) in accordance with the setting of the high-order 4 bits of the timer output control register (TOC). At this time, PWM output or PPG output can be performed according to the specification of the capture/compare control register 2 (CRC2).

Timer output can be disabled/enabled by means of the TOC register. When timer output is disabled, a fixed level is output to the TO2 and TO3 pins (the output level is set by the TOC register).

**(7) Prescaler**

The prescaler generates the count clock from the internal system clock. The clock generated by the prescaler is selected by the selector, and is used as the count clock by the timer register 2 (TM2/TM2W) to perform count operations.

**(8) Selector**

The selector selects a signal resulting from dividing the internal clock or the edge detected by the edge detection circuit as the count clock of timer register 2 (TM2/TM2W).

## 10.3 TIMER/COUNTER 2 CONTROL REGISTERS

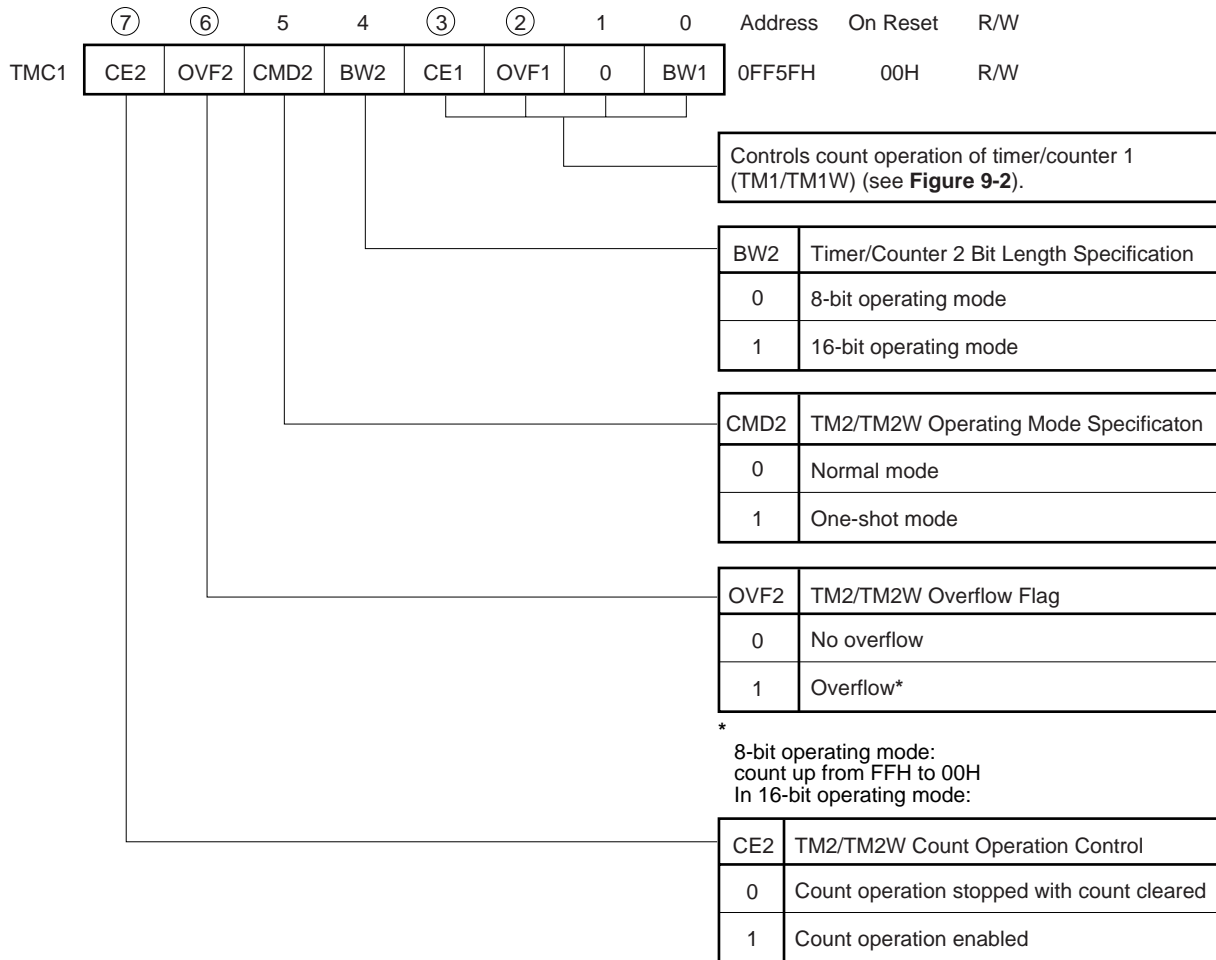
## (1) Timer control register 1 (TMC1)

In TMC1 the timer/counter 2, TM2/TM2W, count operation is controlled by the high-order 4 bits (the low-order 4 bits control the count operation of timer/counter 1, TM1/TM1W).

TMC1 can be read or written to with an 8-bit manipulation instruction or bit manipulation instruction. The format of the TMC1 is shown in Figure 10-2.

RESET input clears TMC1 to 00H.

Figure 10-2 Timer Control Register 1 (TMC1) Format



**Remark** The OVF2 bit is reset by software only.

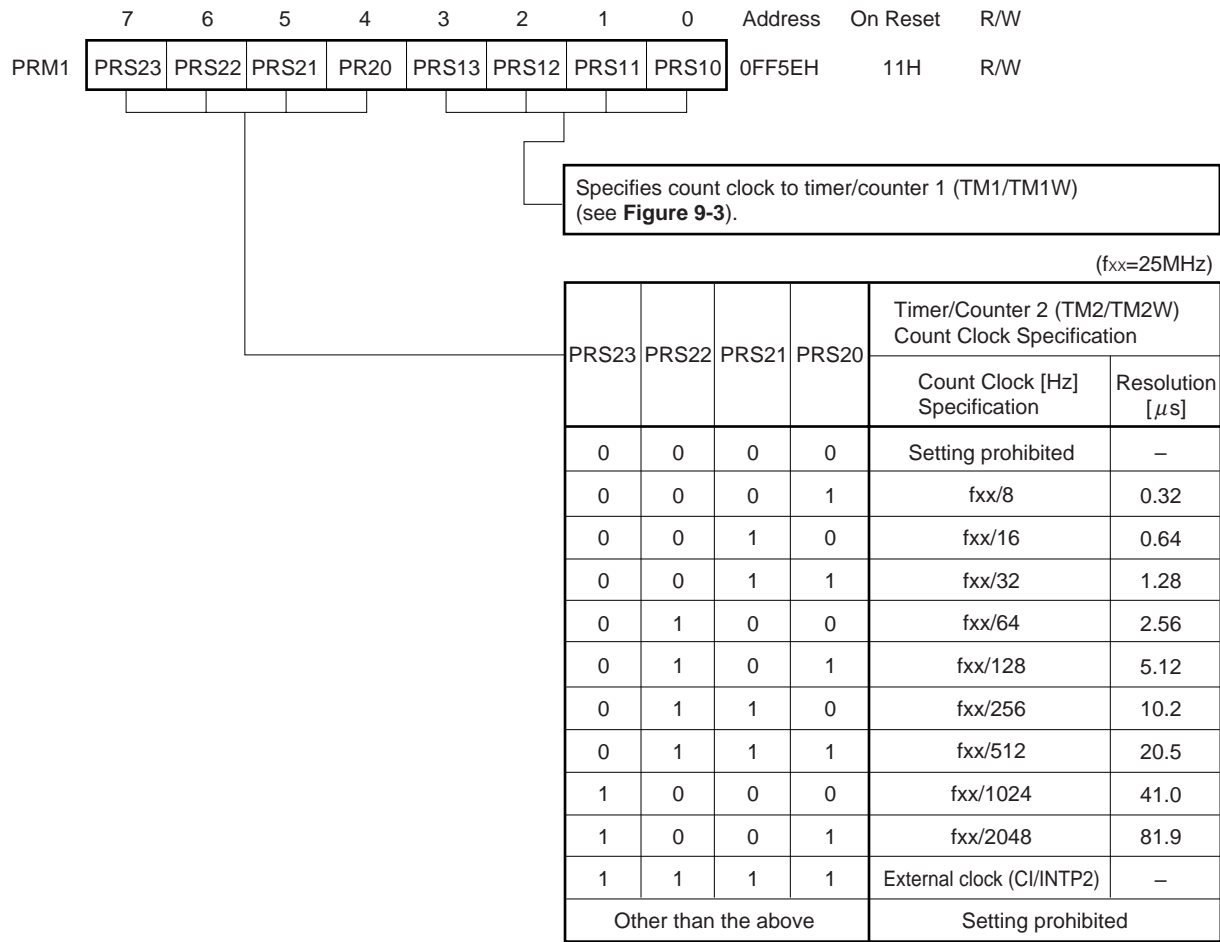
**(2) Prescaler mode register 1 (PRM1)**

In PRM1 the count clock to timer/counter 2, TM2/TM2W, is specified by the high-order 4 bits (the low-order 4 bits specify the count clock to timer/counter 1, TM1/TM1W).

PRM1 can be read or written with an 8-bit manipulation instruction. The format of the PRM1 is shown in Figure 10-3.

$\overline{\text{RESET}}$  input clears PRM1 to 11H.

**Figure 10-3 Prescaler Mode Register 1 (PRM1) Format**



**Remark** fxx: X1 input frequency or oscillation frequency

**(3) Capture/compare control register 2 (CRC2)**

The CRC2 specifies the enabling condition for a timer register 2 (TM2/TM2W) clear operation by the capture/compare register (CR21/CR21W) or the capture register (CR22/CR22W) and the timer output (TO2/TO3) mode.

CRC2 can be read or written with an 8-bit manipulation instruction. The format of the CRC2 is shown in Figure 10-4.

$\overline{\text{RESET}}$  input clears CRC2 to 10H.

**Figure 10-4 Capture/Compare Control Register 2 (CRC2) Format**

	7	6	5	4	3	2	1	0	Address	ON Reset	R/W
CRC2	MOD1	MOD0	CLR22	1	CLR21	CM21	0	0	0FF33H	10H	R/W
<div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div><div></div></div>											
		MOD1	MOD0	CLR22	CLR21	CM21	CR21 Operation Specification		Timer Output Mode Specification		TM2 Clear Operation
								TO2	TO3		
		0	0	0	0	0	Compare operations	Toggle output	Toggle output	Not cleared	
		0	0	0	1	0		Toggle output	Toggle output	Cleared if TM2 and CR21 match	
		0	0	1	0	0		Toggle output	Toggle output	Cleared after TM2 contents are captured in CR22 by INTP1	
		0	0	1	1	0		Toggle output	Toggle output	Cleared by match of TM2 and CR21 or after TM2 contents are captured in CR22 by INTP1	
		0	1	0	0	0		PWM output	Toggle output	Not cleared	
		1	0	0	0	0		PWM output	PWM output	Not cleared	
		1	1	0	1	0		PPG output	Toggle output	Cleared if TM2 and CR21 match	
		0	0	0	0	1	Capture operations	Toggle output	<div></div>	Not cleared	
		0	0	0	1	1		Toggle output		Cleared after TM2 contents are captured in CR21 by INTP2	
		0	1	0	0	1		PWM output		Not cleared	
Other than the above							Setting prohibited				

**Remark** The register names in the 8-bit operating mode are shown in this figure. In the 16-bit operating mode, the register names TM2, CR20, CR21, and CR22 are TM2W, CR20W, CR21W, and CR22W, respectively.

#### (4) Timer output control register (TOC)

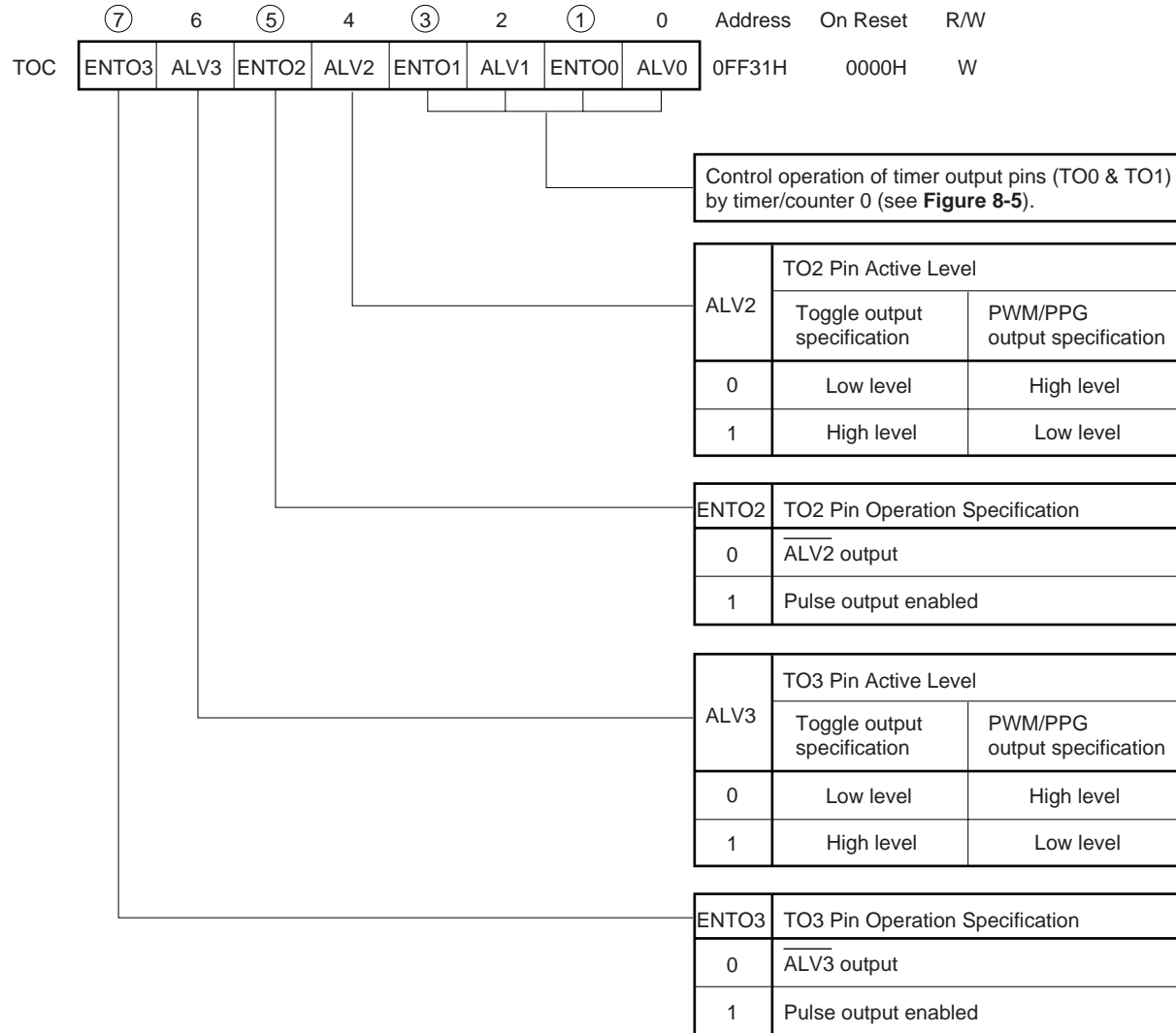
TOC is an 8-bit register that controls output enabling/disabling of the active level of timer output.

The operation of the timer output pins (TO2 and TO3) by timer/counter 2 is controlled by the high-order 4 bits (the low-order 4 bits control the operation of the timer output pins (TO0 and TO1) by timer/counter 0).

TOC can be read or written with an 8-bit manipulation instruction or bit manipulation instruction. The format of the TOC is shown in Figure 10-5.

RESET input clears TOC to 00H.

**Figure 10-5 Timer Output Control Register (TOC) Format**





## 10.4 TIMER REGISTOR 2 (TM2) OPERATION

### 10.4.1 Basic Operation

8-bit operating mode / 16-bit operating mode control can be performed for timer/counter 2 by means of bit 0 (BW2) of timer control register 2 (TMC2).\*

In the timer/counter 2 count operation, an up-count is performed using the count clock specified by the high-order 4 bits of prescaler mode register 1 (PRM1).

Count operation enabling/disabling is controlled by bit 3 (CE2) of TMC2 (timer/counter 2 operation control is performed by the high-order 4 bits of the timer control register 1 (TMC1). When the CE2 bit is set (1) by software, the contents of TM2 are cleared to 0H on the first count clock, and then the up-count operation is performed.

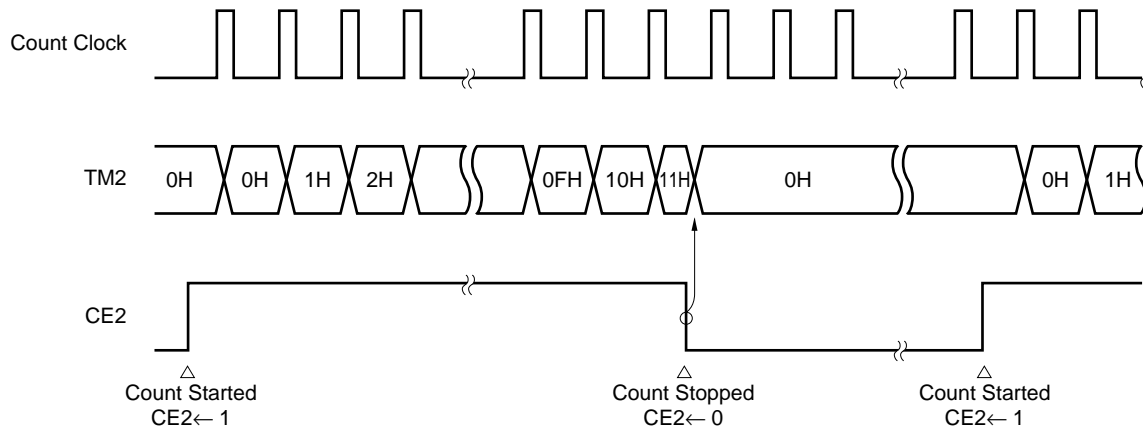
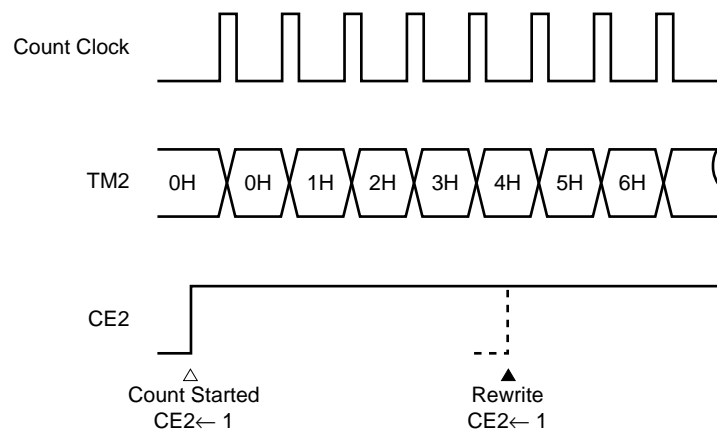
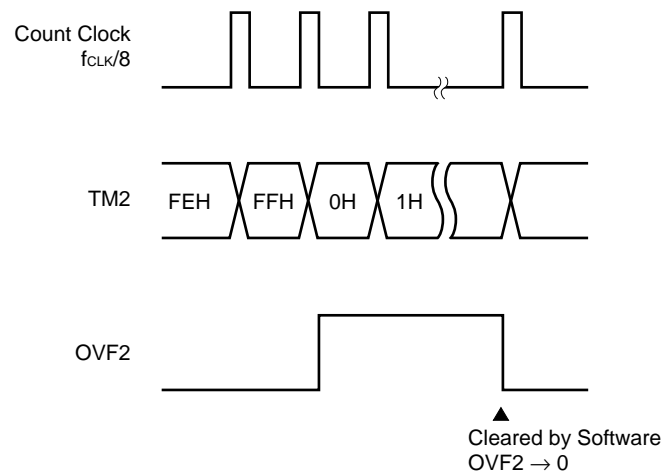
When the CE2 bit is cleared (0) by software, TM2 becomes 0H immediately, and capture operations and match signal generation are stopped.

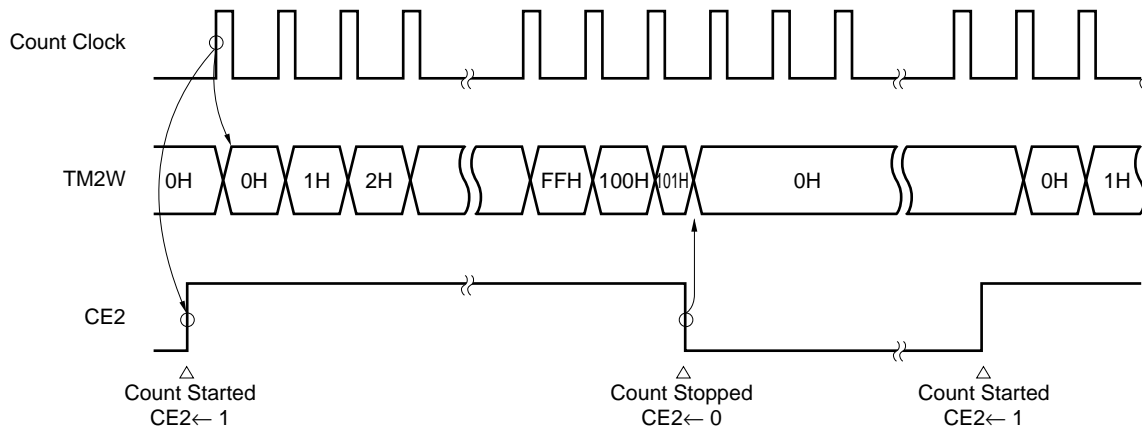
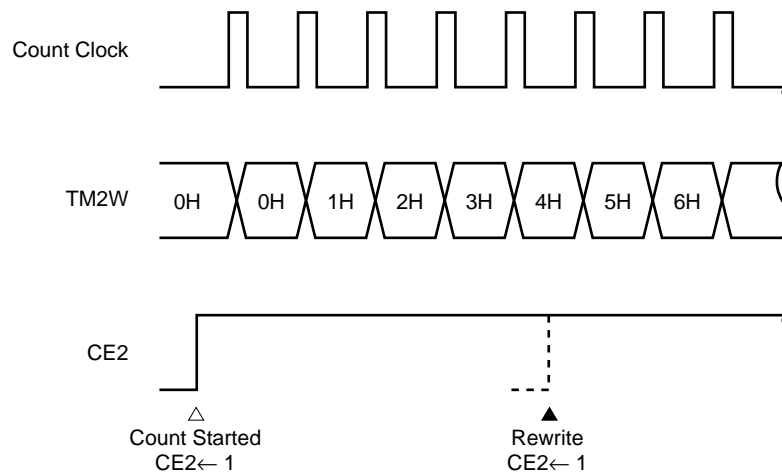
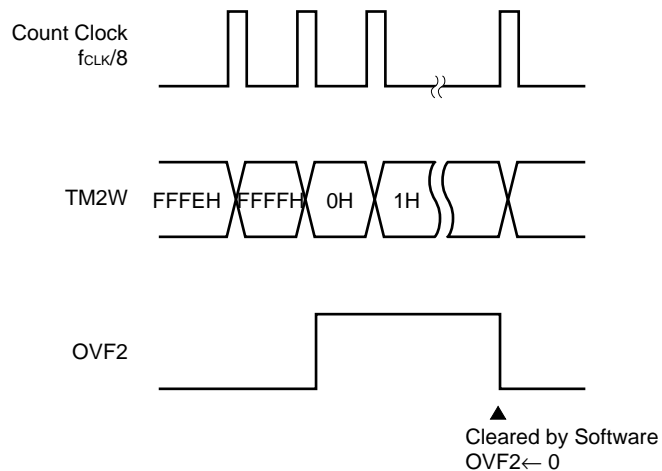
If the CE2 bit is set (1) again when it is already set (1), the TM2 count operation is not affected (see **Figure 10-6 (b)**).

TM2/TM2W is cleared to 0H when the count clock is input while the value of TM2 is FFH in the 8-bit operating mode or while the value of TM2W is FFFFH in the 16-bit operating mode. At this time, OVF2 bit is set and the overflow signal is sent to the output control circuit. OVF2 bit is cleared by software only. The count operation is continued.

When RESET is input, TM2 is cleared to 0H, and the count operation is stopped.

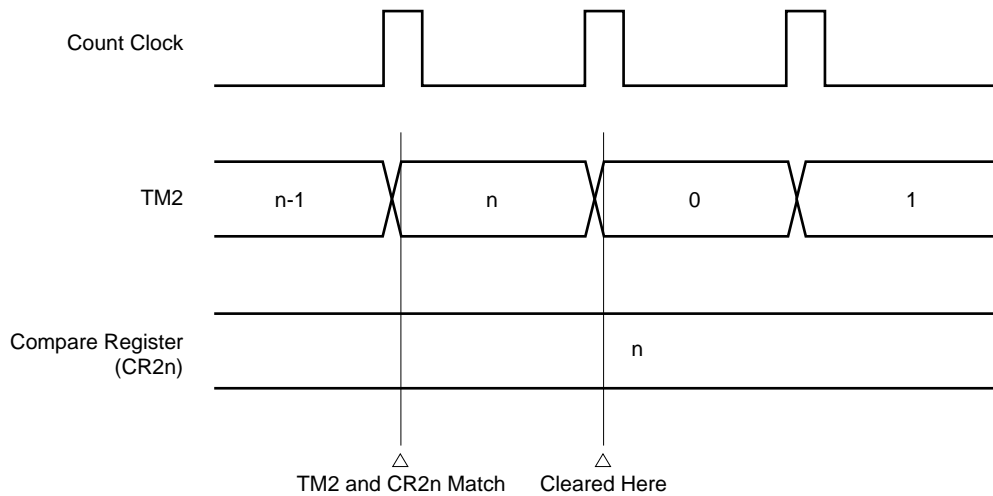
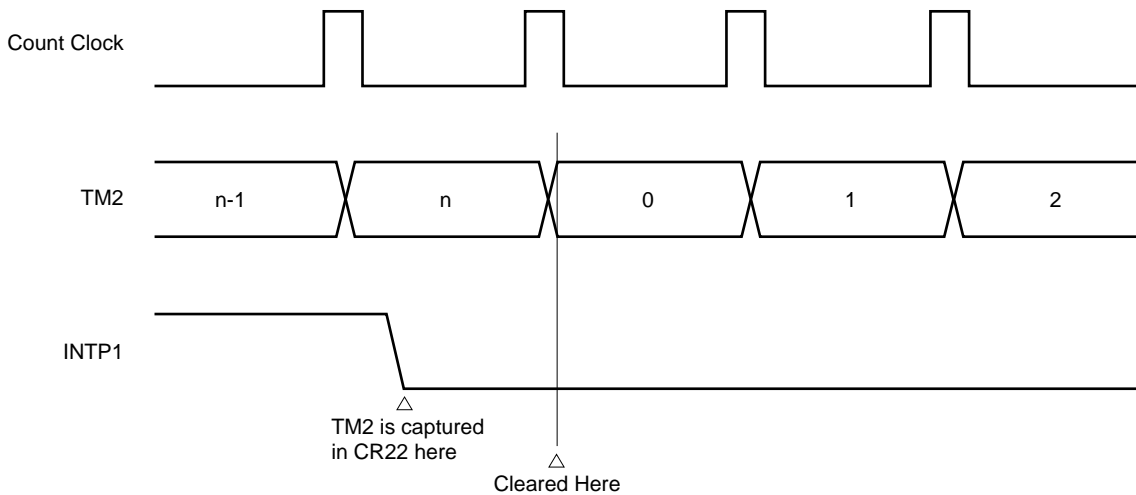
- \* Unless otherwise specified, the functions of timer register 2 in the 8-bit operating mode are described hereafter. In the 16-bit operating mode, TM2, CR20, CR21, and CR22 operate as TM2W, CR20W, CR21W, and CR22W, respectively.

**Figure 10-6 Basic Operation in 8-Bit Operating Mode (BW2 = 0)****(a) Count started → count disabled → count started****(b) When "1" is written to the CE2 bit again after the count starts****(c) Operation when TM2 = FFH**

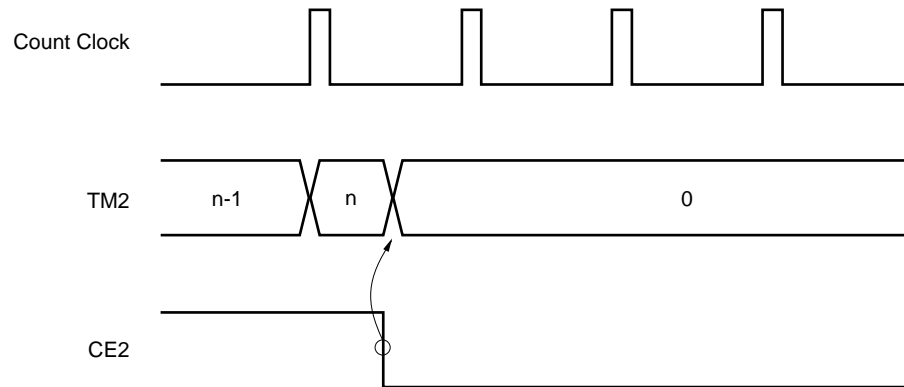
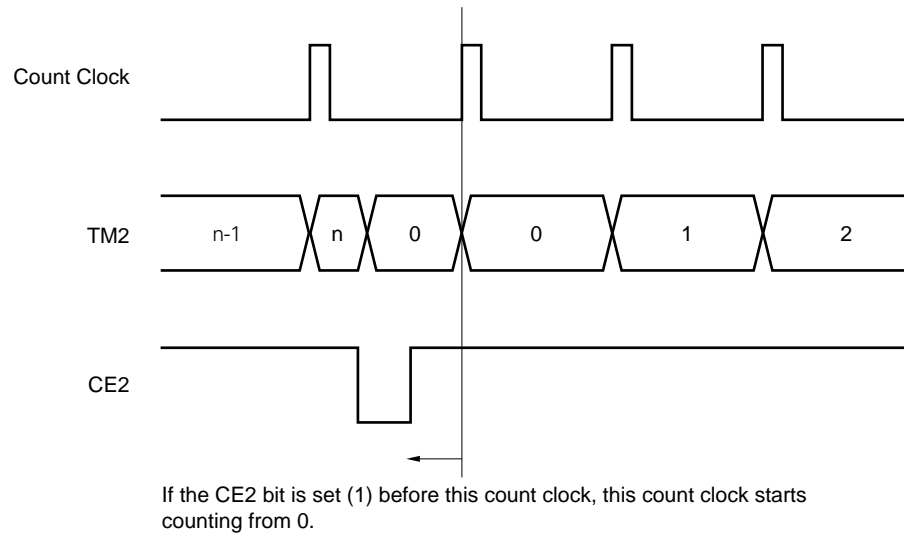
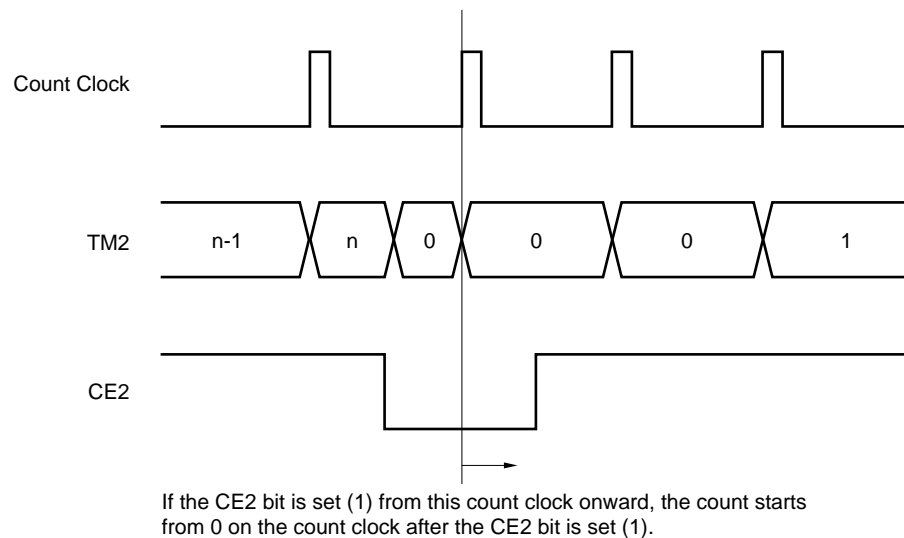
**Figure 10-7 Basic Operation in 16-Bit Operating Mode (BW2 = 1)****(a) Count started → count disabled → count started****(b) When “1” is written to the CE2 bit again after the count starts****(c) Operation when TM2W = FFFFH**

**10.4.2 Clear Operation****(1) Clear operation after match with compare register and capture operation**

Timer register 2 (TM2) can be cleared automatically after a match with the compare register (CR2n:  $n = 0, 1$ ) and a capture operation. When a clearance source arises, TM2 is cleared to 0H on the next count clock. Therefore, even if a clearance source arises, the value at the point at which the clearance source arose is retained until the next count clock arrives.

**Figure 10-8 TM2 Clearance by Match With Compare Register (CR20/CR21)****Figure 10-9 TM2 Clearance after Capture Operation****(2) Clear operation by CE2 bit of timer control register 2 (TMC2)**

Timer register 2 (TM2) is also cleared when the CE2 bit of the TMC1 is cleared (0) by software. The clear operation is performed immediately after clearance (0) of the CE2 bit.

**Figure 10-10 Clear Operation When CE2 Bit is Cleared (0)****(a) Basic operation****(b) Restart before count clock is input after clearance****(c) Restart after count clock is input after clearance**

## 10.5 EXTERNAL EVENT COUNTER FUNCTION

Timer/counter 2 can count clock pulses input from external interrupt request input pin (INTP2/CI).

No special selection method is needed for the external event counter operating mode. When the timer register 2 (TM2) count clock is specified as external clock input by the setting of the high-order 4 bits of prescaler mode register 1 (PRM1), TM2 operates as an external event counter.

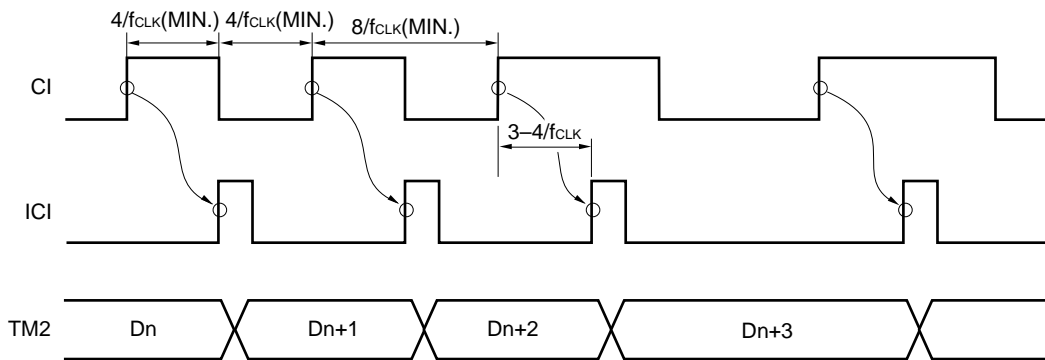
The maximum frequency of external clock pulses that can be counted by TM2 as the external event counter is 1.56 MHz ( $f_{CLK} = 12.5$  MHz) irrespective of whether only one edge or both edges are counted on INTP2/CI input.

The pulse width of INTP2/CI input must be at least 4 system clocks ( $0.32 \mu s$ ;  $f_{CLK} = 12.5$  MHz) for both the high level and low level. If the pulse width is shorter than this, the pulse may not be counted.

The timer/counter 2 external event count timing is shown in Figure 10-11.

★ **Figure 10-11 Timer/Counter 2 External Event Count Timing (1/2)**

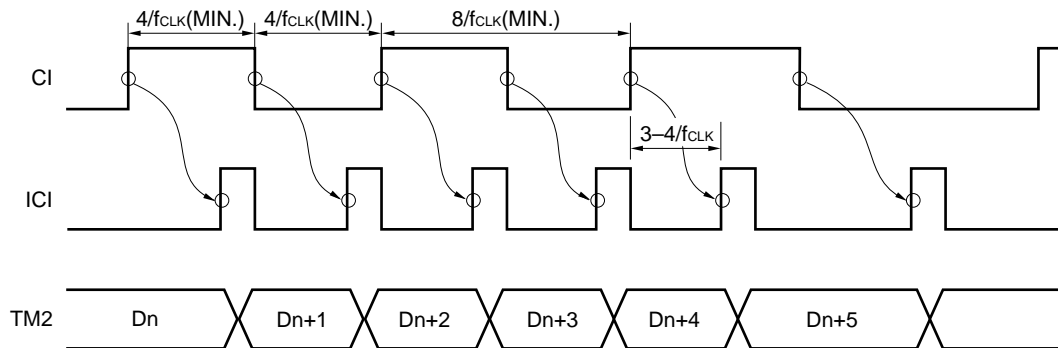
### (1) Counting one edge (maximum frequency = $f_{CLK}/8$ )



**Remark** ICI: CI input signal after passing through edge detection circuit

★

Figure 10-11 Timer/Counter 2 External Event Count Timing (2/2)

(2) Counting both edges (maximum frequency =  $f_{CLK}/8$ )

**Remark** ICI: CI input signal after passing through edge detection circuit

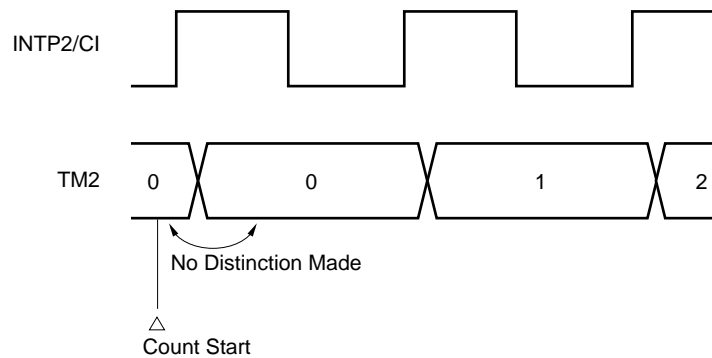
The TM2 count operation is controlled by the CE2 bit of the timer control register 1 (TMC1) in the same way as with the basic operation.

When the CE2 bit is set (1) by software, the contents of TM2 are set to 0H and the up-count operation is started on the initial count clock.

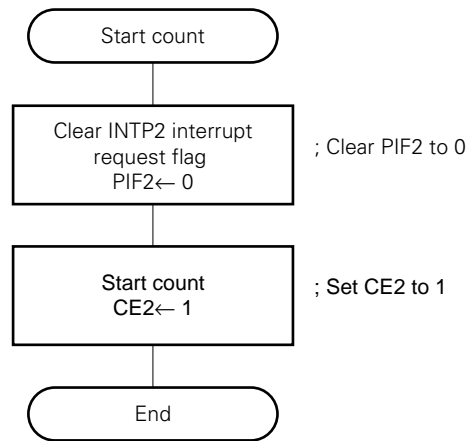
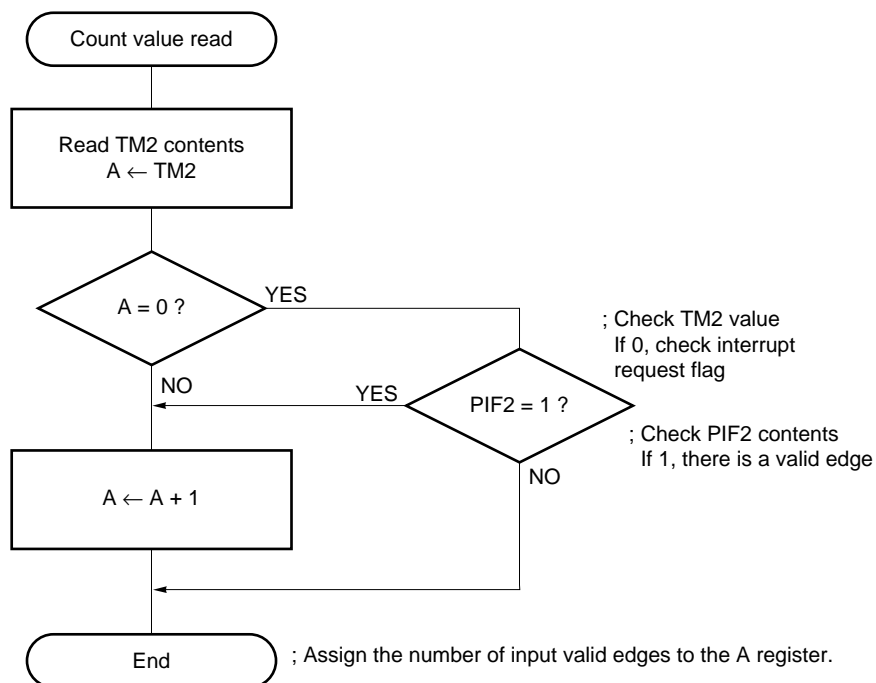
When the CE2 bit is cleared (0) by software during a TM2 count operation, the contents of TM2 are set to 0H immediately and the stopped state is entered. The TM2 count operation is not affected if the CE2 bit is set (1) by software again when it is already set (1).

**Caution** When timer/counter 2 is used as an external event counter, it is not possible to distinguish between the case where there is no valid edge input at all and the case where there is a single valid edge input using timer register 2 (TM2) alone (see Figure 10-12), since the contents of TM2 are 0 in both cases. If it is necessary to make this distinction, the INTP2 interrupt request flag should be used (the INTP2 pin and CI pin have a dual function, and both functions can be used at the same time). An example is shown in Figure 10-13.

**Figure 10-12** Example of the case where the external event counter does not distinguish between one valid edge input and no valid edge input

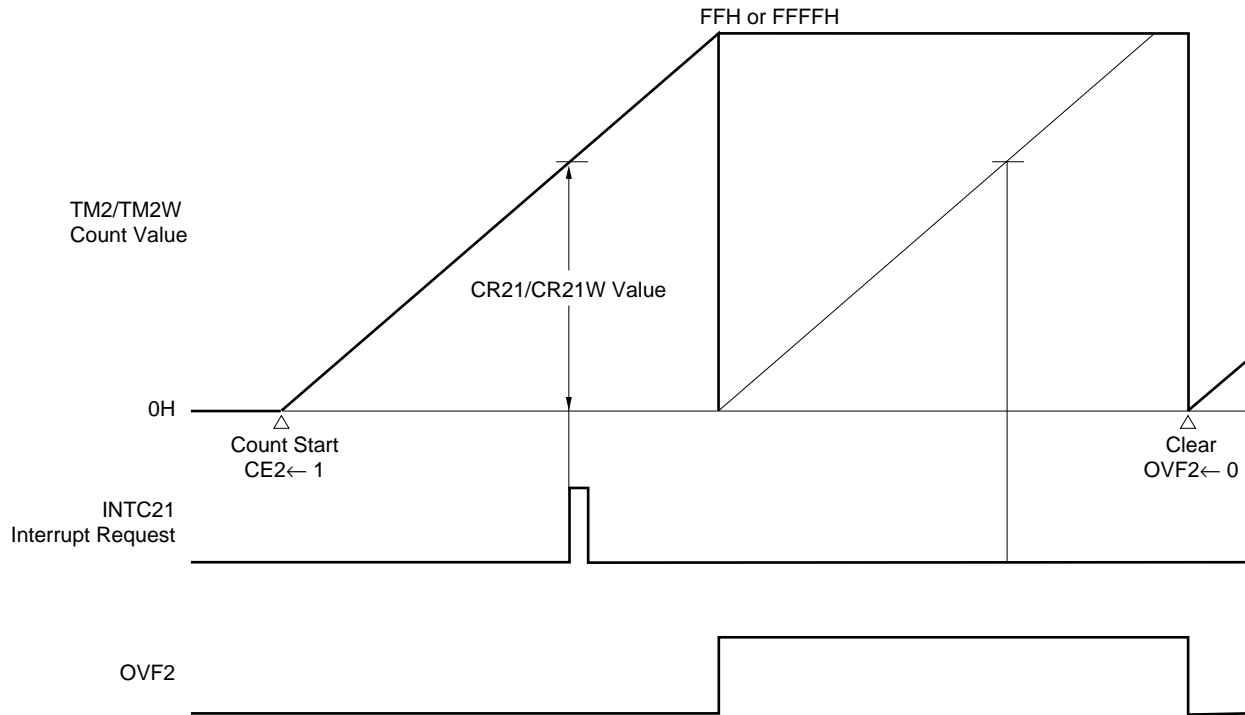




**Figure 10-13 To Distinguish Whether One or No Valid Edge Has Been Input with External Event Counter****(a) Processing when count is started****(b) Processing when count value is read**

**10.6 ONE-SHOT TIMER FUNCTION**

Timer/counter 2 has an operating mode in which it stops automatically when a full count value is reached (FFH/FFFFH) as a result of counting by timer register 2 (TM2/TM2W).

**Figure 10-14 One-Shot Timer Operation**

As shown in Figure 10-14, the respective one-shot interrupt is generated when the value (0H to FFH/FFFFH) set beforehand in the CR20, CR21/CR21W or CR21W and the timer register 2 (TM2/TM2W) value match.

The one-shot timer operating mode is specified by setting (1) bit 5 (CMD2) of timer control register 1 (TMC1) by software.

The TM2/TM2W count operation is controlled by the CE2 bit of the TMC1 as with the basic operation.

When the CE2 bit is set (1) by software, the contents of TM2/TM2W are set to 0H and the up-count operation is started on the initial count clock.

When the contents of TM2/TM2W reach FFH/FFFFH (full count) as a result of the up-count operation, bit 6 (OVF2) of the TMC1 are set (1), and TM2/TM2W stops with the count at FFH/FFFFH.

The one-shot timer operation is started again from the count-stopped state by clearing (0) the OVF2 bit by software. When the OVF2 bit is cleared (0), the contents of TM2/TM2W become 0H and the up-count operation is restarted on the next count clock.

If the CE2 bit is cleared (0) by software during a TM2/TM2W count operation, the contents of TM2/TM2W are set to 0H immediately and the stopped state is entered. The TM2/TM2W count operation is not affected if the CE2 bit is set (1) by software again when it is already set (1).

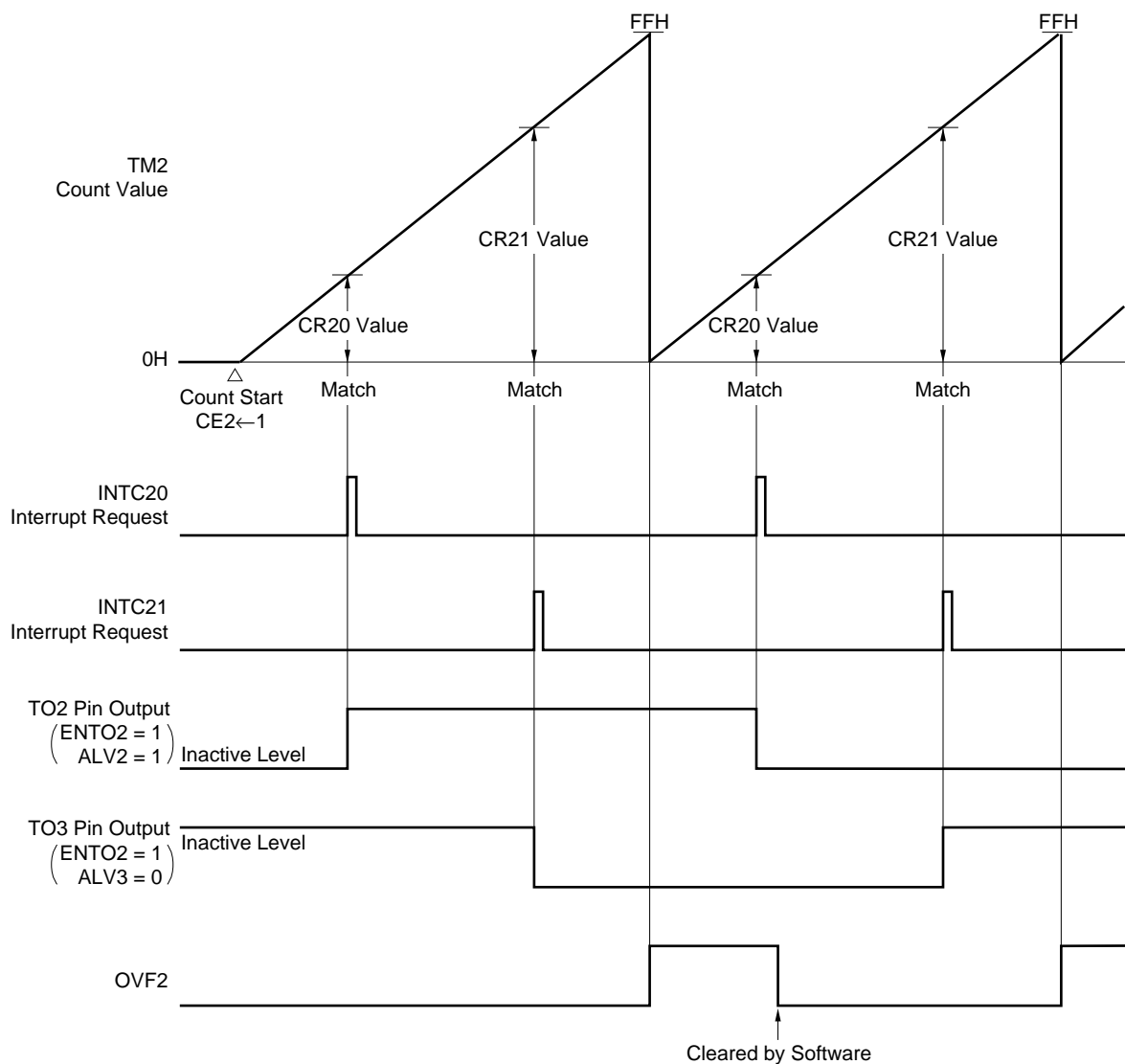
## 10.7 COMPARE REGISTER, CAPTURE/COMPARE REGISTER AND CAPTURE REGISTER OPERATION

### 10.7.1 Compare Operations

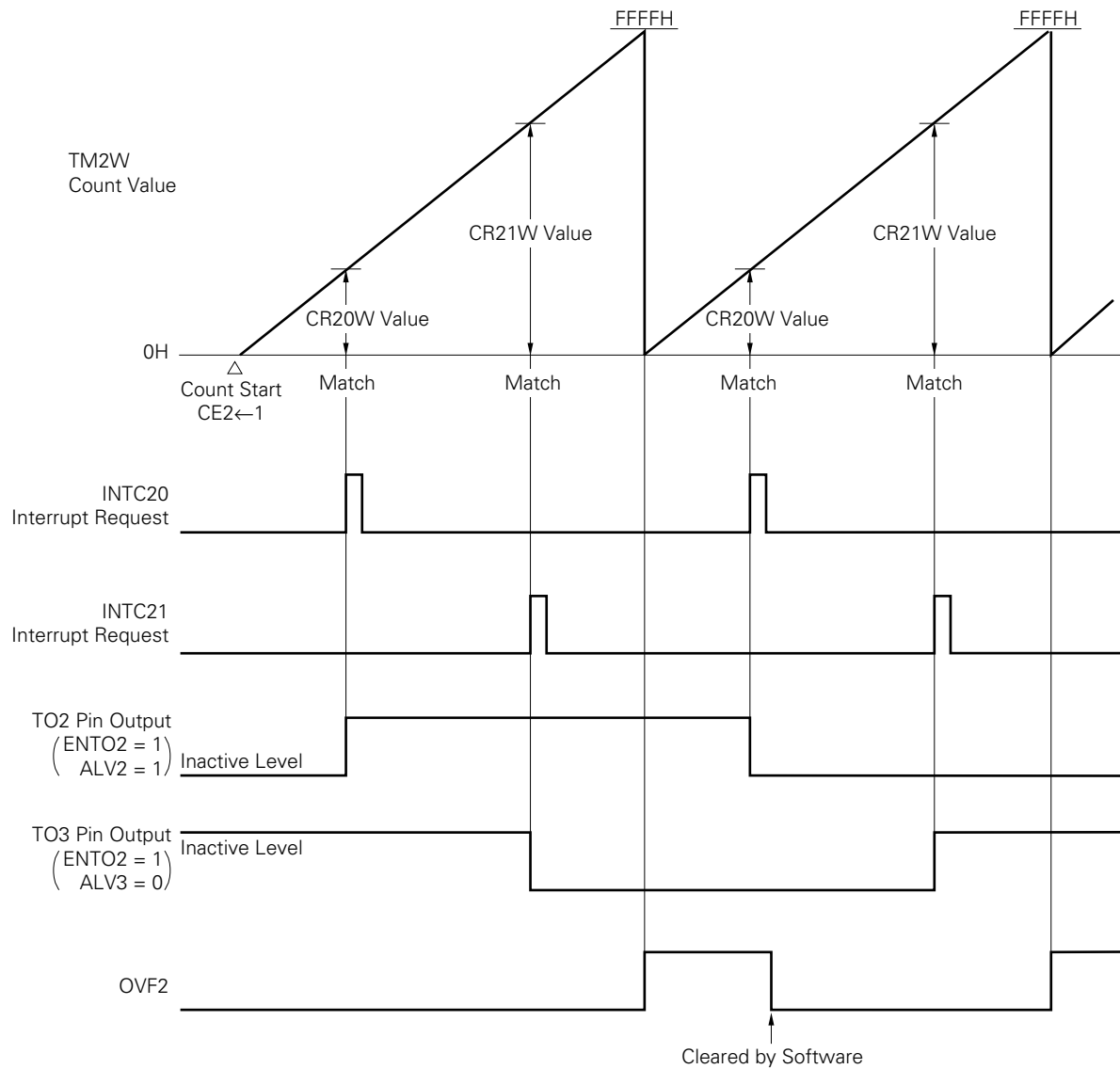
Timer/counter 2 performs compare operations in which the value set in the compare register (CR20) and the capture/compare register (CR21) specified for compare operation is compared with the timer register 2 (TM2) count value.

If the count value of TM2 matches the preset value of the CR20, and CR21 when a compare operation is performed, as the result of the count operation, a match signal is sent to the output control circuit, and an interrupt request signal (INTC20 or INTC21) is generated at the same time.

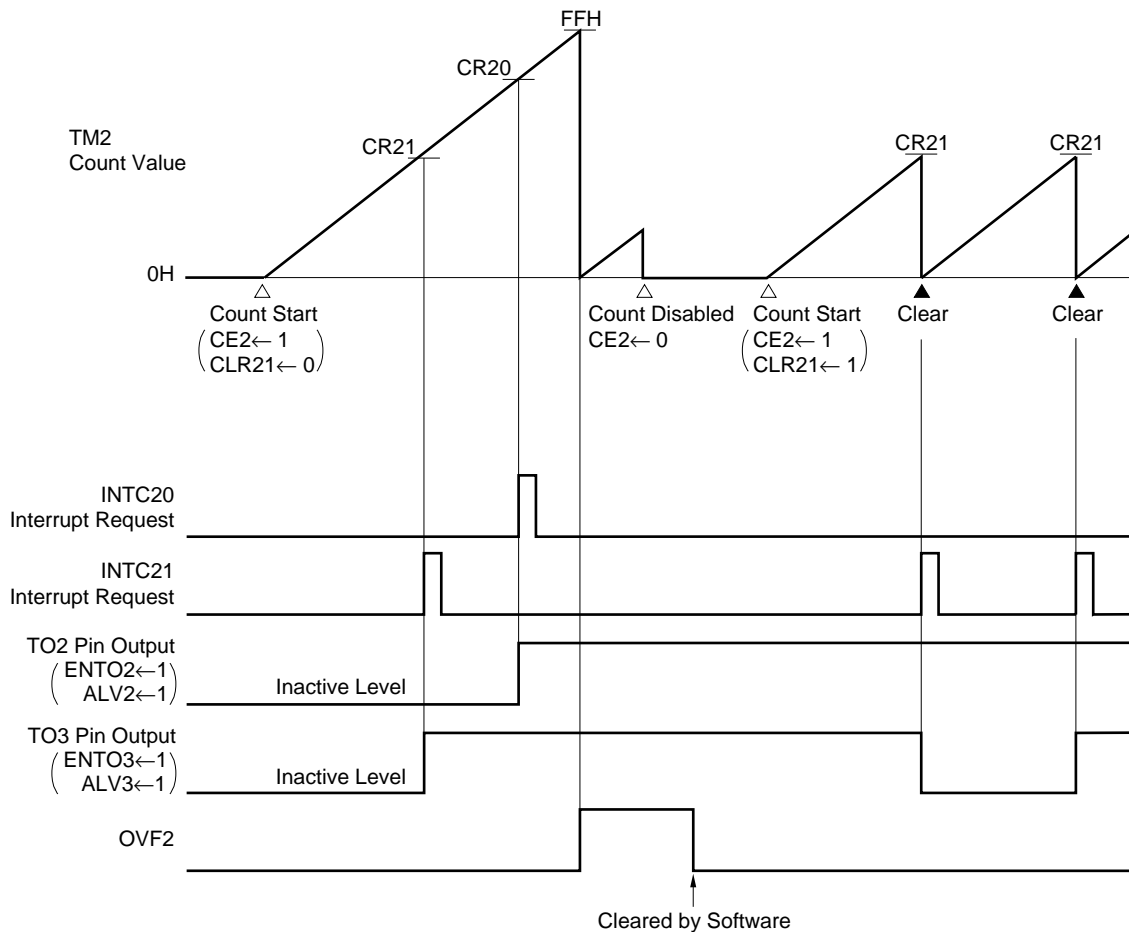
After a match with the CR20 or CR21 value, the TM2 contents can be cleared, and the timer functions as an interval timer that repeatedly counts up to the value set in the CR20 or CR21.

**Figure 10-15 Compare Operation in 8-Bit Operating Mode**

**Remark** CLR21 = 0, CLR22 = 0, BW2 = 0

**Figure 10-16 Compare Operation in 16-Bit Operating Mode**

**Remark** CLR21 = 0, CLR22 = 0, BW2 = 1

**Figure 10-17 TM2 Clearance after Match Detection**

**Remark** CLR22 = 0

### 10.7.2 Capture Operations

Timer/counter 2 performs capture operations in which the timer register 2 (TM2) count value is fetched into the capture register in synchronization with an external trigger, and retained there.

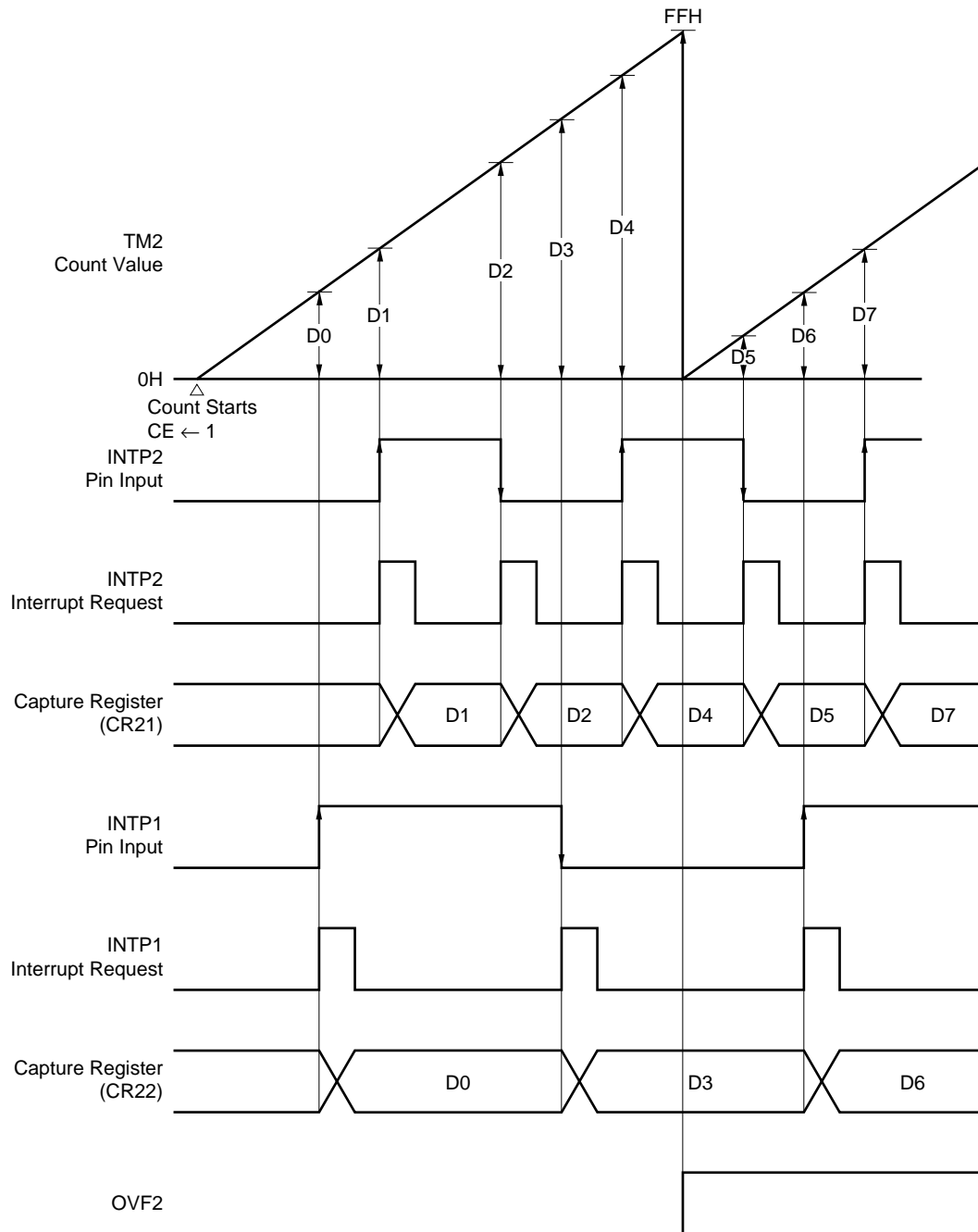
A valid edge detected from the input of the external interrupt request input pins (INTP1/INTP2) is used as the external trigger (capture trigger). The count value of TM2 in the process of being counted in synchronization with the capture trigger is fetched into the capture register (CR22) in synchronization with INTP1, or into the capture/compare register (CR21) when a capture operation is specified in synchronization with INTP2, and is retained there.

The contents of CR21 and CR22 are retained until the next capture triggers corresponding to CR21 and CR22 are generated.

The capture trigger valid edge is set by means of external interrupt mode register 0 (INTM0). If both rising and falling edges are set as capture triggers, the width of pulses input from off-chip can be measured, and if a capture trigger is generated by a single edge, the input pulse cycle can be measured.

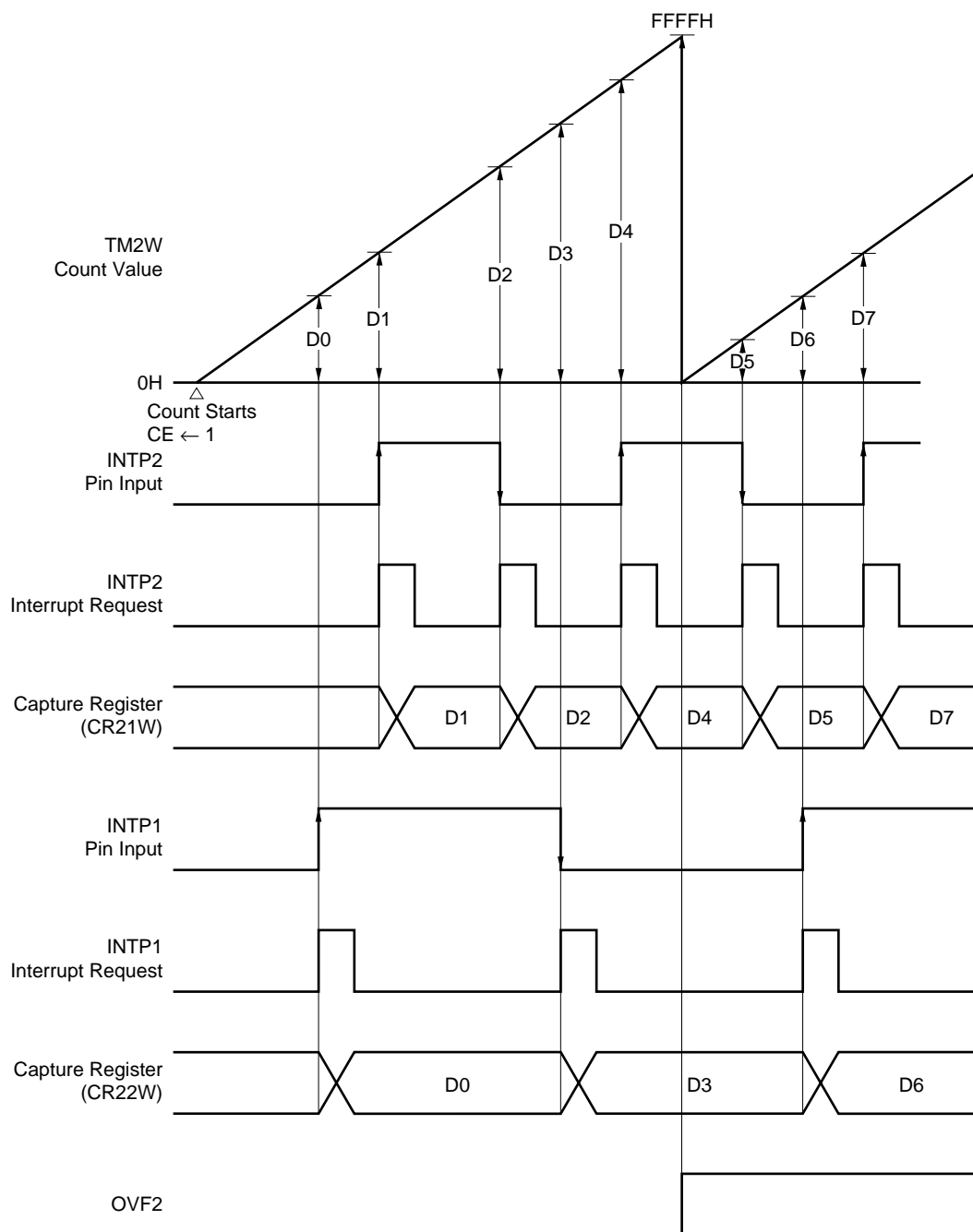
See **Figure 19-1 in CHAPTER 19 EDGE DETECTION FUNCTION** for details of the INTM0 format.

When CR21 is used as a capture register, TM2 can be cleared as soon as the contents of TM2 have been captured by capture trigger to CR21 or CR22.

**Figure 10-18 Capture Operation in 8-Bit Operating Mode**

**Remark** Dn: TM2 count value (n = 0, 1, 2, ...)  
 CM21 = 1, CLR21 = 0, CLR22 = 0, BW2 = 0

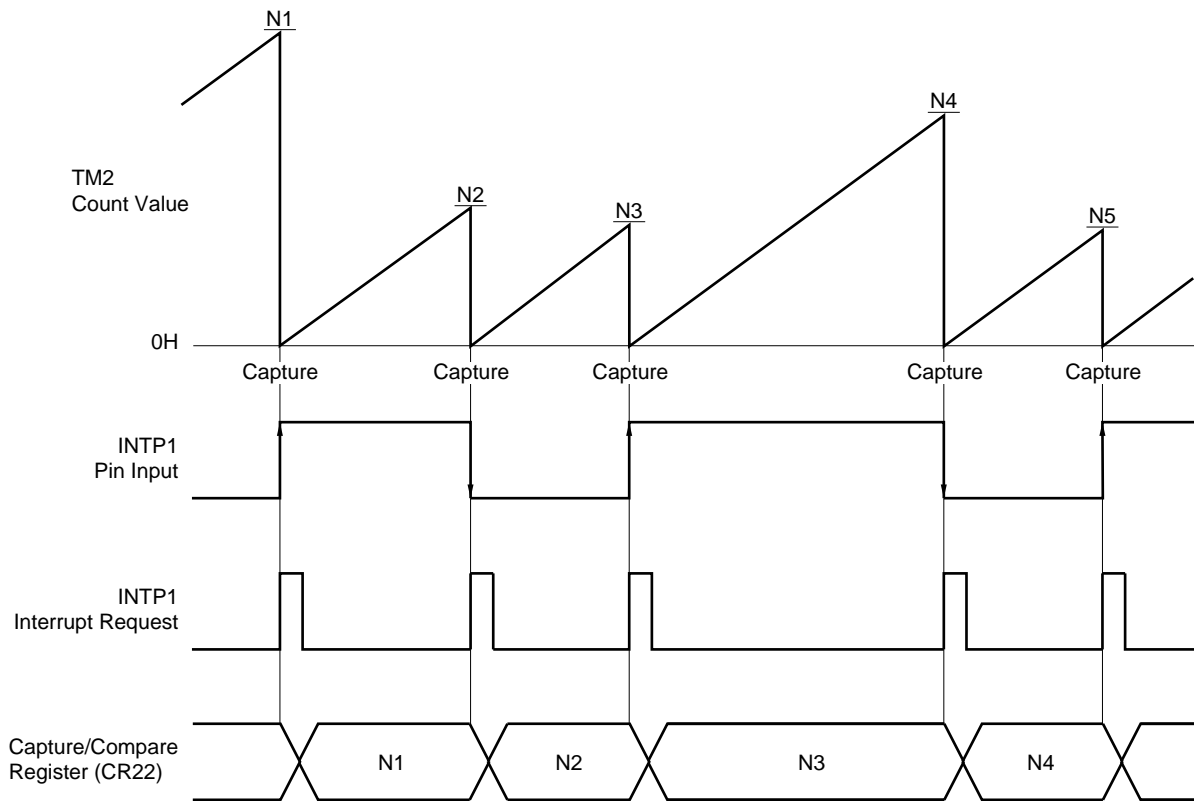
Figure 10-19 Capture Operation in 16-Bit Operating Mode



**Remark** Dn: TM2W count value (n = 0, 1, 2, ...)

CM21 = 1, CLR21 = 0, CLR22 = 0, BW2 = 0



**Figure 10-20 TM2 Clearance after Capture Operation**

**Remark** CLR21 = 0, CLR22 = 1

## 10.8 BASIC OPERATION OF OUTPUT CONTROL CIRCUIT

The output control circuit controls the timer output pins (TO2/TO3) level by means of match signals from the compare register (CR22). The operation of the output control circuit is determined by the timer output control register (TOC) and capture/compare control register 2 (CRC2) (see **Table 10-5**). When TO2/TO3 signal is output to a pin, the relevant pin must be in control mode in the port 3 mode register (PMC3).

Table 10-5 Timer Output (TO2/TO3) Operations

TOC				CRC2				TMC1	TO3	TO2
ENTO3	ALV3	ENTO2	ALV2	MOD1	MOD0	CLR22	CLR21	CMD2		
0	0/1	0	0/1	×	×	×	×	×	High/low level fixed	High/low level fixed
0	0/1	1	0/1	0	0	×	×	×	High/low level fixed	Toggle output (active-low/high)
1	0/1	0	0/1	0	0	×	×	×	Toggle output (active-low/high)	High/low level fixed
1	0/1	1	0/1	0	0	×	×	×	Toggle output (active-low/high)	Toggle output (active-low/high)
0	0/1	1	0/1	0	1	0	0	0	High/low level fixed	PWM output (active-high/low)
1	0/1	0	0/1	0	1	0	0	0	Toggle output (active-low/high)	High/low level fixed
1	0/1	1	0/1	0	1	0	0	0	Toggle output (active-low/high)	PWM output (active-high/low)
0	0/1	1	0/1	1	0	0	0	0	High/low level fixed	PWM output (active-high/low)
1	0/1	0	0/1	1	0	0	0	0	PWM output (active-high/low)	High/low level fixed
1	0/1	1	0/1	1	0	0	0	0	PWM output (active-high/low)	PWM output (active-high/low)
0	0/1	1	0/1	1	1	0	1	0	High/low level fixed	PPG output (active-high/low)
1	0/1	0	0/1	1	1	0	1	0	Toggle output (active-low/high)	High/low level fixed
1	0/1	1	0/1	1	1	0	1	0	Toggle output (active-low/high)	PPG output (active-high/low)

\* CLR22 is normally set to 0 in this case.

- Remarks**
1. 0/1 in the ALVn (n = 2, 3) columns correspond to the items on the left and right of the slash ("/") in the TOn (n = 2, 3) columns respectively.
  2. "×" indicates 0 or 1.
  3. Combinations not shown in this table are prohibited to use in that combination.

### 10.8.1 Basic Operation

Setting (1) the ENTOn (n = 2, 3) bit of the timer output control register (TOC) enables timer output (TON: n = 2, 3) to be varied at a timing in accordance with the settings of MOD0, MOD1 and CLR21 bits of capture/compare control register 2 (CRC2).

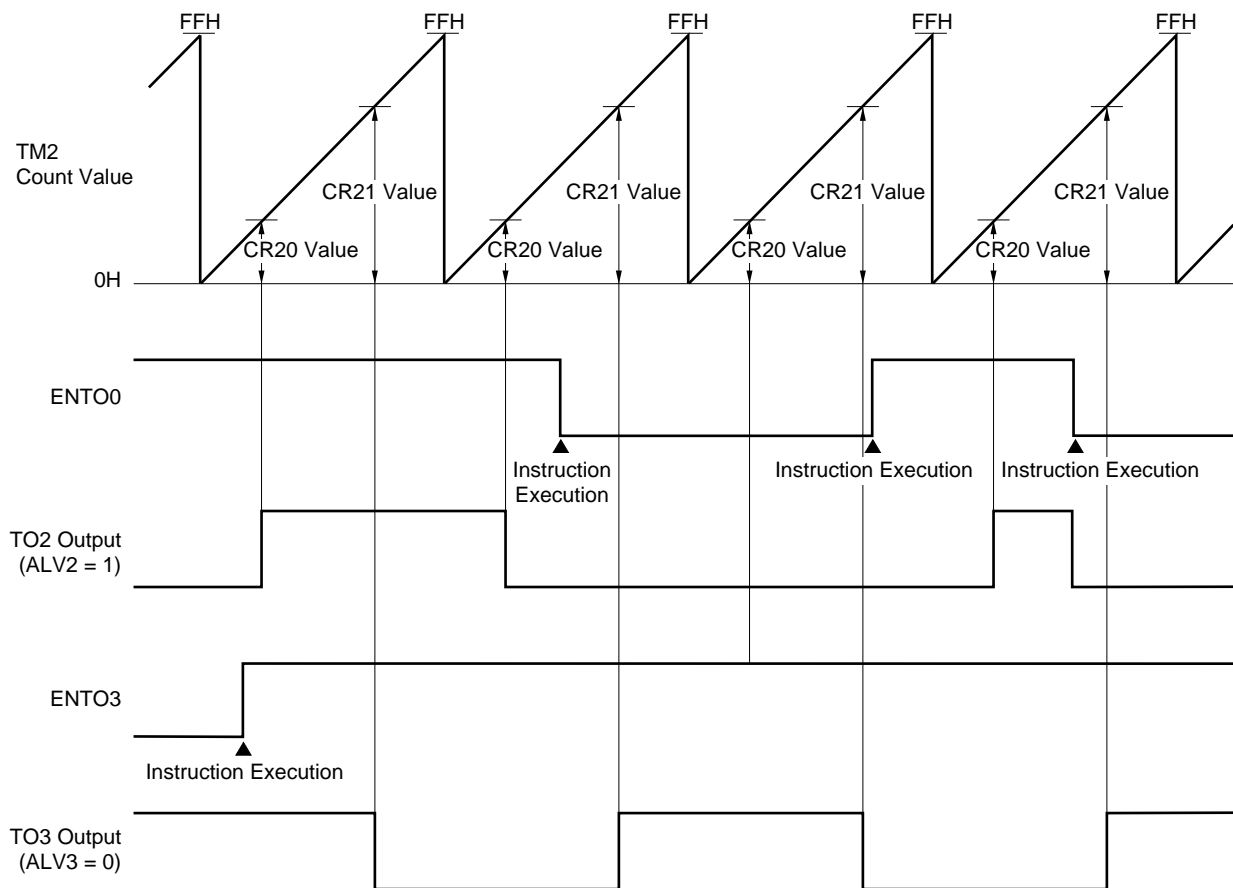
Clearing (0) ENTOn sets the TON to a fixed level. The fixed level is determined by the ALVn (n = 2/3) bit of the TOC. The level is high when ALVn is 0, and low when 1.

### 10.8.2 Toggle Output

Toggle output is an operating mode in which the output level is inverted each time the compare register (CR20/CR21) value coincides with the timer register 2 (TM2) value. The output level of timer output (TO2) is inverted by a match between CR20 and TM2, and the output level of timer output (TO3) is inverted by a match between CR21 and TM2.

- ★ When timer/counter 2 is stopped by clearing (0) the CE2 bit of the timer control register 1 (TMC1), the inactive level ( $\overline{ALVn}$ : n = 0, 1) is output.

Figure 10-21 Toggle Output Operation



**Table 10-6 TO2/TO3 Toggle Output ( $f_{xx} = 25 \text{ MHz}$ )**

Count Clock	Minimum Pulse Width	Maximum Pulse Width
$f_{xx}/8$	$8/f_{xx}$ (0.32 $\mu\text{s}$ )	$2^{16} \times 8/f_{xx}$ (21.0 ms)
$f_{xx}/16$	$16/f_{xx}$ (0.64 $\mu\text{s}$ )	$2^{16} \times 16/f_{xx}$ (41.9 ms)
$f_{xx}/32$	$32/f_{xx}$ (1.28 $\mu\text{s}$ )	$2^{16} \times 32/f_{xx}$ (83.9 ms)
$f_{xx}/64$	$64/f_{xx}$ (2.56 $\mu\text{s}$ )	$2^{16} \times 64/f_{xx}$ (168 ms)
$f_{xx}/128$	$128/f_{xx}$ (5.12 $\mu\text{s}$ )	$2^{16} \times 128/f_{xx}$ (336 ms)
$f_{xx}/256$	$256/f_{xx}$ (10.2 $\mu\text{s}$ )	$2^{16} \times 256/f_{xx}$ (671 ms)
$f_{xx}/512$	$512/f_{xx}$ (20.5 $\mu\text{s}$ )	$2^{16} \times 512/f_{xx}$ (1.34 s)
$f_{xx}/1024$	$1024/f_{xx}$ (41.0 $\mu\text{s}$ )	$2^{16} \times 1024/f_{xx}$ (2.68 s)
$f_{xx}/2048$	$2048/f_{xx}$ (81.9 $\mu\text{s}$ )	$2^{16} \times 2048/f_{xx}$ (5.37 s)

**10.8.3 PWM Output****(1) Basic operation of PWM output**

In this mode, a PWM signal with the period in which timer register 2 (TM2) reaches a full count used as one cycle is output. The timer output (TO2) pulse width is determined by the value of compare register (CR20), and the timer output (TO3) pulse width is determined by the value of compare register (CR21). When this function is used, the CLR21 bit and CLR22 bit of capture/compare control register 2 (CRC2) and the CMD2 bit of timer control register 1 (TMC1) must be set to 0.

The pulse cycle and pulse width are as shown below.

**(a) BW2 = 0**

- PWM cycle =  $256 \times x/f_{xx}$
- PWM pulse width =  $CR2n \times x/f_{xx}^*$ ;  $x = 8, 16, 32, 64, 128, 256, 512, 1024, 2048$

\* The CR2n cannot be set to 0.

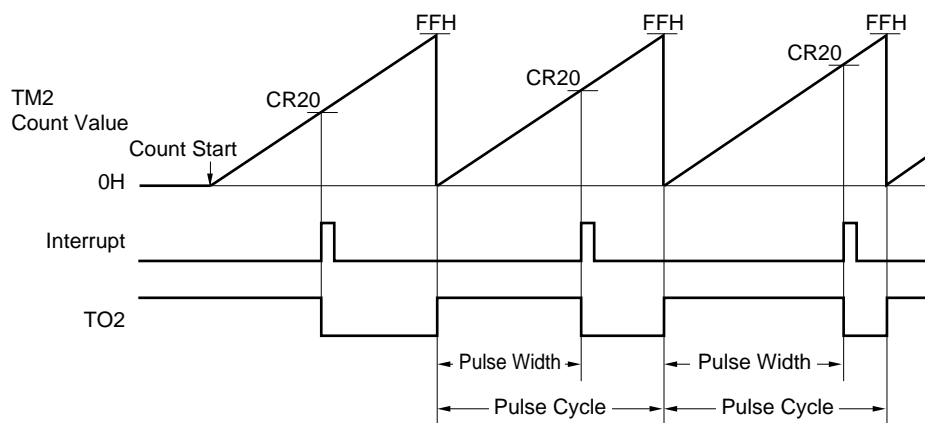
- Duty =  $\frac{\text{PWM pulse width}}{\text{PWM}} = \frac{CR2n}{256}$

**(b) BW2 = 1**

- PWM cycle =  $65536 \times x/f_{xx}$
- PWM pulse width =  $CR2n \times x/f_{xx}^*$ ;  $x = 8, 16, 32, 64, 128, 256, 512, 1024, 2048$

\* The CR2n cannot be set to 0.

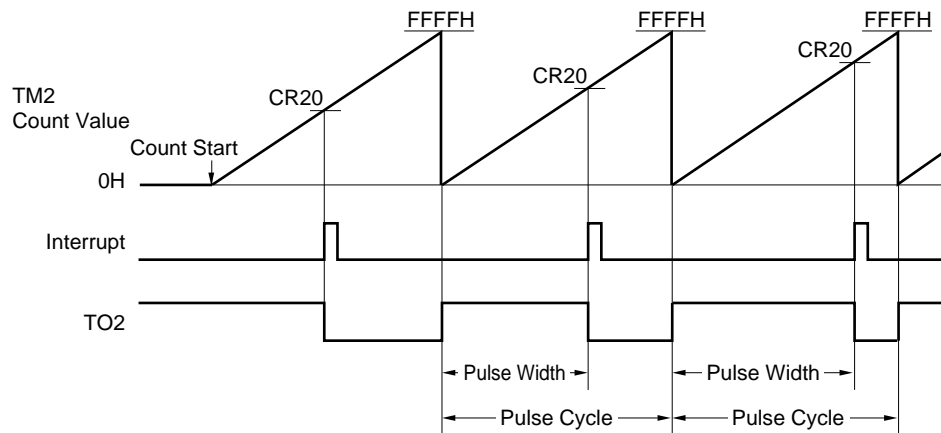
- Duty =  $\frac{\text{PWM pulse width}}{\text{PWM cycle}} = \frac{CR2n}{65536}$

**Figure 10-22 PWM Pulse Output (BW2 = 0)**

**Remark** ALV2 = 0

**Table 10-7 TO2/TO3 PWM Cycle ( $f_{xx} = 25 \text{ MHz}$ , BW2 = 0)**

Count Clock	Minimum Pulse Width [ $\mu\text{s}$ ]	PWM Cycle [ms]	PWM Frequency [Hz]
$f_{xx}/8$	0.32	0.08	12207
$f_{xx}/16$	0.64	0.16	6104
$f_{xx}/32$	1.28	0.33	3052
$f_{xx}/64$	2.56	0.66	1526
$f_{xx}/128$	5.12	1.31	763
$f_{xx}/256$	10.2	2.61	381
$f_{xx}/512$	20.5	5.25	191
$f_{xx}/1024$	41.0	10.5	95
$f_{xx}/2048$	81.9	21.0	48

**Figure 10-23 PWM Pulse Output (BW2 = 1)**

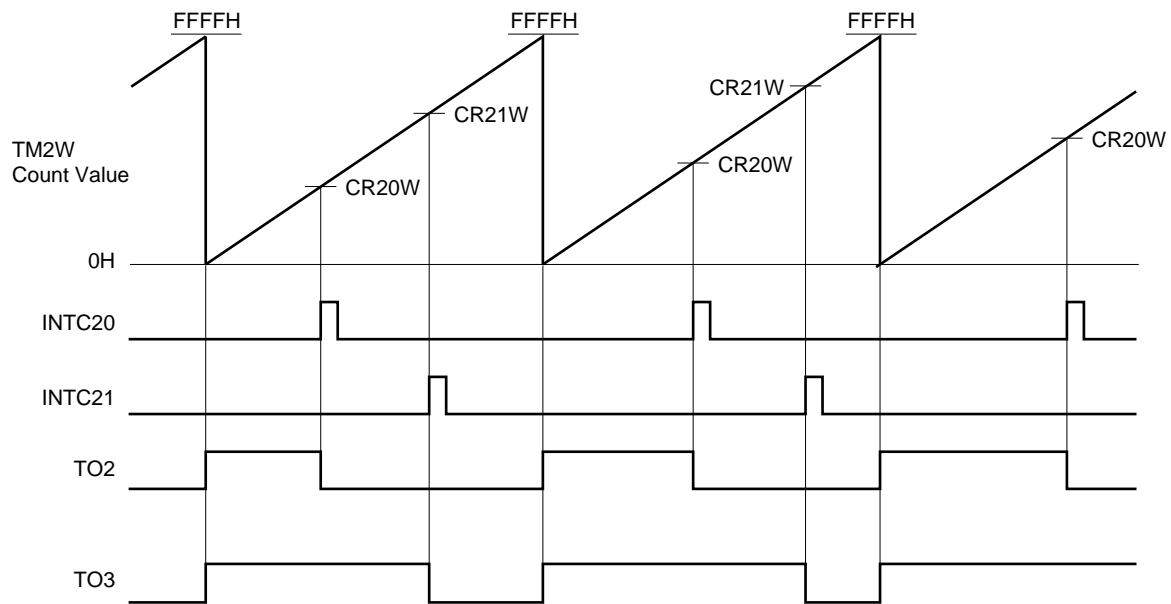
**Remark** ALV2 = 0

**Table 10-8 TO2/TO3 PWM Cycle ( $f_{xx} = 25 \text{ MHz}$ , BW2 = 1)**

Count Clock	Minimum Pulse Width [ $\mu\text{s}$ ]	PWM Cycle [s]	PWM Frequency [Hz]
$f_{xx}/8$	0.32	0.02	47.7
$f_{xx}/16$	0.64	0.04	23.8
$f_{xx}/32$	1.28	0.08	11.9
$f_{xx}/64$	2.56	0.17	6.0
$f_{xx}/128$	5.12	0.34	3.0
$f_{xx}/256$	10.2	0.67	1.5
$f_{xx}/512$	20.5	1.34	0.7
$f_{xx}/1024$	41.0	2.68	0.4
$f_{xx}/2048$	81.9	5.37	0.2

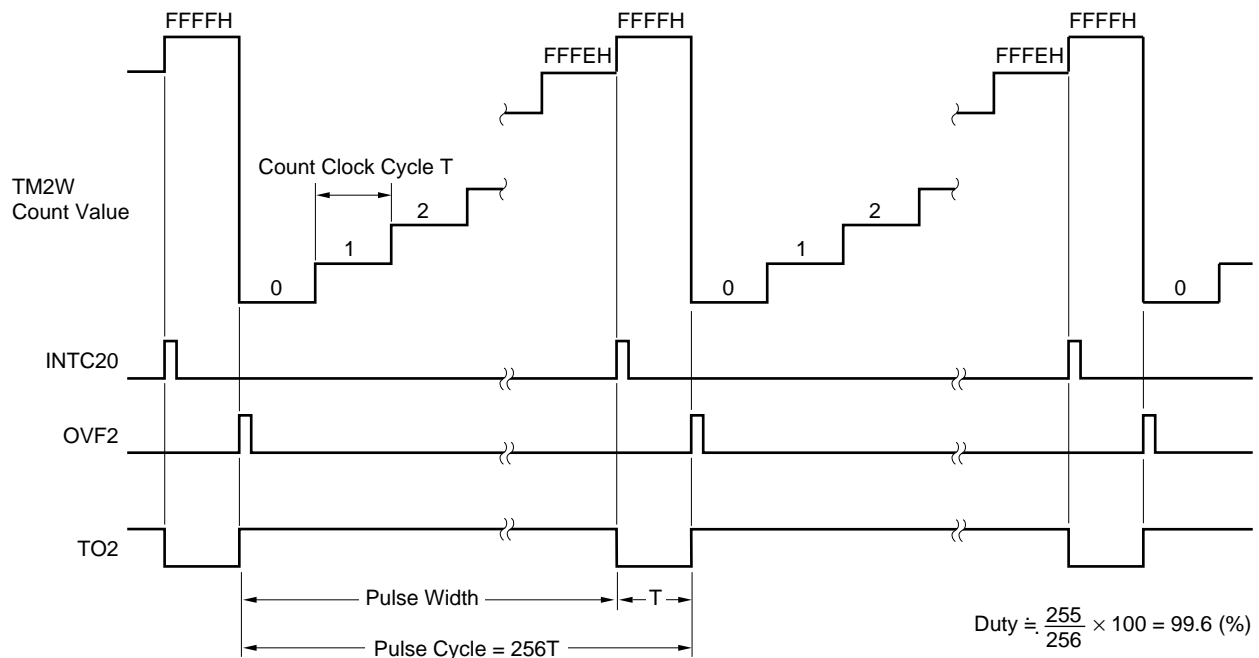
Figure 10-24 shows an example of 2-channel PWM output, and Figure 10-25 shows the case where FFFFH is set in the CR20W.

**Figure 10-24 Example of PWM Output Using TM2W**



**Remark** ALV2 = 0, ALV3 = 0

**Figure 10-25 Example of PWM Output When CR20W = FFFFH**



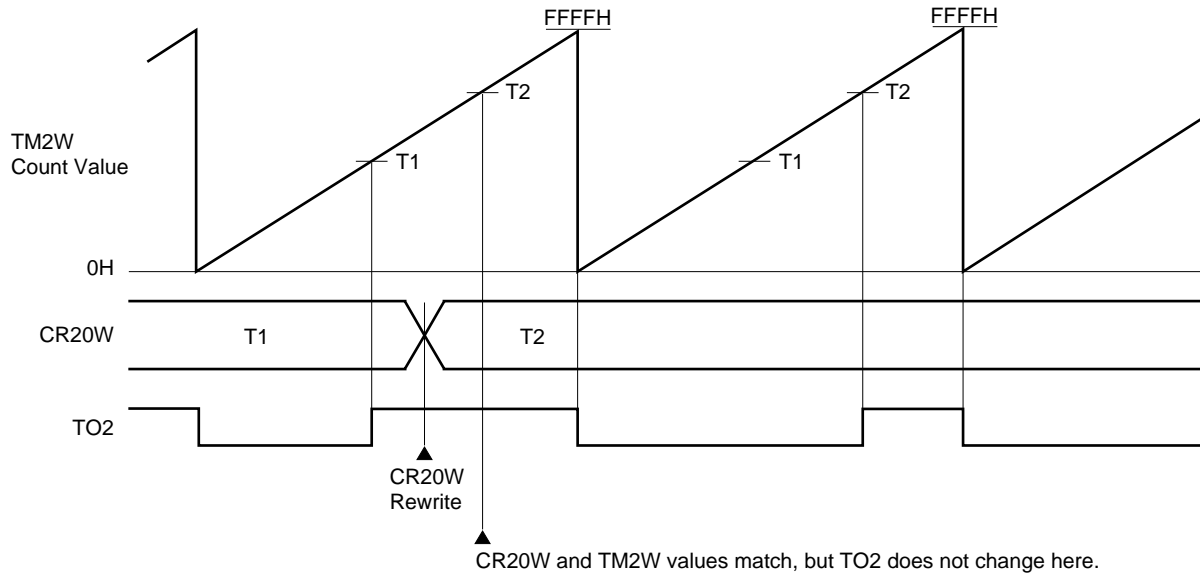
- Remarks**
1. ALV2 = 0
  2.  $T = x/f_{xx}$  ( $x = 8, 16, 32, 64, 128, 256, 512, 1024, 2048$ )



**(2) Rewriting compare registers (CR20, CR21)**

The output level of the timer output (TOn + 2: n + 2 = 2, 3) is not inverted even if the CR2n (n = 0, 1) value matches the timer register 2 (TM2) value more than once during one PWM output cycle.

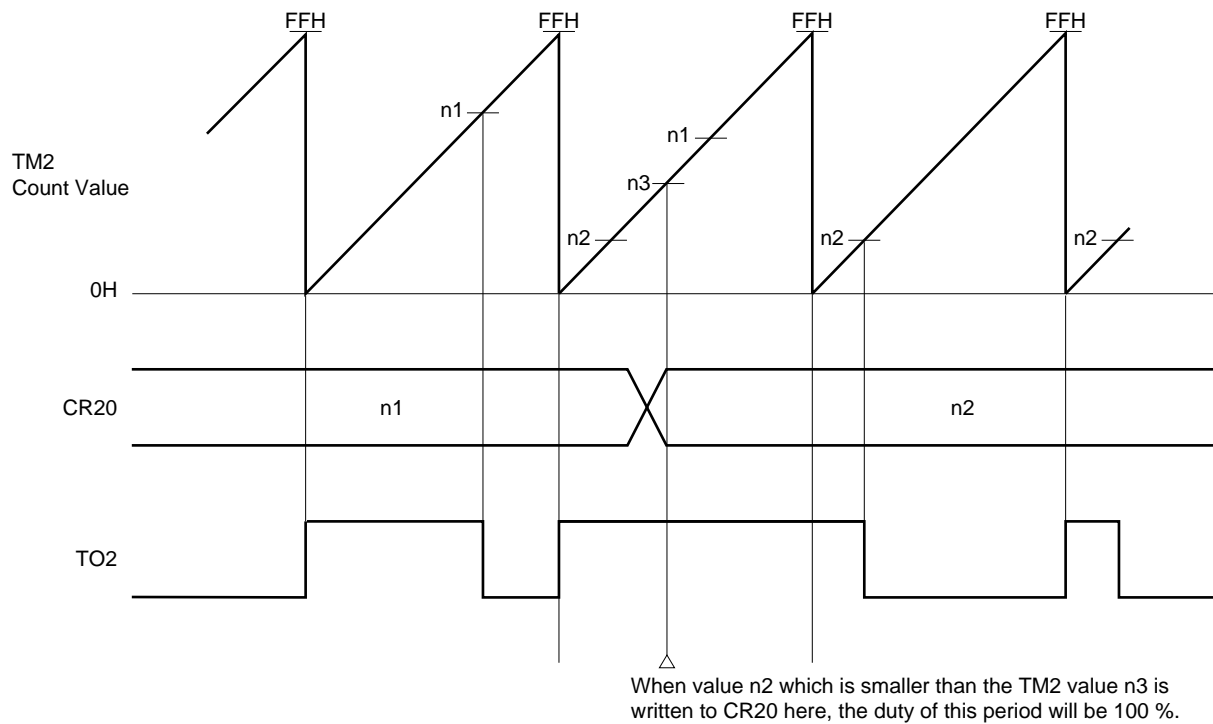
**Figure 10-26 Example of Compare Register (CR20W) Rewrite**



**Remark** ALV2 = 1

If a value smaller than that of the TM2 is set as the CR2n value, a 100 % duty PWM signal will be output. CR2n rewriting should be performed by the interrupt due to a match between TM2 and the CR2n on which the rewrite is performed.

**Figure 10-27 Example of 100 % Duty With PWM Output**

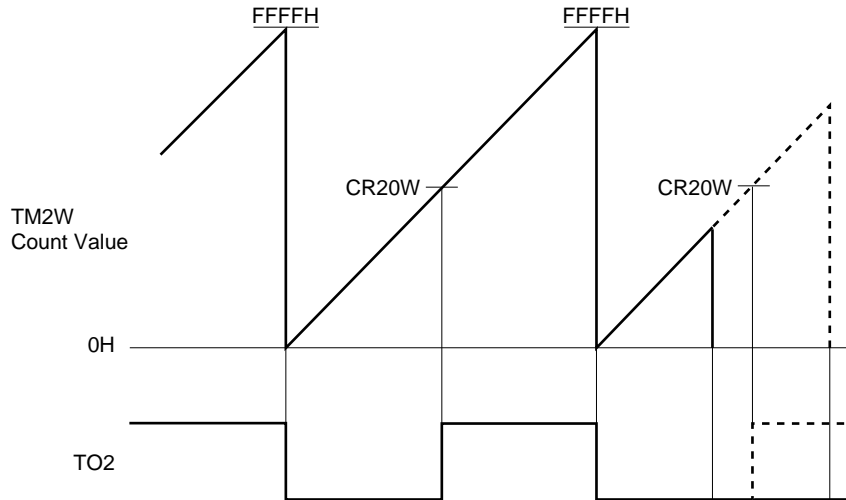


**Remark** ALV2 = 0

**(3) Stopping PWM output**

If timer/counter 2 is stopped by clearing (0) the CE2 bit of the timer control register 1 (TMC1) during PWM signal output, the active level is output.

★

**Figure 10-28 When Timer/Counter 2 is Stopped During PWM Signal Output**

**Remark** ALV2 = 1

**Caution** The output level of the TOn ( $n = 2/3$ ) pin when timer output is disabled ( $ENTOn = 0$ ,  $n = 2/3$ ) is the inverse of the value set in ALVn ( $n = 2/3$ ) bits. Caution is therefore required as the active level is output when timer output is disabled when the PWM output function has been selected.

### 10.8.4 PPG Output

#### (1) Basic operation of PPG output

This function outputs a square-wave with the time determined by compare register CR21 value as one cycle, and the time determined by compare register CR20 value as the pulse width. The PWM output PWM cycle is made variable. This signal can only be output from timer output (TO2).

When this function is used, it is necessary to set the CLR21 bit of capture/compare control register 2 (CRC2) to 1 and the CLR22 bit to 0, and to set the CMD2 bit of timer control register 1 (TMC1) to 0.

The pulse cycle and pulse width are as shown below.

- $\text{PPG cycle} = (\text{CR21} + 1) \times x/f_{xx}; x = 8, 16, 32, 64, 128, 256, 512, 1024, 2048$

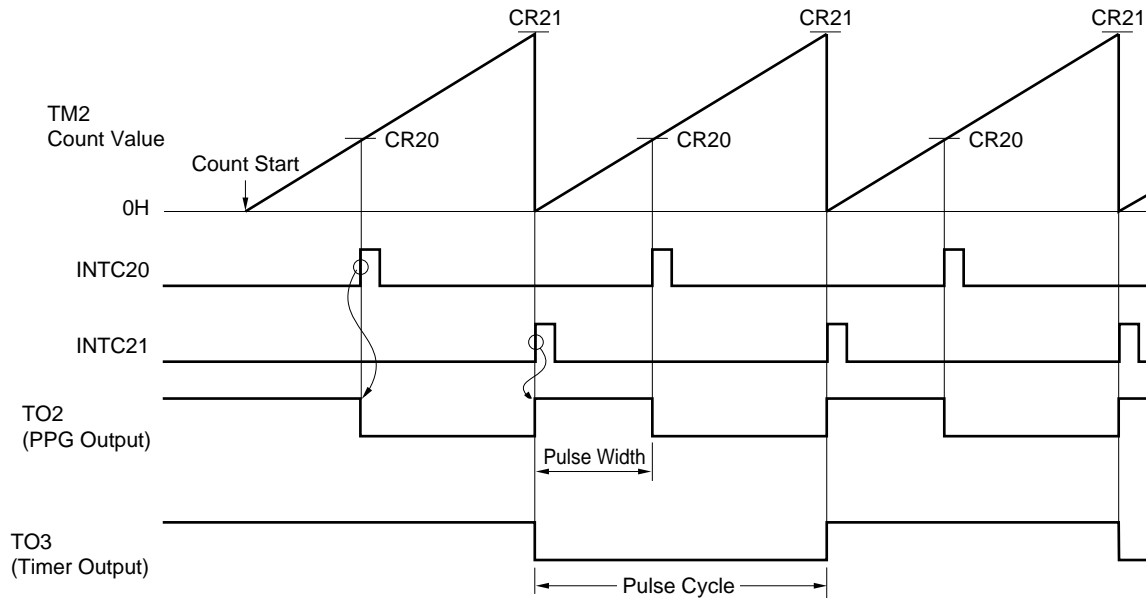
- $\text{PPG pulse width} = \text{CR20} \times x/f_{xx} \text{ where } 1 \leq \text{CR20} \leq \text{CR21}^*$

- $\text{Duty} = \frac{\text{PPG pulse width}}{\text{PPG cycle}} = \frac{\text{CR20}}{\text{CR21} + 1}^*$

\* CR20 and CR21 cannot be set to 0.

Figure 10-29 shows an example of PPG output using timer register 2 (TM2), Figure 10-30 shows an example of the case where CR20 = CR21.

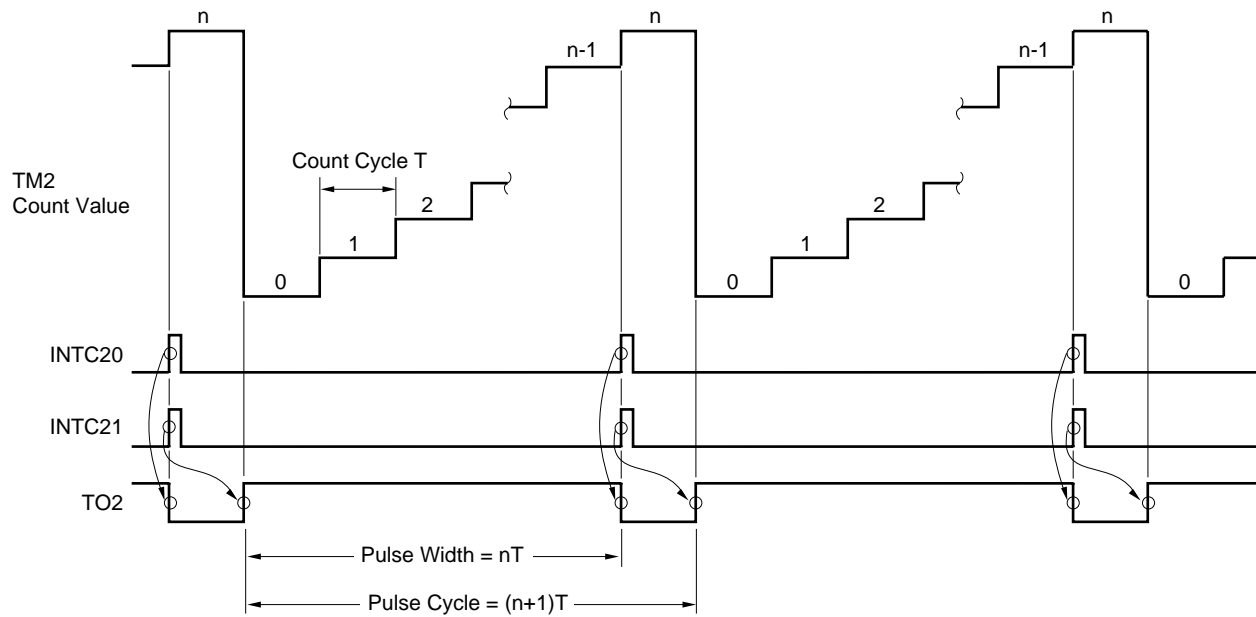
Figure 10-29 Example of PPG Output Using TM2



**Remark** ALV2 = 0, ALV3 = 0

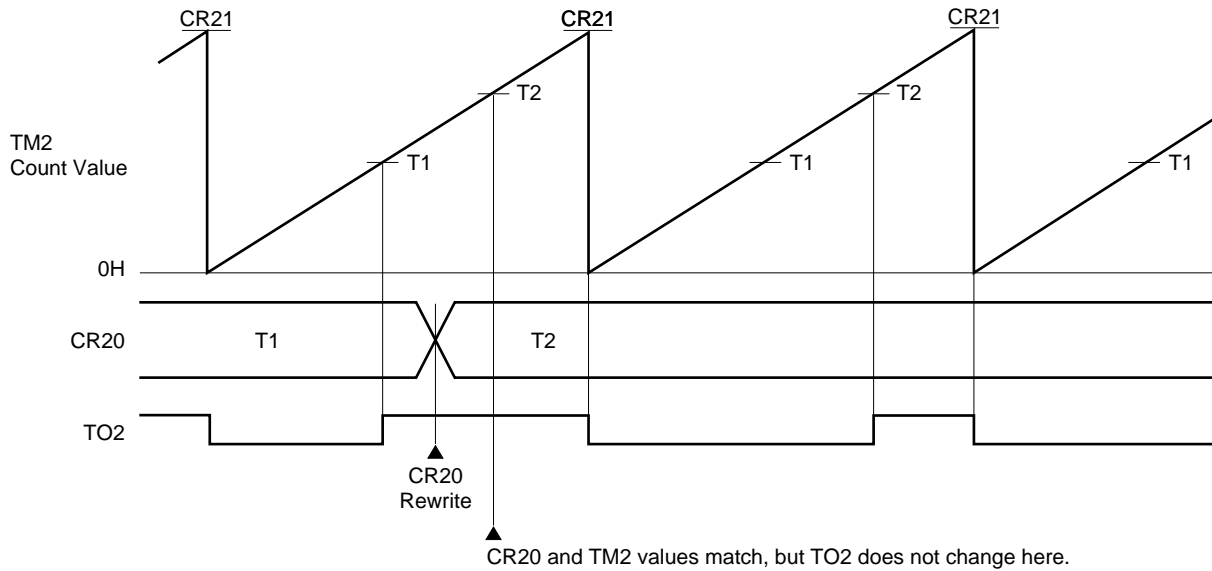
Table 10-9 TO2 PPG Output ( $f_{xx} = 25 \text{ MHz}$ )

Count Clock	Minimum Pulse Width	PPG Cycle	PPG Frequency
$f_{xx}/8$	$0.32 \mu\text{s}$	$0.64 \mu\text{s} - 21.0 \text{ ms}$	$1563 \text{ kHz} - 47.7 \text{ Hz}$
$f_{xx}/16$	$0.64 \mu\text{s}$	$1.28 \mu\text{s} - 41.9 \text{ ms}$	$781 \text{ kHz} - 23.8 \text{ Hz}$
$f_{xx}/32$	$1.28 \mu\text{s}$	$2.56 \mu\text{s} - 83.9 \text{ ms}$	$391 \text{ kHz} - 11.9 \text{ Hz}$
$f_{xx}/64$	$2.56 \mu\text{s}$	$5.12 \mu\text{s} - 168 \text{ ms}$	$195 \text{ kHz} - 5.96 \text{ Hz}$
$f_{xx}/128$	$5.12 \mu\text{s}$	$10.2 \mu\text{s} - 336 \text{ ms}$	$97.7 \text{ kHz} - 2.98 \text{ Hz}$
$f_{xx}/256$	$10.2 \mu\text{s}$	$20.5 \mu\text{s} - 671 \text{ ms}$	$48.8 \text{ kHz} - 1.49 \text{ Hz}$
$f_{xx}/512$	$20.5 \mu\text{s}$	$41.0 \mu\text{s} - 1.34 \text{ s}$	$24.4 \text{ kHz} - 0.75 \text{ Hz}$
$f_{xx}/1024$	$41.0 \mu\text{s}$	$81.9 \mu\text{s} - 2.68 \text{ s}$	$12.2 \text{ kHz} - 0.37 \text{ Hz}$
$f_{xx}/2048$	$81.9 \mu\text{s}$	$164 \mu\text{s} - 5.37 \text{ s}$	$6.10 \text{ kHz} - 0.19 \text{ Hz}$

**Figure 10-30** Example of PPG Output When CR20 = CR21**Remark** ALV2 = 0 $T = x/f_{xx}$  ( $x = 8, 16, 32, 64, 128, 256, 512, 1024, 2048$ )

**(2) Rewriting compare register (CR20)**

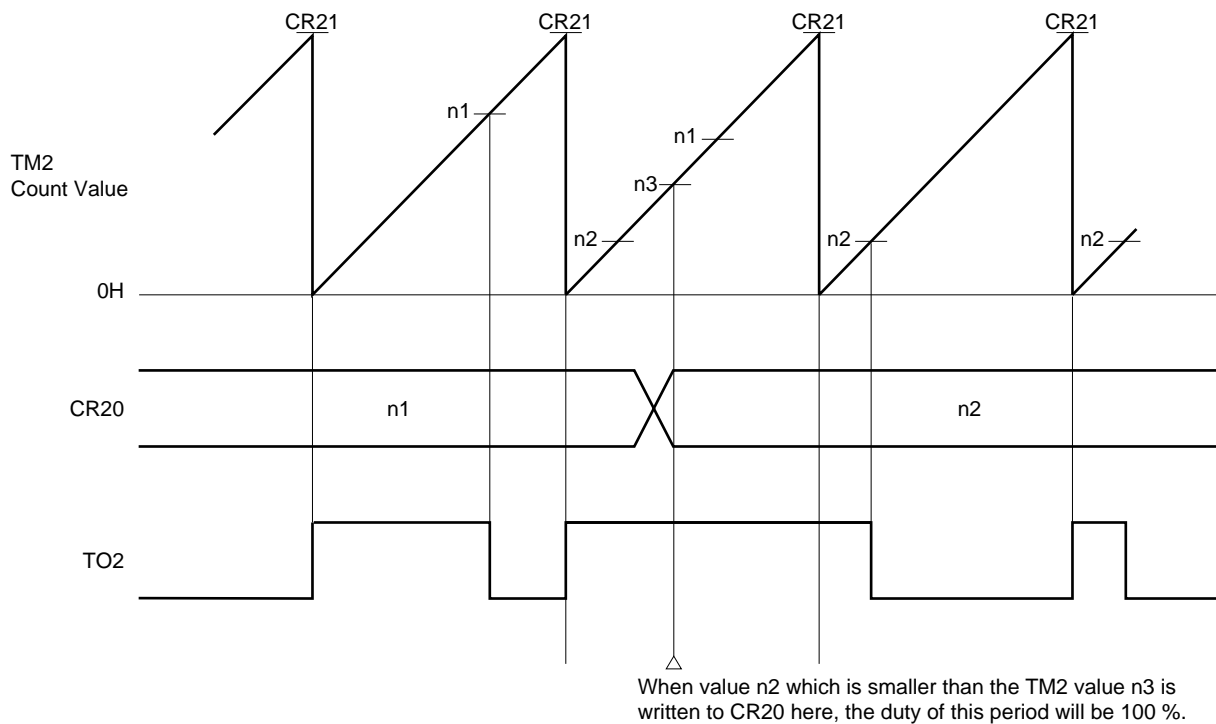
The output level of the timer output (TO2) is not changed even if the CR20 value matches the timer register 2 (TM2) value more than once during one PPG output cycle.

**Figure 10-31 Example of Compare Register Rewrite**

**Remark** ALV2 = 1

If a value equal to or less than the TM2 value is written to CR20 before the CR20 and TM2 match, the duty of that PPG cycle will be 100 %. CR20 rewriting should be performed by the interrupt due to a match between TM2 and CR20.

**Figure 10-32 Example of 100 % Duty With PPG Output**



**Remark** ALV2 = 0

**Caution** If the PPG cycle is extremely short as compared with the time required to acknowledge an interrupt, the value of CR20 cannot be rewritten by interrupt processing that is performed on match between TM2 and CR20. Use another method (for example, to poll the interrupt request flags in software with all the interrupts masked).



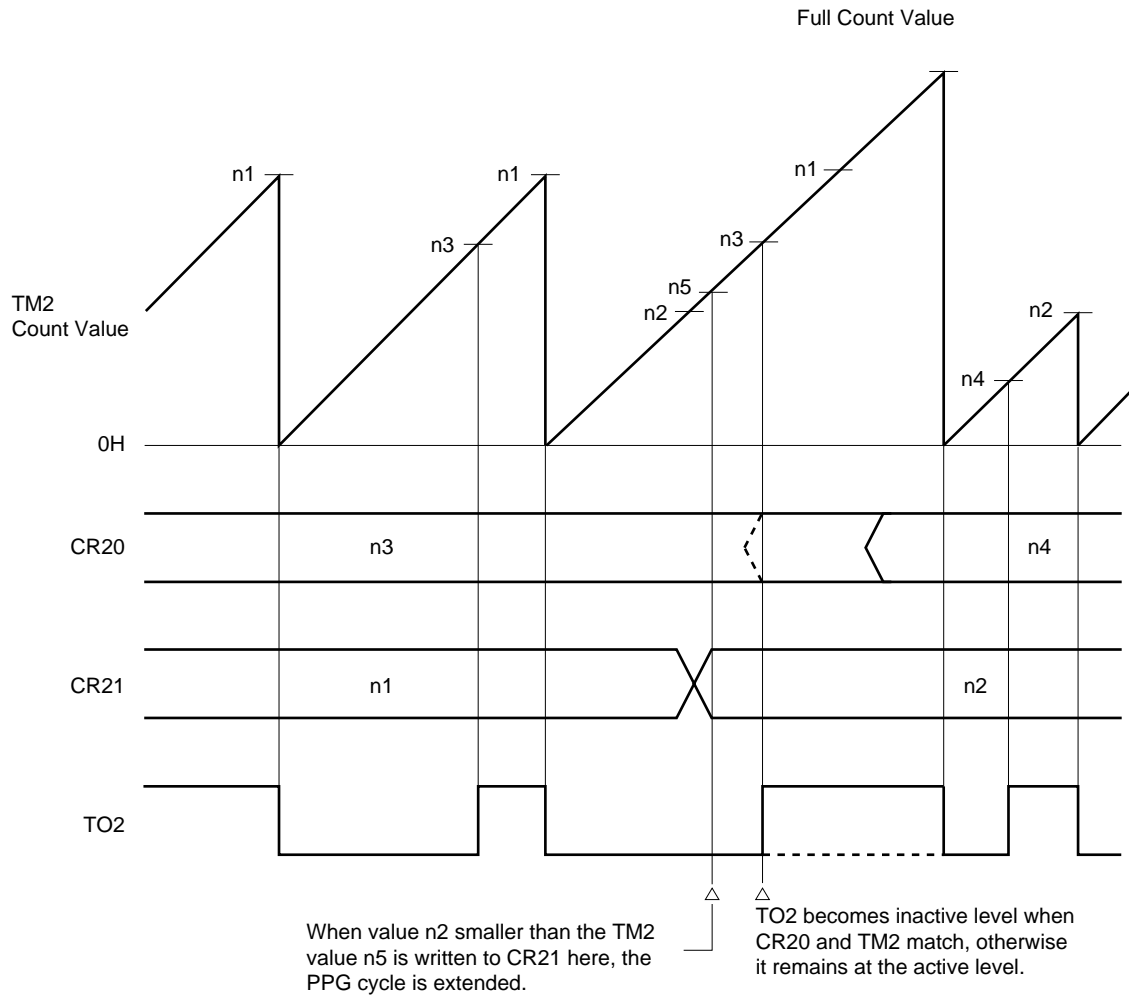
**(3) Rewriting compare register (CR21)**

If the current value of the CR21 is changed to a smaller value, and the CR21 value is made smaller than the register 2 (TM2) value, the PPG cycle at that time will be extended to the time equivalent to a full-count by TM2. If CR21 is rewritten after the compare register (CR20) and TM2 match, the output level at this time will be the inactive level until TM2 overflows and becomes 0, and will then return to normal PPG output.

If CR21 is rewritten before CR20 and TM2 match, the active level will be output until CR20 and TM2 match. If CR20 and TM2 match before TM2 overflows and becomes 0, the inactive level is output at that point. When TM2 overflows and becomes 0, the active level will be output, and normal PPG output will be restored.

CR21 rewriting should be performed by the interrupt due to a match between TM2 and CR21, etc.

**Figure 10-33 Example of Extended PPG Output Cycle**



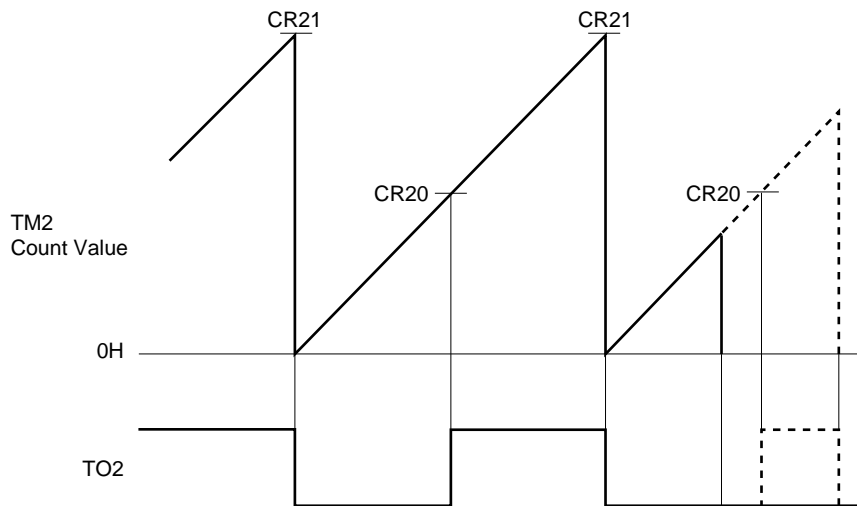
**Remark** ALV2 = 1

**Caution** If the PPG cycle is extremely short as compared with the time required to acknowledge an interrupt, the value of CR2n cannot be rewritten by interrupt processing that is performed on match between timer register 2 (TM2) and compare register (CR2n: n = 0, 1). Use another method (for example, to poll the interrupt request flags in software with all the interrupts masked).

#### (4) Stopping PPG output

If timer/counter 2 is stopped by clearing (0) the CE2 bit of the timer control register 1 (TMC1) during PPG signal output, the active level is output irrespective of the output level at the time timer/counter 2 was stopped.

**Figure 10-34 When Timer/Counter 2 is Stopped During PPG Signal Output**



**Caution** The output level of the TOn (n = 2/3) pin when timer output is disabled (ENTOn = 0, n = 2/3) is the inverse value of the value set in ALVn (n = 2/3) bits. Caution is therefore required as the active level is output when timer output is disabled when the PPG output function has been selected.

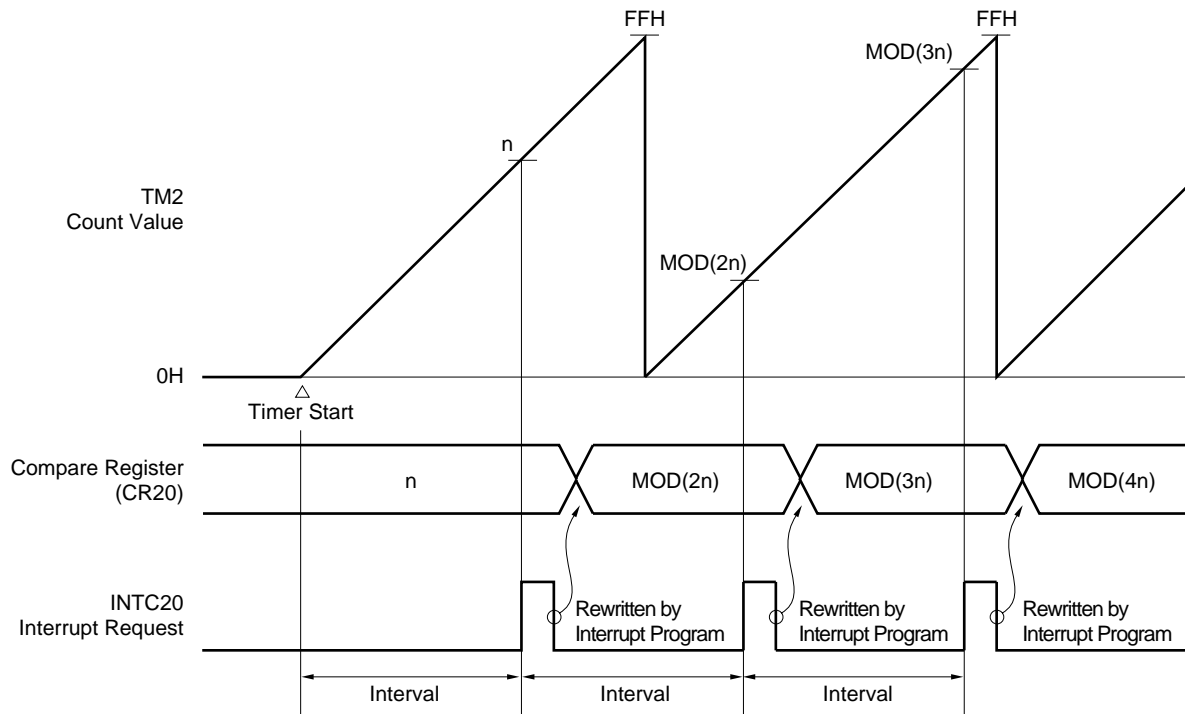
## 10.9 EXAMPLES OF USE

### 10.9.1 Operation as Interval Timer (1)

When timer register 2 (TM2) is made free-running and a fixed value is added to the compare register (CR2n:  $n = 0, 1$ ) in the interrupt service routine, TM2 operates as an interval timer with the added fixed value as the cycle (see **Figure 10-35**).

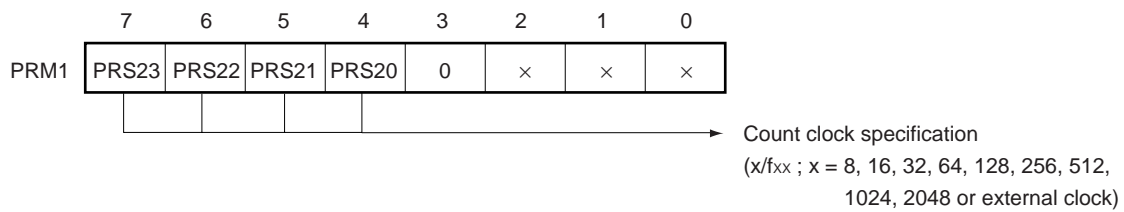
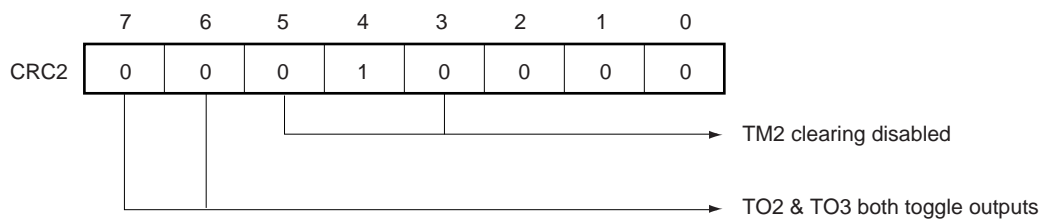
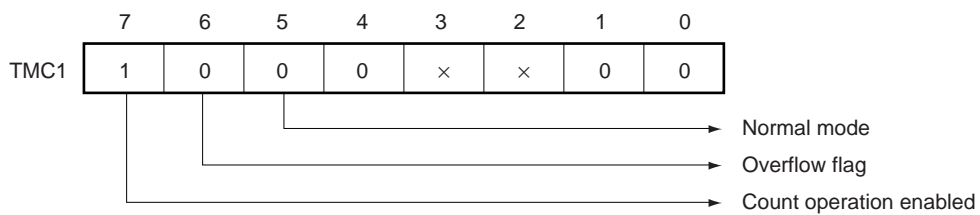
The control register settings are shown in Figure 10-36, the setting procedure in Figure 10-37, and the processing in the interrupt processing routine in Figure 10-38.

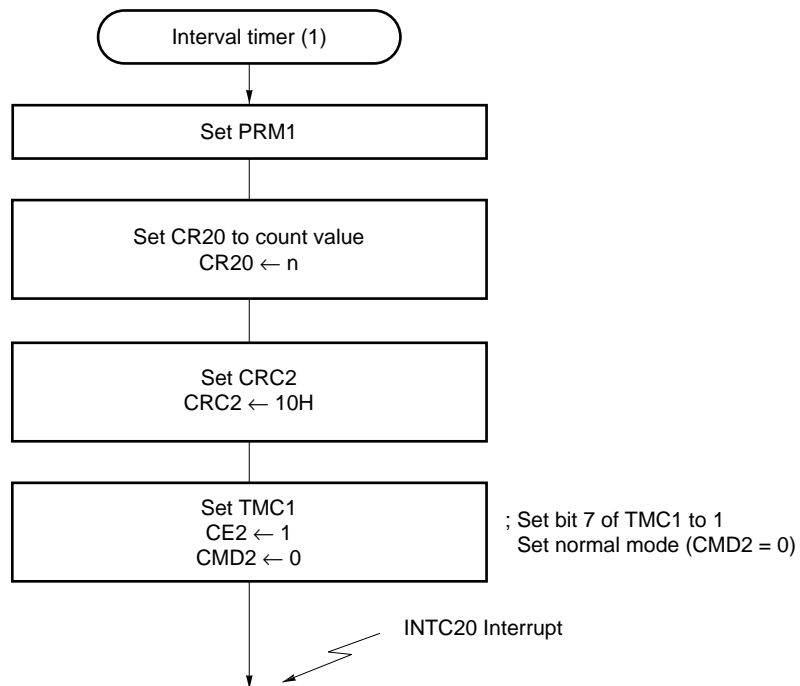
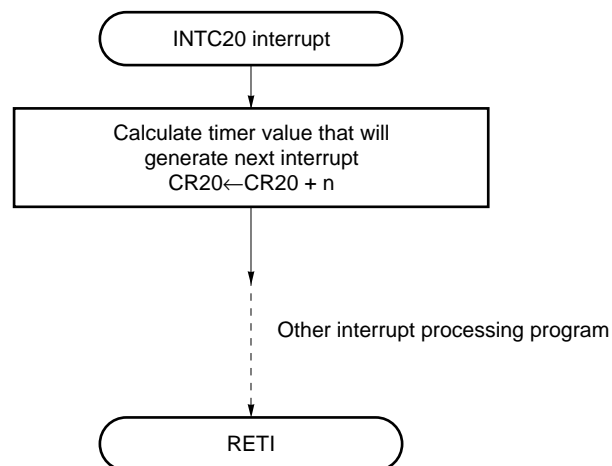
**Figure 10-35 Interval Timer Operation (1) Timing**



**Remark** Interval =  $n \times x / f_{xx}$

$1 \leq n \leq \text{FFH}$ ,  $x = 8, 16, 32, 64, 128, 256, 512, 1024, 2048$

**Figure 10-36 Control Register Settings for Interval Timer Operation (1)****(a) Prescaler mode register 1 (PRM1)****(b) Capture/compare control register 2 (CRC2)****(c) Timer control register 1 (TMC1)**

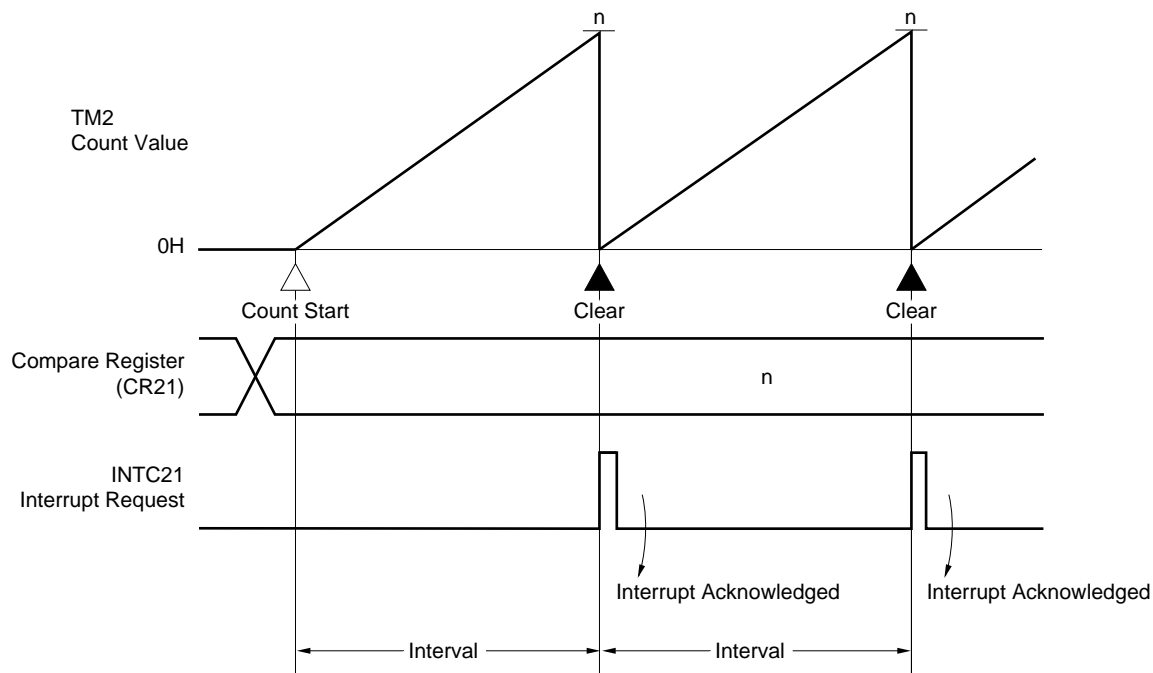
**Figure 10-37 Interval Timer Operation (1) Setting Procedure****Figure 10-38 Interval Timer Operation (1) Interrupt Request Processing**

**10.9.2 Operation as Interval Timer (2)**

TM2 operates as an interval timer that generates interrupts repeatedly with the preset count time as the interval (see **Figure 10-39**).

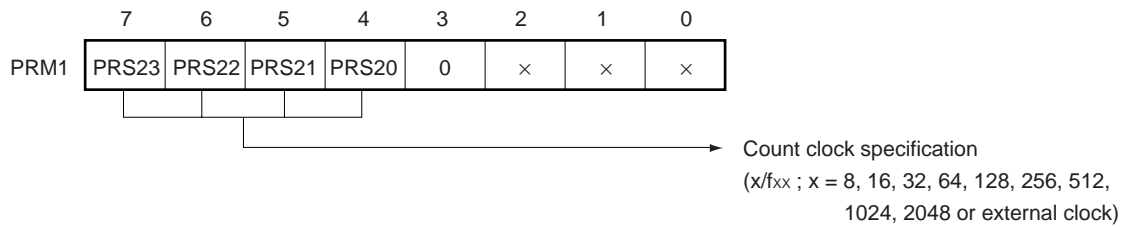
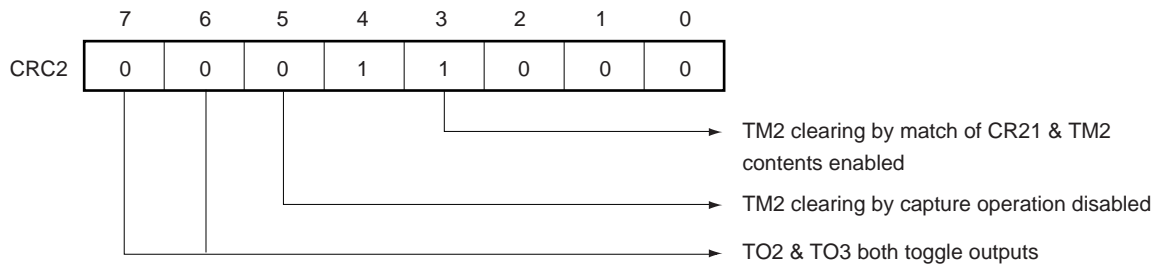
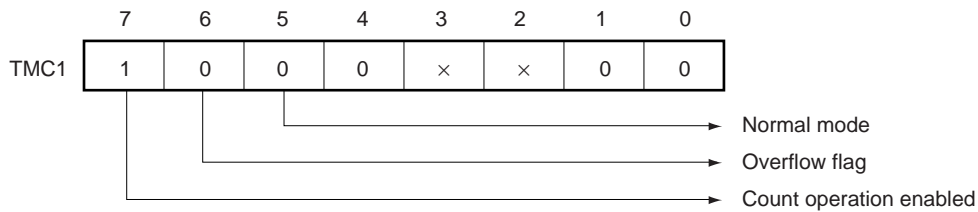
The control register settings are shown in **Figure 10-40**, and the setting procedure in **Figure 10-41**.

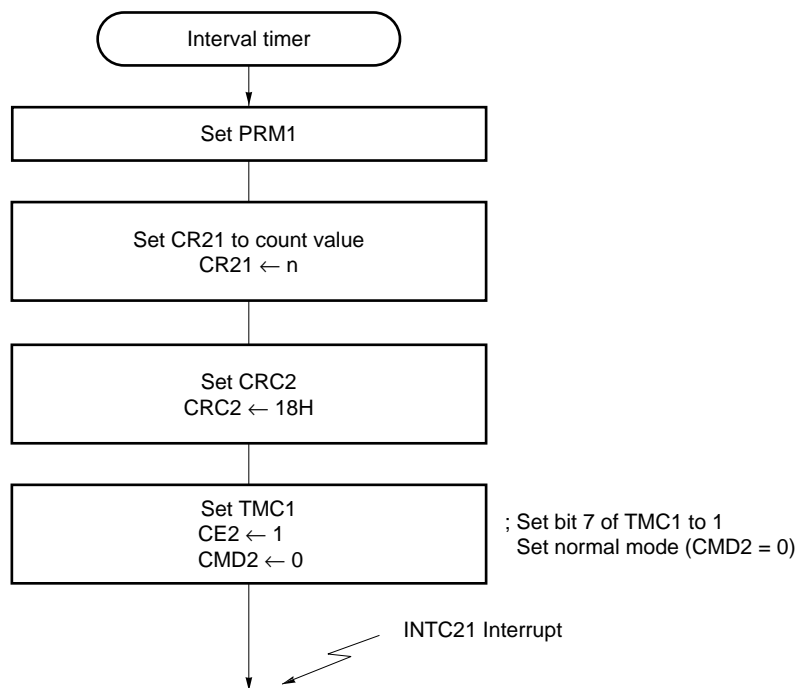
**Figure 10-39 Interval Timer Operation (2) Timing**



**Remark**  $\text{Interval} = (n+1) \times x/f_{xx}$

$0 \leq n \leq \text{FFH}$ ,  $x = 8, 16, 32, 64, 128, 256, 512, 1024, 2048$

**Figure 10-40 Control Register Settings for Interval Timer Operation (2)****(a) Prescaler mode register 1 (PRM1)****(b) Capture/compare control register 2 (CRC2)****(c) Timer control register 1 (TMC1)**

**Figure 10-41 Interval Timer Operation (2) Setting Procedure**



### 10.9.3 Pulse Width Measurement Operation

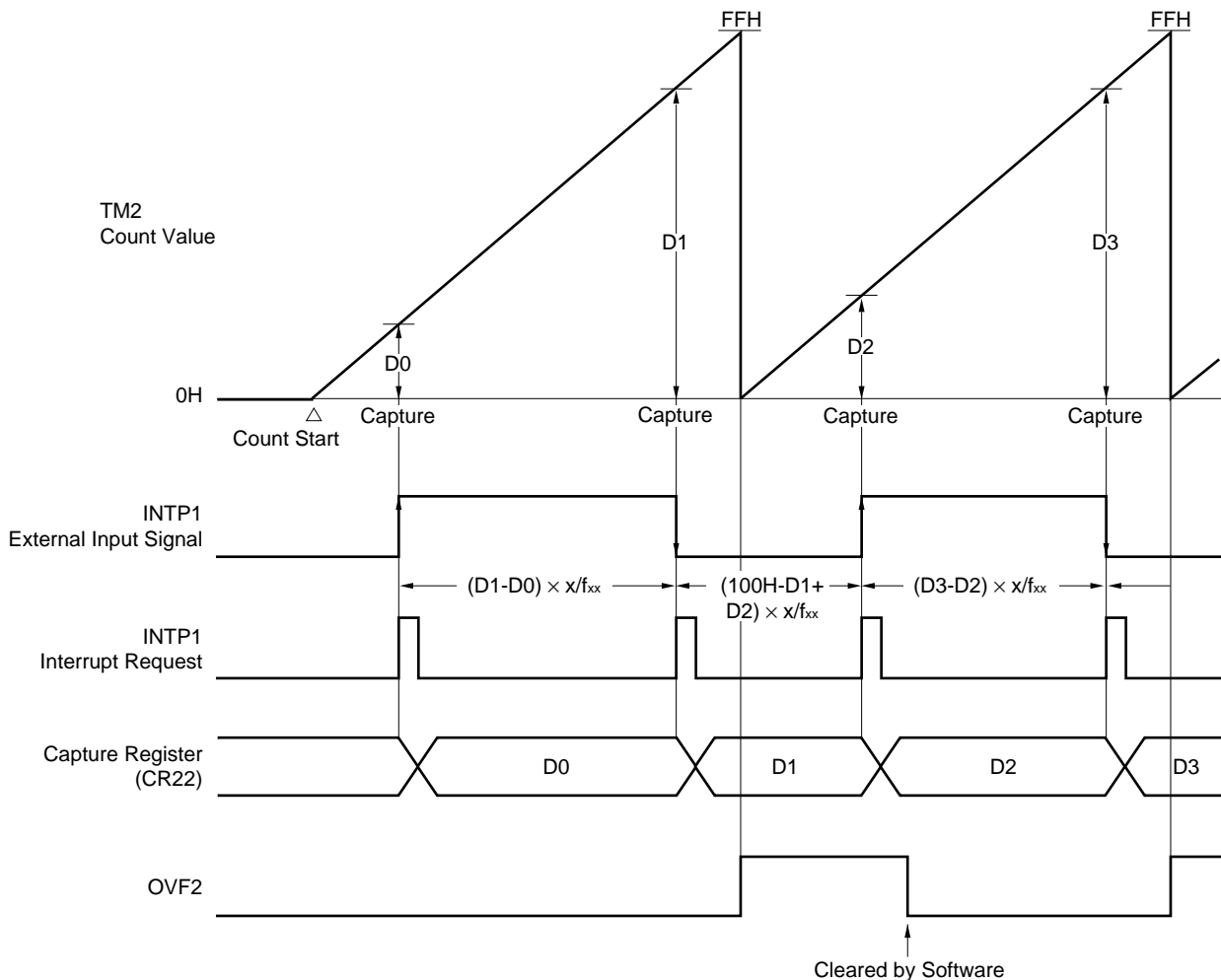
In pulse width measurement, the high-level or low-level width of external pulses input to the external interrupt request input pin (INTP1) pin are measured.

Both the high-level and low-level widths of pulses input to the INTP1 pin must be at least 3 system clocks ( $0.24 \mu\text{s}$ :  $f_{\text{CLK}} = 12.5 \text{ MHz}$ ); if shorter than this, the valid edge will not be detected and a capture operation will not be performed.

As shown in Figure 10-42, the timer register 2 (TM2) value being counted is fetched into the capture register (CR22) in synchronization with a valid edge (specified as both rising and falling edges) in the INTP1 pin input, and held there. The pulse width is obtained from the product of the difference value between the TM2 count value ( $D_n$ ) fetched into and held in the CR22 on detection of the  $n$ th valid edge and the count value ( $D_{n-1}$ ) fetched and held on detection of  $n-1$ th valid edge, and the number of  $n-1$ th count clocks ( $x/f_{xx}$ ;  $x = 8, 16, 32, 64, 128, 256, 512, 1024, 2048$ ).

The control register settings are shown in Figure 10-43, and the setting procedure in Figure 10-44.

Figure 10-42 Pulse Width Measurement Timing



**Remark**  $D_n$ : TM2 count value ( $n = 0, 1, 2, \dots$ )

$x = 8, 16, 32, 64, 128, 256, 512, 1024, 2048$

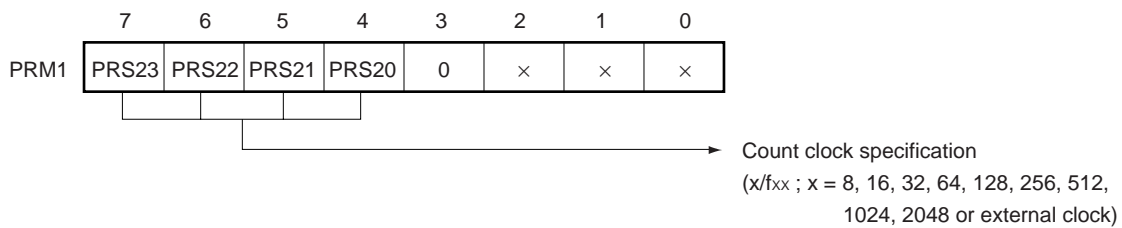
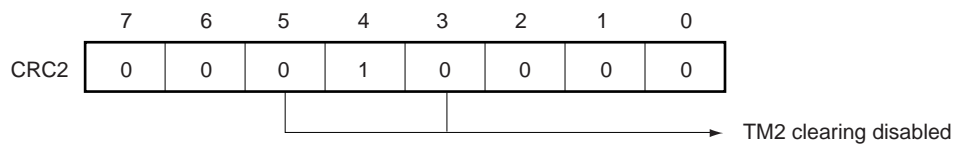
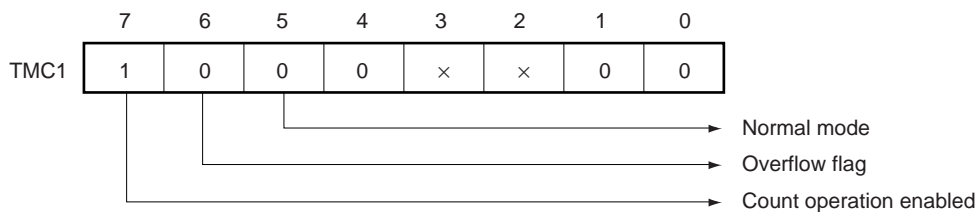
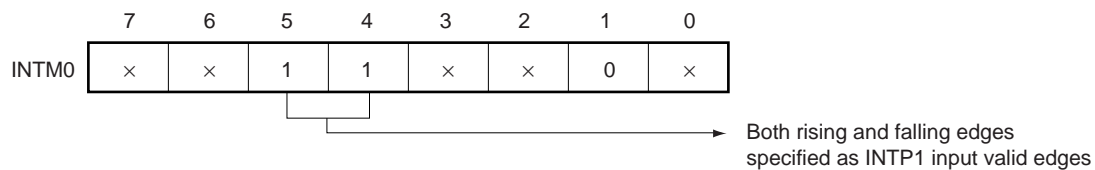
**Figure 10-43 Control Register Settings for Pulse Width Measurement****(a) Prescaler mode register 1 (PRM1)****(b) Capture/compare control register 2 (CRC2)****(c) Timer control register 1 (TMC1)****(d) External interrupt mode register 0 (INTM0)**

Figure 10-44 Pulse Width Measurement Setting Procedure

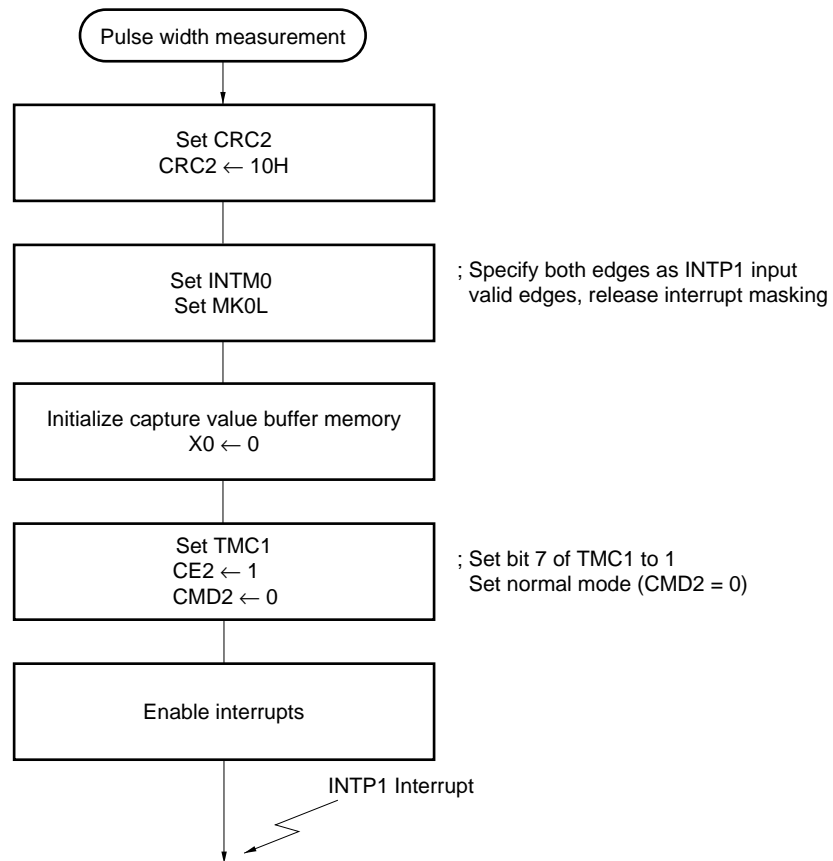
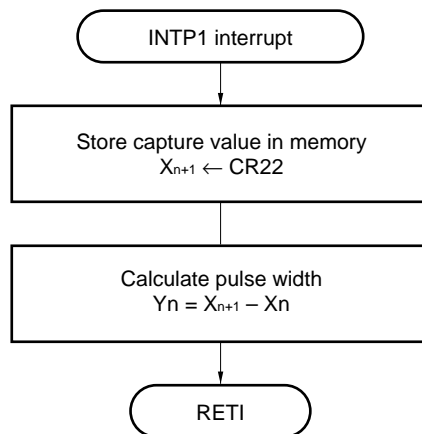


Figure 10-45 Interrupt Request Processing that Calculates Pulse Width



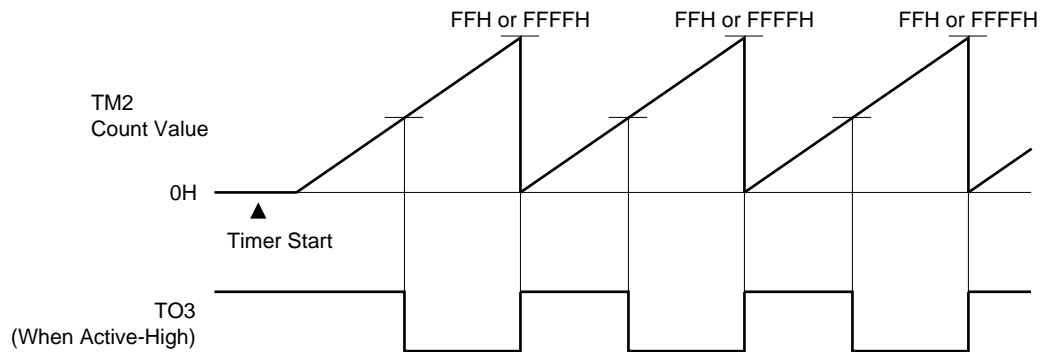
**10.9.4 Operation as PWM Output**

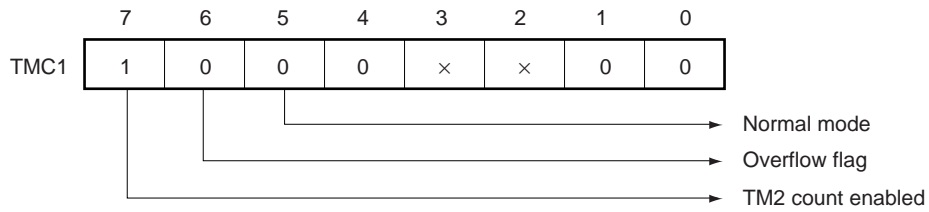
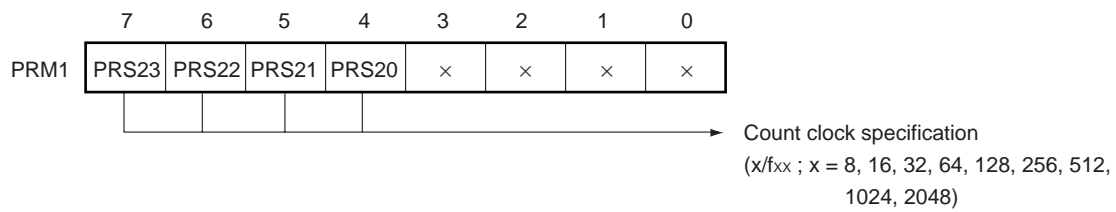
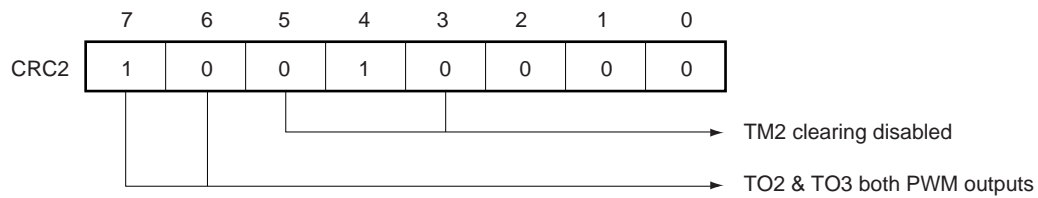
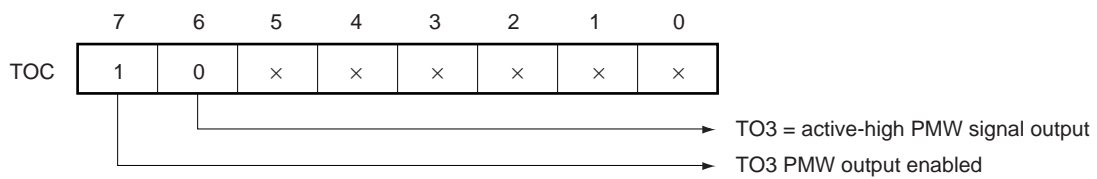
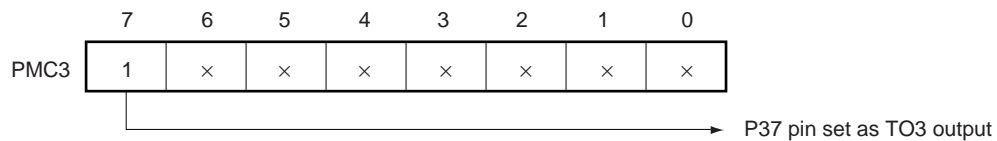
In PWM output, pulses with the duty ratio determined by the value set in the compare register (CR2n: n = 0, 1) are output (see **Figure 10-46**).

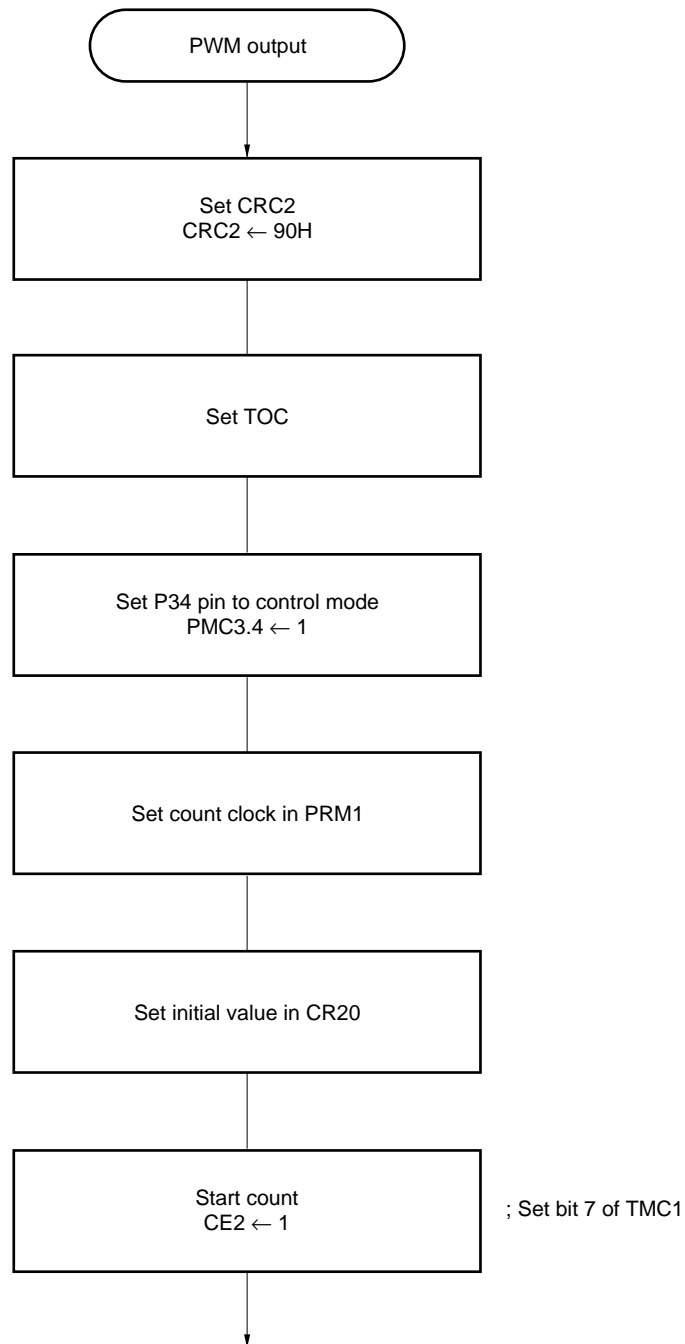
This PWM output duty ratio can be varied in the range 1/256 to 255/256 in 1/256 units.

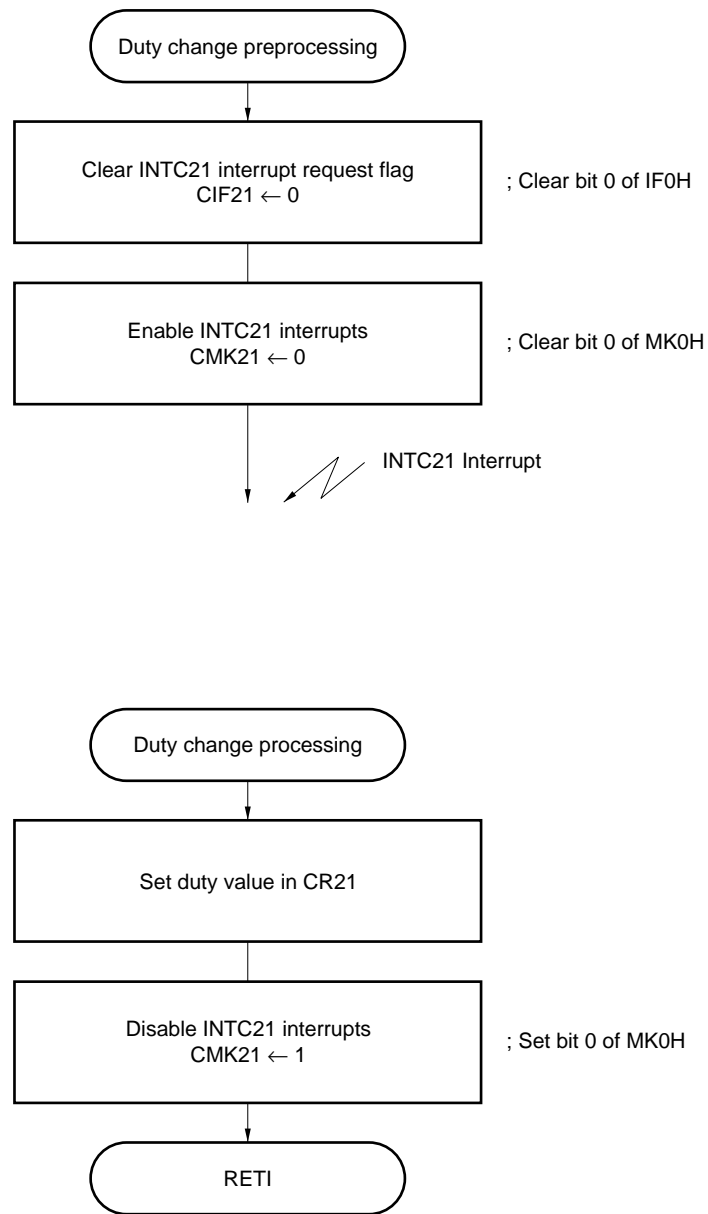
The control register settings are shown in Figure 10-47, the setting procedure in Figure 10-48, and the procedure for varying the duty in Figure 10-49.

**Figure 10-46 Example of Timer/Counter 2 PWM Signal Output**



**Figure 10-47 Control Register Settings for PWM Output Operation****(a) Timer control register 1 (TMC1)****(b) Prescaler mode register 1 (PRM1)****(c) Capture/compare control register 2 (CRC2)****(d) Timer output control register (TOC)****(e) Port 3 mode control register (PMC3)**

**Figure 10-48 PWM Output Setting Procedure**

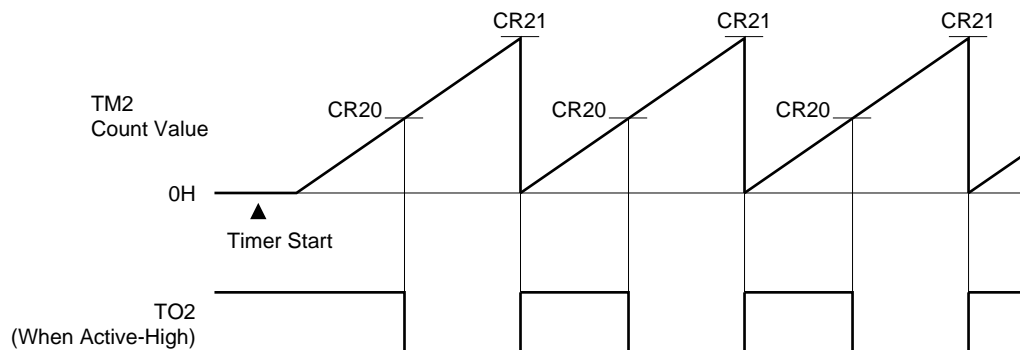
**Figure 10-49 Changing PWM Output Duty**

**10.9.5 Operation as PPG Output**

In PPG output, pulses with the cycle and duty ratio determined by the value set in the compare register (CR2n: n = 0, 1) are output (see **Figure 10-50**).

The control register settings are shown in Figure 10-51, the setting procedure in Figure 10-52, and the procedure for varying the duty in Figure 10-53.

**Figure 10-50 Example of Timer/Counter 2 PPG Signal Output**





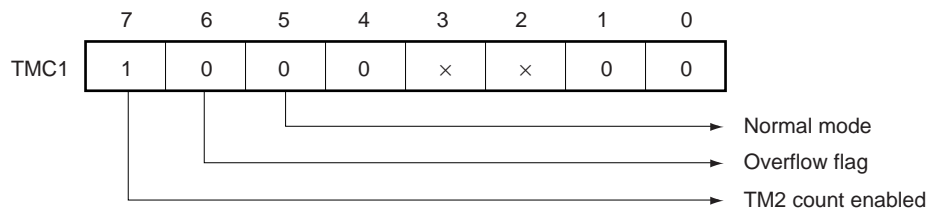
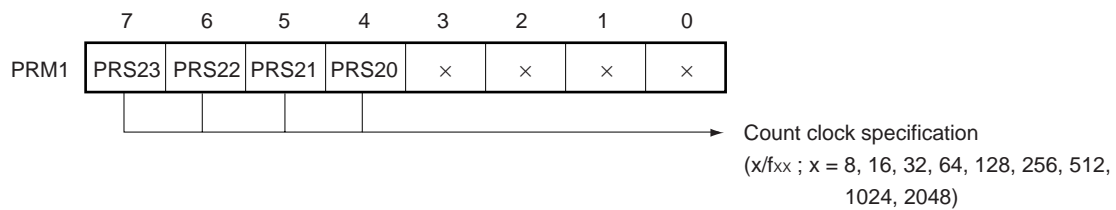
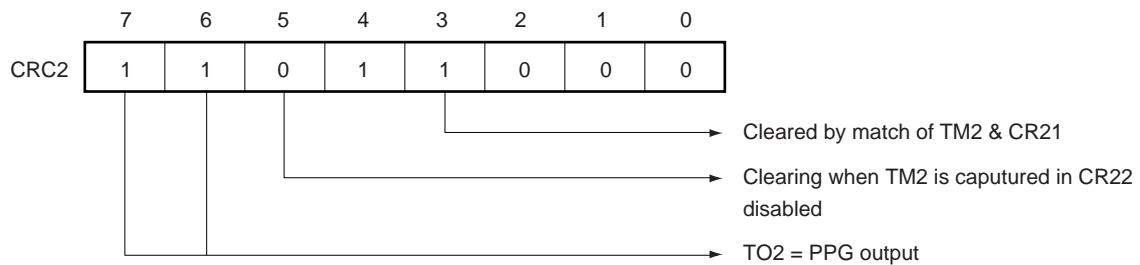
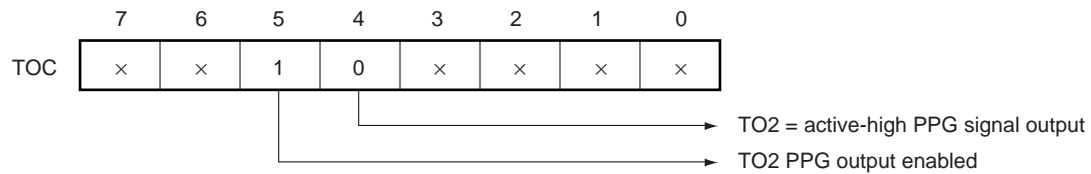
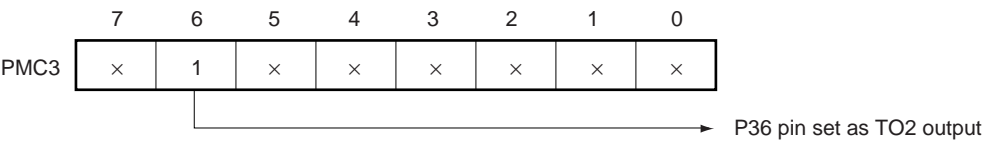
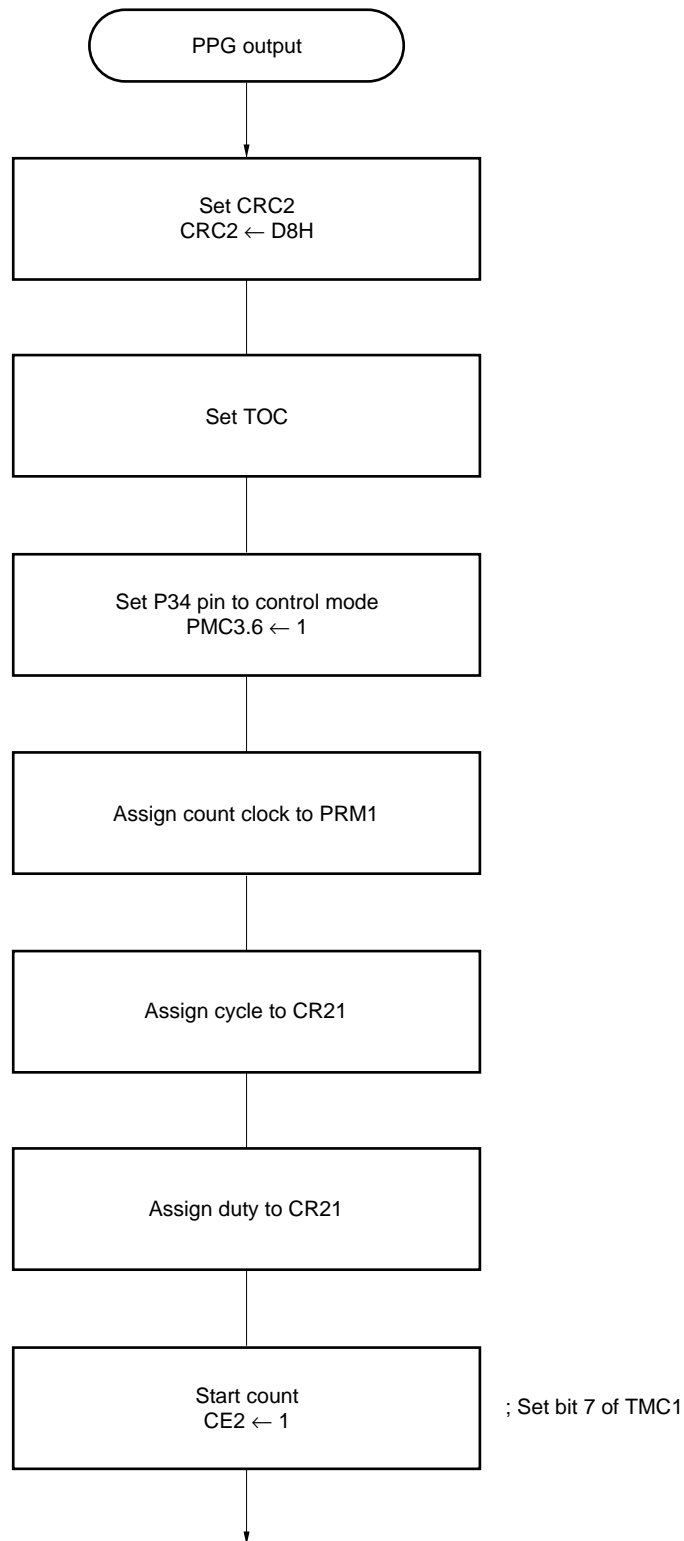
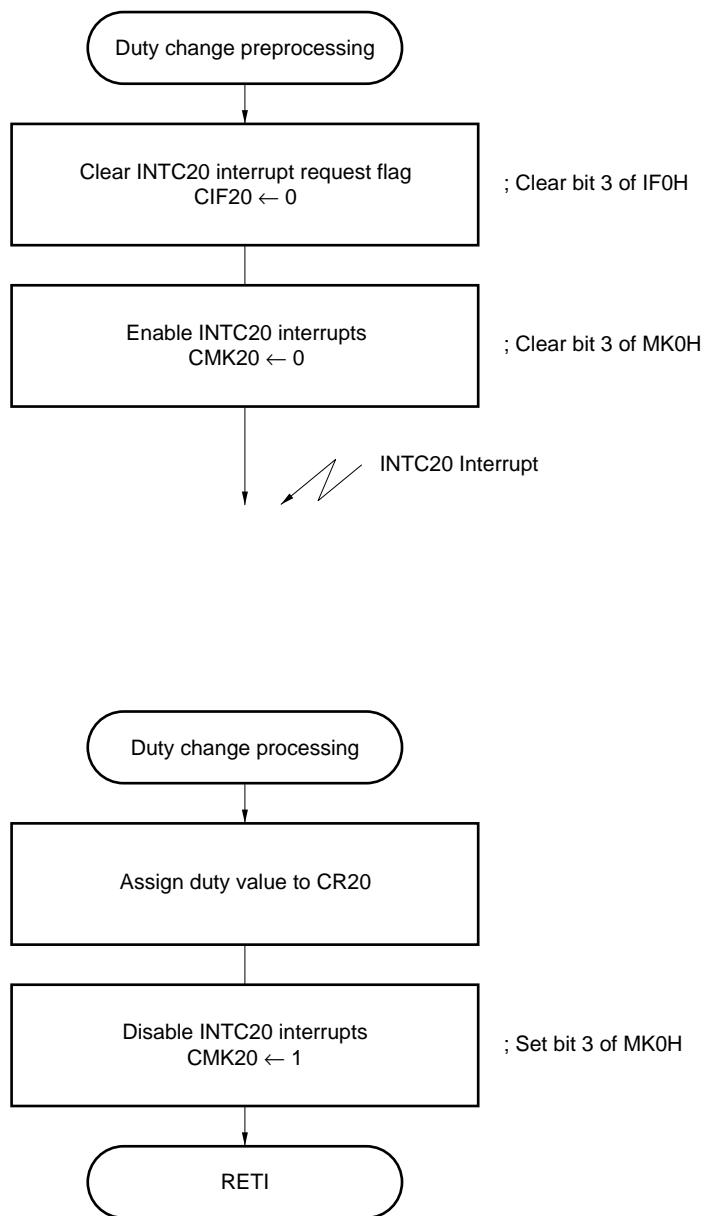
**Figure 10-51 Control Register Settings for PPG Output Operation (1/2)****(a) Timer control register 1 (TMC1)****(b) Prescaler mode register 1 (PRM1)****(c) Capture/compare control register 2 (CRC2)****(d) Timer output control register (TOC)**

Figure 10-51 Control Register Settings for PPG Output Operation (2/2)

(e) Port 3 mode control register (PMC3)



**Figure 10-52 PPG Output Setting Procedure**

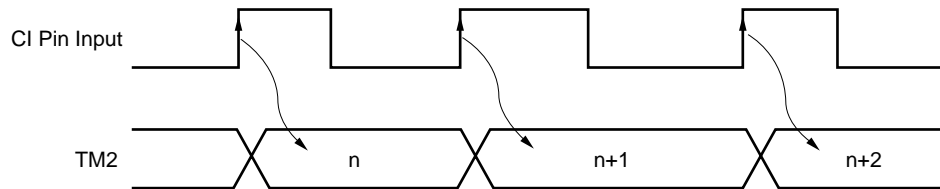
**Figure 10-53 Changing PPG Output Duty**

**10.9.6 Operation as External Event Counter**

An external event counter counts clock pulses (CI pin input pulses) input from off-chip.

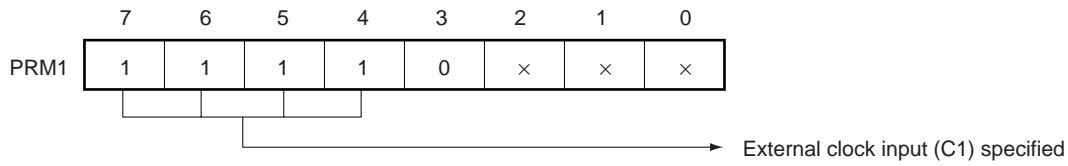
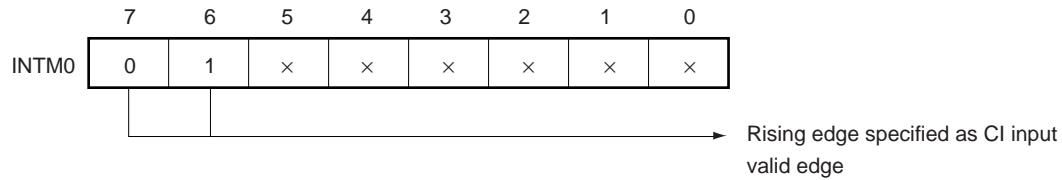
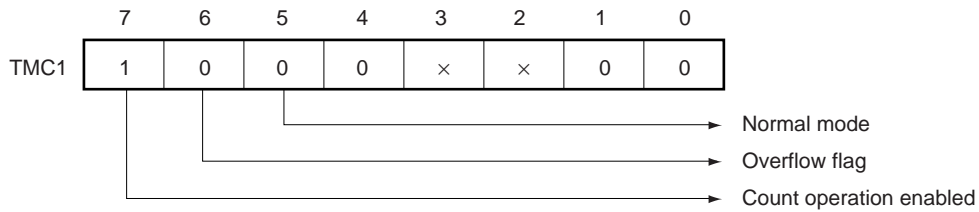
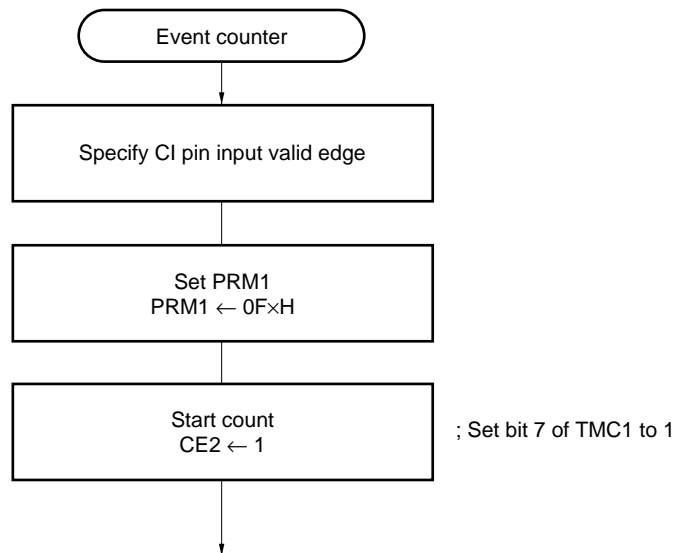
As shown in Figure 10-54, the value of timer register 2 (TM2) is incremented in synchronization with a CI pin input valid edge (specified as rising edge only).

**Figure 10-54 External Event Counter Operation (Single Edge)**



**Remark** The TM2 value is one less than the number of input clock pulses.

The control register settings when TM2 operates as an external event counter are shown in Figure 10-55, and the setting procedure in Figure 10-56.

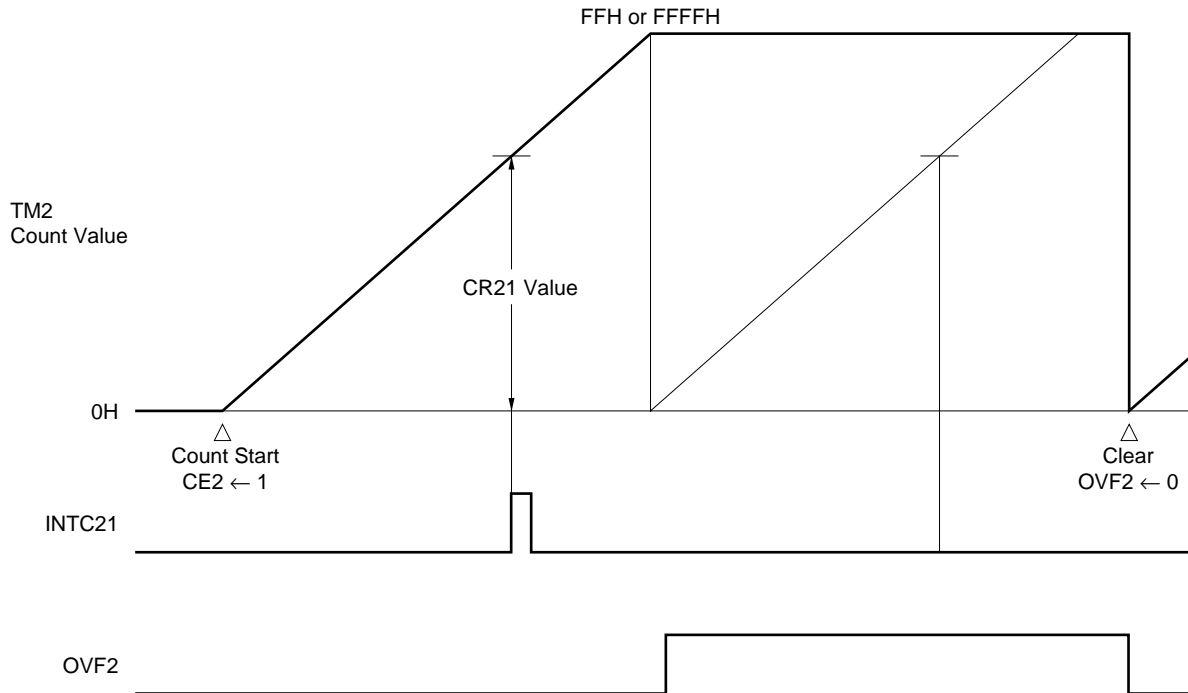
**Figure 10-55 Control Register Settings for External Event Counter Operation****(a) Prescaler mode register 1 (PRM1)****(b) External interrupt mode register 0 (INTM0)****(c) Timer control register 1 (TMC1)****Figure 10-56 External Event Counter Operation Setting Procedure**

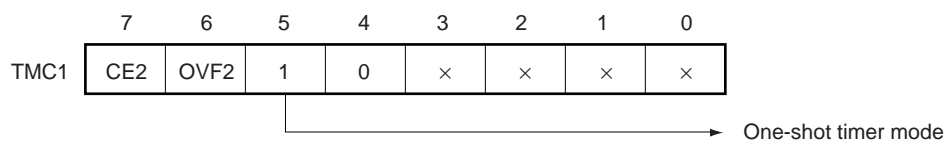
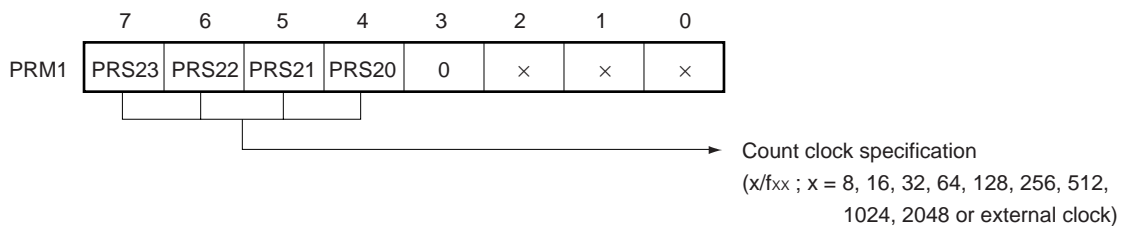
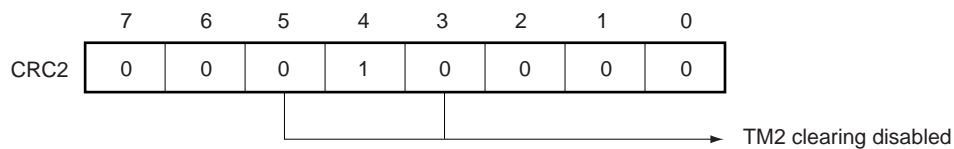
**10.9.7 Operation as One-Shot Timer**

After timer register 2 (TM2) is started, it operates as a one-shot pulse that generates a single interrupt after the preset count time (see **Figure 10-57**).

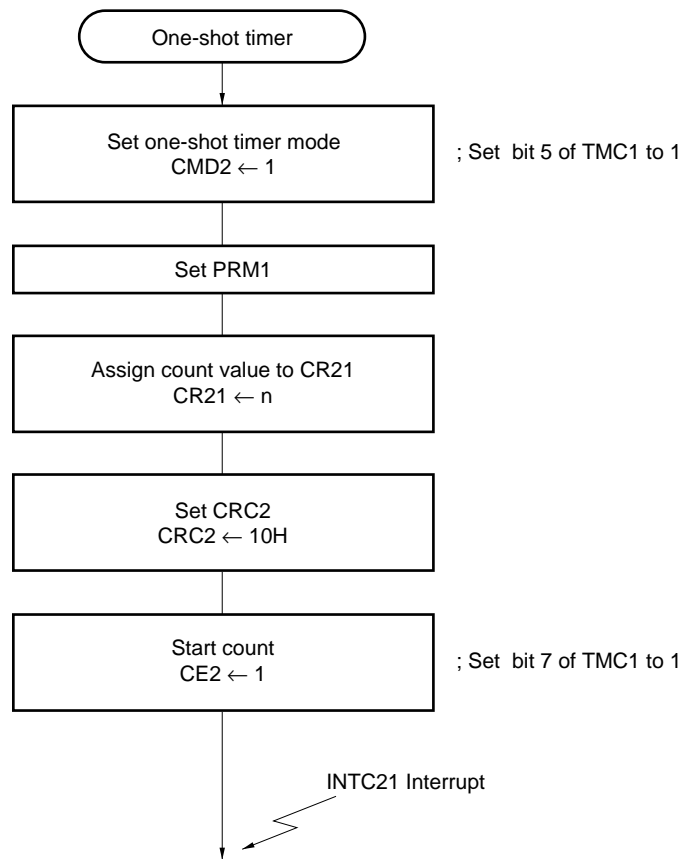
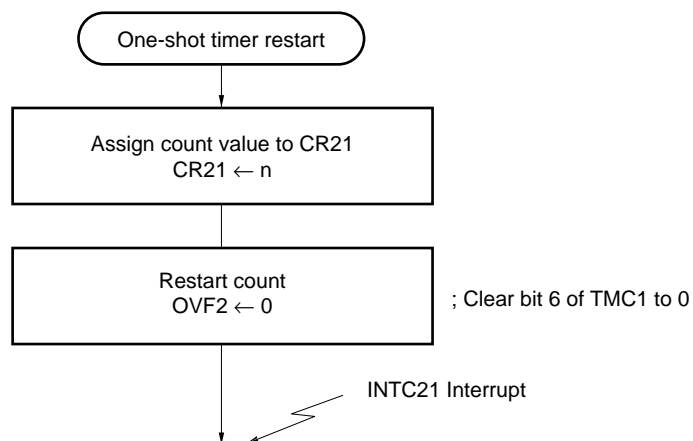
The second and subsequent one-shot timer operations can be started by clearing the OVF2 bit of timer control register 1 (TMC1).

The control register settings are shown in Figure 10-58, the setting procedure in Figure 10-59, and the procedure for starting the one-shot timer from the second time onward in Figure 10-60.

**Figure 10-57 One-Shot Timer Operation**

**Figure 10-58 Control Register Settings for One-Shot Timer Operation****(a) Timer control register 1 (TMC1)****(b) Prescaler mode register 1 (PRM1)****(c) Capture/compare control register 2 (CRC2)**



**Figure 10-59 One-Shot Timer Operation Setting Procedure****Figure 10-60 One-Shot Timer Operation Start Procedure from Second Time Onward**

**10.10 CAUTIONS**

- (1) While timer/counter 2 is operating (while the CE2 bit of the timer control register 1 (TMC1) is set), malfunctioning may occur if the contents of the following registers are rewritten. This is because it is undefined which takes precedence, the change in the hardware functions due to rewriting the register, or the change in the status because of the function before rewriting.

Therefore, be sure to stop the counter operation for the sake of safety before rewriting the contents of the following registers.

- Prescaler mode register 1 (PRM1)
- Capture/compare control register 2 (CRC2)
- Timer output control register (TOC)
- CMD2 bit of timer control register 1 (TMC)

- (2) If the contents of the compare register (CR2n: n = 0, 1) match with those of TM2 when an instruction that stops timer register 2 (TM2) operation is executed, the counting operation of TM2 stops, but an interrupt request is generated. In order not to generate the interrupt when stopping the operation of TM2, mask the interrupt in advance by using the interrupt mask register before stopping TM2.

**Example**

Program that may generate interrupt request

```

      :
      :
CLR1  CE2      : ← Interrupt request
OR    MK0H, #03H ← from timer/counter 2
      :         occurs between
      :         these instructions

```

Program that does not generate interrupt request

```

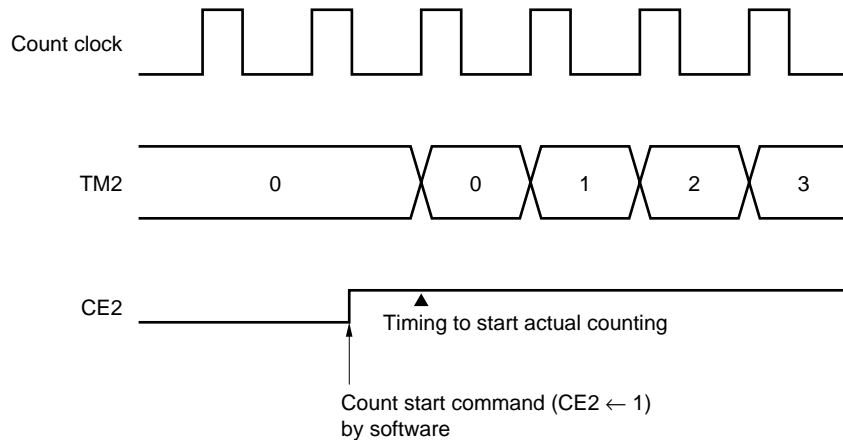
      :
      :
OR    MK0H, #03H ← Disables interrupt from timer/
CLR1  CE2        counter 2
CLR1  CIF20      ← Clears interrupt request flag for timer/
CLR1  CIF21      counter 2
      :

```

- (3) Up to 1 count clock is required after an operation to start timer/counter 2 ( $CE2 \leftarrow 1$ ) has been performed before timer/counter 2 actually starts (refer to **Figure 10-61**).

For example, when using timer/counter 2 as an interval timer, the first interval time is delayed by up to 1 clock. The second and those that follow are at the specified interval.

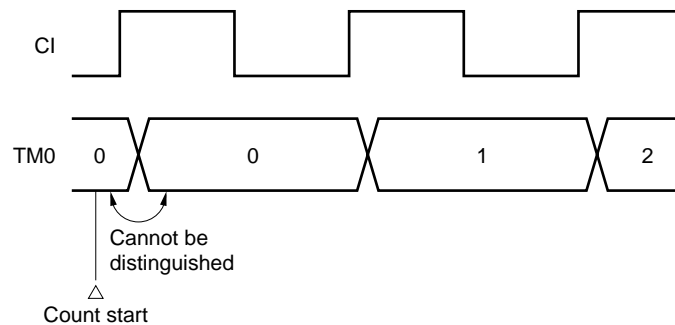
**Figure 10-61 Operation When Counting Is Started**

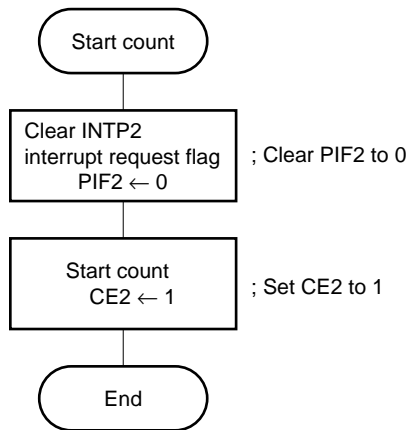
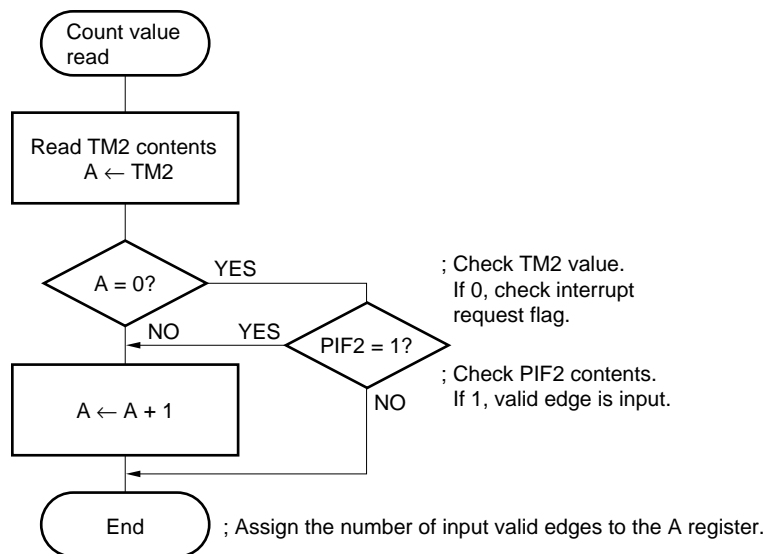


- (4) While an instruction that writes data to the compare register ( $CR2n$ :  $n = 0$  or  $1$ ) is executed, coincidence between  $CR2n$ , to which the data is to be written, and timer register 2 (TM2) is not detected. For example, if the contents of  $CR2n$  do not change before and after the writing, the interrupt request is not generated even if the value of TM2 coincides with the value of  $CR2n$ , nor does the timer output ( $TON + 2$ :  $n + 2 = 2, 3$ ) change.
- Write data to  $CR2n$  when timer/counter 2 is executing counting operation in the manner that the contents of TM2 do not match the value of  $CR2n$  before and after writing (e.g., immediately after an interrupt request has been generated because TM2 and  $CR2n$  have matched).
- (5) Match between timer register 2 (TM2) and compare register ( $CR2n$ :  $n = 0, 1$ ) is detected only when TM2 is incremented. Therefore, the interrupt request is not generated and timer output ( $TON + 2$ :  $n + 2 = 2, 3$ ) does not change even if the same value as TM2 is written to  $CR2n$ .

- (6) During PPG output, if the PPG cycle is extremely short as compared with the time required to acknowledge an interrupt, the value of the compare register (CR2n:  $n = 0, 1$ ) cannot be rewritten by interrupt processing that is performed on match between timer register (TM2) and compare register (CR2n). Use another method (for example, to poll the interrupt request flags by software with all the interrupts masked).
- (7) The output level of the TOn ( $n = 2, 3$ ) when the timer output is disabled (ENTOn = 0:  $n = 2, 3$ ) is the reverse value of the value set to the ALVn ( $n = 2, 3$ ) bit. Note, therefore, that an active level is output when the timer output is disabled with the PWM output function or PPG output function selected.
- (8) When using timer/counter 2 as an external event counter, the status where no valid edge is input cannot be distinguished from the status where only one valid edge has been input, by using timer register 2 (TM2) alone (refer to **Figure 10-62**), because the contents of TM2 are 0 in both the cases. To make a distinction, use the interrupt request flag of INTP2, as shown in Figure 10-63 (the INTP2 pin is multiplexed with the CI pin and both the functions can be used at the same time).

**Figure 10-62 Example Where Whether One or No Valid Edge Has been Input Cannot Be Distinguished with External Event Counter**



**Figure 10-63 To Distinguish Whether One or No Valid Edge Has Been Input with External Event Counter****(a) Processing on starting counting****(b) Processing on reading count value**

- ★ (9) Although the timer register can be read, if counting up the timer and reading the timer register conflict, the read value of the timer register may be illegal. Therefore, do not read the timer register if at all possible. If it is necessary to read the timer register, read it two or more times in a row in software, and assume the value of the timer register is correct only when all the read values match.

[MEMO]

## CHAPTER 11 TIMER 3

### 11.1 FUNCTION

Timer 3 is a 16- or 8-bit timer.

In addition to its function as an interval timer, it can be used as a counter for clocked serial interface (CSI) clock generation.

The interval timer generates internal interrupts at pre-set intervals. The interval setting range is shown in Table 11-1.

**Table 11-1 Timer 3 Intervals**

Minimum Interval	Maximum Interval	Resolution
$8/f_{xx}$ (0.32 $\mu$ s)	$2^{16} \times 8/f_{xx}$ (21.0 ms)	$8/f_{xx}$ (0.32 ms)
$16/f_{xx}$ (0.64 $\mu$ s)	$2^{16} \times 16/f_{xx}$ (41.9 ms)	$16/f_{xx}$ (0.64 ms)
$32/f_{xx}$ (1.28 $\mu$ s)	$2^{16} \times 32/f_{xx}$ (83.9 ms)	$32/f_{xx}$ (1.28 ms)
$64/f_{xx}$ (2.56 $\mu$ s)	$2^{16} \times 64/f_{xx}$ (167 ms)	$64/f_{xx}$ (2.56 ms)
$128/f_{xx}$ (5.12 $\mu$ s)	$2^{16} \times 128/f_{xx}$ (336 ms)	$128/f_{xx}$ (5.12 ms)
$256/f_{xx}$ (10.2 $\mu$ s)	$2^{16} \times 256/f_{xx}$ (671 ms)	$256/f_{xx}$ (10.2 ms)
$512/f_{xx}$ (20.5 $\mu$ s)	$2^{16} \times 512/f_{xx}$ (1.34 s)	$512/f_{xx}$ (20.5 ms)
$1024/f_{xx}$ (41.0 $\mu$ s)	$2^{16} \times 1024/f_{xx}$ (2.68 s)	$1024/f_{xx}$ (41.0 ms)
$2048/f_{xx}$ (81.9 $\mu$ s)	$2^{16} \times 2048/f_{xx}$ (5.37 s)	$2048/f_{xx}$ (81.9 ms)

( ): When  $f_{xx} = 25$  MHz





**(1) Timer register 3 (TM3/TM3W)**

TM3/TM3W are timer registers that count up using the count clock specified by the high-order 4 bits of prescaler mode register 0 (PRM0).

The count operation is stopped or enabled by the timer control register 0 (TMC0). In addition, an 8-bit mode (TM1) or 16-bit mode (TM1W) can be selected.

TM3 can be read only with an 8/16-bit manipulation instruction.

When  $\overline{\text{RESET}}$  is input, TM3 is cleared to 00H and the count is stopped.

★ **Caution** Although the timer register can be read, if counting up the timer and reading the timer register conflict, the read value of the timer register may be illegal. Therefore, do not read the timer register if at all possible.

If it is necessary to read the timer register, read it two or more times in a row in software, and assume the value of the timer register is correct only when all the read values coincide.

**(2) Compare register (CR30/CR30W)**

CR30/CR30W are 8/16-bit registers that hold the value that determines the interval timer frequency.

If the CR30/CR30W contents match the contents of TM3/TM3W, the contents of TM3/TM3W are cleared automatically and an interrupt request (INTC30) is generated.

This compare register operates as CR30 in the 8-bit mode and CR30W in the 16-bit mode.

The CR30 register can be read or written to with an 8/16-bit manipulation instruction. The contents of CR30 are undefined after  $\overline{\text{RESET}}$  input.

**(3) Prescaler**

The prescaler generates the count clock from the internal system clock. The clock generated by the prescaler is selected by the selector, and is used as the count clock by the timer to perform count operations.

**(4) Selector**

The selector selects a signal resulting from dividing the internal clock or the edge detected by the edge detection circuit as the count clock of timer register 3 (TM3/TM3W).

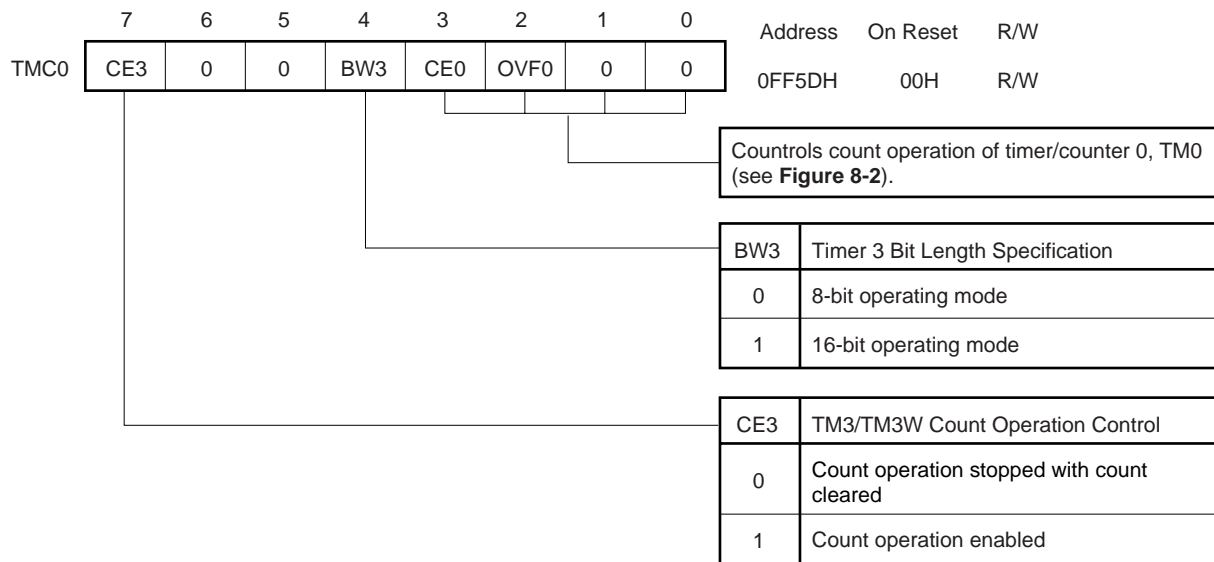
**11.3 TIMER 3 CONTROL REGISTERS****(1) Timer control register 0 (TMC0)**

TMC0 controls the timer 3, TM3/TM3W, count operation by the high-order 4 bits (the low-order 4 bits control the count operation of timer/counter 0, TM0).

TMC0 can be read or written to with an 8-bit manipulation instruction. The format of the TMC0 is shown in Figure 11-2.

RESET input clears TMC0 to 00H.

**Figure 11-2 Timer Control Register 0 (TMC0) Format**



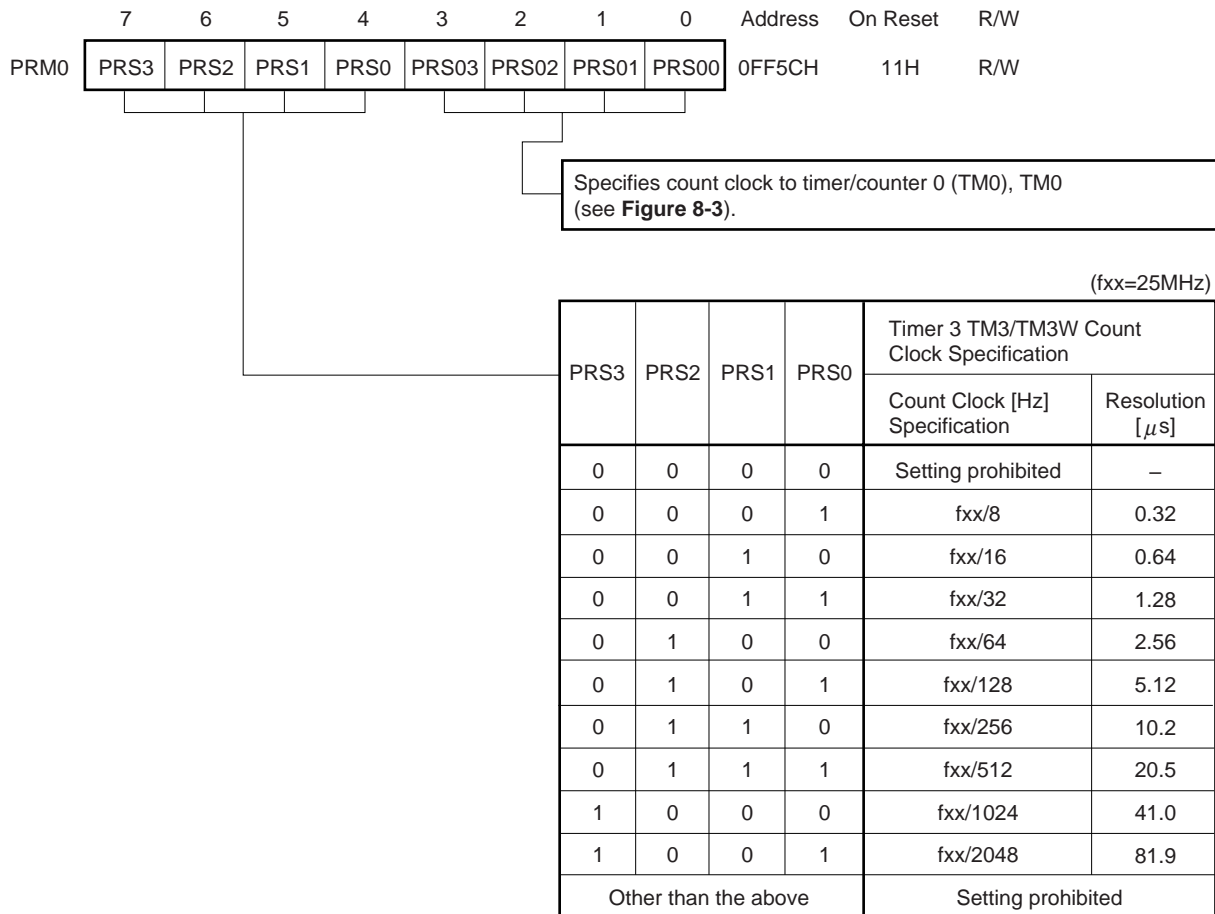
**(2) Prescaler mode register 0 (PRM0)**

PRM0 specifies the count clock to timer/counter 3 TM3/TM3W by the high-order 4 bits (the low-order 4 bits specify the count clock to timer/counter 0, TM0).

PRM0 can be read and written with an 8-bit manipulation instruction. The format of the PRM0 is shown in Figure 11-3.

RESET input clears PRM0 to 11H.

**Figure 11-3 Prescaler Mode Register 0 (PRM0) Format**



## 11.4 TIMER REGISTER 3 (TM3) OPERATION

### 11.4.1 Basic Operation

Timer 3 can operate in an 8-bit or 16-bit mode. These operation modes are selected by bit 4 (BW3) of timer control register 0 (TMC0)\*.

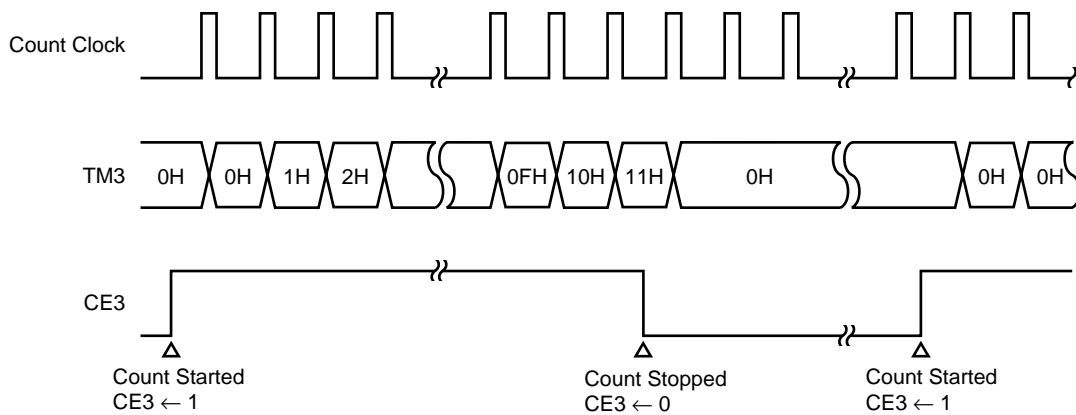
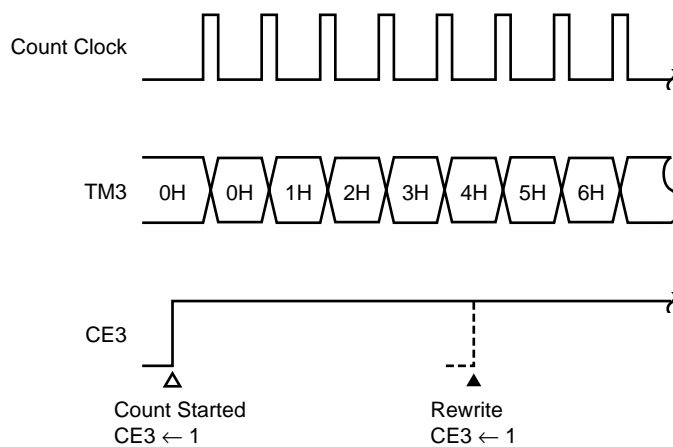
In the timer 3 count operation, an up-count is performed using the count clock specified by the high-order 4 bits of prescaler mode register 0 (PRM0).

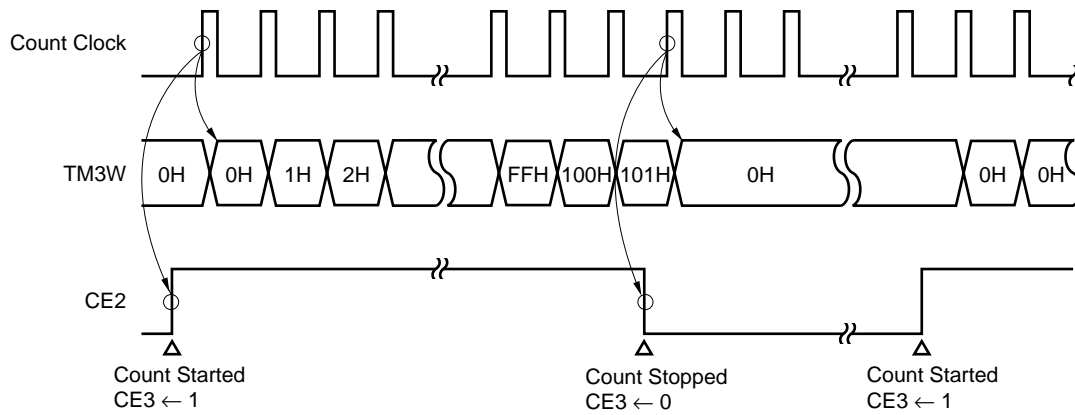
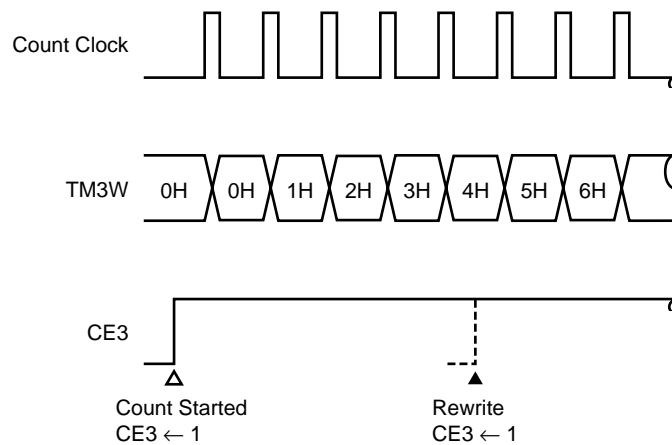
When  $\overline{\text{RESET}}$  is input, TM3 is cleared to 0000H, and the count operation is stopped.

Count operation enabling/disabling is controlled by bit 7 (CE3) of timer control register 0 (TMC0) (the high-order 4 bits of TMC0 control timer 3 operation). When the CE3 bit is set (1) by software, the contents of TM3 are immediately cleared on the first count clock, and then the up-count operation is performed. When the CE3 bit is cleared (0), TM3 becomes 0H immediately, and match signal generation is stopped. If the CE3 bit is set (1) again when it is already set (1), TM3 continues the count operation without being cleared.

\* Unless there functional differences are found, the register names in the 8-bit mode are used.

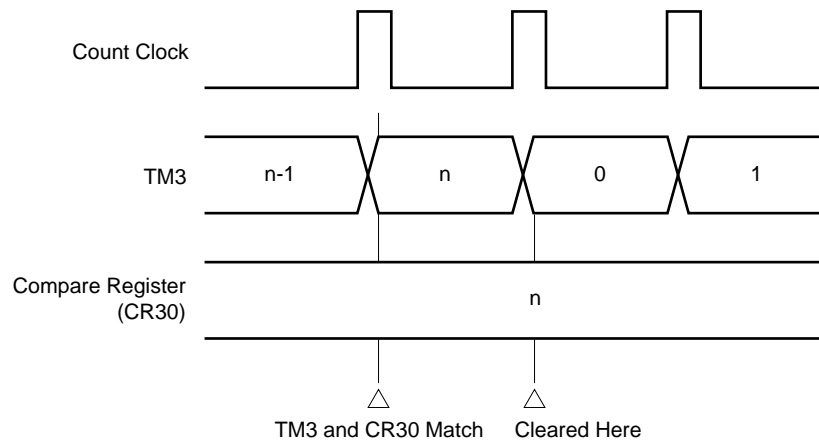
In the 16-bit mode, the register names TM3 and CR30 are TM3W and CR30W, respectively.

**Figure 11-4 Basic Operation in 8-Bit Operating Mode (BW3 = 0)****(a) Count started → count stopped → count started****(b) When “1” is written to the CE3 bit again after the count starts**

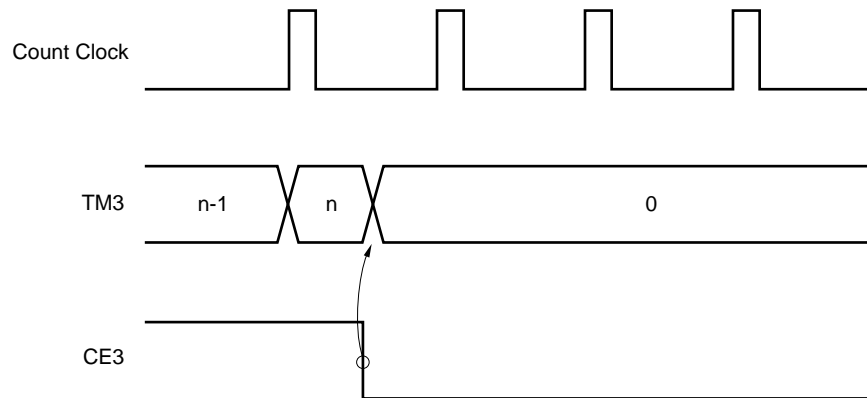
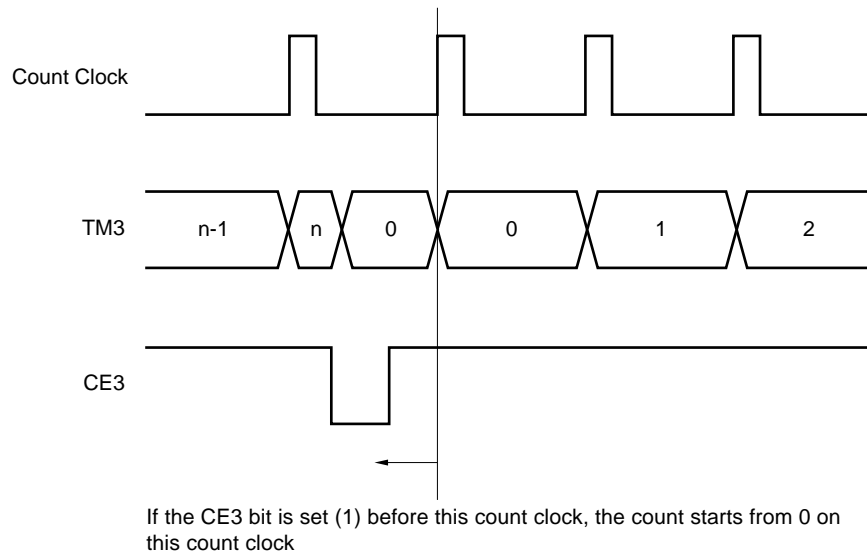
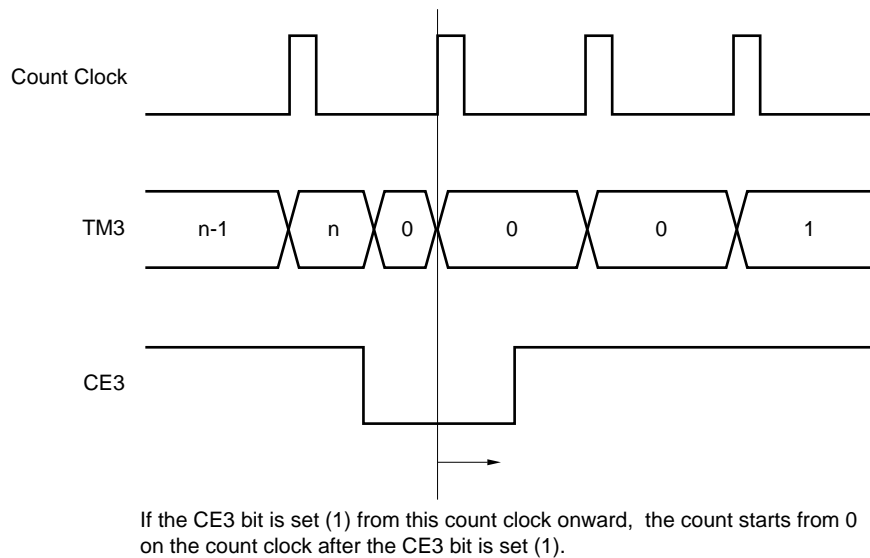
**Figure 11-5 Basic Operation in 16-Bit Operating Mode (BW3 = 1)****(a) Count started → count stopped → count started****(b) When “1” is written to the CE3 bit again after the count starts**

**11.4.2 Clear Operation****(1) Clear operation by match with compare register (CR30)**

16-bit timer 3 (TM3) is cleared automatically after a match with the compare register (CR30). When a clearance source arises, TM3 is cleared to 0H on the next count clock. Therefore, even if a clearance source arises, the value at the point at which the clearance source arose is retained until the next count clock arrives.

**Figure 11-6 TM3 Clearance by Match with Compare Register (CR30)****(2) Clear operation by CE3 bit of timer/control register 0 (TMC0)**

Timer register 3 (TM3) is also cleared when the CE3 bit of TMC0 is cleared (0) by software. The clear operation is performed immediately after the clearance (0) of the CE3 bit.

**Figure 11-7 Clear Operation When CE3 Bit is Cleared (0)****(a) Basic operation****(b) Restart before count clock is input after clearance****(c) Restart when count clock is input after clearance**



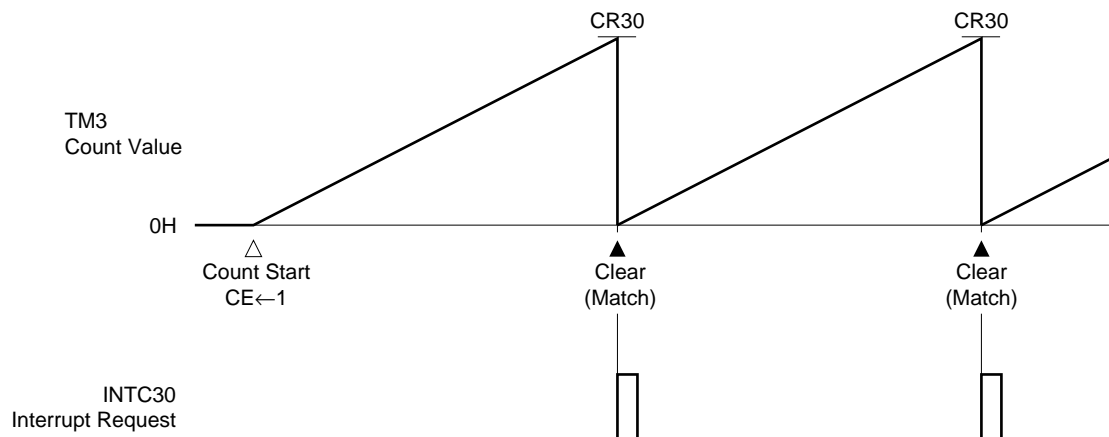
### 11.5 COMPARE REGISTER OPERATION

Timer 3 performs compare operations in which the value set in the compare register (CR30) is compared with the timer register 3 (TM3) count value.

If the count value of TM3 matches the preset CR30 value as the result of the count operation, an interrupt request (INTC30) is generated.

After a match, the TM3 contents are cleared automatically, and therefore TM3 functions as an interval timer that repeatedly counts up to the value set in the CR30.

**Figure 11-8 Compare Operation**



## 11.6 EXAMPLE OF USE

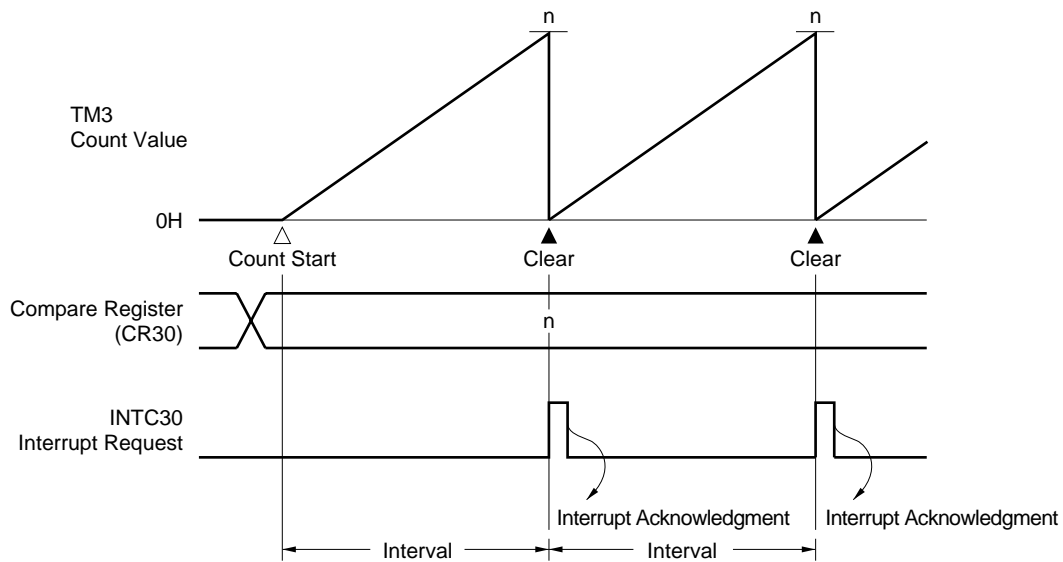
### Operation as interval timer:

TM3 operates as an interval timer that generates interrupts repeatedly with the pre-set count time as the interval (see **Figure 11-9**). TM3 can also be used for baud rate generation.

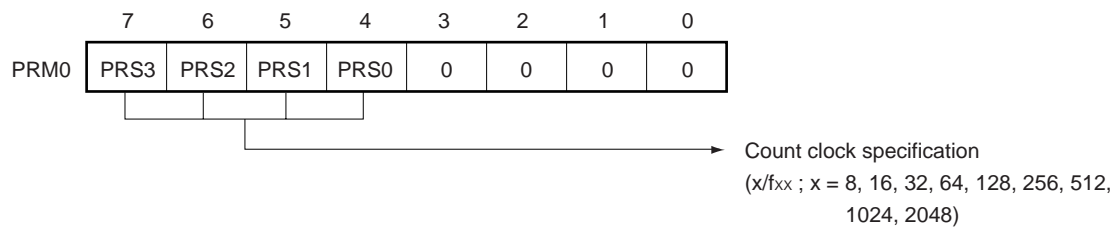
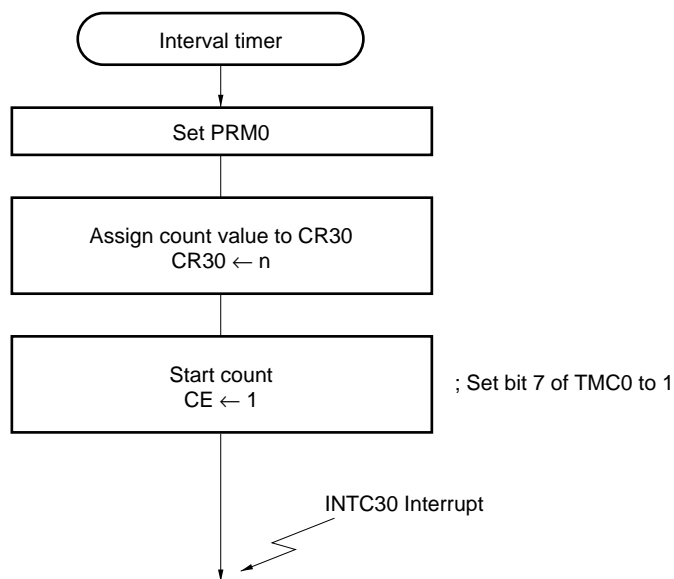
This interval timer can count up to a maximum of 21.0 ms at the minimum resolution of 0.32  $\mu$ s, and up to 5.37 s at the maximum resolution of 81.9  $\mu$ s (internal system clock  $f_{xx} = 25$  MHz).

The control register settings are shown in Figure 11-10, and the setting procedure in Figure 11-11.

**Figure 11-9 Interval Timer Operation Timing**



**Remark**  $\text{Interval} = (n+1) \times x/f_{xx}$ ,  $0 \leq n \leq \text{FFH}$   
 $x = 8, 16, 32, 64, 128, 256, 512, 1024, 2048$

**Figure 11-10 Control Register Settings for Interval Timer Operation****Prescaler mode register 0 (PRM0)****Figure 11-11 Interval Timer Operation Setting Procedure**

**11.7 CAUTIONS**

- (1) There is a possibility of misoperation if the next register contents are rewritten while the timer 3 is running (when the CE3 bit of the timer control register 0 (TMC0) is set). The misoperation occurs as there is no defined order of priority in the event of contention between the timings at which the hardware function changes due to a register rewrite and the status changes in the function prior to the rewrite.

When the contents of following registers are rewritten, counter operations must be stopped first to ensure stability.

- Prescaler mode register 0 (PRM0)

- (2) If the compare register (CR30) and TM3 contents match when an instruction that stops timer register 3 (TM3) operation is executed, the TM3 count operation stops, but an interrupt request is generated.

If you do not want an interrupt to be generated when TM3 operation is stopped, interrupts should be masked by means of interrupt the mask register before stopping the TM3.

**Example**

Program in which an interrupt request may be generated

```

:
:
CLR1 CE3
SET1 CMK30
:
:
: ← Interrupt request generated
:   by timer 3 here

```

Program in which an interrupt request is not generated

```

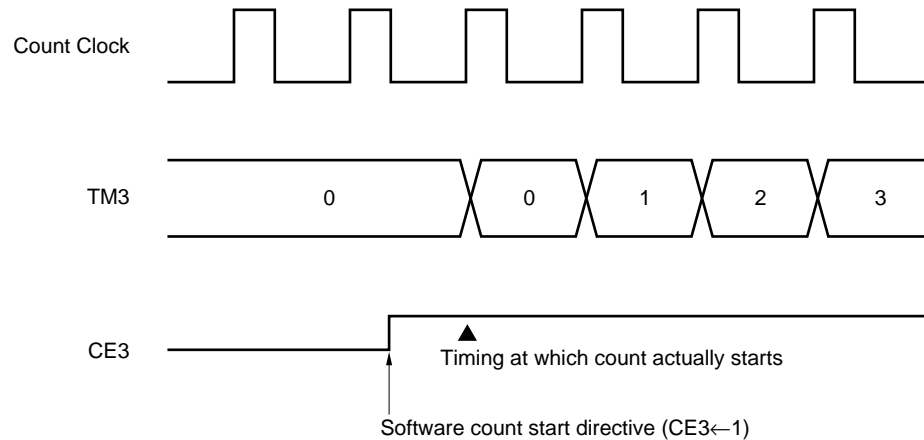
:
:
SET1 CMK30
CLR1 CE3
CLR1 CIF30
:
: ← Disables interrupts from timer 3
:   ← Clears timer 3 interrupt request flag

```

- (3) There is a delay of up to one count clock between the operation that starts a timer 3 ( $CE3 \leftarrow 1$ ) and the actual start of the timer/counter (see **Figure 11-12**).

For example, if a timer/counter is used as an interval timer, the first interval will be extended by up to one clock. The second and subsequent intervals will be as specified.

**Figure 11-12 Operation When Count Starts**



- (4) While an instruction that writes data to the compare register (CR30) is executed, match between CR30, to which the data is to be written, and timer register 3 (TM3) is not detected.  
Write data to CR30 when timer 3 is executing counting operation so that the contents of TM3 do not match the value of CR30 before and after writing (e.g., immediately after an interrupt request has been generated because TM3 and CR30 have matched).
- (5) Match between timer register 3 (TM3) and compare register (CR30) is detected only when TM3 is incremented. Therefore, the interrupt request is not generated even if the same value as TM3 is written to CR30.
- (6) Although the timer register can be read, if counting up the timer and reading the timer register conflict, the read value of the timer register may be illegal. Therefore, do not read the timer register if at all possible.  
If it is necessary to read the timer register, read it two or more times in a row in software, and assume the value of the timer register is correct only when all the read values match.

[MEMO]

## CHAPTER 12 WATCHDOG TIMER FUNCTION

The watchdog timer is a timer that detects inadvertent program loops.

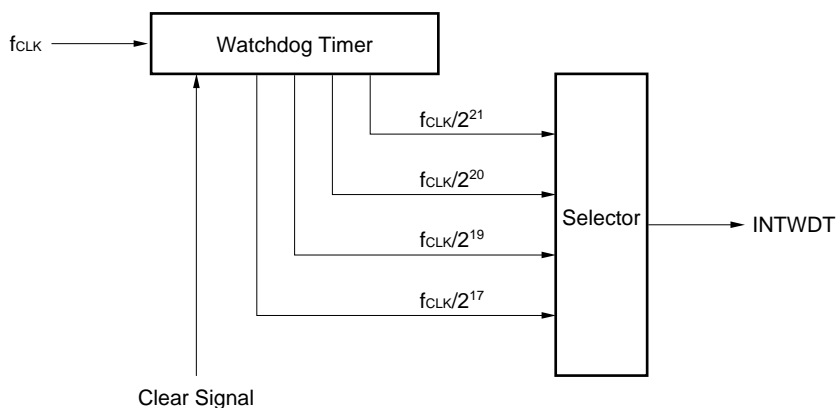
Watchdog timer interrupts are used to detect system or program errors. For this purpose, instructions that clear the watchdog timer (start the count) within a given period are inserted at various places in a program.

If an instruction that clears the watchdog timer is not executed within the set time and the watchdog timer overflows, a watchdog timer interrupt (INTWDT) is generated and a program error is reported.

### 12.1 CONFIGURATION

The watchdog timer block diagram is shown in Figure 12-1.

**Figure 12-1 Watchdog Timer Block Diagram**



## 12.2 WATCHDOG TIMER MODE REGISTER (WDM)

The WDM is an 8-bit register that controls the watchdog timer operation.

To prevent erroneous clearing of the watchdog timer by an inadvertent program loop, writing can only be performed by a dedicated instruction. This dedicated instruction, `MOV WDM,#byte`, has a special code configuration (4 bytes), and a write is not performed unless the 3rd and 4th bytes of the operation code are mutual complements.

If the 3rd and 4th bytes of the operation code are not complements, a write is not performed and an operand error interrupt is generated. In this case, the return address saved in the stack area is the address of the instruction that was the source of the error, and thus the address that was the source of the error can be identified from the return address saved in the stack area.

If recovery from an operand error is simply performed by means of an `RETB` instruction, an endless loop will result.

As an operand error interrupt is only generated in the event of an inadvertent program loop (with the NEC assembler, RA78K/IV, only the correct dedicated instruction is generated when `MOV WDM, #byte` is written), system initialization should be performed by the program.

Other write instructions (`MOV WDM, A`; `AND WDM, #byte`; `SET1 WDM.7`, etc.) are ignored and do not perform any operation. That is, a write is not performed to the WDM, and an interrupt such as an operand error interrupt is not generated.

After a system reset ( $\overline{\text{RESET}}$  input), once the watchdog timer has been started (by setting (1) the RUN bit), the WDM contents cannot be changed. The watchdog timer can only be stopped by a reset, but can be cleared at any time with a dedicated instruction.

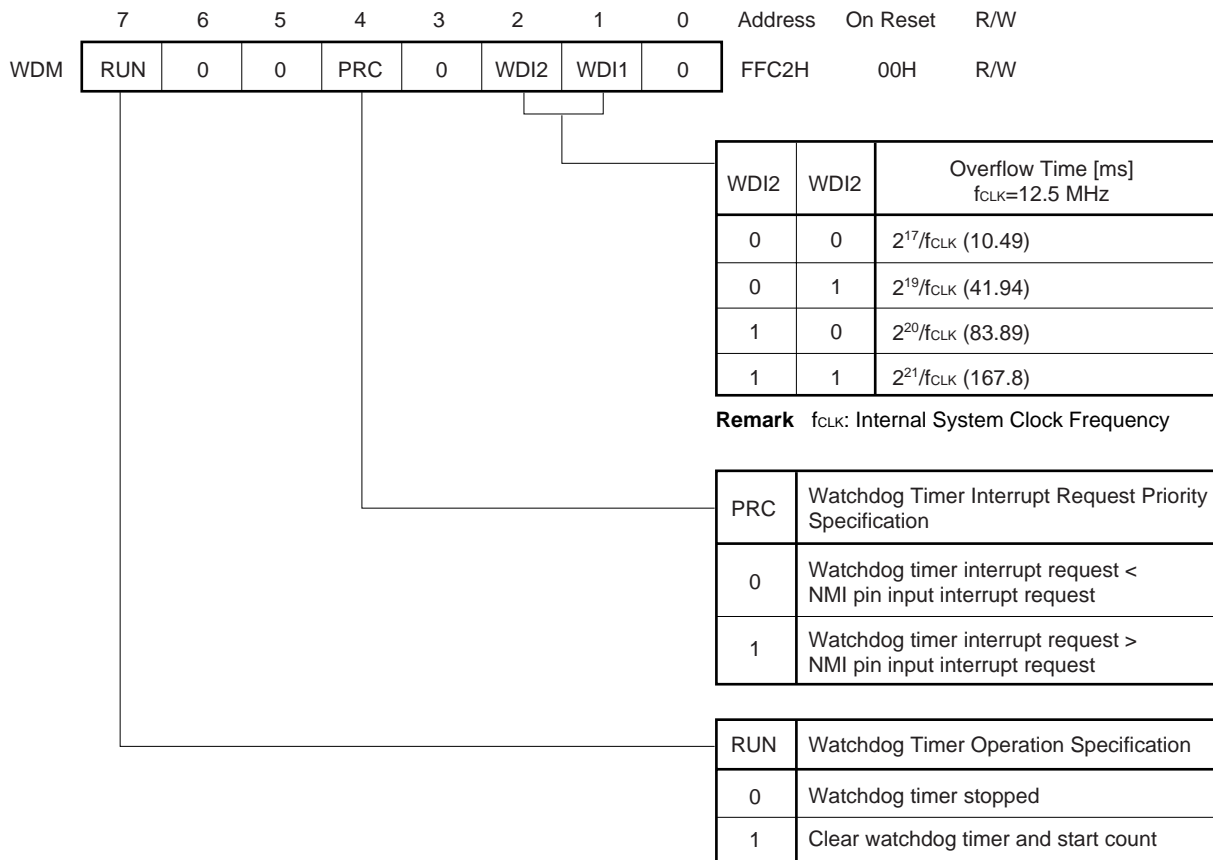
The WDM can be read at any time by a data transfer instruction.

$\overline{\text{RESET}}$  input clears the WDM to 00H.

The WDM format is shown in Figure 12-2.



Figure 12-2 Watchdog Timer Mode Register (WDM) Format



- Cautions**
1. The watchdog timer mode register (WDM) can only be written to with a dedicated instruction (MOV WDM, #byte).
  2. The same value should be written each time in writes to the WDM to set (1) the RUN bit. The contents written the first time cannot be changed even if a different value is written.
  3. Once the RUN bit has been set (1), it cannot be reset (0) by software.

## 12.3 OPERATION

### 12.3.1 Count Operation

The watchdog timer is cleared, and the count started, by setting (1) the RUN bit of the watchdog timer mode register (WDM). When overflow time specified by the WDM2 and WDM1 bits of WDM has elapsed after the RUN bit has been set (1), a non-maskable interrupt (INTWDT) is generated.

If the RUN bit is set (1) again before the overflow time elapses, the watchdog timer is cleared and the count operation is started again.

### 12.3.2 Interrupt Priorities

The watchdog timer interrupt (INTWDT) is a non-maskable interrupt. Other non-maskable interrupts are interrupts from the NMI pin (NMI). The order of acknowledgment when an INTWDT interrupt and NMI interrupt are generated simultaneously can be specified by the setting of bit 4 of the watchdog timer mode register (WDM).

Even if INTWDT is generated while the NMI processing program is executed when NMI acknowledgement is specified to take precedence, INTWDT is not acknowledged until completion of execution of the NMI processing program.

## 12.4 CAUTIONS

### 12.4.1 General Cautions on Use of Watchdog Timer

- (1) The watchdog timer is one means of detecting inadvertent program loops, but it cannot detect all inadvertent program loops. Therefore, in equipment that requires a high level of reliability, you should not rely on the on-chip watchdog timer alone, but should use external circuitry for early detection of inadvertent program loops, to enable processing to be performed that will restore the normal state or establish a stable state and then stop the operation.
- (2) The watchdog timer cannot detect inadvertent program loops in the following cases.
  - <1> If watchdog timer clearance is performed in the timer interrupt processing program
  - <2> If cases where an interrupt request or macro service is held pending (see **20.9**) occur consecutively
  - <3> If the watchdog timer is cleared periodically when inadvertent program looping is due to an error in the program logic (if each module of the program functions normally but the overall program does not)
  - <4> If the watchdog timer is periodically cleared by a group of instructions executed when an inadvertent program loop occurs
  - <5> If the STOP mode, HALT mode, or IDLE mode is entered as the result of an inadvertent program loop
  - <6> If watchdog timer runaway also occurs in the event of CPU runaway due to external noise

In cases <1>, <2> and <3> the program can be amended to allow detection to be performed.

In case <4>, the watchdog timer can only be cleared by a 4-byte dedicated instruction. Similarly, in case <5>, the STOP mode, HALT mode, or IDLE mode cannot be set unless a 4-byte dedicated instruction is used. For state <2> to be entered as the result of an inadvertent program loop, 3 or more consecutive bytes of data must comprise a specific pattern (e.g. BT PSWL. bit, \$\$, etc.). Therefore, the establishment of state <2> as the result of <4>, <5> or an inadvertent program loop is likely to be extremely rare.

### 12.4.2 Cautions on $\mu$ PD784026 Subseries Watchdog Timer

- (1) The watchdog timer mode register (WDM) can only be written to with a dedicated instruction (MOV WDM, #byte).
- (2) The same value should be written each time in writes to the watchdog timer mode register (WDM) to set (1) the RUN bit. The contents written the first time cannot be changed even if a different value is written.
- (3) Once the RUN bit has been set (1), it cannot be reset (0) by software.

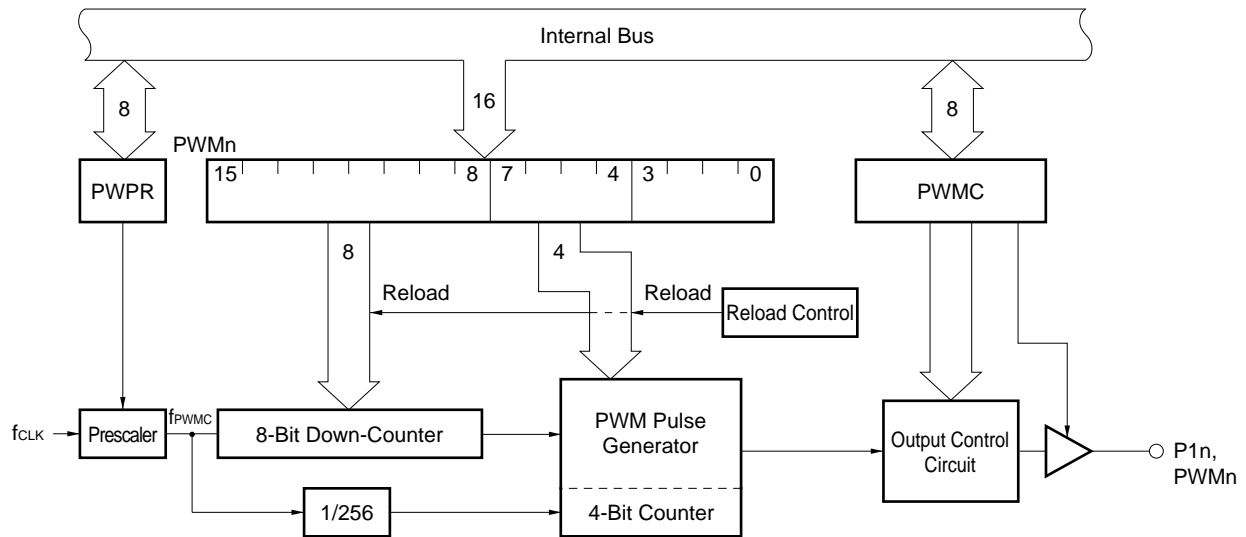
## CHAPTER 13 PWM OUTPUT UNIT

The  $\mu$ PD784026 incorporates two 12-bit resolution PWM (pulse width modulation) output circuit channels. The active level of the PWM output pulses can be selected as high or low. The PWM output ports have a dual function as pins P10 and P11.

### 13.1 PWM OUTPUT UNIT CONFIGURATION

The PWM output unit configuration is shown in Figure 13-1.

**Figure 13-1 PWM Output Unit Configuration**



**Remark**  $n = 0, 1$

**(1) 8-bit down-counter**

Generates the basic PWM signal timing.

**(2) PWM pulse generator (including 4-bit counter)**

Controls addition of extra pulses and generates the PWM pulses to be output.

**(3) Reload control**

Controls 8-bit down counter and 4-bit count modulo value reloading.

**(4) Output control circuit**

Controls the active level of the PWM signal.

**(5) Prescaler**

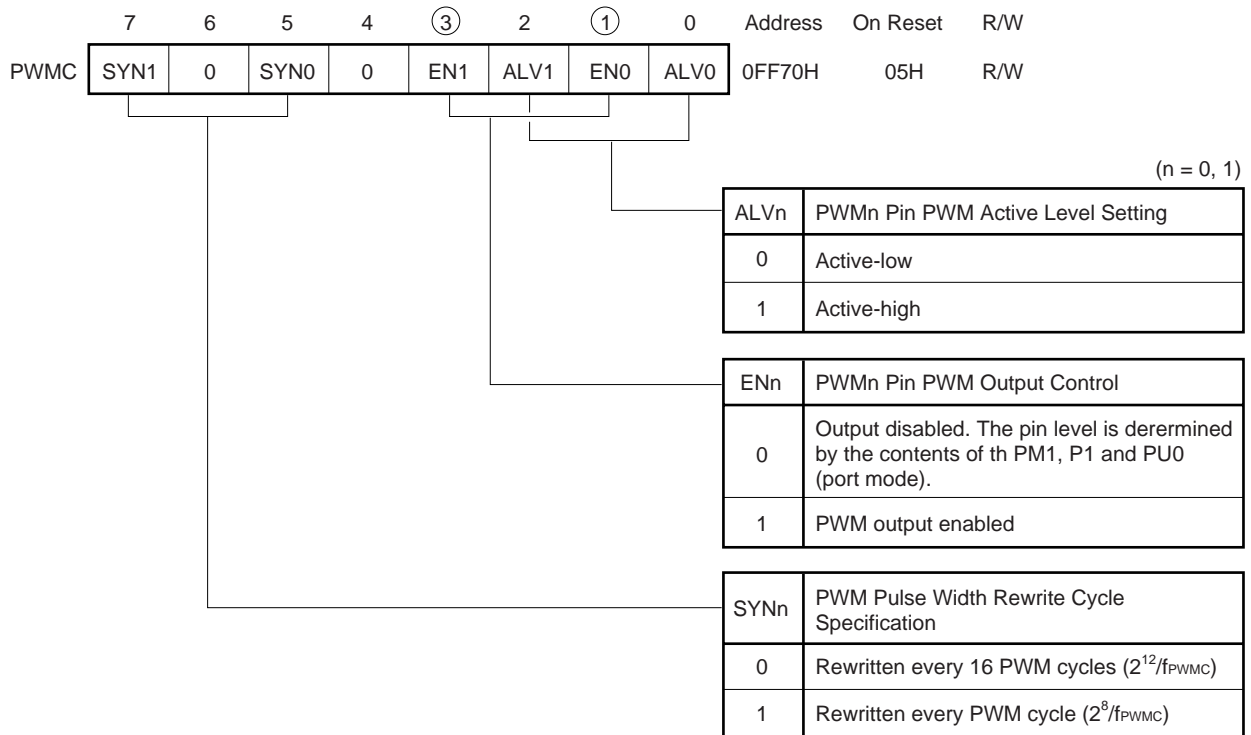
Scales  $f_{CLK}$ , and generates the reference clock.

**13.2 PWM OUTPUT UNIT CONTROL REGISTERS****13.2.1 PWM Control Register (PWMC)**

The PWMC is an 8-bit register that controls the operating status of the PWM output pins (PWMn: n = 0, 1).

The PWMC can be read or written to with an 8-bit manipulation instruction or bit manipulation instruction. Its format is shown in Figure 13-2.

When  $\overline{\text{RESET}}$  is input, PWMC is set to 05H, the PWMn pin is set to port mode, and the input state (output high impedance) is set.

**Figure 13-2 PWM Control Register (PWMC) Format**

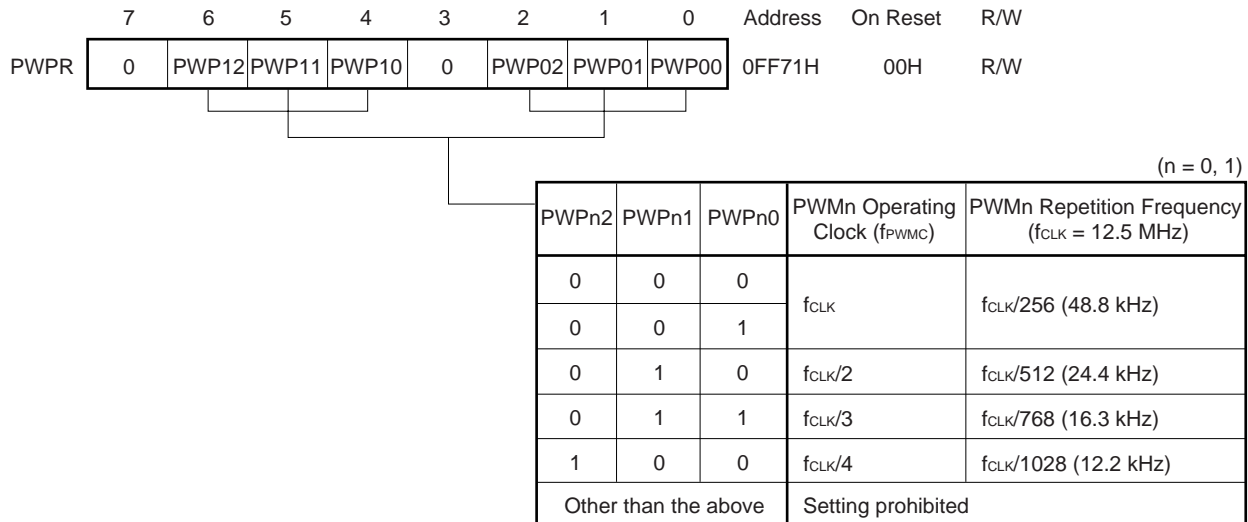
### 13.2.2 PWM Prescaler Register (PWPR)

The PWPR is an 8-bit register that selects the PWM output circuit operating clock ( $f_{PWM}$ ).

The PWPR can be read or written to with an 8-bit manipulation instruction. Its format is shown in Figure 13-3.

When  $\overline{RESET}$  is input, PWPR is cleared to 00H, and  $f_{CLK}$  is selected as  $f_{PWM}$  for both channels.

Figure 13-3 PWM Prescaler Register (PWPR) Format



### 13.2.3 PWM Modulo Registers (PWM0, PWM1)

The PWM modulo register 16-bit register (PWMn:  $n = 0, 1$ ) is a 16-bit register that determines the PWM pulse width. Reads/writes by a 16-bit manipulation instruction only are possible for data setting.

The contents of bits 4 to 15 of the PWMn determines the 12-bit PWM pulse width (12-bit resolution). Bits 3 to 0 have no meaning, and PWM output is not affected whether 1 or 0 is written to these bits.

When  $\overline{RESET}$  is input, the PWMn content are undefined, and therefore data must be set by the program before PWM output is enabled.

**Caution** A value between 0000H and 00FFH should not be set in the PWM modulo registers (PWMn:  $n = 0, 1$ ). A value between 0100H and FFFFH should be set in the PWMn registers. Outputtable PWM signal duty values are 17/4096 to 4096/4096.



### 13.3 PWM OUTPUT UNIT OPERATION

#### 13.3.1 Basic PWM Output Operation

The PWM pulse output duty is determined by the value set in bits 4 to 15 of the PWM modulo register (PWMn: n = 0, 1) as shown below.

$$\text{PWM pulse output duty} = \frac{(\text{Value of PWMn bits 4 to 15})^* + 1}{4096}$$

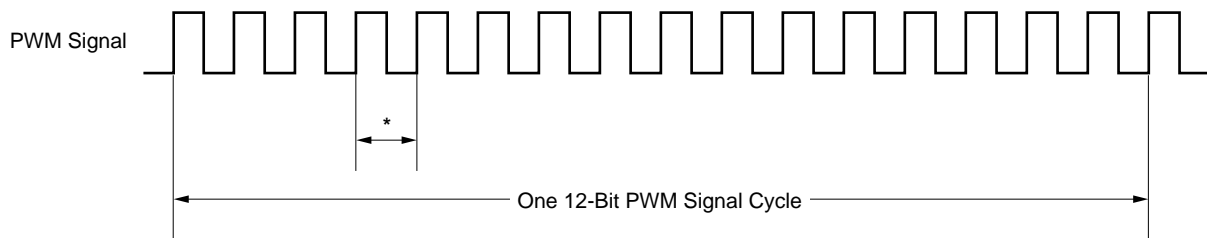
$$* \quad 16 \leq (\text{Value of PWMn bits 4 to 15}) \leq 4095$$

The PWM pulse output repetition frequency is the frequency obtained by division-by-256 of the PWM clock  $f_{\text{CLK}}/1$  to  $f_{\text{CLK}}/4$  set by the PWM prescaler register (PWPR) ( $=f_{\text{PWM}}/256$ ), and the minimum pulse width is  $1/f_{\text{PWM}}$ .

In PWM pulse output, 12-bit resolution is achieved by repeating output of a  $f_{\text{PWM}}/256$  repetition frequency 8-bit resolution PWM signal 16 times.

The addition of extra pulses ( $1/f_{\text{PWM}}$ ) to the 8-bit resolution PWM pulses determined by bits 8 to 15 of the PWMn every cycle is controlled in accordance with the value of bits 4 to 7 of the PWMn to implement a PWM pulse signal once every 16 cycles.

**Figure 13-4 Basic PWM Output Operation**



\* 8-bit resolution per PWM pulse cycle

### 13.3.2 PWM Pulse Output Enabling/Disabling

When PWM pulses are output, the ENn (n = 0, 1) bits of the PMC register are set (1) after data is set in the PWM prescaler register (PWPR) and PWM modulo register (PWMn: n = 0, 1). As a result, PWM pulses with the active level specified by ALVn (n = 0, 1) bit of the PWM control register (PWMC) are output from the PWM output pin.

When the ENn bits of the PWMC are cleared (0), the PWM output unit immediately stops the PWM output operation and the PWM output pins are set to the state specified by the PM1, P1 and PUO registers.

That is, when PM1n (n = 0, 1) in the port 1 mode register (PM1) is 0, the output state is set and the contents of P1n (n = 0, 1) are specified. When PM1n = 1 (n = 0, 1) the input state is set, when PUO1 in the pull-up resistor option register (PUO) is 1 the high level is set by the on-chip pull-up resistor, and when PUO1 = 0 the output high-impedance state is set.

### 13.3.3 PWM Pulse Active Level Specification

The ALVn (n = 0, 1) bit of the PWM control register (PWMC) specify the active level of PWM pulses output from the PWM output pins.

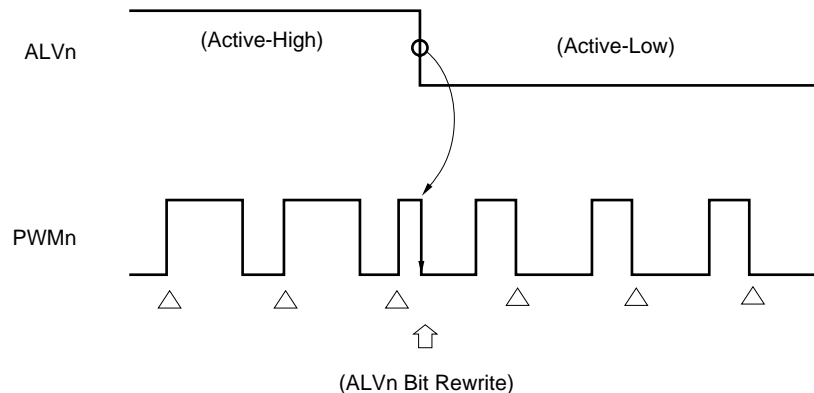
When ALVn bit is set (1), active-high level pulses are output, and when cleared (0), active-low level pulses are output.

When ALVn bit is rewritten, the PWM active level changes immediately. PWM output active level setting and pin states are shown in Figure 13-5.

Figure 13-5 shows the case where ALVn bit is switched when the ENn (n = 0/1) bit of the PWMC is set (1) and PWM output is enabled.

The pin state does not change if ALVn is rewritten when ENn bit is in the cleared (0) state.

Figure 13-5 PWM Output Active Level Setting



**Remark** ENn = 1 (n = 0, 1)

### 13.3.4 PWM Pulse Width Rewrite Cycle Specification

The start of PWM output and pulse width changes are performed in synchronization either with every 16 PWM pulse cycles ( $2^{12}/f_{\text{PWMC}}$ ) or with every PWM pulse cycle ( $2^8/f_{\text{PWMC}}$ ). This PWM pulse width rewrite cycle specification is performed by means of the SYNn bits of the PWM control register (PWMC).

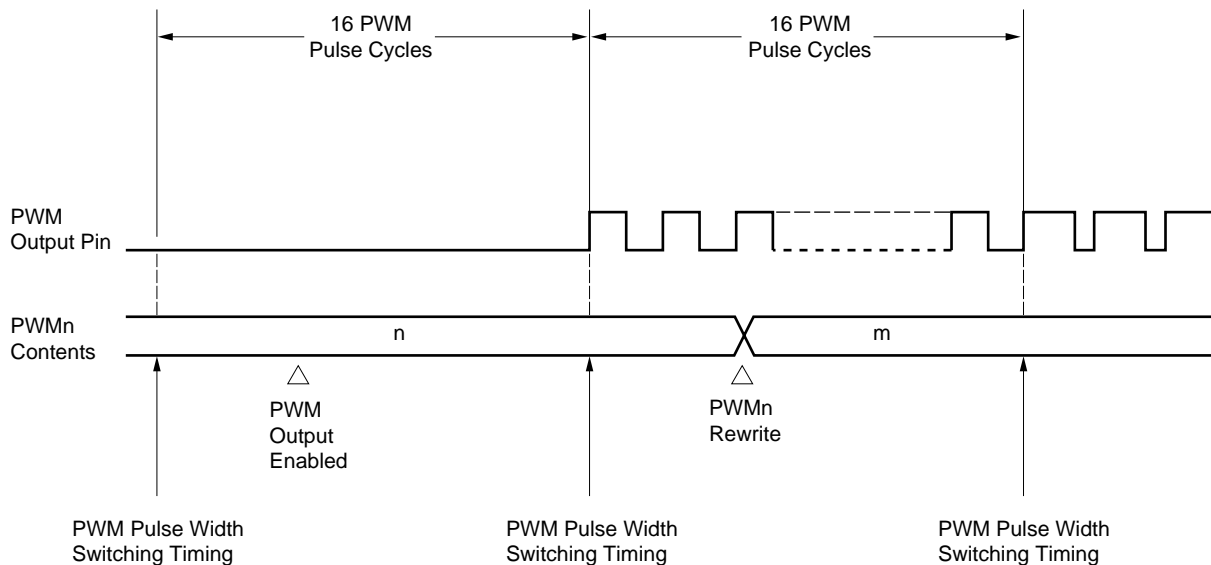
When the SYNn bit is cleared (0), a pulse width change is performed every 16 PWM pulse cycles ( $2^{12}/f_{\text{PWMC}}$ ). It therefore takes a maximum of  $2^{12}$  clocks ( $328\ \mu\text{s}$  when  $f_{\text{PWMC}} = 12.5\ \text{MHz}$ ) until a pulse of a width corresponding to the data written in the PWM modulo register (PWMn:  $n = 0, 1$ ) is output. An example of the PWM output timing at this time is shown in Figure 13-6.

When the SYNn bit is set (1), on the other hand, a pulse width change is performed every PWM pulse cycle ( $2^8/f_{\text{PWMC}}$ ). In this case, it takes a maximum of  $2^8$  clocks ( $20.5\ \mu\text{s}$  when  $f_{\text{PWMC}} = 12.5\ \text{MHz}$ ) until a pulse of a width corresponding to the data written in the PWMn in is output.

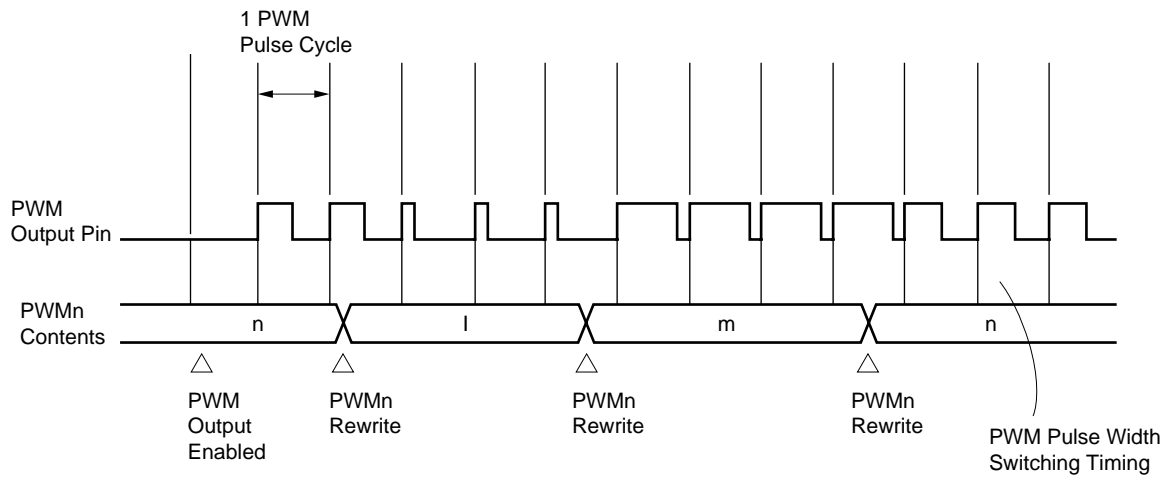
However, caution is required since, if the PWM pulse rewrite cycle is specified as every  $2^8/f_{\text{PWMC}}$ , (if the SYNn bit is set (1)), the obtained PWM pulse precision is between 8 bits and 12 bits, and is lower than when the PWM pulse rewrite cycle is specified as  $2^{12}/f_{\text{PWMC}}$ .

An example of the PWM output timing when the rewrite timing is  $2^8/f_{\text{PWMC}}$  is shown in Figure 13-7.

**Figure 13-6 PWM Output Timing Example 1 (PWM Pulse Width Rewrite Cycle =  $2^{12}/f_{\text{PWMC}}$ )**



- Cautions**
1. Pulse width rewriting is performed every PWM pulse cycle.
  2. The PWM pulse precision is 12 bits.

**Figure 13-7 PWM Output Timing Example 2 (PWM Pulse Width Rewrite Cycle =  $2^8/f_{\text{PWMC}}$ )**

- Cautions**
1. Pulse width rewriting is performed every PWM pulse cycle.
  2. The PWM pulse precision is between 8 and 12 bits.

**Remark** 1, m and n mean the PWMn contents.

#### 13.4 CAUTION

A value between 0000H and 00FFH should not be set in the PWM modulo registers (PWMn: n = 0, 1). A value between 0100H and FFFFH should be set in the PWMn. Outputtable PWM signal duty values are 17/4096 to 4096/4096.

## CHAPTER 14 A/D CONVERTER

The  $\mu$ PD784026 incorporates an analog/digital (A/D) converter with 8 multiplexed analog inputs (ANI0 to ANI7).

The successive approximation conversion method is used, and the conversion result is held in the 8-bit A/D conversion result register (ADCR). This allows fast, high-precision conversion to be performed (conversion time of 9.6  $\mu$ s when  $f_{CLK} = 12.5$  MHz and high-speed conversion is used).

There are two modes for starting A/D conversion, as follows:

- Hardware start : Conversion started by trigger input (INTP5).
- Software start : Conversion started in accordance with A/D converter mode register (ADM) bit setting.

After start-up, there are two operating modes, as follows:

- Scan mode : Multiple analog inputs are selected in order, and conversion data is obtained from all pins.
- Select mode: One pin is used as the analog input, and conversion values are obtained in succession.

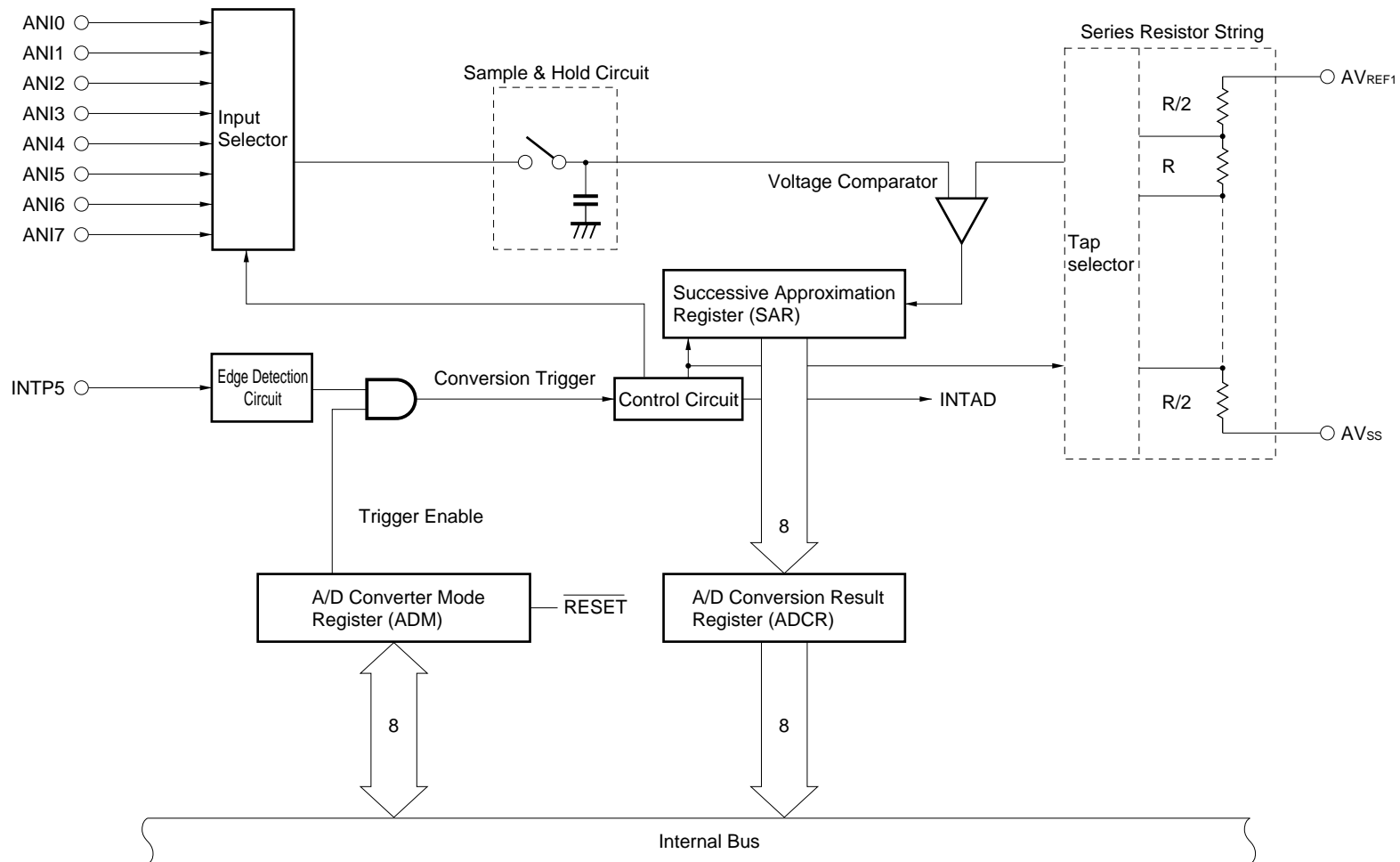
Stoppage of all the above modes and conversion operations is specified by the ADM register.

When the conversion result is transferred to the ADCR, an INTAD interrupt request is generated. This allows conversion values to be transferred to memory in succession by means of macro service.

### 14.1 CONFIGURATION

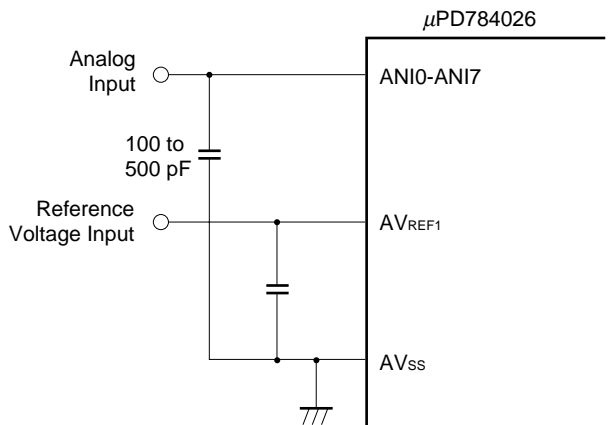
The A/D converter configuration is shown in Figure 14-1.

Figure 14-1 A/D Converter Block Diagram



**Cautions 1.** A capacitor should be connected between the analog input pins (ANI0 to ANI7) and  $AV_{SS}$  and between the reference voltage input pin ( $AV_{REF}$ ) and  $AV_{SS}$  to prevent misoperation due to noise. Be sure to connect the capacitor as closely to ANI0 through ANI7 and  $AV_{REF1}$  as possible.

**Figure 14-2 Example of Capacitor Connection on A/D Converter Pins**



**2.** A voltage outside the range  $AV_{SS}$  to  $AV_{REF1}$  should not be applied to pins used as A/D converter input pins. See 14.5 CAUTIONS for details.

**(1) Input circuit**

The input circuit selects the analog input in accordance with the specification of the A/D converter mode register (ADM), and sends the analog input to the sample & hold circuit according to the operating mode,

**(2) Sample & hold circuit**

The sample & hold circuit samples the analog inputs arriving sequentially one by one and holds the analog input in the process of A/D conversion.

**(3) Voltage comparator**

The voltage comparator determines the voltage difference between the analog input and the series resistor string value tap.

**(4) Series resistor string**

The series resistor string is used to generate voltages that match the analog inputs.

The series resistor string is connected between the A/D converter reference voltage pin ( $AV_{REF1}$ ) and the A/D converter GND pin ( $AV_{SS}$ ). To provide 256 equal voltage steps between the two pins, it is made up of 255 equal resistors and two resistors with half that resistance value.

The series resistor string voltage tap is selected by a tap selector controlled by the successive approximation register (SAR).

**(5) SAR: Successive Approximation Register**

The SAR is an 8-bit register in which the data for which the series resistor string voltage tap value matches the analog input voltage value is set bit by bit starting from the most significant bit (MSB).

When data has been set up to the least significant bit (LSB) of the SAR (when A/D conversion is completed), the SAR contents (conversion result) are stored in the A/D conversion result register (ADCR).

**(6) ADCR: A/D Conversion Result Register**

The ADCR is an 8-bit register that holds the A/D conversion result. The conversion result is loaded into this register from the successive approximation register (SAR) each time A/D conversion finishes.

The contents of this register approximation are undefined when  $\overline{\text{RESET}}$  is input.

**(7) Edge detection circuit**

The edge detection circuit detects a valid edge from the interrupt request input pin (INTP5) input, and generates an external interrupt request signal (INTP5) and A/D conversion operation external trigger.

The INTP5 pin input valid edge is specified by external interrupt mode register 1 (INTM1) (see **Figure 19-2**). External trigger enabling/disabling is set by means of the A/D converter mode register (ADM) (see **14.2 A/D Converter Mode Register (ADM)**).



## 14.2 A/D CONVERTER MODE REGISTER (ADM)

ADM is an 8-bit register that controls A/D converter operations.

The ADM register can be read or written to with an 8-bit manipulation instruction or bit manipulation instruction. Its format is shown in Figure 14-3.

Bit 0 (MS) controls the operating mode.

Bits 1, 2 and 3 (ANI0, 1, 2) select the analog inputs for A/D conversion.

Bit 5 (SCMD) controls the A/D conversion operation in scan mode.

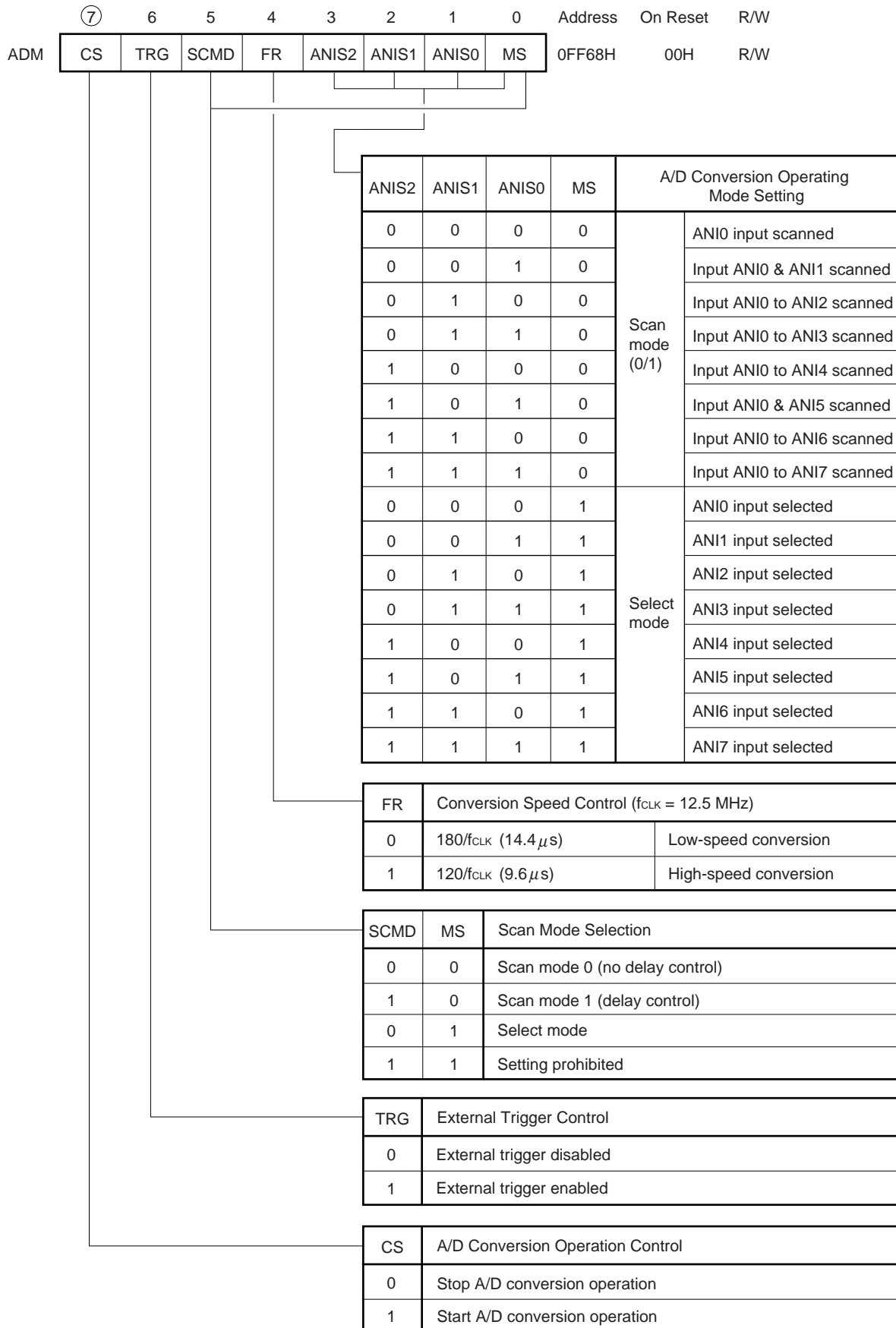
Bit 6 (TRG) enables external synchronization of the A/D conversion operation. If the TRG bit is set (1) when the CS bit is set (1), the conversion operation is initialized with each input of a valid edge as an external trigger to the INTP5 pin. When the TRG bit is cleared (0), the conversion operation is performed without regard to the INTP5 pin.

Bit 7 (CS) controls the A/D conversion operation. When the CS bit is set (1) the conversion operation is started, and when cleared (0), all conversion operations are stopped even if conversion is in progress. In this case, the A/D conversion result register (ADCR) is not updated and an INTPAD interrupt request is not generated. Also, the power supply to the voltage comparator is stopped, and the A/D converter consumption current is reduced.

$\overline{\text{RESET}}$  input clears the ADM register to 00H.

**Caution** When the STOP mode or IDLE mode is used, the consumption current should be reduced by clearing (0) the CS bit before entering the STOP or IDLE mode. If the CS bit remains set (1), the conversion operation will be stopped by entering the STOP or IDLE mode, but the power supply to the voltage comparator will not be stopped, and therefore the A/D converter consumption current will not be reduced.

Figure 14-3 A/D Converter Mode Register (ADM) Format



**Caution** Once the A/D converter starts operating, conversion operations are performed repeatedly until the CS bit of the A/D converter mode register (ADM) is cleared (0). Therefore, a superfluous interrupt may be generated if ADM setting is performed after interrupt-related registers, etc., when A/D converter mode conversion, etc., is performed. The result of this superfluous interrupt is that the conversion result storage address appears to have been shifted when the scan mode is used. Also, when the select mode is used, the first conversion result appears to have been an abnormal value, such as the conversion result for the other channel. It is therefore recommended that A/D converter mode conversion be carried out using the following procedure.

- <1> Write to the ADM (CS bit must be set (1))
- <2> Interrupt request flag (ADIF) clearance (0)
- <3> Interrupt mask flag or interrupt service mode flag setting

Operations <1> to <3> should not be divided by an interrupt or macro service. When scan mode 0 (no delay control) is used, in particular, you should ensure that the time between <1> and <2> is less than the time taken by one A/D conversion operation.

Alternatively, the following procedure is recommended.

- <1> Stop the A/D conversion operation by clearing (0) the CS bit of the ADM.
- <2> Interrupt request flag (ADIF) clearance (0).
- <3> Interrupt mask flag or interrupt service mode flag setting
- <4> Write to the ADM

## 14.3 OPERATION

### 14.3.1 Basic A/D Converter Operation

#### (1) A/D Conversion Operation procedure

A/D conversion is performed by means of the following procedure:

- (a) Analog pin selection and operating mode specification are set with the A/D converter mode register (ADM).
- (b) Bit 7 (CS) of the ADM is set (1), and A/D conversion is started.
- (c) When conversion starts, the MSB (bit 7) of the successive approximation register (SAR) is set (1) automatically.
- (d) When bit 7 of the SAR is set (1), the tap selector sets the series resistor string voltage tap to

$$\frac{255}{512} AV_{REF1} (\approx 1/2 AV_{REF1}).$$

- (e) The voltage difference between the series resistor string voltage tap and the analog input is determined by the voltage comparator. If the analog input is greater than  $(1/2) AV_{REF1}$ , the MSB of the SAR remains set (1), and if it is less than  $(1/2) AV_{REF1}$ , the MSB is cleared (0).
- (f) Next, bit 6 of the SAR is set (1) automatically, and the next comparison is performed. Here, the series resistor string voltage tap is selected according to the value of bit 7 for which the result has already been set, as shown below.

- Bit 7 = 1 .....  $\frac{383}{512} AV_{REF1} \approx \frac{3}{4} AV_{REF1}$
- Bit 7 = 0 .....  $\frac{127}{512} AV_{REF1} \approx \frac{1}{4} AV_{REF1}$

This voltage tap is compared with the analog input voltage, and bit 6 of the SAR is manipulated as follows according to the result:

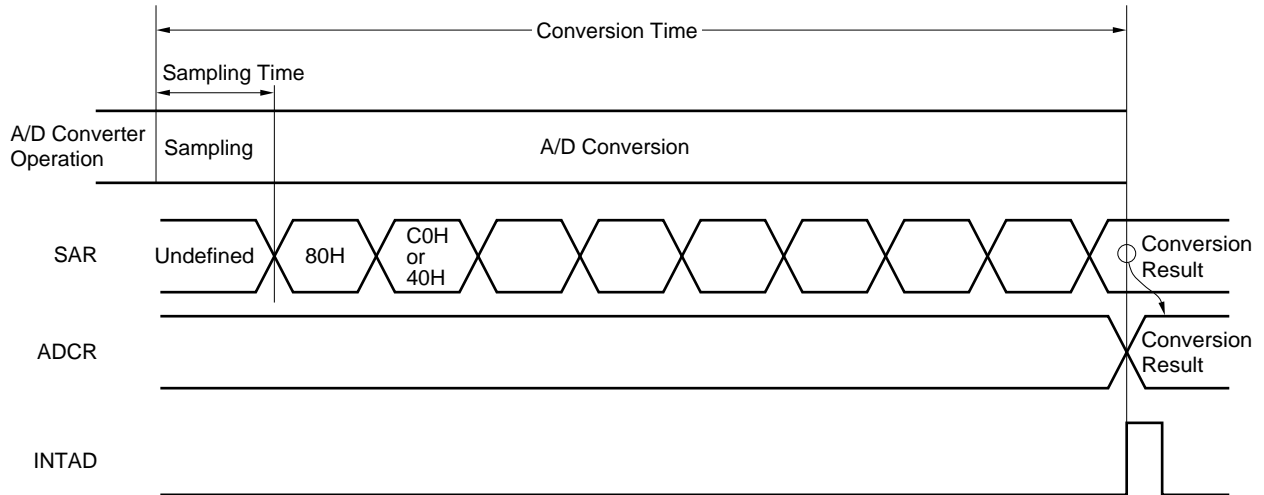
- Analog input voltage  $\geq$  voltage tap: Bit 6 = 1
- Analog input voltage  $<$  voltage tap: Bit 6 = 0

- (g) The same kind of comparison is continued up to the LSB (bit 0) of the SAR (binary search method).

- (h) When comparison of the 8 bits is completed, a valid digital result is left in the SAR, and that value is transferred to the A/D conversion result register (ADCR) and latched.

An A/D conversion operation end interrupt request (INTAD) can be generated at the same time.

**Figure 14-4 Basic A/D Converter Operation**



A/D conversion operations are performed successively until the CS bit is cleared (0) by software. If a write operation is performed on the ADM during an A/D conversion operation, the conversion operation is initialized, and if the CS bit is set (1), conversion will be started from the beginning.

The contents of the ADCR are undefined after RESET input.

**(2) Input voltage and conversion result**

The relationship between the analog input voltage input to an analog input pin (ANI0 to ANI7) and the A/D conversion result (value stored in ADCR) is shown by the following expression:

$$\text{ADCR} = \text{INT}\left(\frac{V_{\text{IN}}}{\text{AV}_{\text{REF1}}} \times 256 + 0.5\right)$$

or

$$(\text{ADCR} - 0.5) \times \frac{\text{AV}_{\text{REF1}}}{256} \leq V_{\text{IN}} < (\text{ADCR} + 0.5) \times \frac{\text{AV}_{\text{REF1}}}{256}$$

**Remark** INT( ) : Function that returns the integer part of the value in ( )

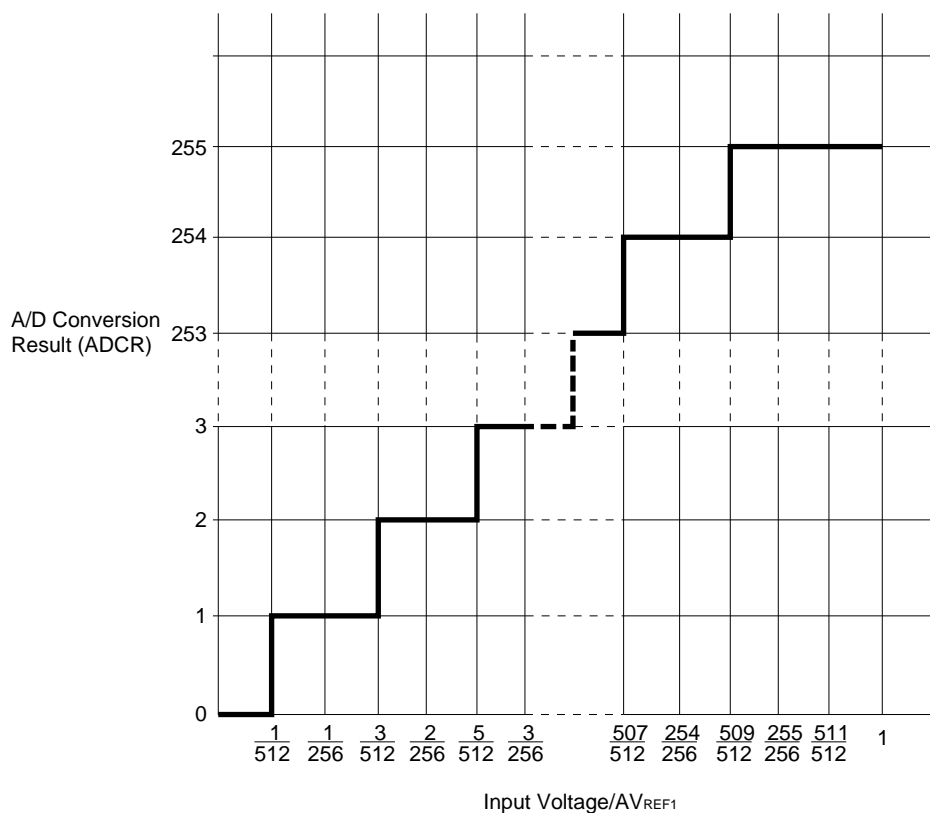
$V_{\text{IN}}$  : Analog input voltage

$\text{AV}_{\text{REF1}}$  :  $\text{AV}_{\text{REF1}}$  pin voltage

ADCR : ADCR value

Figure 14-5 shows the relationship between the analog input voltage and the A/D conversion result in graphic form.

**Figure 14-5 Relationship Between Analog Input Voltage and A/D Conversion Result**



**(3) A/D conversion time**

The A/D conversion time is determined by the system clock frequency ( $f_{CLK}$ ) and the FR bit of the A/D converter mode register (ADM).

The A/D conversion time includes the entire time required for one A/D conversion operation, and the sampling time is also included in the A/D conversion time.

These values are shown in Table 14-1.

**Table 14-1 A/D Conversion Time**

System Clock ( $f_{CLK}$ ) Range	FR Bit	Conversion Time	Sampling Time
$0.25 \text{ MHz} \leq f_{CLK} \leq 12.5 \text{ MHz}$	0	$240/f_{CLK}$ ( $19 \mu\text{s} - 120 \mu\text{s}$ )	$48/f_{CLK}$ ( $3.8 \mu\text{s} - 24 \mu\text{s}$ )
$0.25 \text{ MHz} \leq f_{CLK} \leq 12.5 \text{ MHz}$	1	$120/f_{CLK}$ ( $9.6 \mu\text{s} - 60 \mu\text{s}$ )	$24/f_{CLK}$ ( $1.9 \mu\text{s} - 12 \mu\text{s}$ )

**(4) A/D converter operating modes**

There are two A/D converter operating modes, scan mode and select mode. These modes are selected according to the setting of bit 0 (MS) of the A/D converter mode register (ADM).

In addition, scan mode 0 or 1 can be selected by bit 5 (SCMD) of the ADM.

Operation in either mode continues until the ADM is rewritten.

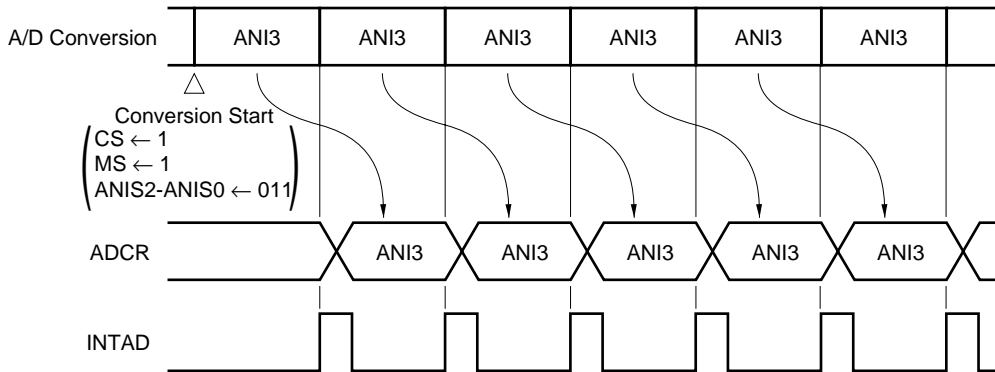
### 14.3.2 Select Mode

One analog input is specified by bits 1 to 3 (ANIS0 to ANIS2) of the A/D converter mode register (ADM), and A/D conversion of the specified analog input pin is started. The conversion result is stored in the A/D conversion result register (ADCR).

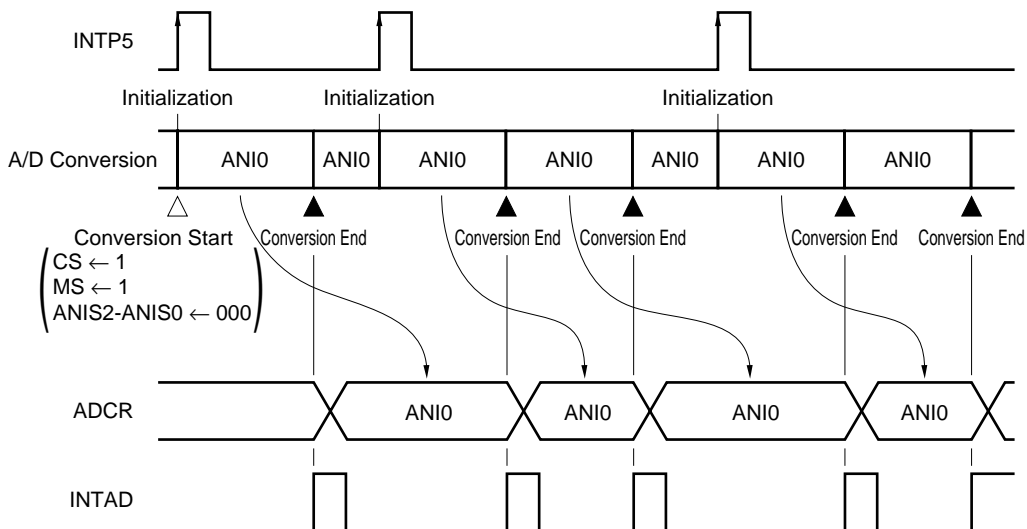
An A/D conversion end interrupt request (INTAD) is generated at the end of each conversion operation.

Figure 14-6 Select Mode Operation Timing

(a) TRG bit  $\leftarrow 0$



(b) TRG bit  $\leftarrow 1$





### 14.3.3 Scan Mode

Two scan modes, 1 and 0, are available. In scan mode 0, delay control that takes delay in reading the A/D conversion result by the CPU into consideration can be performed. In scan mode 1, no delay control is performed but the A/D conversion interval is fixed.

Generally, use of scan mode 1 is recommended.

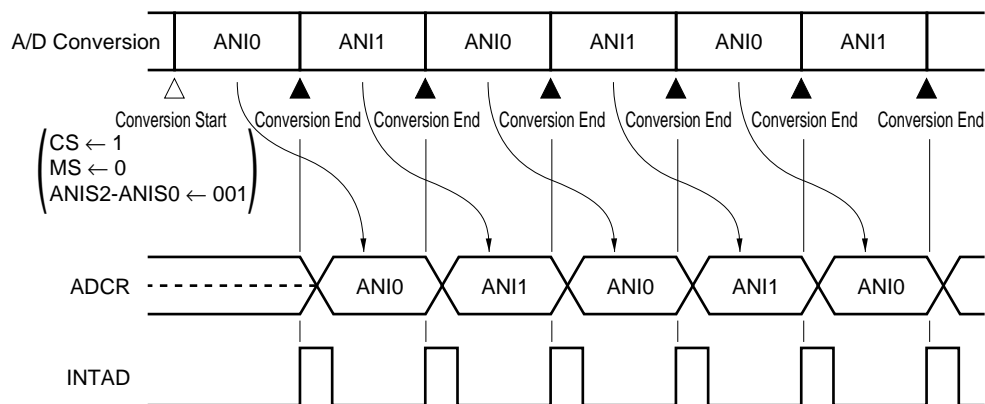
#### (1) Scan mode 0 (bit 5 (SCMD) of A/D converter mode register (ADM) = 0)

Input from the analog input pins specified by bits 1 to 3 (ANIS0 to ANIS2) of the ADM is selected and converted in order.

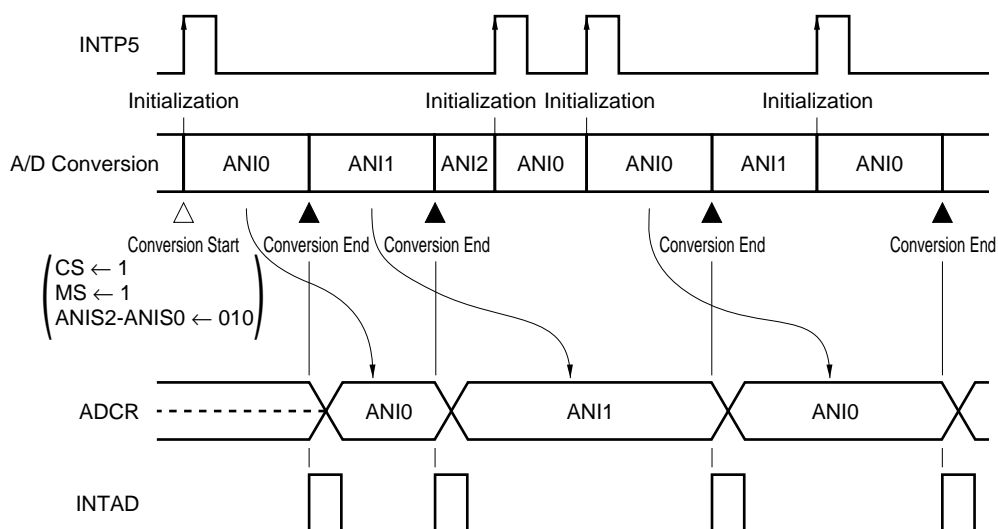
For example, if ANIS2 to ANIS0 of the ADM = 001, ANI0 and ANI1 will be scanned repeatedly (ANI0 → ANI1 → ANI0 → ANI1 → ...). In the scan mode, at the end of the conversion operation for each input the conversion value is stored in the A/D conversion result register (ADCR) and an A/D conversion end interrupt request (INTAD) is generated.

Figure 14-7 Scan Mode 0 Operation Timing

#### (a) TRG bit ← 0



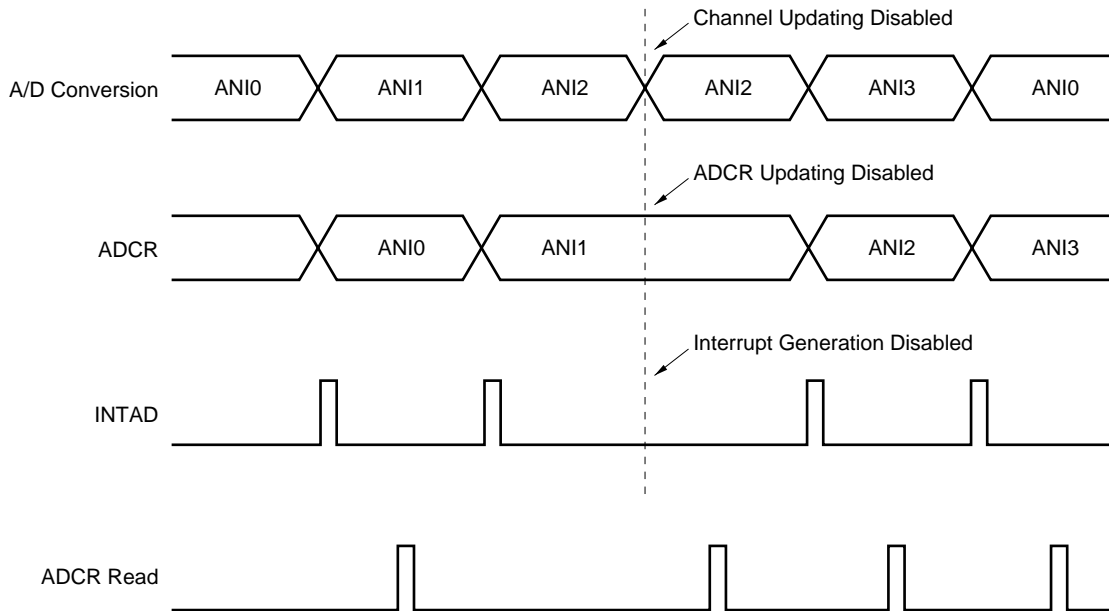
#### (b) TRG bit ← 1



**(2) Scan mode 1 (bit 5 (SCMD) of A/D converter mode register (ADM) = 1)**

When bit 5 of the ADM is set (1), the analog input pins specified by bits 1 to 3 (ANIS0 to ANIS2) are selected, and subjected to conversion, in order. If an A/D conversion result register (ADCR) read is not performed by the CPU by the end of the next A/D conversion after A/D conversion end (INTAD) generation, conversion is restarted without performing INTAD generation, ADCR updating or channel updating (see **Figure 14-8**).

If an ADCR read is performed by the CPU before the end of the next A/D conversion, the same operation as in scan mode 0 is performed.

**Figure 14-8 Scan Mode 1 Operation Timing**

**14.3.4 A/D Conversion Operation Start by Software**

An A/D conversion operation start by software is performed by writing a value to the A/D converter mode register (ADM) that sets the TRG bit of the ADM register to 0 and the CS bit to 1.

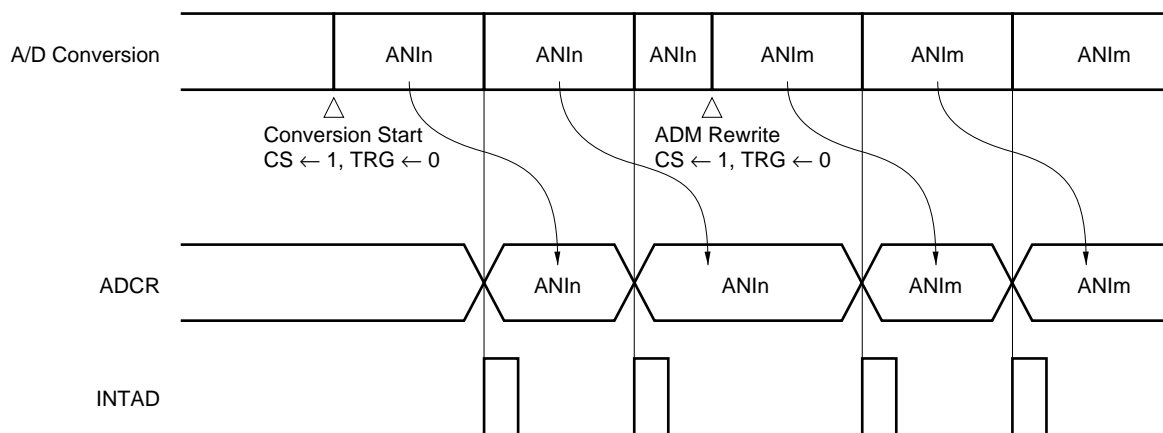
If a value is written to the ADM during an A/D conversion operation (CS bit = 1) such that the TRG bit is set to 0 and the CS bit to 1 again, the A/D conversion operation being performed at that time is suspended, and A/D conversion is started immediately in accordance with the written value.

Once A/D conversion operation is started, as soon as one A/D conversion operation ends the next A/D conversion operation is started in accordance with the operating mode set by the ADM, and conversion operations continue repeatedly until an instruction that writes to the ADM is executed.

When A/D conversion operation is started by software (TRG bit = 0), INTP5 pin (P26 pin) input does not affect the A/D conversion operation.

**(1) Select mode A/D conversion operation**

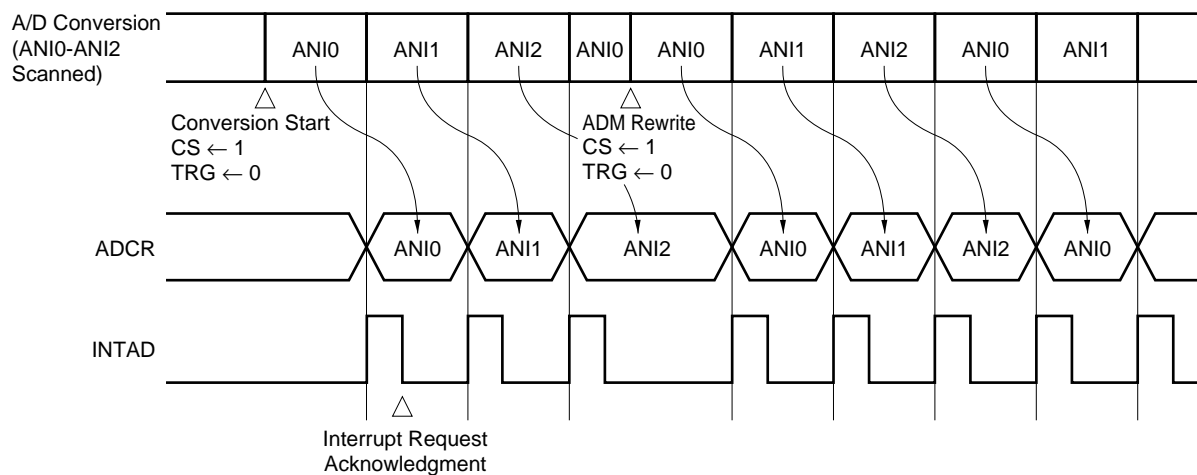
An A/D conversion operation is started on the analog input pin set by the A/D converter mode register (ADM). As soon as the A/D conversion operation ends, another A/D conversion operation is performed on the same analog input pin. An A/D conversion end interrupt request (INTAD) is generated at the end of each A/D conversion operation.

**Figure 14-9 Software Start Select Mode A/D Conversion Operation**

**Remark**  $n = 0, 1, \dots, 7$   
 $m = 0, 1, \dots, 7$

**(2) Scan mode A/D conversion operation**

When conversion operation is started, an A/D conversion operation is started on the ANI0 pin input. When the A/D conversion operation ends, an A/D conversion operation is started on the next analog input pin. An A/D conversion end interrupt request (INTAD) is generated at the end of each A/D conversion operation.

**Figure 14-10 Software Start Scan Mode A/D Conversion Operation**

### 14.3.5 A/D Conversion Operation Start by Hardware

An A/D conversion operation start by hardware is made possible by setting both the TRG bit and the CS bit of the A/D converter mode register (ADM) to 1. When the TRG bit and the CS bit of the ADM are both set to 1, external signals are placed in the standby state, and an A/D conversion operation is started when a valid edge is input to the INTP5 pin (P26 pin).

If another valid edge is input to the INTP5 pin after the A/D conversion operation has been started by a valid edge input to the INTP5 pin, the A/D conversion operation being performed at that time is suspended, and A/D conversion is performed from the beginning in accordance with the contents set in the ADM.

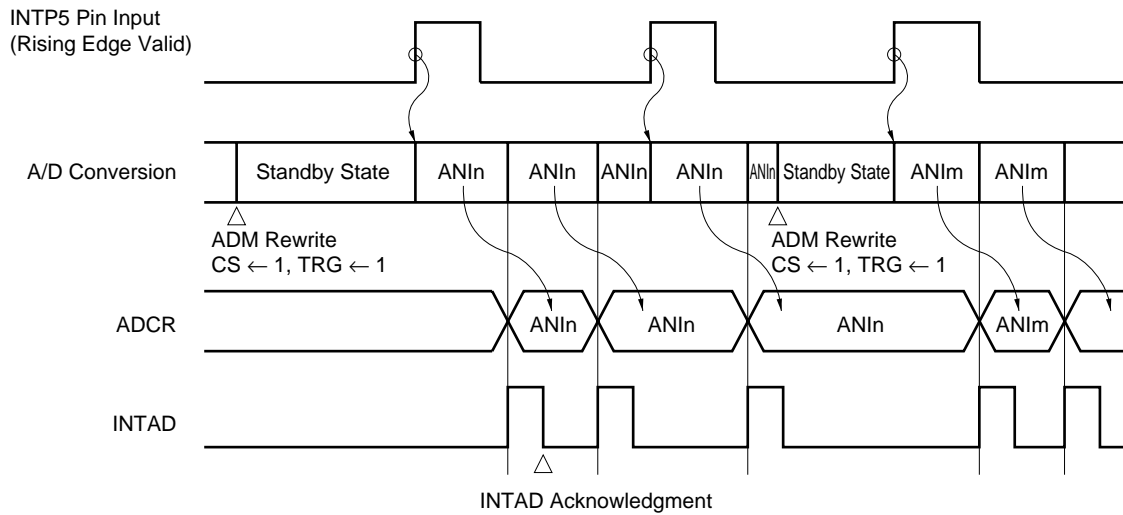
If a value is written to the ADM during an A/D conversion operation (CS bit = 1) such that the TRG bit and CS bit are both set to 1 again, the A/D conversion operation being performed at that time is suspended (the standby state is also suspended), and a standby state is entered in which the A/D converter waits for input of a valid edge to the INTP5 pin in the A/D conversion operation mode in accordance with the written value, and a conversion operation is started when a valid edge is input.

Use of this function allows A/D conversion operations to be synchronized with external signals. Once A/D conversion operation is started, as soon as one A/D conversion operation ends the next A/D conversion operation is started in accordance with the operating mode set by the ADM (the A/D converter does not wait for INTP5 pin input), and conversion operations continue repeatedly until an instruction that writes to the ADM is executed, or a valid edge is input to the INTP5 pin.

**Caution** Approximately 10  $\mu$ s is required from the time a valid edge is input to the INTP5 pin until the A/D conversion operation is actually started. This delay must be taken into account in the design stage. See CHAPTER 19 EDGE DETECTION FUNCTION for details of the edge detection function.

**(1) Select mode A/D conversion operation**

An A/D conversion operation is started on the analog input pin set by the A/D converter mode register (ADM). As soon as the A/D conversion operation ends, another A/D conversion operation is performed on the same analog input pin. An A/D conversion end interrupt request (INTAD) is generated at the end of each A/D conversion operation. If a valid edge is input to the INTP5 pin during an A/D conversion operation, the A/D conversion operation being performed at that time is suspended, and a new A/D conversion operation is started.

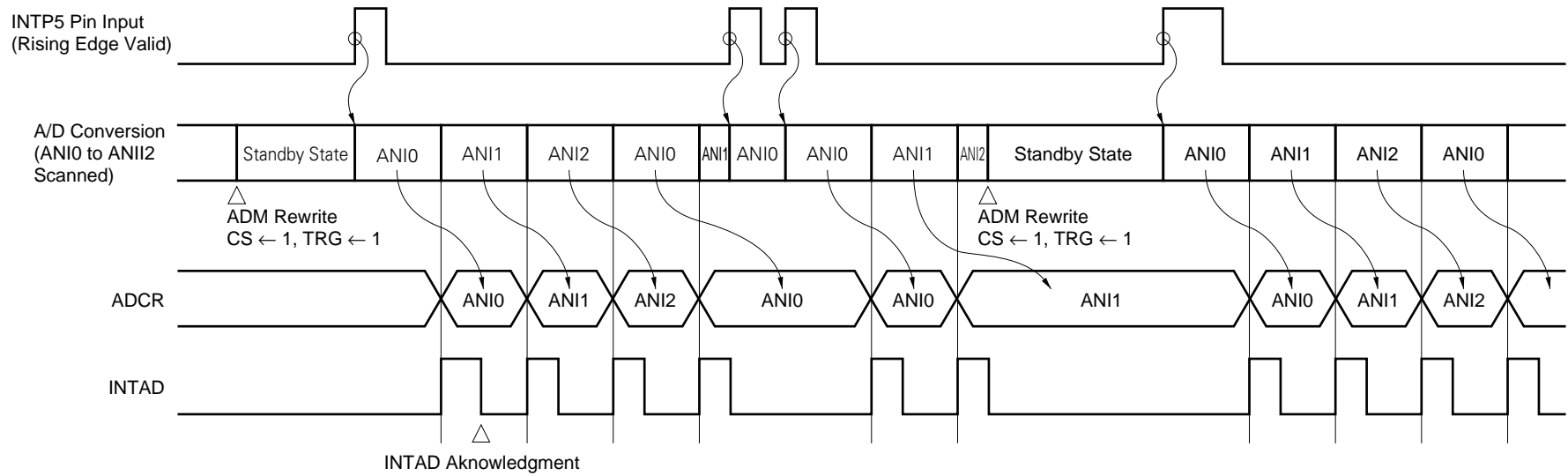
**Figure 14-11 Hardware Start Select Mode A/D Conversion Operation**

**Remark**  $n = 0, 1, \dots, 7$   
 $m = 0, 1, \dots, 7$

**(2) Scan mode A/D conversion operation**

When conversion operation is started, an A/D conversion operation is started on the ANI0 pin input. When the A/D conversion operation ends, an A/D conversion operation is started on the next analog input pin. An A/D conversion end interrupt request (INTAD) is generated at the end of each A/D conversion operation. If a valid edge is input to the INTP5 pin during an A/D conversion operation, the A/D conversion operation being performed at that time is suspended, and a new A/D conversion operation is started on the ANI0 pin input.

**Figure 14-12 Hardware Start Scan Mode A/D Conversion Operation**



## 14.4 EXTERNAL CIRCUIT OF A/D CONVERTER

The A/D converter is provided with a sample & hold circuit to stabilize its conversion operation. This sample & hold circuit outputs sampling noise during sampling immediately after an A/D conversion channel has been changed.

To absorb this sampling noise, an external capacitor must be connected. If the impedance of the signal source is high, an error may occur in the conversion result due to the sampling noise. Especially when the scan mode is used, the impedance of the signal source must be kept low because the channel whose signal is to be converted changes one after another.

One way to absorb the sampling noise is to increase the capacitance of the capacitor. However, if the capacitance is increased too much, the sampling noise is accumulated. Therefore, the most effective way is to reduce the resistance component.

## 14.5 CAUTIONS

### (1) Range of voltages applied to analog input pins

The following must be noted concerning A/D converter analog input pins ANI0 to ANI7 (P70 to P77).

- A voltage outside the range  $AV_{SS}$  to  $AV_{REF1}$  should not be applied to pins subject to A/D conversion during an A/D conversion operation.

If this restriction is not observed, the  $\mu$ PD784026 may be damaged.

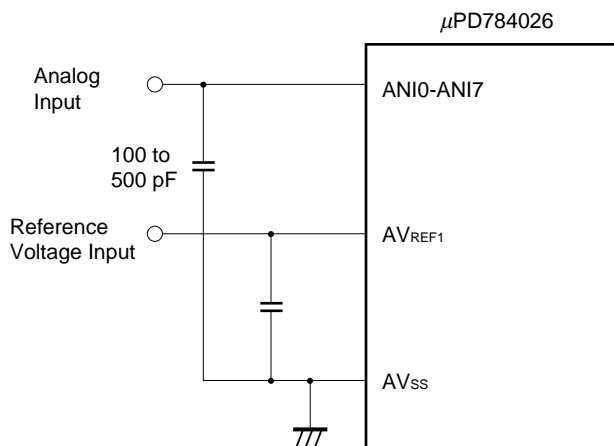
### (2) Hardware start A/D conversion

Approximately 10  $\mu$ s is required from the time a valid edge is input to the INTP5 pin until the A/D conversion operation is actually started. This delay must be taken into account in the design stage. See **CHAPTER 19 EDGE DETECTION FUNCTION** for details of the edge detection function.

### (3) Connecting capacitor to analog input pins

A capacitor should be connected between the analog input pins (ANI0 to ANI7) and  $AV_{SS}$  and between the reference voltage input pin ( $AV_{REF1}$ ) and  $AV_{SS}$  to prevent misoperation due to noise.



**Figure 14-13 Example of Capacitor Connection on A/D Converter Pins**

- (4) When the STOP mode or IDLE mode is used, the consumption current should be reduced by clearing (0) the CS bit before entering the STOP or IDLE mode. If the CS bit remains set (1), the conversion operation will be stopped by entering the STOP or IDLE mode, but the power supply to the voltage comparator will not be stopped, and therefore the A/D converter consumption current will not be reduced.
- (5) Once the A/D converter starts operating, conversion operations are performed repeatedly until the CS bit of the A/D converter mode (ADM) is cleared (0). Therefore, a superfluous interrupt may be generated if ADM setting is performed after interrupt-related registers, etc., are set when A/D converter mode conversion, etc., is performed. The result of this superfluous interrupt is that the conversion result storage address appears to have been shifted when the scan mode is used. Also, when the select mode is used, the first conversion result appears to have been an abnormal value, such as the conversion result for the other channel. It is therefore recommended that A/D converter mode conversion be carried out using the following procedure.

- <1> Write to the ADM (CS bit must be set (1))
- <2> Interrupt request flag (ADIF) clearance (0)
- <3> Interrupt mask flag or interrupt service mode flag setting

Operations <1> to <3> should not be divided by an interrupt or macro service. When scan mode 0 (no delay control) is used, in particular, you should ensure that the time between <1> and <2> is less than the time taken by one A/D conversion operation.

Alternatively, the following procedure is recommended.

- <1> Stop the A/D conversion operation by clearing (0) the CS bit of the ADM.
- <2> Interrupt request flag (ADIF) clearance (0).
- <3> Interrupt mask flag or interrupt service mode flag setting
- <4> Write to the ADM

[MEMO]

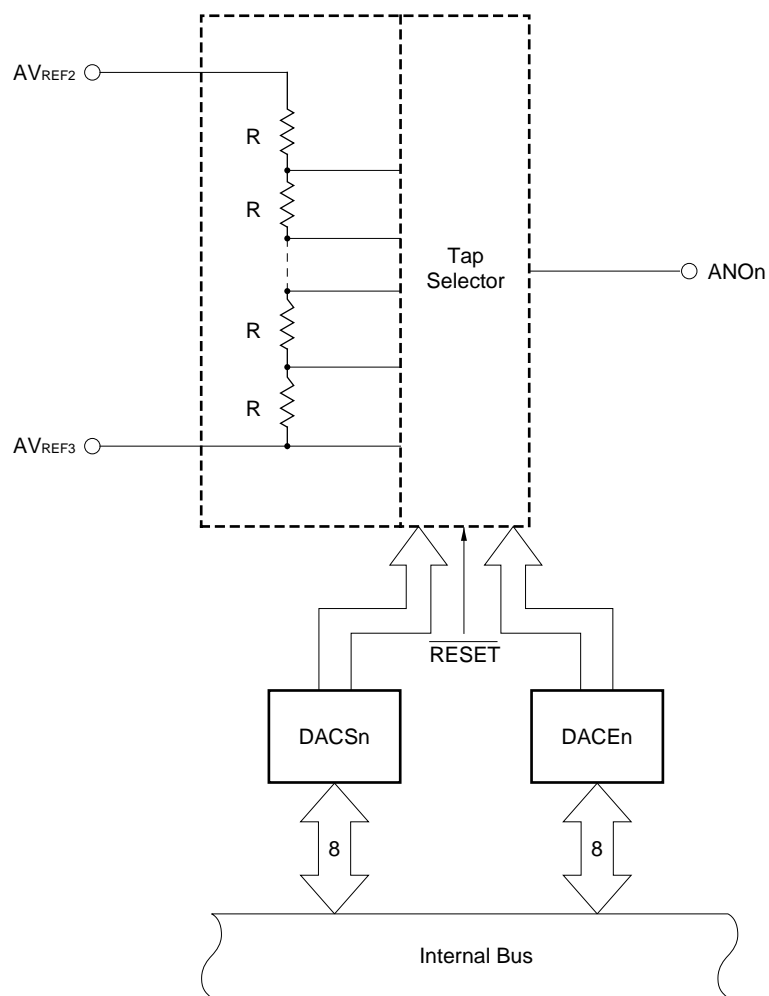
## CHAPTER 15 D/A CONVERTER

The  $\mu$ PD784026 incorporates an 8-bit resolution voltage output type digital/analog (D/A) converter, which uses the resistor string conversion method.

### 15.1 CONFIGURATION

The D/A converter block diagram is shown in Figure 15-1.

**Figure 15-1 D/A Converter Block Diagram**



**Remark**  $n = 0, 1$

**(1) D/A conversion value setting registers (DACS0, DACS1)**

These registers are used to set the voltage values to be output to the ANOn pins ( $n = 0, 1$ ). The voltage value output to the ANOn pin is given by the following expression:

$$ANOn = \frac{AV_{REF2} - AV_{REF3}}{256} \times DACSn + AV_{REF3} [V]$$

$\overline{RESET}$  input initializes these registers to 00H.

**(2) Resistor string**

256 resistors with the same resistance value are connected in series, and the ends are connected to the  $AV_{REF2}$  pin and the  $AV_{REF3}$  pin respectively. Two independent resistor strings are incorporated, for the ANO0 pin and the ANO1 pin.

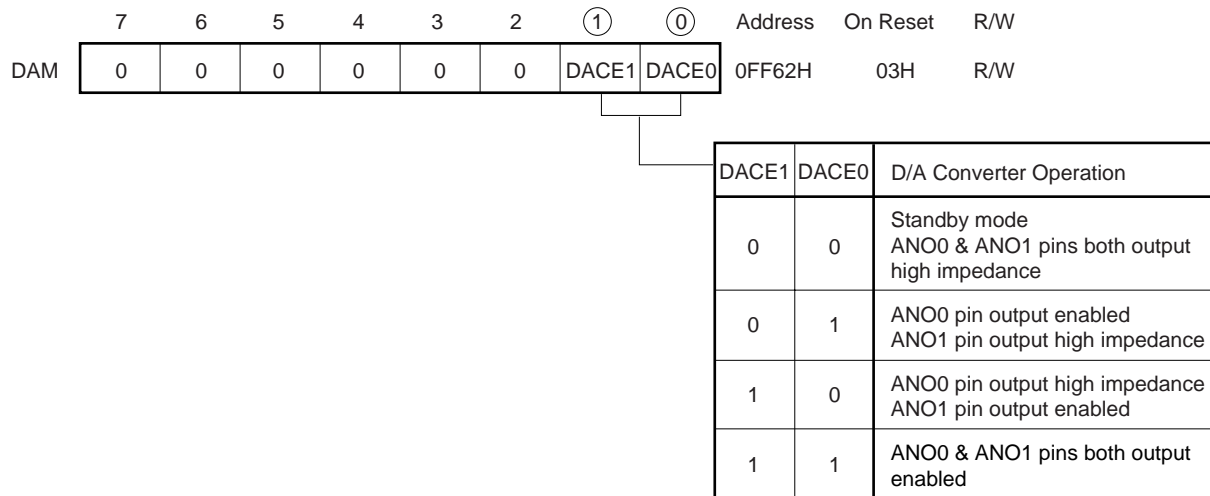
**(3) Tap selector**

One of the 256 resistor string taps is selected in accordance with the value of the D/A conversion value setting register ( $DACS_n$ :  $n = 0, 1$ ), and is connected to the ANOn ( $n = 0, 1$ ) pin.

When  $\overline{RESET}$  is low (0), and when  $DACEn = 0$ , the ANOn pin is not connected to any tap; that is, it is placed in the output high impedance state.

**15.2 D/A CONVERTER MODE REGISTER (DAM)**

The D/A converter mode register (DAM) is an 8-bit register that controls D/A converter operations. The DAM register can be read or written to with an 8-bit manipulation instruction or bit manipulation instruction. DMA format is shown in Figure 15-2.  $\overline{\text{RESET}}$  input sets the DAM register to 03H, enabling D/A conversion output for both channels.

**Figure 15-2 D/A Converter Mode Register (DAM) Format**

## 15.3 D/A CONVERTER OPERATION

### 15.3.1 Basic Operation

When the value to be output is written to the D/A conversion value setting register (DACS<sub>n</sub>,  $n = 0, 1$ ) while the D/A conversion enable bit (DACEn,  $n = 0, 1$ ) of the D/A converter mode register (DAM) is set (1), an analog voltage corresponding to the value written is output from the ANOn pin ( $n = 0, 1$ ). The output voltage is retained until the next value is written to the DACS<sub>n</sub>.

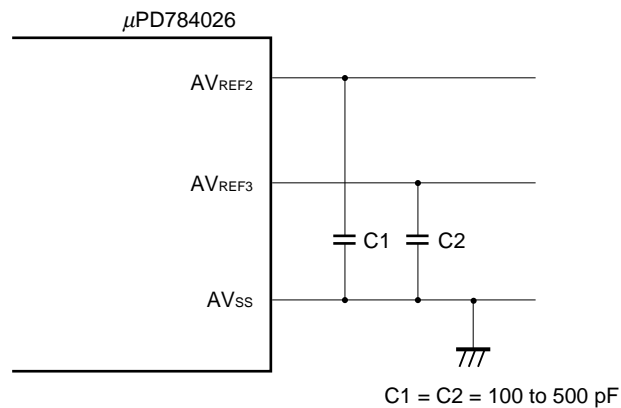
The voltage output from the ANOn pin is determined by the following expression:

$$ANOn = \frac{AV_{REF2} - AV_{REF3}}{256} \times DACSn + AV_{REF3} \text{ [V]}$$

While the  $\overline{RESET}$  input is low, ANOn is in the output high impedance state, and the DACS<sub>n</sub> is initialized to 00H. After  $\overline{RESET}$  release, the same level as the AVREF3 pin is output from the ANOn pin.

Connect capacitors between the reference voltage input pins (AVREF2 and AVREF3) and AVSS to stabilize the operation of the D/A converter.

**Figure 15-3 Example of Connecting Capacitors to Reference Voltage Input Pins of D/A Converter**



### 15.3.2 D/A Converter Standby Operation

When the D/A conversion enable bit (DACEn,  $n = 0, 1$ ) of the D/A converter mode register (DAM) is cleared (0), the ANOn pin ( $n = 0, 1$ ) is set to the output high impedance state.

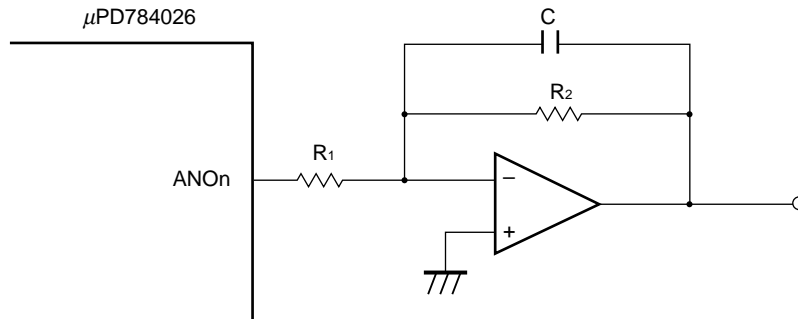
When both DACEn bits are cleared (0), the D/A converter enters standby mode, enabling the consumption current to be reduced.

## 15.4 CAUTIONS

- (1) As the D/A converter output impedance is high, a current cannot be taken from the ANOn pin ( $n = 0, 1$ ). If the load input impedance is low, a buffer amp should be inserted between the load and the ANOn pin. Also, the wiring to the buffer amp and load should be kept as short as possible (since the output impedance is high). If the wiring is long, measures such as enclosure with a ground pattern should be taken.
- (2) As the D/A converter output voltage varies in steps, the signal output by the D/A converter should generally be passed through a low-pass filter before use.
- (3) The D/A converter incorporated in the  $\mu$ PD784026 is in the output high impedance state while  $\overline{\text{RESET}}$  is low. The design should therefore make provision for high impedance input in the load side circuitry.

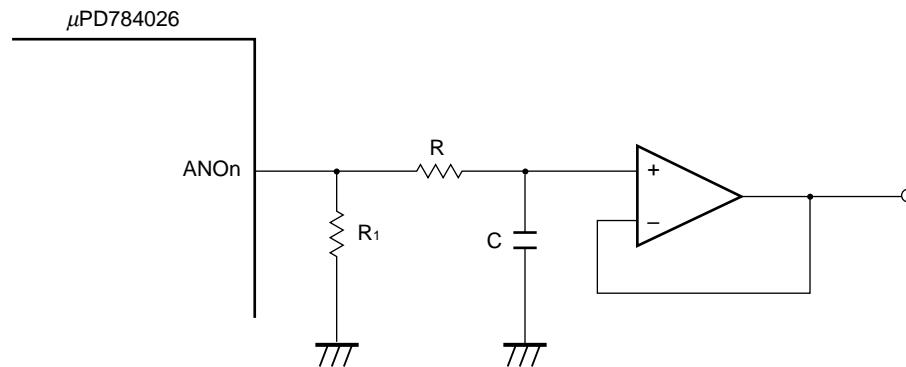
Figure 15-4 Example of Buffer Amp Insertion

## (a) Inverting amp



- Buffer amp input impedance =  $R_1$

## (b) Voltage floor



- Buffer amp input impedance =  $R_1$
- If  $R_1$  were omitted, the output would be undefined when  $\overline{\text{RESET}}$  is low.

- (4) Since the D/A converter output is at the same level as the  $AV_{REF3}$  pin after reset release, the design should allow for  $AV_{REF3}$  pin level output after reset release.



## CHAPTER 16 ASYNCHRONOUS SERIAL INTERFACE/3-WIRE SERIAL I/O

The  $\mu$ PD784026 incorporates two serial interface channels for which asynchronous serial interface (UART) mode or 3-wire serial I/O (IOE) mode can be selected.

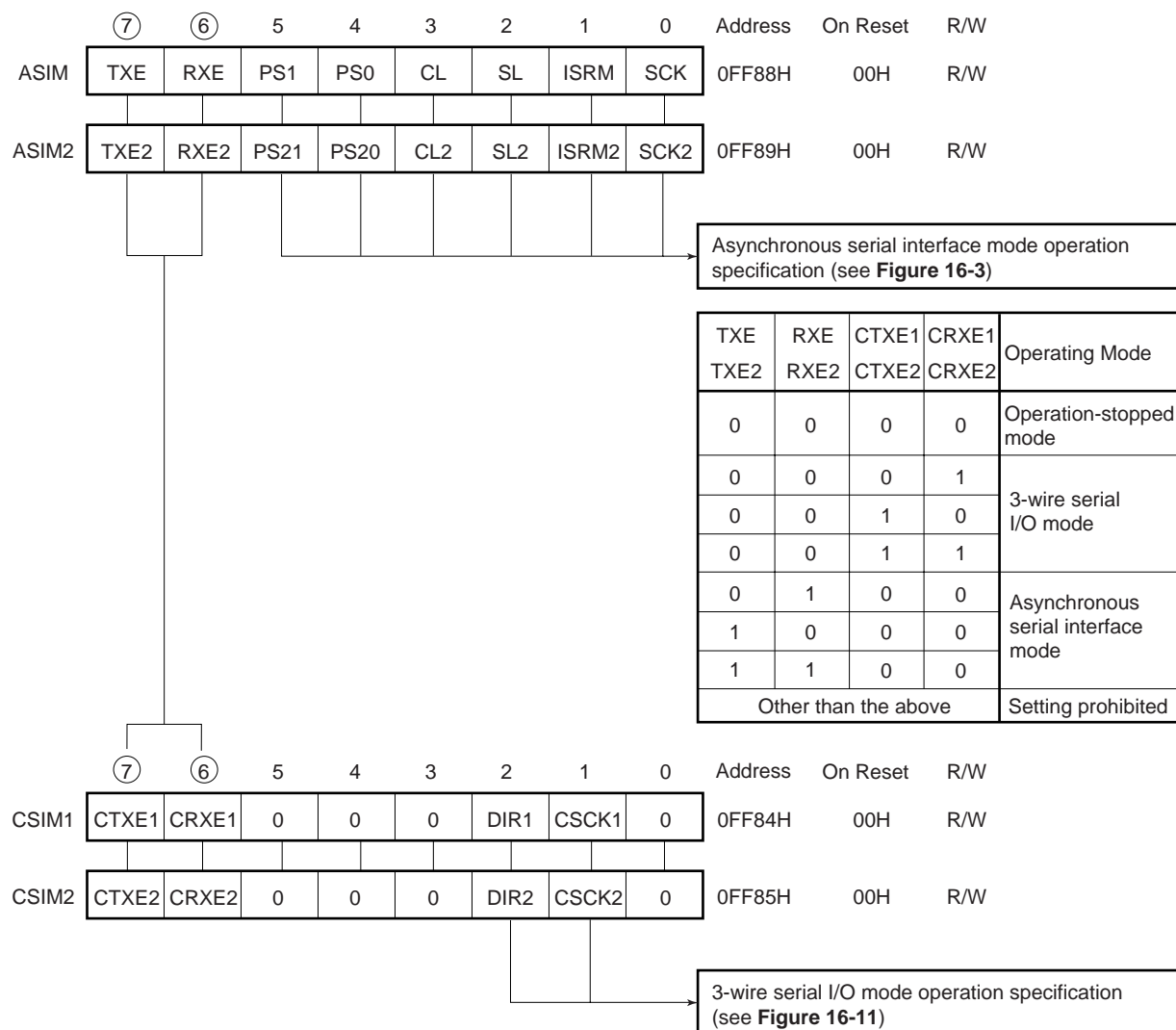
The two UART/IOE channels have completely identical functions. In this chapter, therefore, unless stated otherwise, UART/IOE1 will be described as representative of both UART/IOEs. When used as UART2/IOE2, the UART/IOE1 register names, bit names and pin names should be read as their UART2/IOE2 equivalents as shown in Table 16-1.

**Table 16-1 Differences Between UART/IOE1 and UART2/IOE2 Names**

Item	UART/IOE1	UART2/IOE2
Pin names	P25/ASCK/SCK1, P30/RxD/SI1, P31/TxD/SO2	P12/ASCK2/SCK2, P13/RxD2/SI2, P14/TxD2/SO2
Asynchronous serial interface mode register	ASIM	ASIM2
Asynchronous serial interface mode register bit names	TXE, RXE, PS1, PS0, CL, SL, ISRM, SCK	TXE2, RXE2, PS21, PS20, CL2, SL2, ISRM2, SCK2
Asynchronous serial interface status register	ASIS	ASIS2
Asynchronous serial interface status register bit names	PE, FE, OVE	PE2, FE2, OVE2
Clocked serial interface mode register	CSIM1	CSIM2
Clocked serial interface mode register bit names	CTXE1, CRXE1, DIR1, CSCK1	CTXE2, CRXE2, DIR2, CSCK2
Baud rate generator control register	BRGC	BRGC2
Baud rate generator control register bit names	TPS0-TPS3, MDL0-MDL3	TPS20-TPS23, MDL20-MDL23
Interrupt request names	INTSR/ITCSI1, INTSER, INTST	INTSR2/INTCSI2, INTSER2, INTST2
Interrupt control registers and bit names used in this chapter	SRIC, CSIIC1, SERIC, STIC, SRIF, CSIF1, SERIF, STIF	SRIC2, CSIIC2, SERIC2, STIC2, SRIF2, CSIF2, SERIF2, STIF2

**16.1 SWITCHING BETWEEN ASYNCHRONOUS SERIAL INTERFACE MODE AND 3-WIRE SERIAL I/O MODE**

The asynchronous serial interface mode and 3-wire serial I/O mode cannot be used simultaneously. Switching between these modes is performed in accordance with the settings of the asynchronous serial interface mode register (ASIM/ASIM2) and the clocked serial interface mode register (CSIM1/CSIM2) as shown in Figure 16-1.

**Figure 16-1 Switching Between Asynchronous Serial Interface Mode and 3-Wire Serial I/O Mode**

## 16.2 ASYNCHRONOUS SERIAL INTERFACE MODE

A UART (Universal Asynchronous Receiver Transmitter) mode is incorporated as the asynchronous serial interface. With this method, one byte of data is transmitted following a start bit, and full-duplex operation is possible.

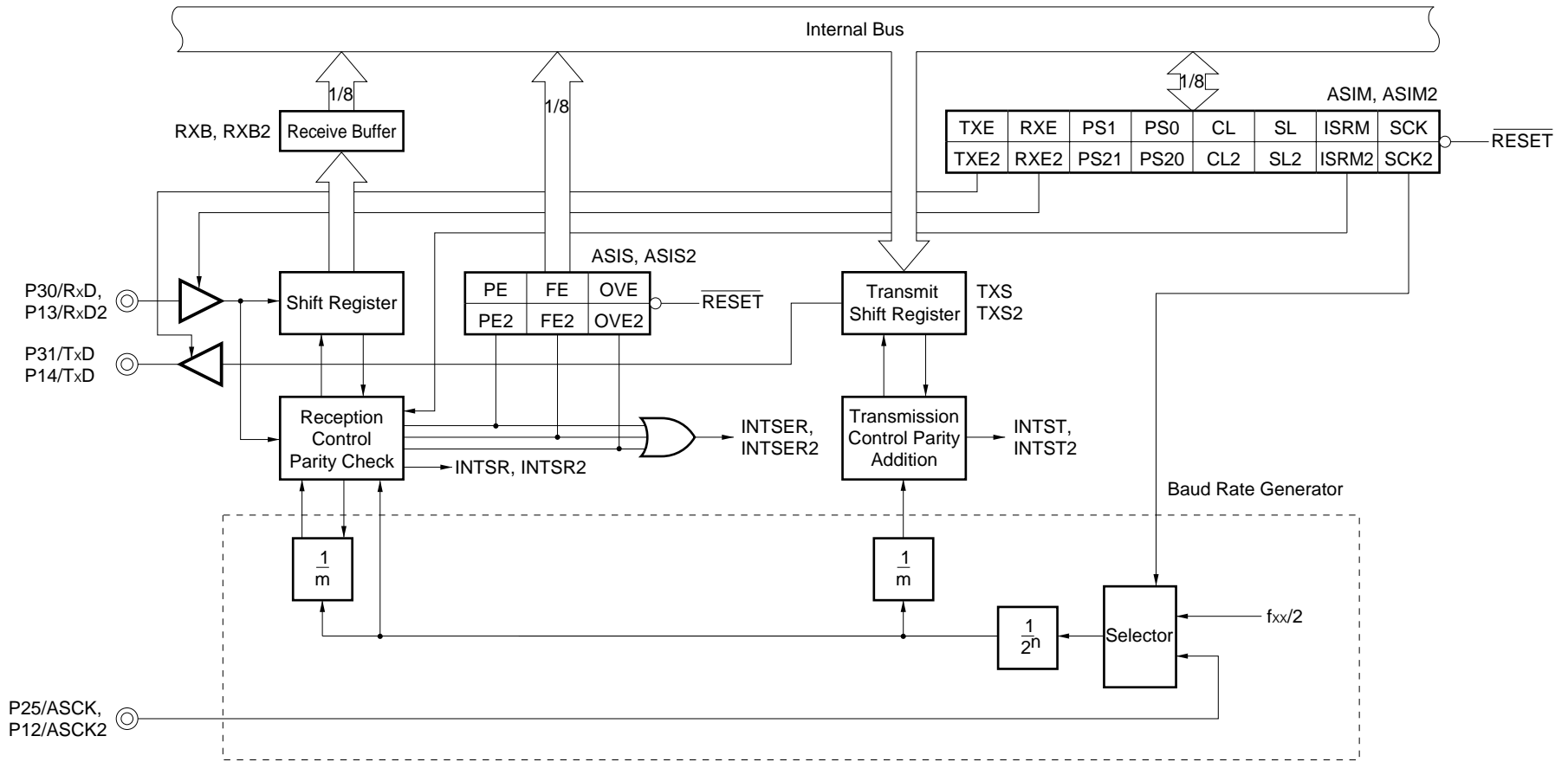
A baud rate generator is incorporated, enabling communication to be performed at any of a wide range of baud rates. Also, the baud rate can be defined by scaling the clock input to the ASCK pin.

### 16.2.1 Configuration in Asynchronous Serial Interface Mode

The block diagram of the asynchronous serial interface is described in Figure 16-2.

See **16.4 Baud Rate Generator** for details of the baud rate generator.

Figure 16-2 Asynchronous Serial Interface Block Diagram



**(1) Receive buffer (RXB/RXB2)**

This is the register that holds the receive data. Each time one byte of data is received, the receive data is transferred from the shift register.

If a 7-bit data length is specified, receive data is transferred to bits 0 to 6 of RXB/RXB2, and the MSB of RXB/RXB2 is always "0".

RXB/RXB2 can be read only by an 8-bit manipulation instruction. The contents of RXB/RXB2 are undefined after  $\overline{\text{RESET}}$  input.

**(2) Transmit shift register (TXS/TXS2)**

This is the register in which the data to be transmitted is set. Data written to the TXS/TXS2 is transmitted as serial data.

If a 7-bit data length is specified, bits 0 to 6 of the data written in the TXS/TXS2 are treated as transmit data. A transmit operation starts when a write to the TXS/TXS2 is performed. The TXS/TXS2 cannot be written to during a transmit operation.

TXS/TXS2 can be written to only by an 8-bit manipulation instruction. The contents of TXS/TXS2 are undefined after  $\overline{\text{RESET}}$  input.

**(3) Shift register**

This is the shift register that converts the serial data input to the RxD pin to parallel data. When one byte of data is received, the receive data is transferred to the receive buffer.

The shift register cannot be manipulated directly by the CPU.

**(4) Reception control parity check**

Receive operations are controlled in accordance with the contents set in the asynchronous serial interface mode register (ASIM/ASIM2). In addition, parity error and other error checks are performed during receive operations, and if an error is detected, a value is set in the asynchronous serial interface status register (ASIS/ASIS2) according to the type of error.

**(5) Transmission control parity addition**

Transmission operation is controlled by appending a start bit, parity bit, and stop bit to the data written to the transmit shift registers (TXS and TXS2) in accordance with the contents set to the asynchronous serial interface mode registers (ASIM and ASIM2).

**(6) Selector**

Selects the baud rate clock source.

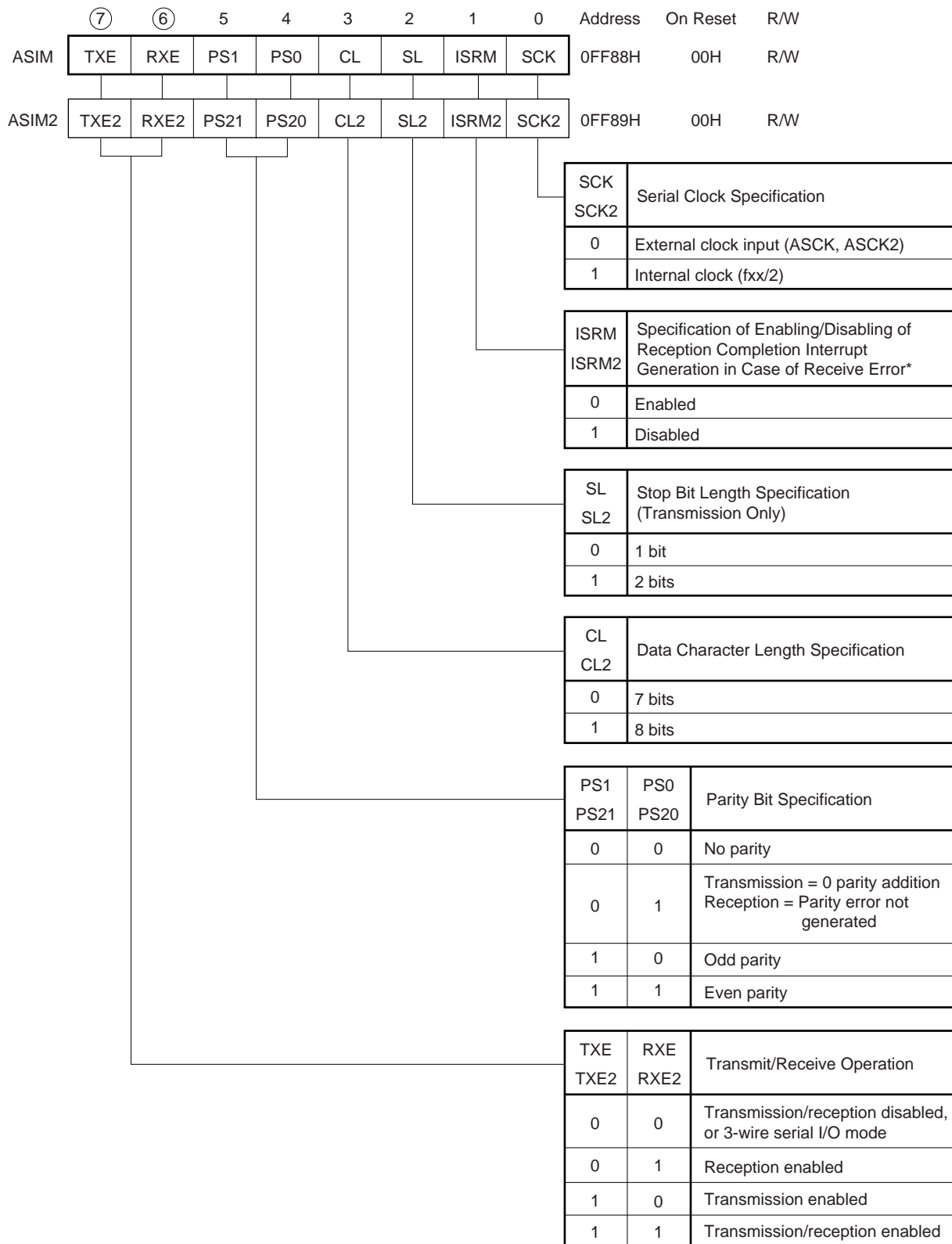
### 16.2.2 Asynchronous Serial Interface Control Registers

#### (1) Asynchronous serial interface mode register (ASIM), Asynchronous serial interface mode register 2 (ASIM2)

The ASIM and ASIM2 are 8-bit registers that specify the UART mode operation.

These registers can be read or written to by an 8-bit manipulation instruction or bit manipulation instruction. The format of ASIM and ASIM is shown in Figure 16-3.

These registers are cleared to 00H by  $\overline{\text{RESET}}$  input.

**Figure 16-3 Format of Asynchronous Serial Interface Mode Register (ASIM) and Asynchronous Serial Interface Mode Register 2 (ASIM2)**

- \* To disable the reception completion interrupt when a reception error occurs, make sure that wait time equivalent to two pulses of the clock that serves as the reference of the baud rate clock elapse after the reception error occurs until the receive buffers (RXB, RXB2) are read. If the wait time is not inserted, the reception completion interrupt occurs even when it is disabled.

The wait time equivalent to two pulses of the clock that serves as the reference of the baud rate clock can be calculated by the following expression:

$$\text{Wait time} = \frac{2^{n+3}}{f_{\text{xx}}}$$

**Remark**  $f_{\text{xx}}$ : Oscillation frequency

$n$  : Value of baud rate generator control registers (BRGC, BRGC2) to select tap of 12-bit prescaler ( $n = 0$  to 11)

**Caution** An asynchronous serial interface mode register (ASIM/ASIM2) rewrite should not be performed during a transmit operation. If an ASIM/ASIM2 register rewrite is performed during a transmit operation, subsequent transmit operations may not be possible (normal operation is restored by  $\overline{\text{RESET}}$  input). Software can determine whether transmission is in progress by using a transmission completion interrupt (INTST/INTST2) or the interrupt request flag (STIF/STIF2) set by INTST/INTST2.

## (2) Asynchronous serial interface status register (ASIS), Asynchronous serial interface status register 2 (ASIS2)

The ASIS and ASIS2 contain flags that indicate the error contents when a receive error occurs. Flags are set (1) when a receive error occurs, and cleared (0) when data is read from the receive buffer (RXB/RXB2). If the next data is received before RXB/RXB2 is read, the overrun error flag (OVE/OVE2) is set (1), and the other error flags are cleared (0) (if there is an error in the next data, the corresponding error flag is set (1)).

These registers can be read only by an 8-bit manipulation instruction or bit manipulation instruction. The format of ASIS and ASIS2 is shown in Figure 16-4.

These registers are cleared to 00H by  $\overline{\text{RESET}}$  input.



**Figure 16-4 Format of Asynchronous Serial Interface Status Register (ASIS) and Asynchronous Serial Interface Status Register 2 (ASIS2)**



- Cautions**
1. The receive buffer (RXB/RXB2) must be read even if there is a receive error.  
If RXB/RXB2 is not read, an overrun error will occur when the next data is received, and the receive error state will continue indefinitely.
  2. To disable the reception completion interrupt when a reception error occurs, make sure that wait time equivalent to two pulses of the clock that serves as the reference of the baud rate clock elapse after the reception error occurs until the receive buffers (RXB, RXB2) are read. If the wait time is not inserted, the reception completion interrupt occurs even when it is disabled.  
The wait time equivalent to two pulses of the clock that serves as the reference of the baud rate clock can be calculated by the following expression:

$$\text{Wait time} = \frac{2^{n+3}}{f_{xx}}$$

**Remark**  $f_{xx}$ : Oscillation frequency

$n$  : Value of baud rate generator control registers (BRGC, BRGC2) to select tap of 12-bit prescaler ( $n = 0$  to 11)

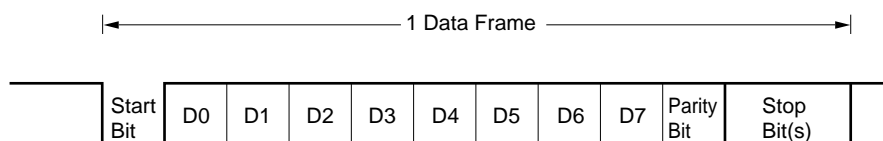
### 16.2.3 Data Format

Serial data transmission/reception is performed in full-duplex asynchronous mode.

The transmit/receive data format is shown in Figure 16-5. One data frame is made up of a start bit, character bits, parity bit, and stop bit(s).

Character bit length specification, parity selection and stop bit length specification for one data frame are performed by means of the asynchronous serial interface mode register (ASIM).

**Figure 16-5 Asynchronous Serial Interface Transmit/Receive Data Format**



- Start bit..... 1 bit
- Character bits ..... 7 bits/8 bits
- Parity bit ..... Even parity/odd parity/0 parity/no parity
- Stop bit s ..... 1 bit/2 bits

The serial transfer rate is selected in accordance with the asynchronous serial interface mode register and baud rate generator settings.

If a serial data receive error occurs, the nature of the receive error can be determined by reading the asynchronous serial interface status register (ASIS) status.

#### 16.2.4 Parity Types and Operations

The parity bit is used to detect a bit error in the communication data. Normally, the same kind of parity bit is used on the transmission side and the reception side. With even parity and odd parity, 1 bit (odd number) errors can be detected. With 0 parity and no parity, errors cannot be detected.

- Even parity

If the number of bits with a value of “1” in the transmit data is odd, the parity bit is set to “1”, and if the number of “1” bits is even, the parity bit is set to “0”. Control is thus performed to make the number of “1” bits in the transmit data plus the parity bit an even number. In reception, the number of “1” bits in the receive data plus the parity bit is counted, and if this number is odd, a parity error is generated.

- Odd parity

Conversely to the case of even parity, control is performed to make the number of “1” bits in the transmit data plus the parity bit an odd number.

In reception, a parity error is generated if the number of “1” bits in the receive data plus the parity bit is even.

- 0 parity

In transmission, the parity bit is set to “0” irrespective of the receive data.

In reception, parity bit detection is not performed. Therefore, no parity error is generated irrespective of whether the parity bit is “0” or “1”.

- No parity

In transmission, a parity bit is not added.

In reception, reception is performed on the assumption that there is no parity bit. Since there is no parity bit, no parity error is generated.

### 16.2.5 Transmission

The  $\mu$ PD784026's asynchronous serial interface is set to the transmission enabled state when the TXE bit of the asynchronous serial interface mode register (ASIM) is set (1). A transmit operation is started by writing transmit data to the transmit shift register (TXS) when transmission is enabled. The start bit, parity bit and stop bit(s) are added automatically.

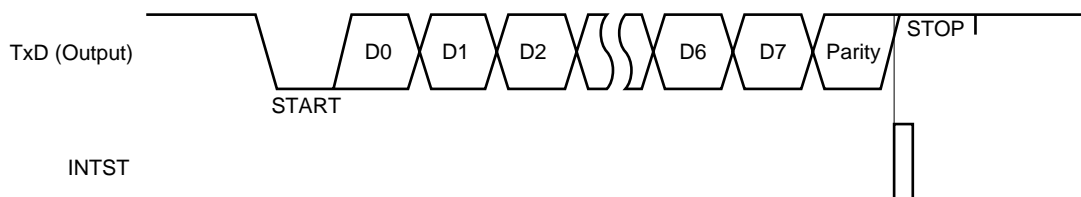
When a transmit operation is started, the data in the TXS is shifted out, and a transmission completion interrupt (INTST) is generated when the TXS is empty.

If no more data is written to the TXS, the transmit operation is discontinued.

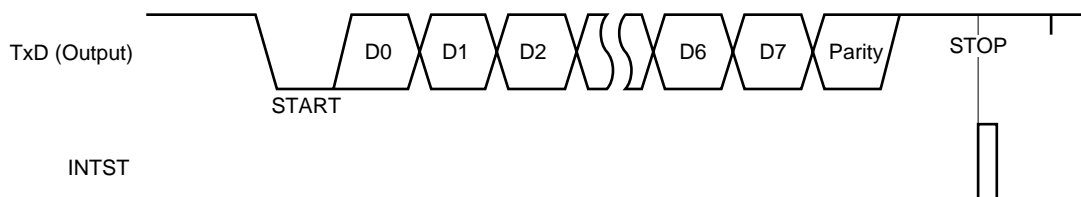
If the TXE bit is cleared (0) during a transmit operation, the transmit operation is discontinued immediately.

**Figure 16-6 Asynchronous Serial Interface Transmission Completion Interrupt Timing**

**(a) Stop bit length: 1**



**(b) Stop bit length: 2**



- Cautions**
1. After  $\overline{\text{RESET}}$  input the transmit shift register (TXS) is emptied but a transmission completion interrupt is not generated. A transmit operation can be started by writing transmit data to the TXS.
  2. An asynchronous serial interface mode register (ASIM) rewrite should not be performed during a transmit operation. If an ASIM rewrite is performed during a transmit operation, subsequent transmit operations may not be possible (normal operation is restored by  $\overline{\text{RESET}}$  input). Software can determine whether transmission is in progress by using a transmission completion interrupt (INTST) or the interrupt request flag (STIF) set by INTST.

### 16.2.6 Reception

When the RXE bit of the asynchronous serial interface mode register (ASIM) is set (1), receive operations are enabled and sampling of the RxD input pin is performed.

RxD input pin sampling is performed using the serial clock (divide-by-m counter input clock) specified by ASIM and baud rate generator control register (BRGC).

When the RxD pin input is driven low, the divide-by-m counter starts counting and a data sampling start timing signal is output on the m'th count. If the RxD pin input is low when sampled again by this start timing signal, the input is recognized as a start bit, the divide-by-m counter is initialized and the count is started, and data sampling is performed. When the character data, parity bit and stop bit are detected following the start bit, reception of one data frame ends.

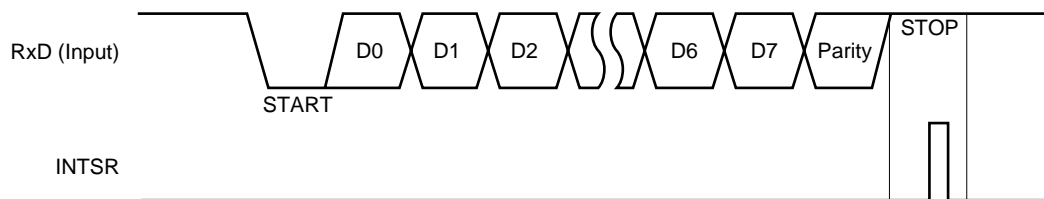
When reception of one data frame ends, the receive data in the shift register is transferred to the receive buffer, RXB, and a reception completion interrupt (INTSR) is generated.

If an error occurs, the receive data in which the error occurred is still transferred to RXB. If bit 1 (ISRM) of the ASIM was cleared (0) when the error occurred,

INTSR is generated. If the ISRM was set (1), INTSR is not generated.

If the RXE bit is cleared (0) during a receive operation, the receive operation is stopped immediately. In this case the contents of RXB and ASIS are not changed, and no INTSR or INTSER interrupt is generated.

**Figure 16-7 Asynchronous Serial Interface Reception Completion Interrupt Timing**



- Cautions**
1. The receive buffer (RXB) must be read even if there is a receive error. If RXB is not read, an overrun error will occur when the next data is received, and the receive error state will continue indefinitely.
  2. To disable the reception completion interrupt when a reception error occurs, make sure that wait time equivalent to two pulses of the clock that serves as the reference of the baud rate clock elapse after the reception error occurs until the receive buffers (RXB, RXB2) are read. If the wait time is not inserted, the reception completion interrupt occurs even when it is disabled. The wait time equivalent to two pulses of the clock that serves as the reference of the baud rate clock can be calculated by the following expression:

$$\text{Wait time} = \frac{2^{n+3}}{f_{xx}}$$

**Remark**  $f_{xx}$ : Oscillation frequency

$n$  : Value of baud rate generator control registers (BRGC, BRGC2) to select tap of 12-bit prescaler ( $n = 0$  to  $11$ )

### 16.2.7 Receive Errors

Three kinds of errors can occur in a receive operation: parity errors, framing errors and overrun errors. As the result of data reception, an error flag is raised in the asynchronous serial interface status register (ASIS) and a receive error interrupt (INTSER) is generated. Receive error causes are shown in Table 16-2.

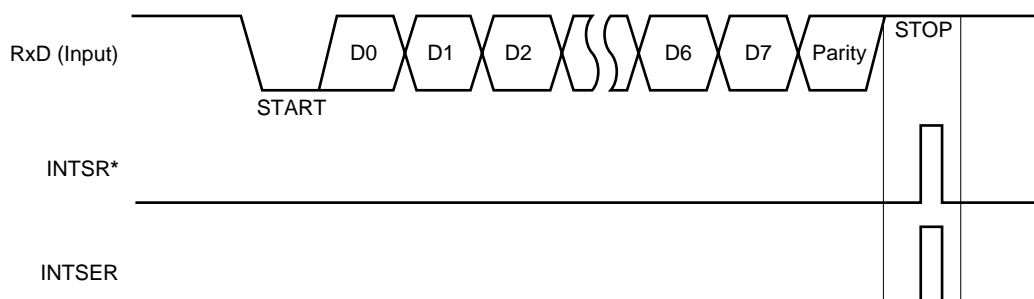
It is possible to detect the occurrence of any of the above errors during reception by reading the contents of the ASIS (see **Figures 16-4** and **16-8**).

The contents of the ASIS register are cleared (0) by reading the receive buffer (RXB) or by reception of the next data (if there is an error in the next data, the corresponding error flag is set).

**Table 16-2 Receive Error Causes**

Receive Error	Cause
Parity error	Transmit data parity specification and receive data parity do not match
Framing error	Stop bit not detected
Overrun error	Reception of next data completed before data is read from receive buffer

**Figure 16-8 Receive Error Timing**



\* If a receive error occurs while the ISRM bit is set (1), INTSR is not generated.

**Remark** In the  $\mu$ PD784026, a break signal cannot be detected by hardware. As a break signal is a low-level signal of two characters or more, a break signal may be judged to have been input if software detects the occurrence of two consecutive framing errors in which the receive data was 00H. The chance occurrence of two consecutive framing errors can be distinguished from a break signal by having the RxD pin level read by software (confirmation is possible by setting "1" in bit 0 of the port 3 mode register (PM3) and reading port 3 (P3)) and confirming that it is "0".

- Cautions**
1. The contents of the asynchronous serial interface status register (ASIS) are cleared (0) by reading the receive buffer (RXB) or by reception of the next data. If you want to find the details of an error, therefore, ASIS must be read before reading RXB.
  2. The RXB must be read even if there is a receive error. If RXB is not read, an overrun error will occur when the next data is received, and the receive error state will continue indefinitely.
  3. To disable the reception completion interrupt when a reception error occurs, make sure that wait time equivalent to two pulses of the clock that serves as the reference of the baud rate clock elapse after the reception error occurs until the receive buffers (RXB, RXB2) are read. If the wait time is not inserted, the reception completion interrupt occurs even when it is disabled.
- The wait time equivalent to two pulses of the clock that serves as the reference of the baud rate clock can be calculated by the following expression:

$$\text{Wait time} = \frac{2^{n+3}}{f_{xx}}$$

**Remark**  $f_{xx}$ : Oscillation frequency

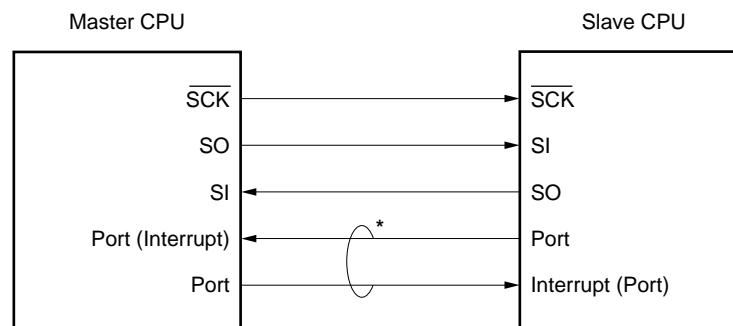
$n$  : Value of baud rate generator control registers (BRGC, BRGC2) to select tap of 12-bit prescaler ( $n = 0$  to 11)

### 16.3 3-WIRE SERIAL I/O MODE

The 3-wire serial I/O mode is used to communicate with devices that incorporate a conventional clocked serial interface. Basically, communication is performed using three lines: the serial clock ( $\overline{\text{SCK}}$ ), serial data output (SO), and serial data input (SI). Handshaking lines are required when a number of devices are connected.

**Figure 16-9 Example of 3-Wire Serial I/O System Configuration**

**3-wire serial I/O ↔ 3-wire serial I/O**

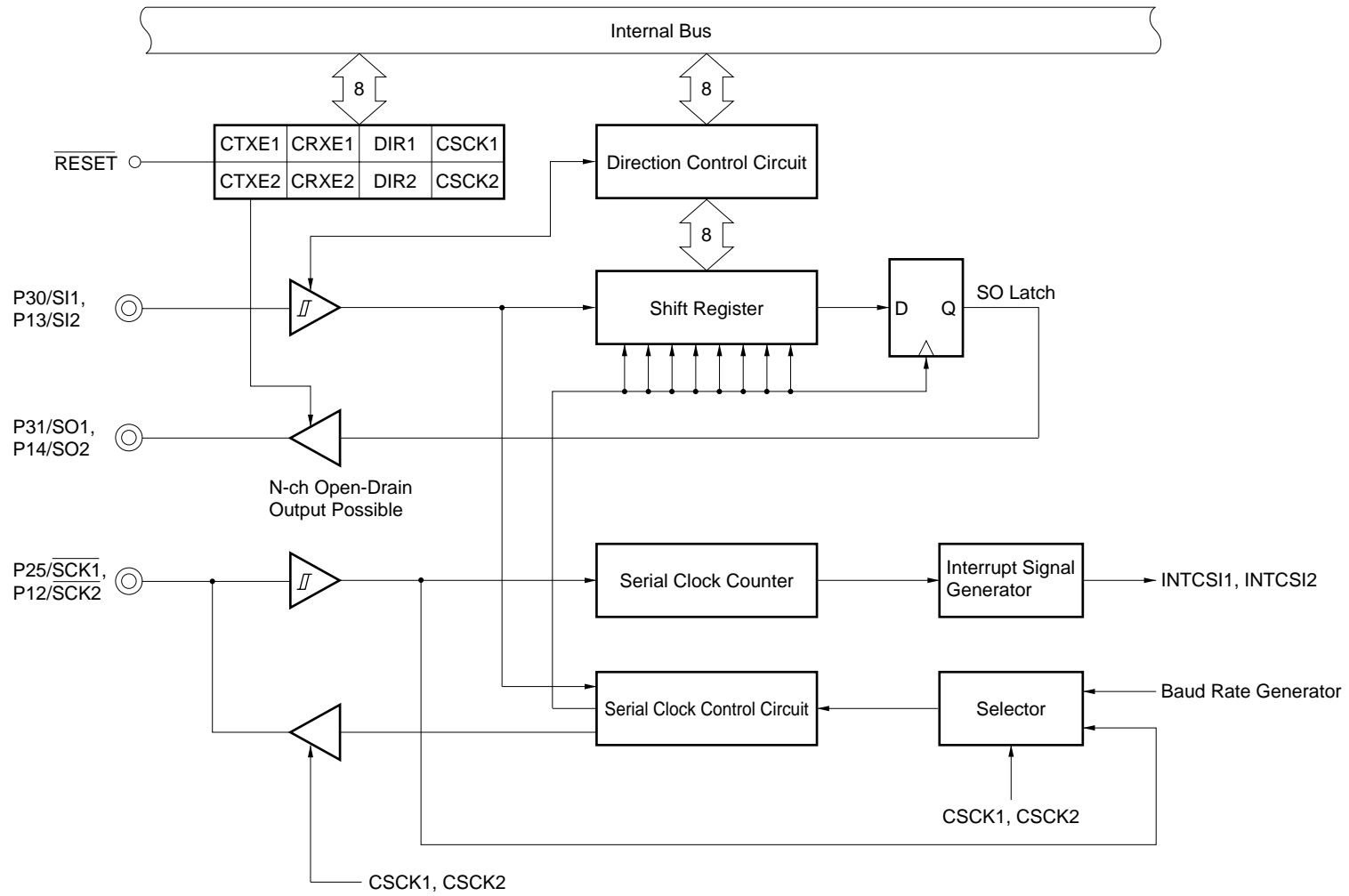


\* Handshaking lines

#### 16.3.1 Configuration in 3-Wire Serial I/O Mode

The block diagram in the 3-wire serial I/O mode is shown in Figure 16-10.

Figure 16-10 3-Wire Serial I/O Mode Block Diagram





**(1) Shift register (SIO1/SIO2)**

The SIO1 and SIO2 converts 8-bit serial data to 8-bit parallel data, and vice versa. SIO1/SIO2 is used for both transmission and reception.

Actual transmit/receive operations are controlled by writing to/reading from SIO1/SIO2.

Reading/writing can be performed by 8-bit manipulation instruction.

The contents of SIO1/SIO2 are undefined after  $\overline{\text{RESET}}$  input.

**(2) SO latch**

The SO latch holds the SO1/SO2 pin output level.

**(3) Serial clock selector**

Selects the serial clock to be used.

**(4) Serial clock counter**

Counts the serial clocks output or input in a transmit/receive operation, and checks that 8-bit data transmission/reception has been performed.

**(5) Interrupt signal generator**

Generates an interrupt request when 8 serial clocks have been counted by the serial clock counter.

**(6) Serial clock control circuit**

Controls the supply of the serial clock to the shift register, and also controls the clock output to the  $\overline{\text{SCK1}}$ / $\overline{\text{SCK2}}$  pins when the internal clock is used.

**(7) Direction control circuit**

Switches between MSB-first and LSB-first modes.

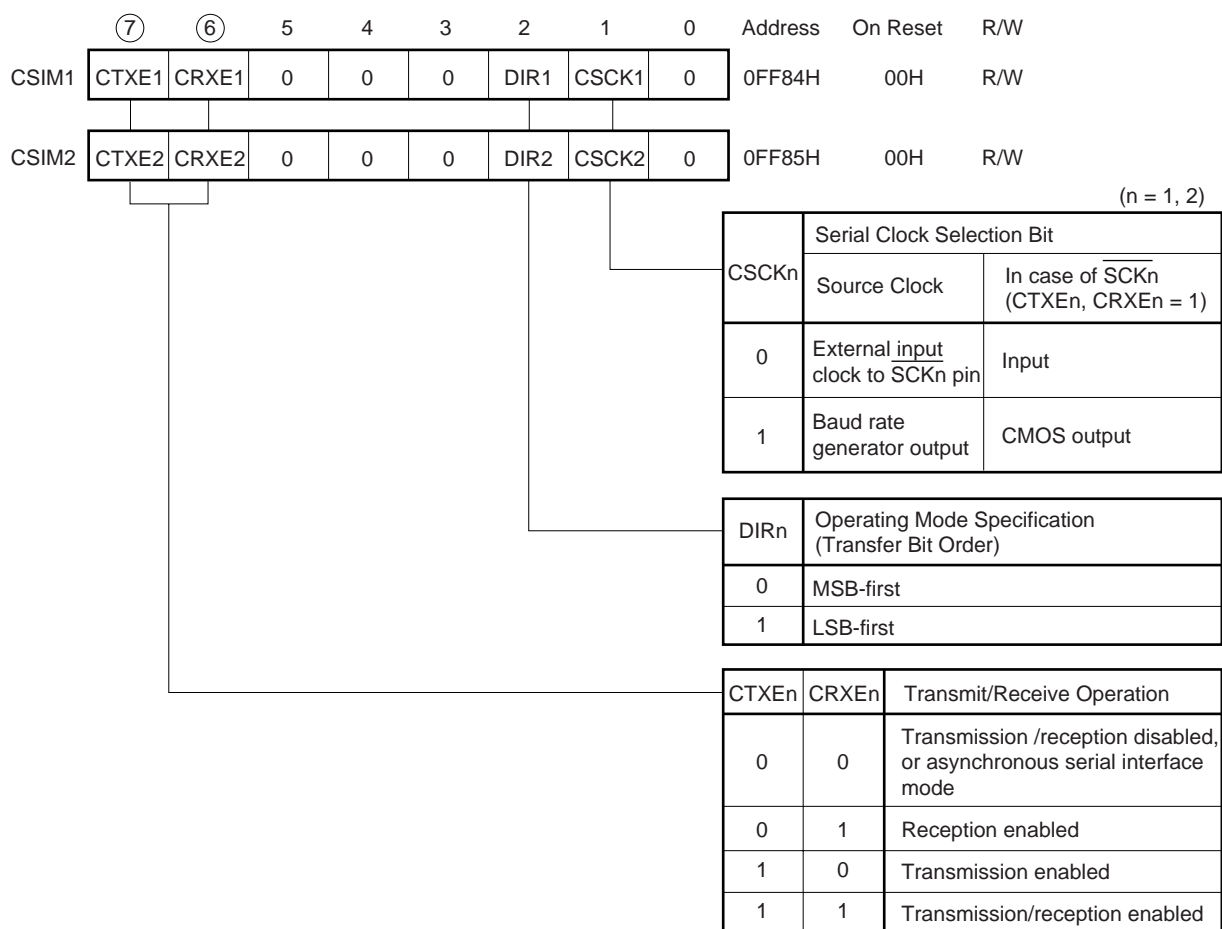
**16.3.2 Clocked Serial Interface Mode Registers (CSIM1, CSIM2)**

The CSIM1 and CSIM2 are 8-bit registers that specify operations in the 3-wire serial I/O mode.

These registers can be read or written to by an 8-bit manipulation instruction or bit manipulation instruction. The CSIM1 and CSIM2 format is shown in Figure 16-11.

These registers are cleared to 00H by  $\overline{\text{RESET}}$  input.

**Figure 16-11 Format of Clocked Serial Interface Mode Register 1 (CSIM1) and Clocked Serial Interface Mode Register 2 (CSIM2)**



- ★ **Caution** Select whether data is transferred from the MSB first or LSB first before writing data to SIO. If this selection is made after data has been written to SIO, the transfer sequence of the data already stored in SIO is not changed.

### 16.3.3 Basic Operation Timing

In the 3-wire serial I/O mode, data transmission/reception is performed in 8-bit units. Data is transmitted/received bit by bit in MSB-first or LSB-first order in synchronization with the serial clock.

MSB/LSB switching is specified by the DIR1 bit of the clock serial interface mode register (CSIM1).

Transmit data is output in synchronization with the fall of  $\overline{\text{SCK1}}$ , and receive data is sampled on the rise of  $\overline{\text{SCK1}}$ .

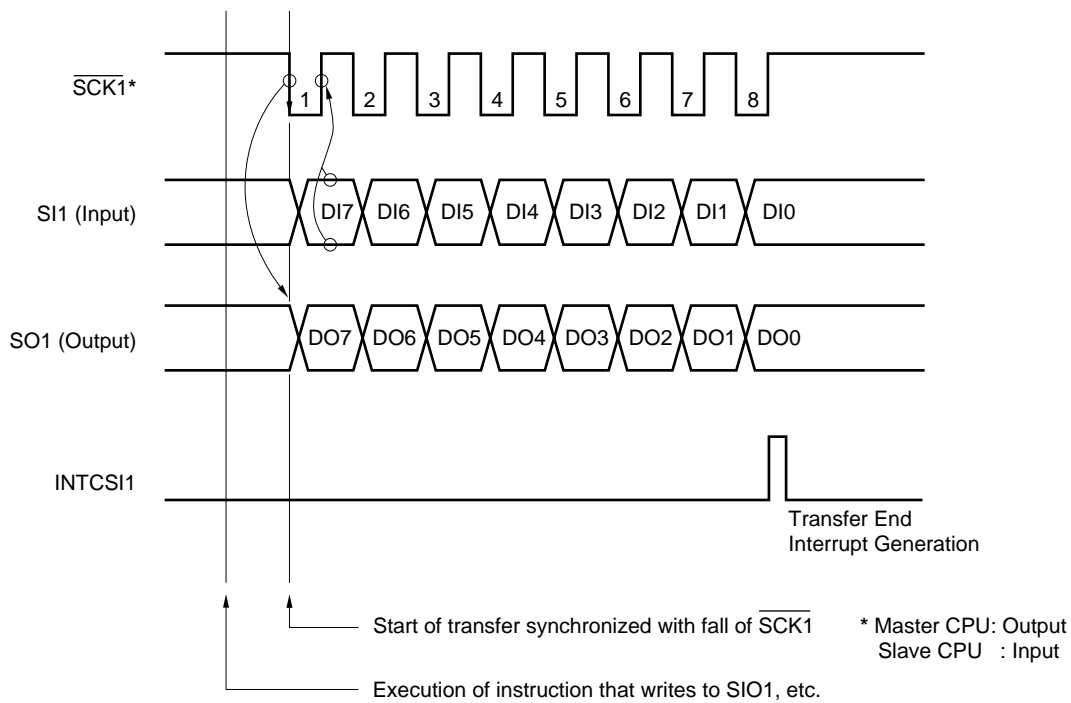
An interrupt request (INTCSI1) is generated on the 8th rise of  $\overline{\text{SCK1}}$ .

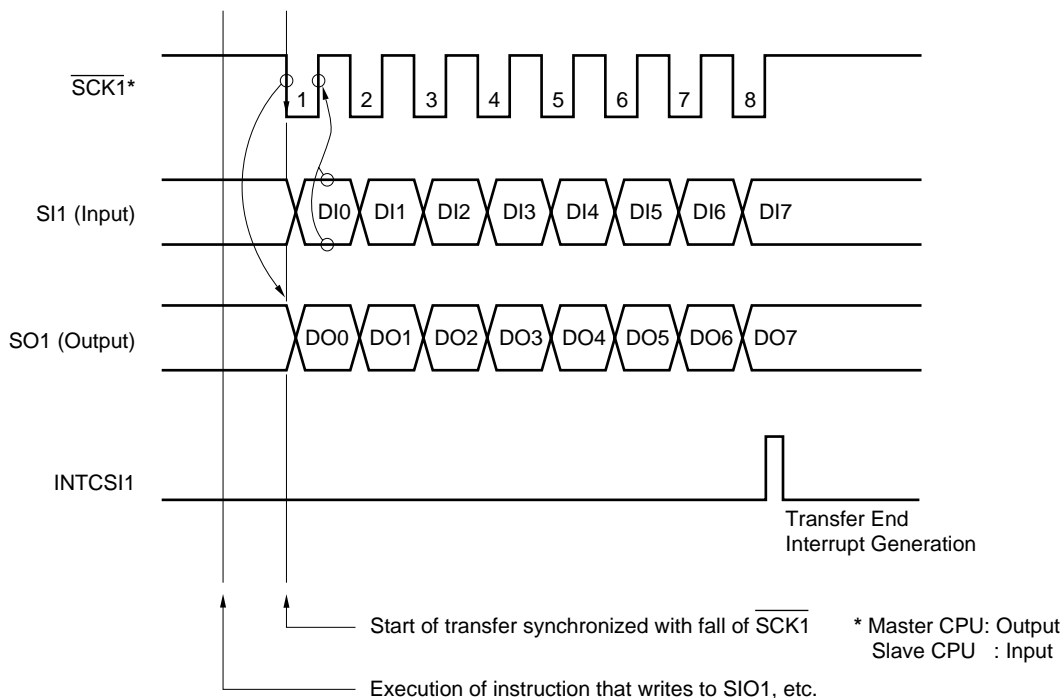
When the internal clock is used as  $\overline{\text{SCK1}}$ ,  $\overline{\text{SCK1}}$  output is stopped on the 8th rise of  $\overline{\text{SCK1}}$  and  $\overline{\text{SCK1}}$  remains high until the next data transmit or receive operation is started.

3-wire serial I/O mode timing is shown in Figure 16-12.

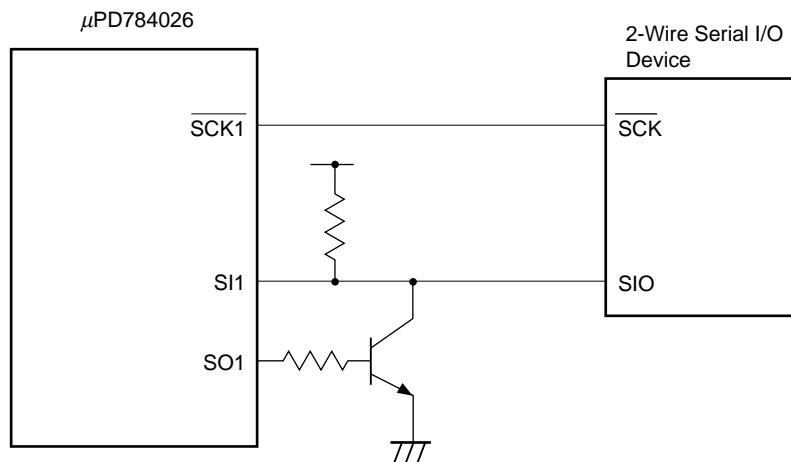
**Figure 16-12 3-Wire Serial I/O Mode Timing (1/2)**

**(a) MSB-first**



**Figure 16-12 3-Wire Serial I/O Mode Timing (2/2)****(b) LSB-first**

**Remark** If the  $\mu\text{PD784026}$  is connected to a 2-wire serial I/O device, a buffer should be connected to the SO1 pin as shown in Figure 16-13. In the example shown in Figure 12-13, the output level is inverted by the buffer, and therefore the inverse of the data to be output should be written to SIO1. In addition, non-connection of the internal pull-up resistor should be specified for the P33/SO1 pin.

**Figure 16-13 Example of Connection to 2-Wire Serial I/O**

#### 16.3.4 Operation When Transmission Only is Enabled

A transmit operation is performed when the CTXE1 bit of clocked serial interface mode register (CSIM1) is set (1). The transmit operation starts when a write to the shift register (SIO1) is performed while the CTXE1 bit is set (1).

When the CTXE1 bit is cleared (0), the SO1 pin is in the output high level.

##### (1) When the internal clock is selected as the serial clock

When transmission starts, the serial clock is output from the  $\overline{\text{SCK1}}$  pin and data is output in sequence from SIO1 to the SO1 pin in synchronization with the fall of the serial clock, and SI1 pin signals are shifted into SIO1 in synchronization with the rise of the serial clock.

There is a delay of up to one  $\overline{\text{SCK1}}$  clock cycle between the start of transmission and the first fall of  $\overline{\text{SCK1}}$ .

If transmission is disabled during the transmit operation (by clearing (0) the CTXE1 bit),  $\overline{\text{SCK1}}$  clock output is stopped and the transmit operation is discontinued on the next rise of  $\overline{\text{SCK1}}$ . In this case an interrupt request (INTCSI1) is not generated, and the SO1 pin becomes output high level.

##### (2) When an external clock is selected as the serial clock

When transmission starts, data is output in sequence from SIO1 to the SO1 pin in synchronization with the fall of the serial clock input to the  $\overline{\text{SCK1}}$  pin after the start of transmission, and SI1 pin signals are shifted into SIO1 in synchronization with the rise of the  $\overline{\text{SCK1}}$  pin input. If transmission has not started, shift operations are not performed and the SO1 pin output level does not change even if the serial clock is input to the  $\overline{\text{SCK1}}$  pin.

If transmission is disabled during the transmit operation (by clearing (0) the CTXE1 bit), the transmit operation is discontinued and subsequent  $\overline{\text{SCK1}}$  input is ignored. In this case an interrupt request (INTCSI1) is not generated, and the SO1 pin becomes output high level.

### 16.3.5 Operation When Reception Only is Enabled

A receive operation is performed when the CRXE1 bit of the clocked serial interface mode register (CSIM1) is set (1). The receive operation starts when the CRXE1 changes from “0” to “1”, or when a read from shift register (SIO1) is performed.

#### (1) When the internal clock is selected as the serial clock

When reception starts, the serial clock is output from the  $\overline{\text{SCK1}}$  pin and the S11 pin data is fetched in sequence into shift register (SIO1) in synchronization with the rise of the serial clock.

There is a delay of up to one  $\overline{\text{SCK1}}$  clock cycle between the start of reception and the first fall of  $\overline{\text{SCK1}}$ .

If reception is disabled during the receive operation (by clearing (0) the CRXE1 bit),  $\overline{\text{SCK1}}$  clock output is stopped and the receive operation is discontinued on the next rise of  $\overline{\text{SCK1}}$ . In this case an interrupt request (INTCSI1) is not generated, and the contents of the SIO1 are undefined.

#### (2) When an external clock is selected as the serial clock

When reception starts, the S11 pin data is fetched into shift register (SIO1) in synchronization with the rise of the serial clock input to the  $\overline{\text{SCK1}}$  pin after the start of reception. If reception has not started, shift operations are not performed even if the serial clock is input to the  $\overline{\text{SCK1}}$  pin.

If reception is disabled during the receive operation (by clearing (0) the CRXE1 bit), the receive operation is discontinued and subsequent  $\overline{\text{SCK1}}$  input is ignored. In this case an interrupt request (INTCSI1) is not generated.

**16.3.6 Operation When Transmission/Reception is Enabled**

When the CTXE1 bit and CRXE1 bit of the clocked serial interface mode register (CSIM1) register are both set (1), a transmit operation and receive operation can be performed simultaneously (transmit/receive operation). The transmit/receive operation is started when the CRXE1 bit is changed from "0" to "1", or by performing a write to shift register (SIO1).

When a transmit/receive operation is started for the first time, the CRXE1 bit always changes from "0" to "1", and there is thus a possibility that the transmit/receive operation will start immediately, and undefined data will be output. The first transmit data should therefore be written to SIO1 beforehand when both transmission and reception are disabled (when the CTXE1 bit and CRXE1 bit are both cleared (0)), before enabling transmission/reception.

When transmission/reception is disabled (CTXE1 = CRXE1 = 0), the SO1 pin is in the output high level.

**(1) When the internal clock is selected as the serial clock**

When transmission/reception starts, the serial clock is output from the  $\overline{\text{SCK1}}$  pin, data is output in sequence from shift register (SIO1) to the (SO1) pin in synchronization with the fall of the serial clock, and SI1 pin data is shifted in order into SIO1 in synchronization with the rise of the serial clock.

There is a delay of up to one  $\overline{\text{SCK1}}$  clock cycle between the start of transmission and the first fall of  $\overline{\text{SCK1}}$ .

If either transmission or reception is disabled during the transmit/receive operation, only the disabled operation is discontinued. If transmission only is disabled, the SO1 pin becomes output high level. If reception only is disabled, the contents of the SIO1 will be undefined.

If transmission and reception are disabled simultaneously,  $\overline{\text{SCK1}}$  clock output is stopped and the transmit and receive operations are discontinued on the next rise of  $\overline{\text{SCK1}}$ . When transmission and reception are disabled simultaneously, the contents of SIO1 are undefined, an interrupt request (INTCSI1) is not generated, and the SO1 pin becomes output high level.

**(2) When an external clock is selected as the serial clock**

When transmission/reception starts, data is output in sequence from shift register (SIO1) to the SO1 pin in synchronization with the fall of the serial clock input to the  $\overline{\text{SCK1}}$  pin after the start of transmission/reception, and SI1 pin data is shifted in order into SIO1 in synchronization with the rise of the serial clock. If transmission/reception has not started, the SIO1 shift operations are not performed and the SO1 pin output level does not change even if the serial clock is input to the  $\overline{\text{SCK1}}$  pin.

If either transmission or reception is disabled during the transmit/receive operation, only the disabled operation is discontinued. If transmission only is disabled, the SO1 pin becomes output high level. If reception only is disabled, the contents of the SIO1 will be undefined.

If transmission and reception are disabled simultaneously, the transmit and receive operations are discontinued and subsequent  $\overline{\text{SCK1}}$  input is ignored. When transmission and reception are disabled simultaneously, the contents of SIO1 are undefined, an interrupt request (INTCSI1) is not generated, and the SO1 pin becomes output high level.

### 16.3.7 Corrective Action in Case of Slippage of Serial Clock and Shift Operations

When an external clock is selected as the serial clock, there may be slippage between the number of serial clocks and shift operations due to noise, etc. In this case, since the serial clock counter is initialized by disabling both transmit operations and receive operations (by clearing (0) the CTXE1 bit and CRXE1 bit), synchronization of the shift operations and the serial clock can be restored by using the first serial clock input after reception or transmission is next enabled as the first clock.

## 16.4 BAUD RATE GENERATOR

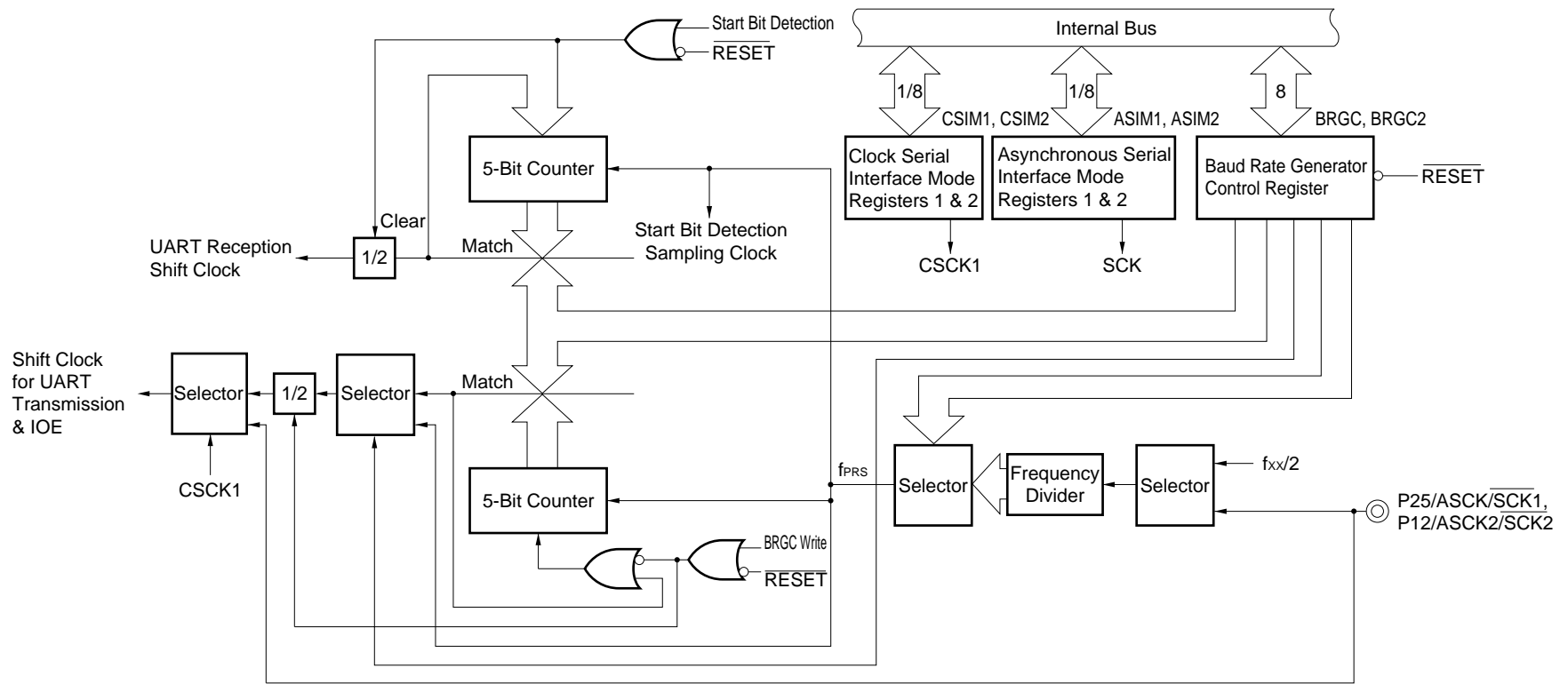
The baud rate generator is the circuit that generates the UART/IOE serial clock. Two independent circuits are incorporated, one for each serial interface.

### 16.4.1 Baud Rate Generator Configuration

The baud rate generator block diagram is shown in Figure 16-14.



Figure 16-14 Baud Rate Generator block Diagram



**(1) 5-bit counter**

Counter that counts the clock ( $f_{PRS}$ ) by which the output from the frequency divider is selected. Generates a signal with the frequency selected by the low-order 4 bits of the baud rate generator control registers (BRGC/BRGC2).

**(2) Frequency divider**

Scales the internal clock ( $f_{xx}/2$ ) or, in asynchronous serial interface mode, a clock that is twice the external baud rate input (ASCK/ASCK2), and selects  $f_{PRS}$  with the next-stage selector.

**(3) Both-edge detection circuit**

Detects both edges of the ASCK/ASCK2 pin input signal and generates a signal with a frequency twice that of the ASCK/ASCK2 input clock.

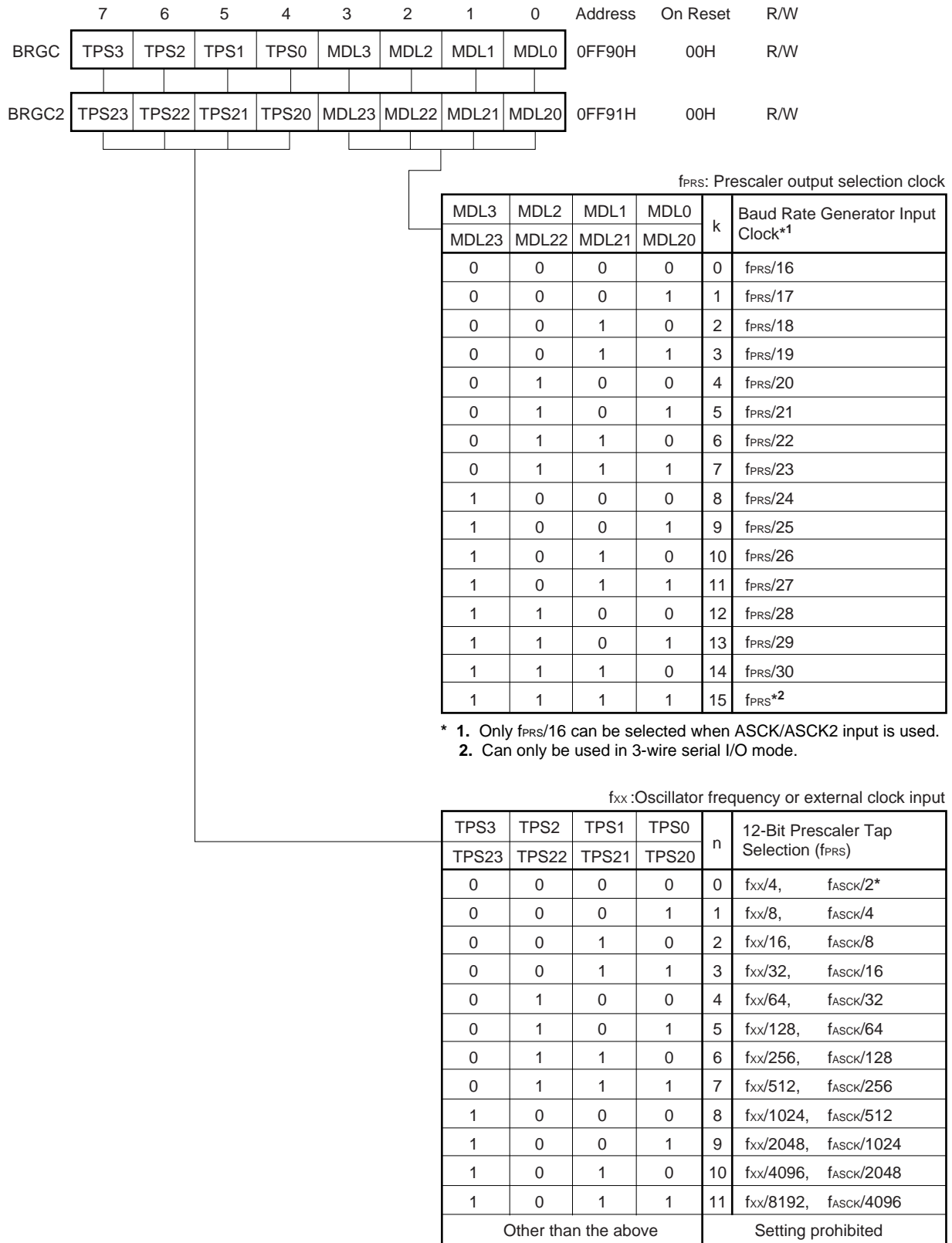
**16.4.2 Baud Rate Generator Control Register (BRGC, BRGC2)**

The BRGC and BRGC2 are 8-bit registers that set the baud rate clock in asynchronous serial interface mode or the shift clock in 3-wire serial I/O mode.

These registers can be written to only with an 8-bit manipulation instruction. The BRGC and BRGC2 format is shown in Figure 16-15.

$\overline{\text{RESET}}$  input clears the BRGC register to 00H.

**Caution** When a baud rate generator control register (BRGC, BRGC2) write instruction is executed, the 5-bit counter and 1/2 frequency divider operations are reset. Consequently, if a write to the BRGC and BRGC2 is performed during communication, the generated baud rate clock may be disrupted, preventing normal communication from continuing. The BRGC and BRGC2 should therefore not be written to during communication.

**Figure 16-15 Baud Rate Generator Control Register (BRGC) Format and  
Baud Rate Generator Control Register 2 (BRGC2) Format**

### 16.4.3 Baud Rate Generator Operation

The baud rate generator only operates when UART/IOE transmit/receive operations are enabled. The generated baud rate clock is a signal scaled from the internal clock ( $f_{xx}/2$ ) or a signal scaled from the clock input from the external baud rate input (ASCK) pin.

**Caution** If a write to the baud rate generator control register (BRGC) is performed during communication, the generated baud rate clock may be disrupted, preventing normal communication from continuing. The BRGC should therefore not be written to during communication.

#### (1) Baud rate clock generation in UART mode

##### (a) Using internal clock ( $f_{xx}/2$ )

This function is selected by setting (1) bit 0 (SCK) of the asynchronous serial interface mode register (ASIM). The internal clock ( $f_{xx}/2$ ) is scaled by the frequency divider, this signal ( $f_{PRS}$ ) is scaled by the 5-bit counter, and the signal further divided by 2 is used as the baud rate. The baud rate is given by the following expression:

$$(\text{Baud rate}) = \frac{f_{xx}}{(k + 16) \cdot 2^{n+3}}$$

$f_{xx}$  : Oscillator frequency or external clock input

$k$  : Value set in bits MDL3 to MDL0 of BRGC ( $k = 0$  to 14)

$n$  : Value set in bits TPS3 to TPS0 of BRGC ( $n = 0$  to 11)

##### (b) Using external baud rate input

This function is selected by clearing (0) bit 0 (SCK) of the asynchronous serial interface mode register (ASIM). When this function is used, bits MDL3 to MDL0 of the baud rate generator control register (BRGC) must all be cleared (0) ( $k = 0$ ).

When this function is used with UART2, it is necessary to set (1) bit 2 of the port 3 mode control register (PMC3) and set the P12 pin to control mode.

The ASCK pin input clock is scaled by the frequency divider, and the signal obtained by dividing this signal by 32 ( $f_{PRS}$ ) (division by 16 and division by 2) is used as the baud rate. The baud rate is given by the following expression:

$$(\text{Baud rate}) = \frac{f_{\text{ASCK}}}{2^{n+6}}$$

$f_{\text{ASCK}}$  : ASCK pin input clock frequency

$n$  : Value set in bits TPS3 to TPS0 of BRGC ( $n = 0$  to 11)

When this function is used, a number of baud rates can be generated by one external input clock.

## (2) Serial clock generation in 3-wire serial I/O mode

Selected when the CSM1 bit of the clocked serial interface mode register (CSIM1) is set (1) and SCK1 is output.

### (a) Normal mode

The internal clock ( $f_{\text{xx}}/2$ ) is scaled by the frequency divider, this signal ( $f_{\text{PRS}}$ ) is scaled by the 5-bit counter, and the signal further divided by 2 is used as the serial clock. The serial clock is given by the following expression:

$$(\text{Serial clock}) = \frac{f_{\text{xx}}}{(k+16) \cdot 2^{n+3}}$$

$f_{\text{xx}}$  : Oscillator frequency or external clock input

$k$  : Value set in bits MDL3 to MDL0 of BRGC ( $k = 0$  to 14)

$n$  : Value set in bits TPS3 to TPS0 of BRGC ( $n = 0$  to 11)

### (b) High-speed mode

When this function is used, bits MDL3 to MDL0 of the baud rate generator control register (BRGC) are all set (1) ( $k = 15$ ).

The internal clock ( $f_{\text{xx}}/2$ ) is scaled by the frequency divider, and this signal ( $f_{\text{PRS}}$ ) divided by 2 is used as the serial clock. The serial clock is given by the following expression:

$$(\text{Serial clock}) = \frac{f_{\text{xx}}}{2^{n+3}}$$

$f_{\text{xx}}$  : Oscillator frequency or external clock input

$n$  : Value set in bits TPS3 to TPS0 of BRGC ( $n = 1$  to 11)

**16.4.4 Baud Rate Setting in Asynchronous Serial Interface Mode**

There are two methods of setting the baud rate, as shown in Table 16-3.

This table shows the range of baud rates that can be generated, the baud rate calculation expression and selection method for each case.

**Table 16-3 Baud Rate Setting Methods**

Baud Rate Clock Source		Selection Method	Baud Rate Calculation Expression	Baud Rate Range
Baud rate generator	Internal system clock	SCK in ASIM = 1	$\frac{f_{xx}}{(k + 16) \cdot 2^{n+3}}$	$\frac{f_{xx}}{491520} - \frac{f_{xx}}{128}$
	ASCK input	SCK in ASIM = 0	$\frac{f_{ASCK}}{2^{n+6}}$	$\frac{f_{ASCK}}{131072} - \frac{f_{ASCK}^*}{64}$

$f_{xx}$  : Oscillator frequency or external clock input

$k$  : Value set in bits MDL3 to MDL0 of BRGC ( $k = 0$  to  $14$ ; see **Figure 16-15**)

$n$  : Value set in bits TPS3 to TPS0 of BRGC ( $n = 0$  to  $11$ ; see **Figure 16-15**)

$f_{ASCK}$  : ASCK input clock frequency ( $0 - f_{xx}/4$ )

\* Including  $f_{ASCK}$  input range: ( $0 - f_{xx}/256$ )

**(1) Examples of settings when baud rate generator is used**

Examples of baud rate generator control register (BRGC) settings when the baud rate generator is used are shown below.

When the baud rate generator is used, the SCK bit of the asynchronous serial interface mode register (ASIM) should be set (1).

**Table 16-4 Examples of BRGC Settings When Baud Rate Generator is Used**

Oscillator Frequency ( $f_{ox}$ ) or External Clock ( $f_x$ )	25.0 MHz		24.576 MHz		12.0 MHz		11.0592 MHz	
Baud Rate [bps]	BRGC Value	Baud Rate Error (%)	BRGC Value	Baud Rate Error (%)	BRGC Value	Baud Rate Error (%)	BRGC Value	Baud Rate Error (%)
75	B4H	1.73	B4H	0.00	A4H	2.34	A2H	0.00
110	ACH	0.92	ABH	1.01	9BH	1.36	99H	1.82
150	A4H	1.73	A4H	0.00	94H	2.34	92H	0.00
300	94H	1.73	94H	0.00	84H	2.34	82H	0.00
600	84H	1.73	84H	0.00	74H	2.34	72H	0.00
1200	74H	1.73	74H	0.00	64H	2.34	62H	0.00
2400	64H	1.73	64H	0.00	54H	2.34	52H	0.00
4800	54H	1.73	54H	0.00	44H	2.34	42H	0.00
9600	44H	1.73	44H	0.00	34H	2.34	32H	0.00
19200	34H	1.73	34H	0.00	24H	2.34	22H	0.00
31520	29H	0.00	29H	1.70	19H	0.00	16H	0.54
38400	24H	1.73	24H	0.00	14H	2.34	12H	0.00
76800	14H	1.73	14H	0.00	04H	2.34	02H	0.00

**(2) Examples of settings when external baud rate input (ASCK) is used**

Table 16-5 shows an example of setting when external baud rate input (ASCK) is used. When using the ASCK input, clear the SCK bit of the asynchronous serial interface mode register (ASIM) to 0, and set the corresponding pin in the control mode by using PMC3 or PMC1.

**Table 16-5 Examples of Settings When External Baud Rate Input (ASCK) is Used**

$f_{ASCK}$ (ASCK Input Frequency)	153.6 kHz	4.9152 MHz
Baud Rate [bps]	BRGC Value	BRGC Value
75	50H	A0H
150	40H	90H
300	30H	80H
600	20H	70H
1200	10H	60H
2400	00H	50H
4800	—	40H
9600	—	30H
19200	—	20H
38400	—	10H
76800	—	00H



**16.5 CAUTIONS**

- (1) An asynchronous serial interface mode register (ASIM) rewrite should not be performed during a transmit operation. If an ASIM rewrite is performed during a transmit operation, subsequent transmit operations may not be possible (normal operation is restored by  $\overline{\text{RESET}}$  input).  
Software can determine whether transmission is in progress by using a transmission completion interrupt (INTST) or the interrupt request flag (STIF) set by INTST.
- (2) After  $\overline{\text{RESET}}$  input the transmit shift register (TXS) is emptied but a transmission completion interrupt is not generated. A transmit operation can be started by writing transmit data to the TXS.
- (3) The receive buffer (RXB) must be read even if there is a receive error. If RXB is not read, an overrun error will occur when the next data is received, and the receive error state will continue indefinitely.
- (4) To disable the reception completion interrupt when a reception error occurs, make sure that wait time equivalent to two pulses of the clock that serves as the reference of the baud rate clock elapse after the reception error occurs until the receive buffers (RXB, RXB2) are read. If the wait time is not inserted, the reception completion interrupt occurs even when it is disabled.  
The wait time equivalent to two pulses of the clock that serves as the reference of the baud rate clock can be calculated by the following expression:

$$\text{Wait time} = \frac{2^{n+3}}{f_{xx}}$$

**Remark**  $f_{xx}$ : Oscillation frequency

$n$  : Value of baud rate generator control registers (BRGC, BRGC2) to select tap of 12-bit prescaler ( $n = 0$  to 11)

- (5) The contents of the asynchronous serial interface status register (ASIS) are cleared (0) by reading the receive buffer (RXB) or by reception of the next data. If you want to find the details of an error, therefore, ASIS must be read before reading RXB.
- (6) The baud rate generator control register (BRGC) should not be written to during communication. If a write instruction is executed, the 5-bit counter and 1/2 frequency divider operations will be reset, and the generated baud rate clock may be disrupted, preventing normal communication from continuing.
- (7) Select whether data is transferred from the MSB first or LSB first before writing data to SIO. If this selection is made after data has been written to SIO, the transfer sequence of the data already stored in SIO is not changed.

[MEMO]

## CHAPTER 17 CLOCKED SERIAL INTERFACE

### 17.1 FUNCTIONS

The configuration of the  $\mu$ PD784026's clocked serial interface is shown in Figure 17-1.

The  $\mu$ PD784026's clocked serial interface uses the following two operating modes:

**(1) 3-wire serial I/O mode (MSB/LSB-first)**

In this mode, 8-bit data transfers are performed using three lines: the serial clock ( $\overline{\text{SCK0}}$ ) and serial bus (SO0, SI0).

This mode is useful when connecting a peripheral

I/O device, display controller, etc., that incorporates a conventional clocked serial interface.

**(2) Serial bus interface (SBI) mode (MSB-first)**

In the SBI mode, communication is performed with multiple devices by means of two lines: the serial clock ( $\overline{\text{SCK0}}$ ) and the serial data bus (SB0).

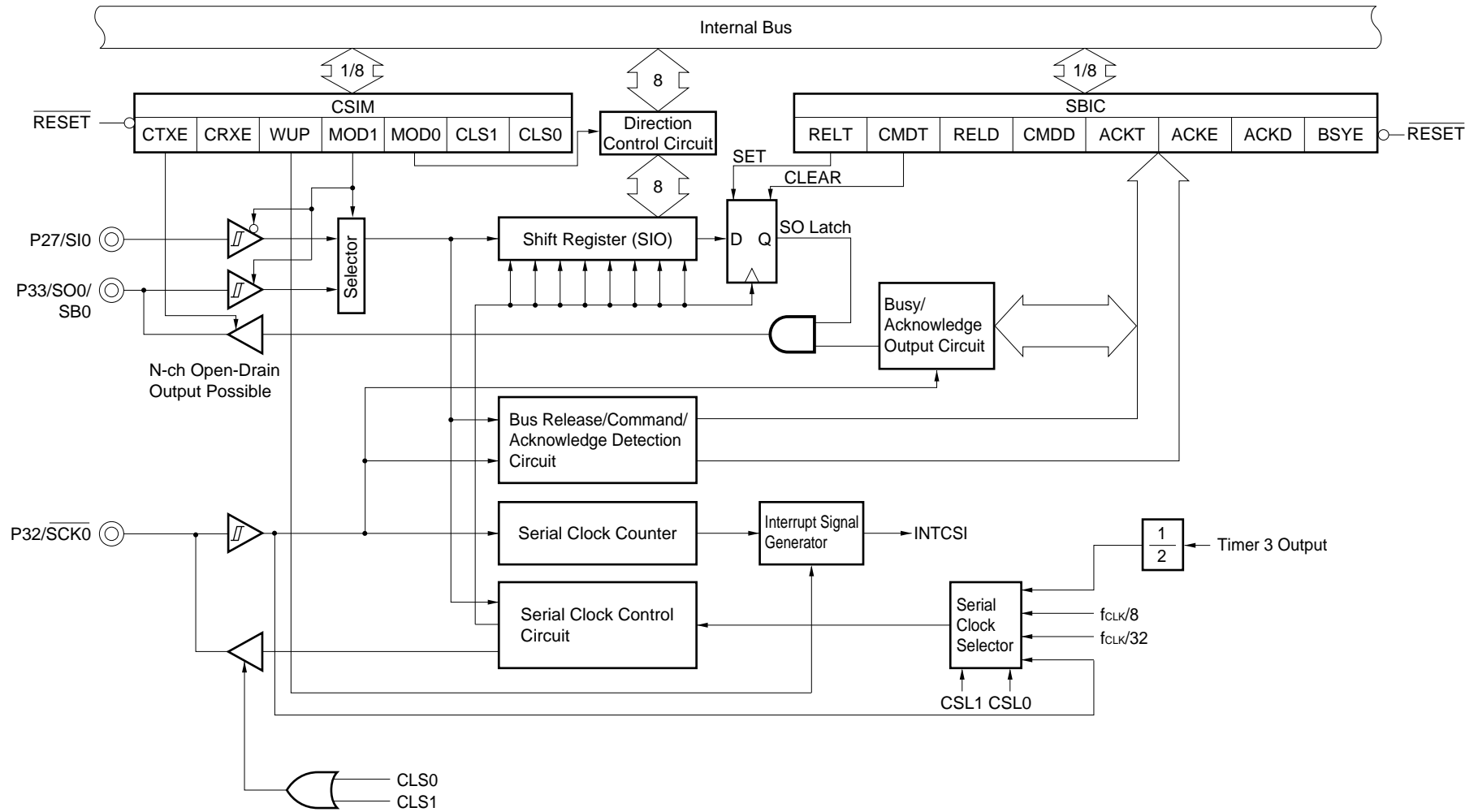
This mode conforms to the NEC serial bus format.

In the SBI mode, it is possible to output to the serial data bus an address to select the target device for serial communication, a command which gives a directive to the target device, and actual data. It is therefore possible to eliminate the handshaking lines required when connecting multiple devices on a conventional clocked serial interface, allowing efficient use of input/output ports.

### 17.2 CONFIGURATION

The block diagram of clocked serial interface (CSI) is shown in Figure 17-1.

Figure 17-1 Clocked Serial Interface (CSI) Block Diagram



**(1) Shift register (SIO)**

The SIO converts 8-bit serial data to 8-bit parallel data, and vice versa. SIO is used for both transmission and reception. Actual transmit/receive operations are controlled by writing to/reading from SIO.

SIO can be read or written to with an 8-bit manipulation instruction. The contents of SIO are undefined after  $\overline{\text{RESET}}$  input.

**(2) SO latch**

The SO latch holds the SO0/SB0 pin output level. This latch can also be directly controlled by software.

**(3) Serial clock selector**

Selects the serial clock to be used.

**(4) Serial clock counter**

Counts the serial clocks output or input in a transmit/receive operation, and checks that 8-bit data transmission/reception has been performed.

**(5) Interrupt signal generator**

Controls whether or not an interrupt request is generated when 8 serial clocks have been counted by the serial clock counter. An interrupt request is generated every 8 clocks in 3-wire serial I/O mode, and when the condition is met in SBI mode.

**(6) Serial clock control circuit**

Controls the supply of the serial clock to the shift register (SIO), and also controls the clock output to the  $\overline{\text{SCK0}}$  pin when the internal clock is used.

**(7) Busy/acknowledge output circuit, bus release/command/acknowledge detection circuit**

Performs output and detection of various control signals in serial bus interface (SBI) mode. Does not operate in 3-wire serial I/O mode.

**(8) Direction control circuit**

Controls the transmit/receive data shift direction.

## 17.3 CONTROL REGISTERS

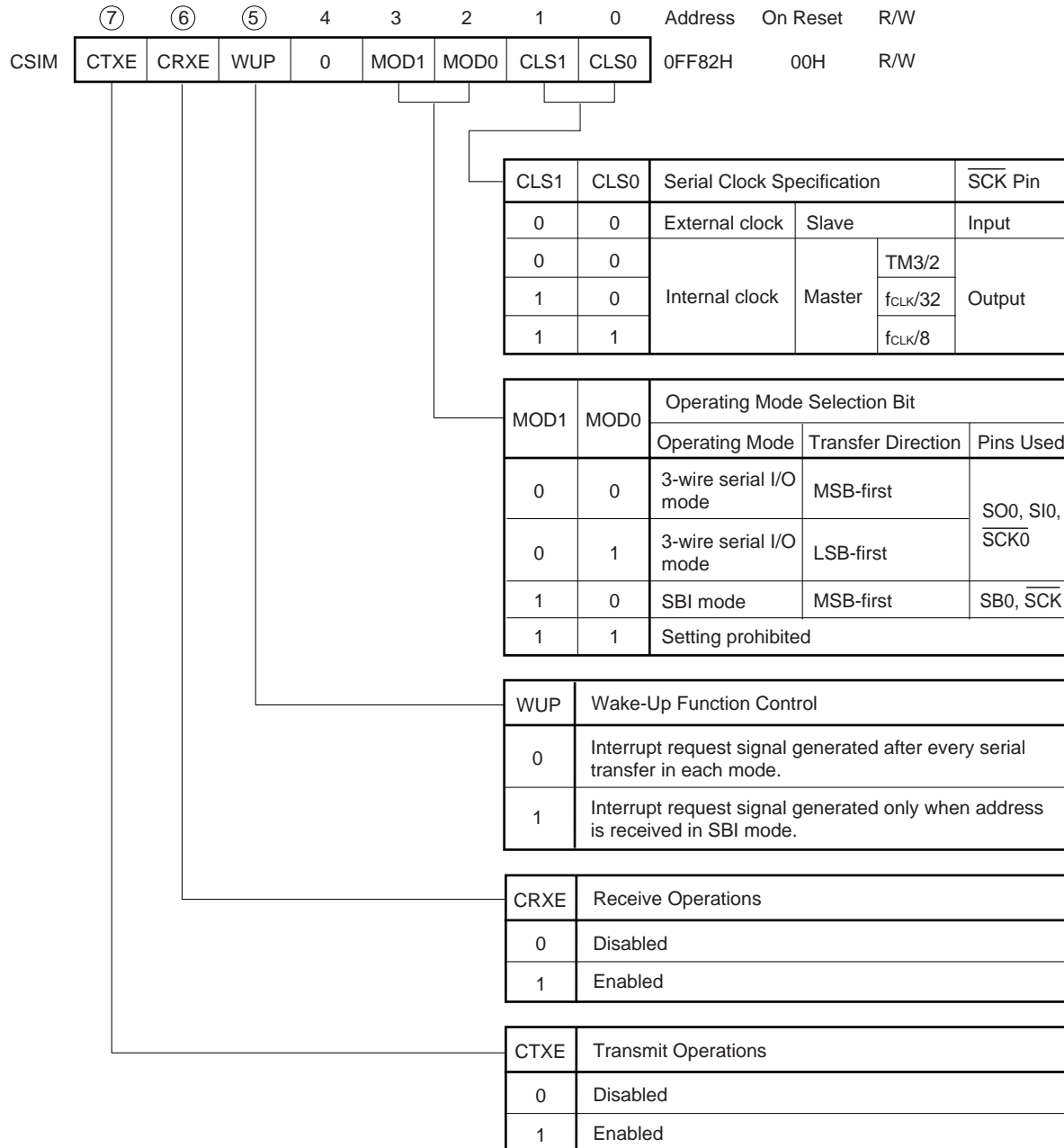
## 17.3.1 Clocked Serial Interface Mode Register (CSIM)

The CSIM is an 8-bit register that specifies the serial interface operating mode, serial clock, wake-up function, etc.

This register can be read or written to with an 8-bit manipulation instruction or bit manipulation instruction. The CSIM format is shown in Figure 17-2.

$\overline{\text{RESET}}$  input clears the CSIM register to 00H.

Figure 17-2 Clocked Serial Interface Mode Register (CSIM) Format



**Caution** When changing from “CTXE = 0, CRXE = 1” to “CTXE = 1, CRXE = 0”, and when changing from “CTXE = 1, CRXE = 0” to “CTXE = 0, CRXE = 1”, ensure that this is not done with a single instruction, as this will result in misoperation of the serial clock counter, and the first communication after the change will finish in fewer than 8 bits. Instead, two instructions should be used as shown below.

**Example** To change “CTXE = 1, CRXE = 0” to “CTXE = 0, CRXE = 1”

CLR1 CTXE

SET1 CRXE

### 17.3.2 Serial Bus Interface Control Register (SBIC)

SBIC is an 8-bit register composed of bits which control the serial bus status and bits which indicate various statuses of the input data from the serial bus. In the 3-wire serial I/O mode, only CMDT bit and RELT bit can be manipulated.

SBIC is manipulated by 8-bit manipulation instructions and bit manipulation instructions. Read/write capability varies from bit to bit as shown in Figure 17-1. When a read is performed, write-only bits are read as “0”. The format of the SBIC register is shown in Figure 17-3. The SBIC register must not be written to during a transmit, receive, or transmit/receive operation.

RESET input clears SBIC to 00H.

The ACKD, CMDD and RELD flags are cleared when transmit/receive operations are disabled (CTXE = CRXE = 0).

**Figure 17-3 Serial Bus Interface Control Register (SBIC) Reading/Writing**

	⑦	⑥	⑤	④	③	②	①	①
SBIC	BSYE	ACKD	ACE	ACKT	CMDD	RELD	CMDT	RELT
	R/W	R	R/W	W	R	R	W	W

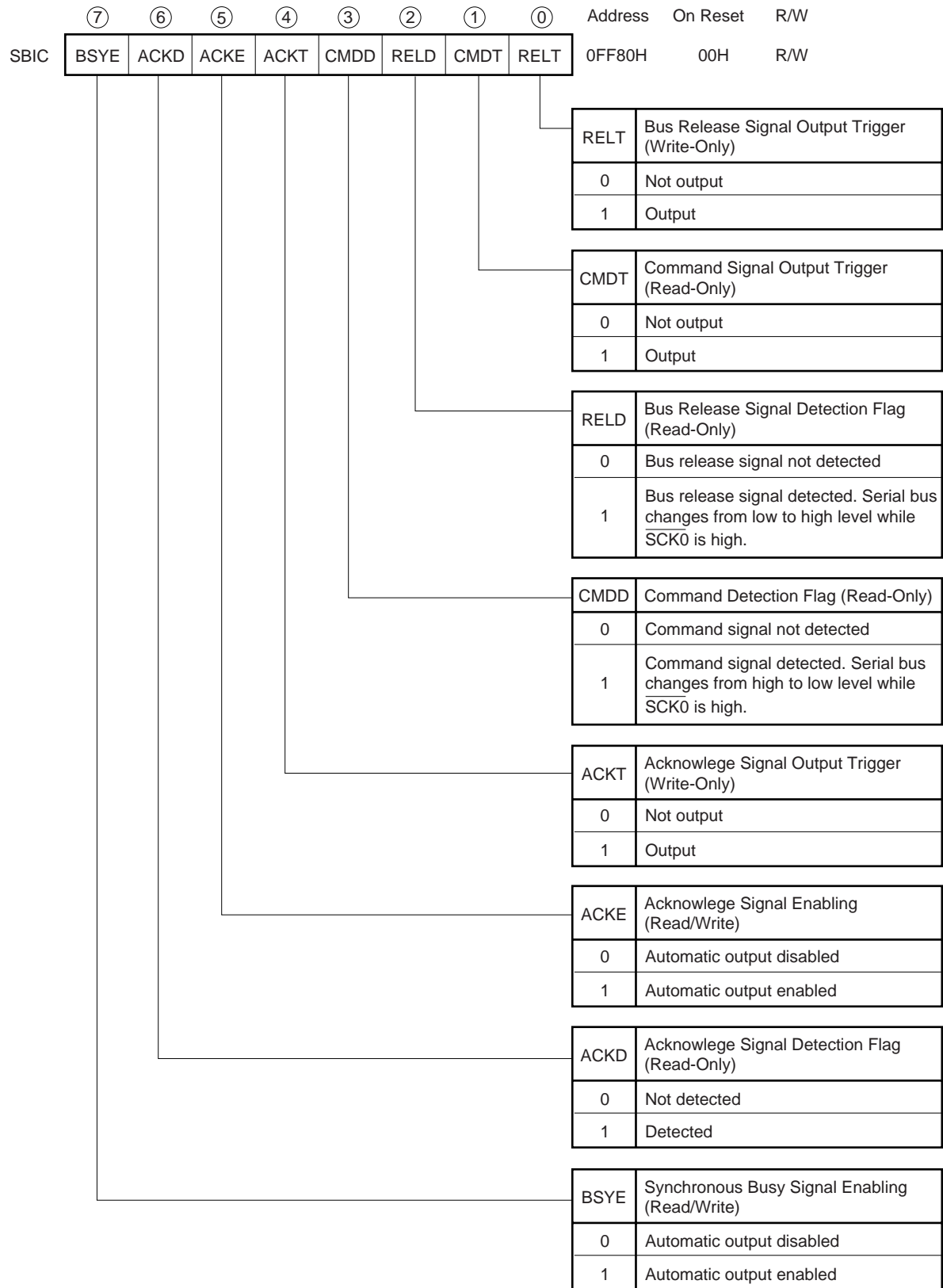
  

CMDT	RELT	SO0 Latch Operations
0	0	Unchanged
0	1	SO latch driven high
1	0	SO latch driven low
1	1	Setting prohibited

**Remark** R/W: Read/write

R : Read only

W : Write only

**Figure 17-4 Serial Bus Interface Control Register (SBIC) Format**



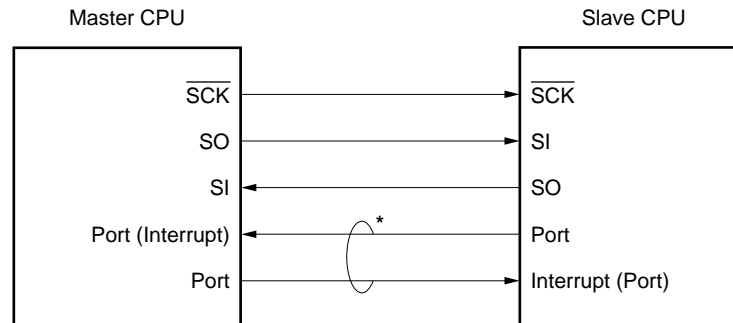
**17.4 3-WIRE SERIAL I/O MODE**

The 3-wire serial I/O mode is used to communicate with devices that incorporate a conventional clocked serial interface.

Basically, communication is performed using three lines: the serial clock ( $\overline{\text{SCK}}$ ), serial data output (SO), and serial data input (SI). Handshaking lines are required when a number of devices are connected.

**Figure 17-5 Example of 3-Wire Serial I/O System Configuration**

**3-wire serial I/O ↔ 3-wire serial I/O**



\* Handshaking lines

**17.4.1 Basic Operation Timing**

In the 3-wire serial I/O mode, data transmission/ reception is performed in 8-bit units. Data is transmitted/received bit by bit in MSB-first or LSB-first order in synchronization with the serial clock.

MSB first/LSB first switching is specified by the mode 0 bit of the clocked serial interface mode register (CSIM).

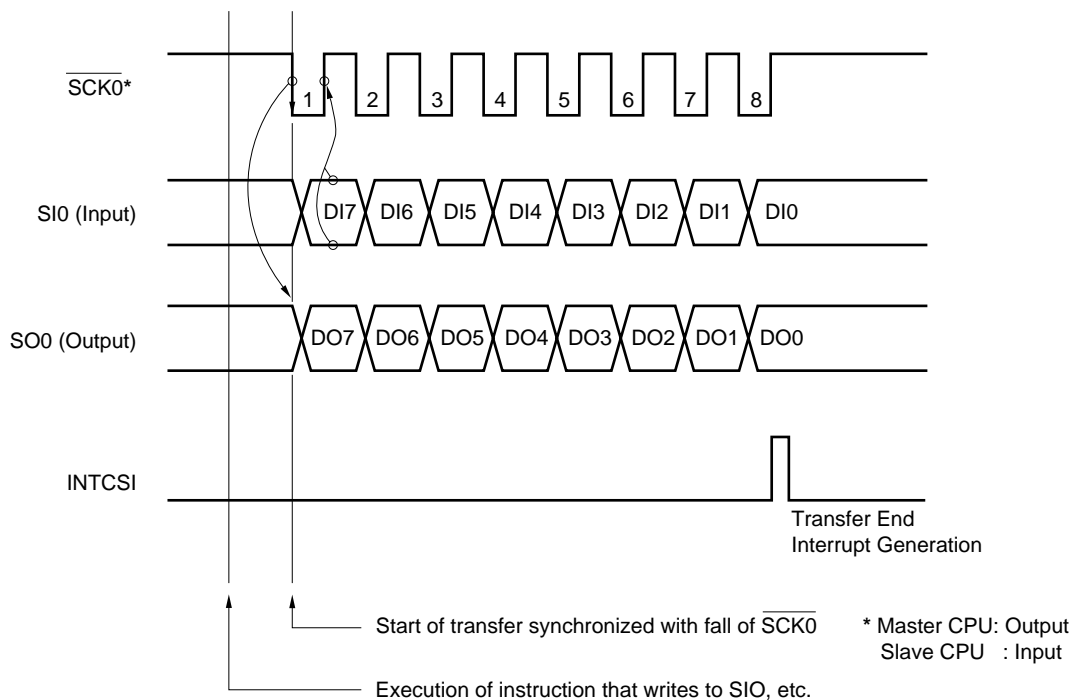
Transmit data is output in synchronization with the fall of  $\overline{\text{SCK0}}$ , and receive data is sampled on the rise of  $\overline{\text{SCK0}}$ .

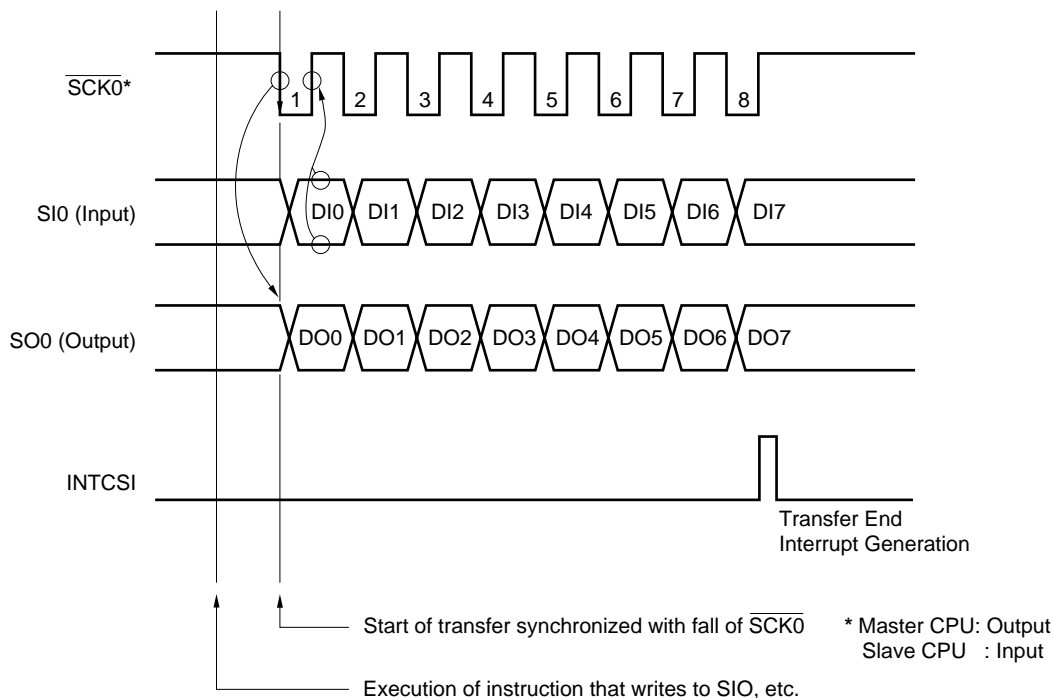
An interrupt request (INTCSI) is generated on the 8th rise of  $\overline{\text{SCK0}}$ .

When the internal clock is used as  $\overline{\text{SCK0}}$ ,  $\overline{\text{SCK0}}$  output is stopped on the 8th rise of  $\overline{\text{SCK0}}$  and  $\overline{\text{SCK0}}$  remains high until the next data transmit or receive operation is started.

3-wire serial I/O mode timing is shown in Figure 17-6.

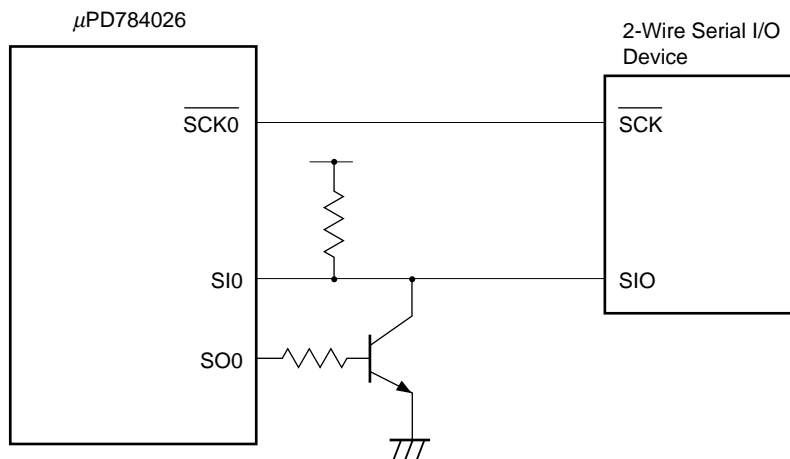
**Figure 17-6 3-Wire Serial I/O Mode Timing (1/2)**

**(a) MSB-first**

**Figure 17-6 3-Wire Serial I/O Mode Timing (2/2)****(b) LSB-first**

In the 3-wire serial I/O mode, the SO0 pin functions as a CMOS push-pull output.

**Remark** If the  $\mu\text{PD784026}$  is connected to a 2-wire serial I/O device, a buffer should be connected to the SO0 pin as shown in Figure 17-7. In the example shown in Figure 17-7, the output level is inverted by the buffer, and therefore the inverse of the data to be output should be written to SIO. Also, non-connection of an internal pull-up resistor should be specified for the P33/SO0 pin.

**Figure 17-7 Example of Connection to 2-Wire Serial I/O Device**

### 17.4.2 Operation When Transmission Only Is Enabled

A transmit operation is performed when the CTXE bit of the clocked serial interface mode register (CSIM) is set (1). The transmit operation starts when a write to the shift register (SIO) is performed while the CTXE1 bit is set (1).

When the CTXE bit is cleared (0), the SO pin is in the output high impedance state.

#### (1) When the internal clock is selected as the serial clock

When transmission starts, the serial clock is output from the  $\overline{\text{SCK0}}$  pin and data is output in sequence from SIO to the SO0 pin in synchronization with the fall of the serial clock, and SIO pin signals are shifted into SIO in synchronization with the rise of the serial clock.

There is a delay of up to one  $\overline{\text{SCK0}}$  clock cycle between the start of transmission and the first fall of  $\overline{\text{SCK0}}$ .

If transmission is disabled during the transmit operation (by clearing (0) the CTXE bit),  $\overline{\text{SCK0}}$  clock output is stopped and the transmit operation is discontinued on the next rise of  $\overline{\text{SCK0}}$ . In this case an interrupt request (INTCSI) is not generated, and the SO0 pin becomes output high impedance.

#### (2) When an external clock is selected as the serial clock

When transmission starts, data is output in sequence from SIO to the SO0 pin in synchronization with the fall of the serial clock input to the  $\overline{\text{SCK0}}$  pin after the start of transmission, and SIO pin signals are shifted into SIO in synchronization with the rise of the  $\overline{\text{SCK0}}$  pin input. If transmission has not started, shift operations are not performed and the SO0 pin output level does not change even if the serial clock is input to the  $\overline{\text{SCK0}}$  pin.

If transmission is disabled during the transmit operation (by clearing (0) the CTXE bit), the transmit operation is discontinued and subsequent  $\overline{\text{SCK0}}$  input is ignored. In this case an interrupt request (INTCSI) is not generated, and the SO0 pin becomes output high impedance.

### 17.4.3 Operation When Reception Only Is Enabled

A receive operation is performed when the CRXE bit of the clocked serial interface mode register (CSIM) is set (1). The receive operation starts when the CRXE changes from “0” to “1”, or when a read from shift register (SIO) is performed.

#### (1) When the internal clock is selected as the serial clock

When reception starts, the serial clock is output from the  $\overline{\text{SCK0}}$  pin and the SIO pin data is fetched in sequence into shift register (SIO) in synchronization with the rise of the serial clock.

There is a delay of up to one  $\overline{\text{SCK0}}$  clock cycle between the start of reception and the first fall of  $\overline{\text{SCK0}}$ .

If reception is disabled during the receive operation (by clearing (0) the CRXE bit),  $\overline{\text{SCK0}}$  clock output is stopped and the receive operation is discontinued on the next rise of  $\overline{\text{SCK0}}$ . In this case an interrupt request (INTCSI) is not generated, and the contents of the SIO register will be undefined.

#### (2) When an external clock is selected as the serial clock

When reception starts, the SIO pin data is fetched into shift register (SIO) in synchronization with the rise of the serial clock input to the  $\overline{\text{SCK0}}$  pin after the start of reception. If reception has not started, shift operations are not performed even if the serial clock is input to the  $\overline{\text{SCK0}}$  pin.

If reception is disabled during the receive operation (by clearing (0) the CRXE bit), the receive operation is discontinued and subsequent  $\overline{\text{SCK0}}$  input is ignored. In this case an interrupt request (INTCSI) is not generated.

**17.4.4 Operation When Transmission/Reception Is Enabled**

When the CTXE bit and CRXE bit of the clocked serial interface mode register (CSIM) are both set (1), a transmit operation and receive operation can be performed simultaneously (transmit/receive operation). The transmit/receive operation is started when the CRXE bit is changed from “0” to “1”, or by performing a write to shift register (SIO).

When a transmit operation is started for the first time, the CRXE bit always changes from “0” to “1”, and there is thus a possibility that the transmit/receive operation will start immediately, and undefined data will be output. The first transmit data should therefore be written to SIO beforehand when both transmission and reception are disabled (when the CTXE bit and CRXE bit are both cleared (0)), before enabling transmission/reception.

When transmission/reception is disabled (CTXE = CRXE = 0), the SO0 pin is in the output high impedance state.

**(1) When the internal clock is selected as the serial clock**

When transmission/reception starts, the serial clock is output from the  $\overline{\text{SCK0}}$  pin, data is output in sequence from shift register (SIO) to the SO0 pin in synchronization with the fall of the serial clock, and SIO pin data is shifted in order into SIO in synchronization with the rise of the serial clock.

There is a delay of up to one  $\overline{\text{SCK0}}$  clock cycle between the start of transmission and the first fall of  $\overline{\text{SCK0}}$ .

If either transmission or reception is disabled during the transmit/receive operation, only the disabled operation is discontinued. If transmission only is disabled, the SO0 pin becomes output high impedance. If reception only is disabled, the contents of the SIO register will be undefined.

If transmission and reception are disabled simultaneously,  $\overline{\text{SCK0}}$  clock output is stopped and the transmit and receive operations are discontinued on the next rise of  $\overline{\text{SCK0}}$ . When transmission and reception are disabled simultaneously, the contents of SIO are undefined, an interrupt request (INTCSI) is not generated, and the SO0 pin becomes output high impedance.

**(2) When an external clock is selected as the serial clock**

When transmission/reception starts, data is output in sequence from shift register (SIO) to the SO0 pin in synchronization with the fall of the serial clock input to the  $\overline{\text{SCK0}}$  pin after the start of transmission/reception, and SIO pin data is shifted in order into SIO in synchronization with the rise of the serial clock. If transmission/reception has not started, shift operations are not performed and the SO0 pin output level does not change even if the serial clock is input to the  $\overline{\text{SCK0}}$  pin.

If either transmission or reception is disabled during the transmit/receive operation, only the disabled operation is discontinued. If transmission only is disabled, the SO0 pin becomes output high impedance. If reception only is disabled, the contents of the SIO register will be undefined.

If transmission and reception are disabled simultaneously, the transmit and receive operations are discontinued and subsequent  $\overline{\text{SCK0}}$  input is ignored. When transmission and reception are disabled simultaneously, the contents of SIO are undefined, an interrupt request (INTCSI) is not generated, and the SO0 pin becomes output high impedance.

#### 17.4.5 Corrective Action in Case of Slippage of Serial Clock and Shift Operations

When an external clock is selected as the serial clock, there may be slippage between the number of serial clocks and shift operations due to noise, etc. In this case, since the serial clock counter is initialized by disabling both transmit operations and receive operations (by clearing (0) the CTXE bit and CRXE bit), synchronization of the shift operations and the serial clock can be restored by using the first serial clock input after reception or transmission is next enabled as the first clock.

### 17.5 SBI MODE

The SBI (serial bus interface) is a high-speed serial interface system that conforms to the NEC serial bus format.

The SBI is a single-master high-speed serial bus, with a format in which bus configuration functions have been added to the clocked serial I/O method to allow communication with multiple devices using two signal lines. This allows a reduction in the number of ports used and in the wiring on the board when a serial bus is configured with multiple microcomputers and peripheral ICs.

Please refer to the **Serial Bus Interface (SBI) User's Manual (EEU-1303)** for details of SBI functions, etc.

### 17.5.1 SBI Features

Since conventional serial I/O methods only have data transfer functions, many ports and wires are required to distinguish between the chip select signal and commands/data, to identify the busy state, etc., when the serial bus is configured with a number of devices connected. Moreover, if the associated control is performed by software, this represents a considerable burden for software.

In SBI, a serial bus can be configured with two signal lines of serial clock  $\overline{\text{SCK0}}$  and serial data bus SB0. Thus, use of SBI leads to reduction in the number of microcontroller ports and that of wirings and routings on the board.

SBI functions are described below.

#### (1) Address/command/data differentiation function

Identifies serial data as an address, command or actual data.

#### (2) Chip selection by address

The master performs slave chip selection by address transmission.

#### (3) Wake-up function

A slave can identify address reception (chip selection) easily by means of the wake-up function (settable/ releasable in software).

When the wake-up function is set, a serial reception interrupt (INTCSI) is generated only when an address is received.

As a result, non-selected CPUs can operate without regard to serial communications even when communication with multiple devices is performed.

#### (4) Acknowledge signal ( $\overline{\text{ACK}}$ ) control function

Controls the acknowledge signal used to confirm serial data reception.

#### (5) Busy signal ( $\overline{\text{BUSY}}$ ) control function

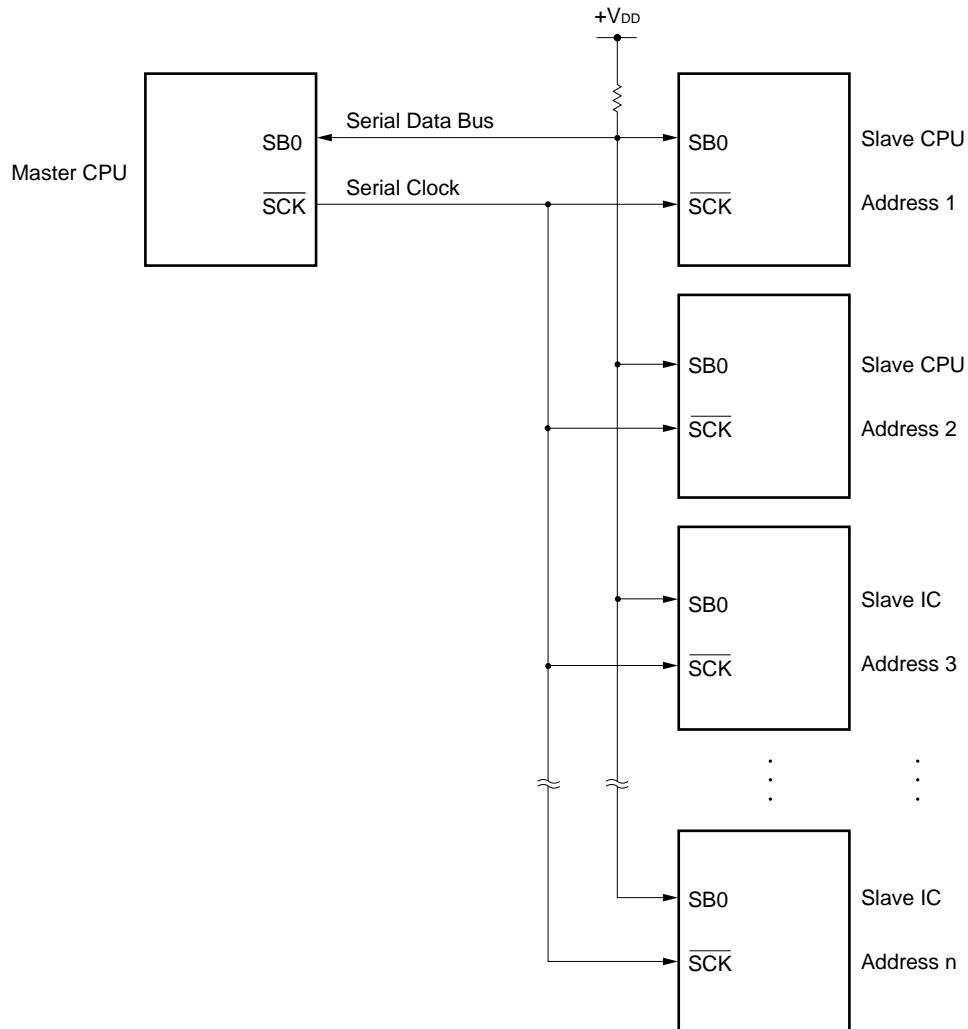
Controls the busy signal used to give notification of a slave busy status.



Figure 17-8 shows an example of a serial bus configuration in which CPUs and peripheral ICs that have a serial interface conforming to the SBI are used.

With the SBI, since the SB0 serial data bus pin is an open-drain output, the serial data bus line is in the wired-OR state. Also, a pull-up resistor is necessary for the serial data bus line.

**Figure 17-8 Example of SBI Serial Bus Configuration**

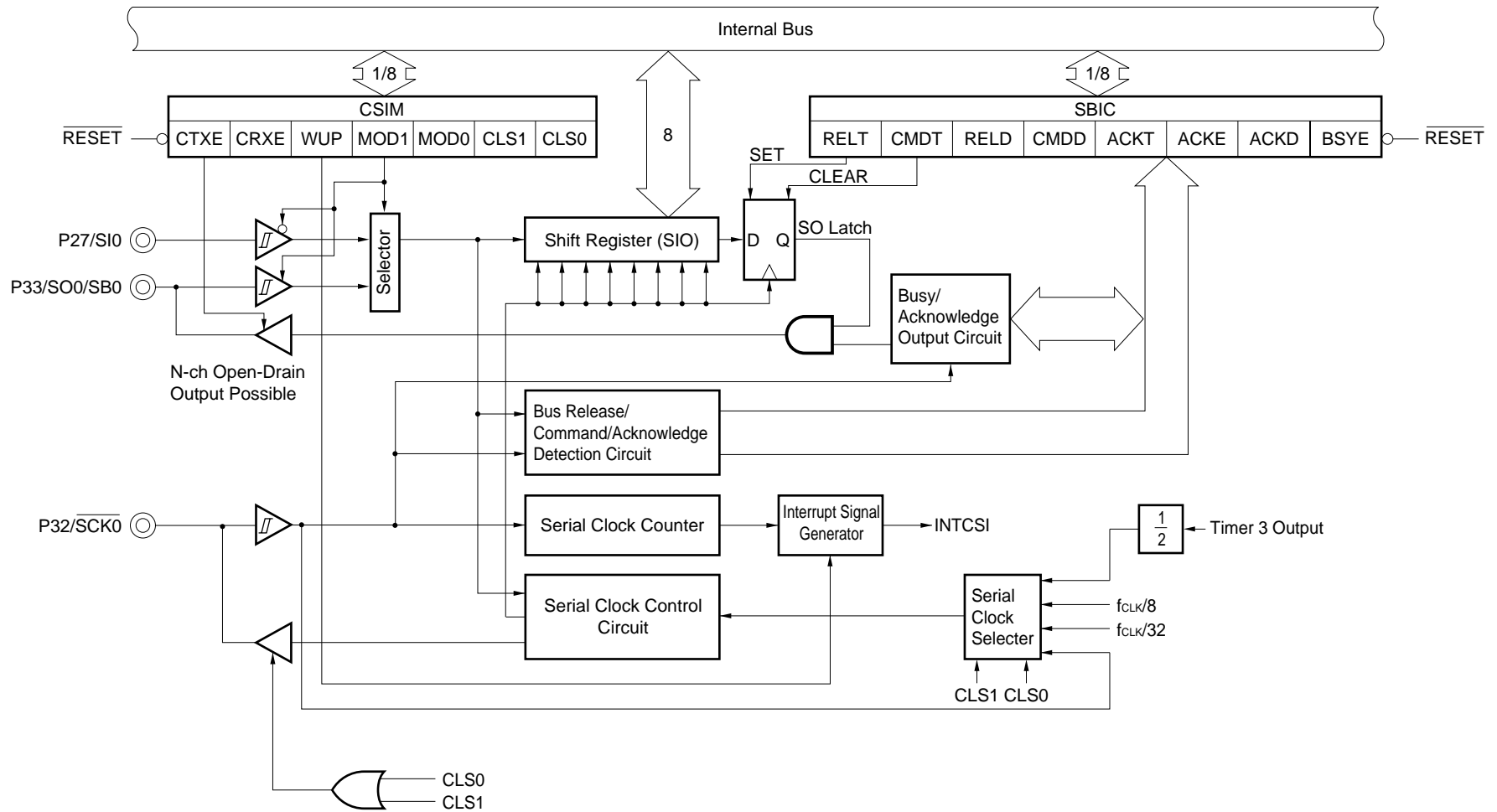


**Caution** When master/slave exchange processing is performed, since serial clock line (**SCK**) input/output switching is performed asynchronously between master and slave, a pull-up resistor is also required for the serial clock line (**SCK**).

### 17.5.2 Serial Interface Configuration

The block diagram of the clocked serial interface (CSI) is shown in Figure 17-9.

Figure 17-9 Clocked Serial Interface (CSI) Block Diagram

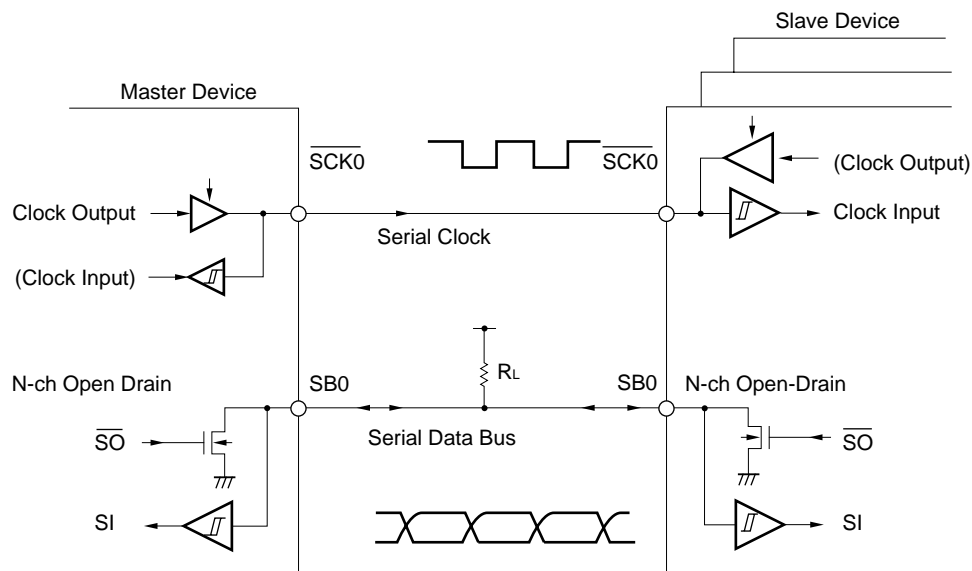


The serial clock pin ( $\overline{\text{SCK0}}$ ) and serial data bus pin (SB0) configuration is as shown below.

- (1) SCK0 ..... Pin for inputting/outputting the serial clock
  - Master ..... CMOS, push-pull output
  - Slave ..... Schmitt input
- (2) SB0 ..... Serial data input/output dual-function pin  
 For both master and slave, output is N-ch open-drain, and input is Schmitt input.

As the serial data bus line output is N-ch open-drain, an external pull-up resistor is required.

**Figure 17-10 Pin Configuration Diagram**



### 17.5.3 Address Match Detection

In SBI communication, a specific slave device is selected and communication is started by master address transmission.

Slave device address match detection is performed by software. In the wake-up state ( $WUP = 1$ ), a slave generates a serial transfer completion interrupt request only when it receives an address.

In match address reception processing by software, the wake-up state is released ( $WUP \leftarrow 0$ ) in preparation for subsequent command or data reception.

### 17.5.4 SBI Mode Control Registers

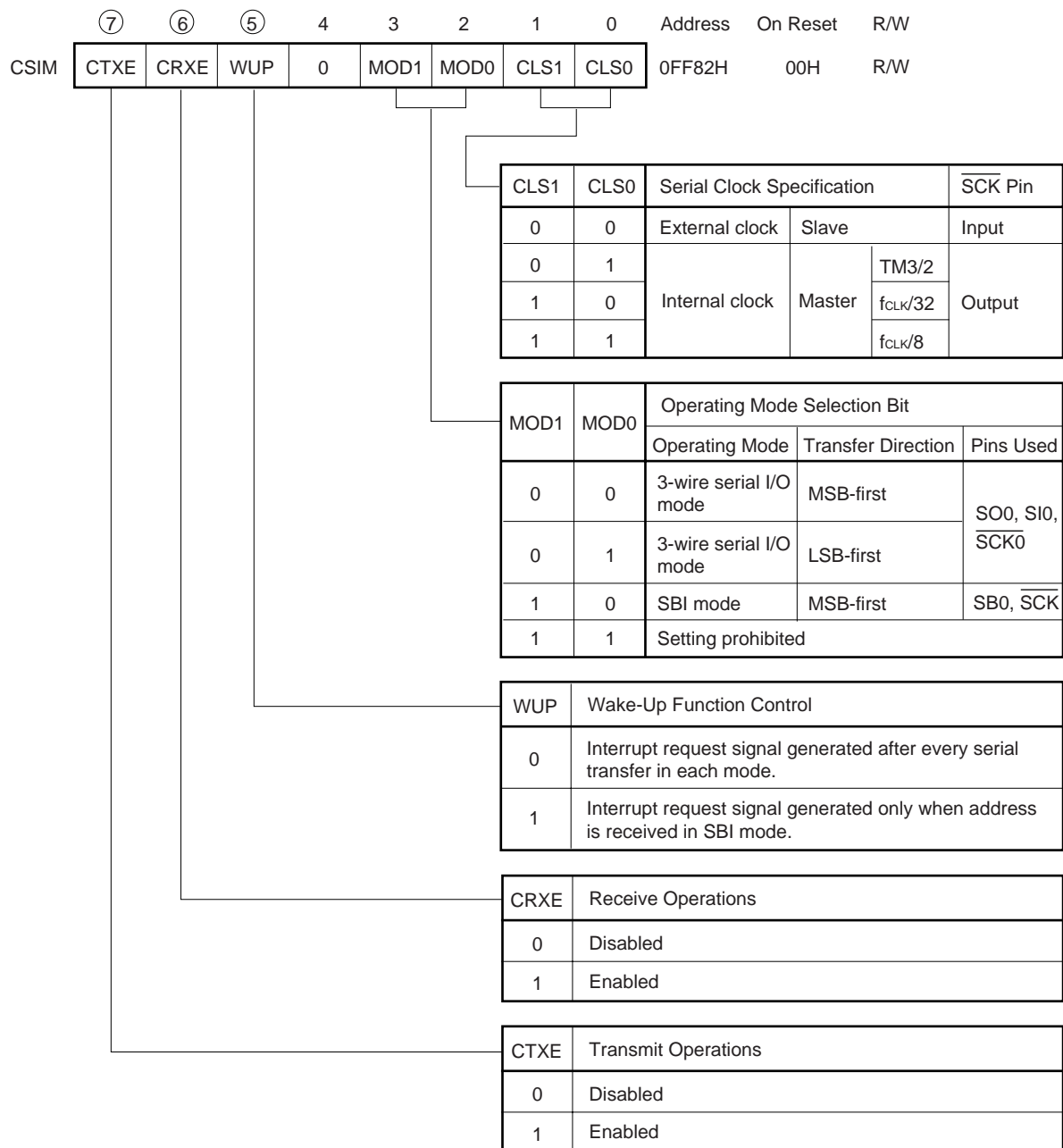
#### (1) Clocked serial interface mode register (CSIM)

The CSIM is an 8-bit register that specifies the serial interface operating mode, serial clock, wake-up function, etc.

The format of the CSIM is shown in Figure 17-11.

CSIM can be read or written to with an 8-bit manipulation instruction or bit manipulation instruction, but read/write attributes are assigned bit by bit.

$\overline{RESET}$  input clears the value of CSIM to 00H.

**Figure 17-11 Clocked Serial Interface Mode Register (CSIM) Format**

**Caution** When changing from “CTXE = 0, CRXE = 1” to “CTXE = 1, CRXE = 0”, and when changing from “CTXE = 1, CRXE = 0” to “CTXE = 0, CRXE = 1”, ensure that this is not done with a single instruction, as this will result in misoperation of the serial clock counter, and the first communication after the change will finish in fewer than 8 bits. Instead, two instructions should be used as shown below.

**Example** To change “CTXE = 1, CRXE = 0” to “CTXE = 0, CRXE = 1”

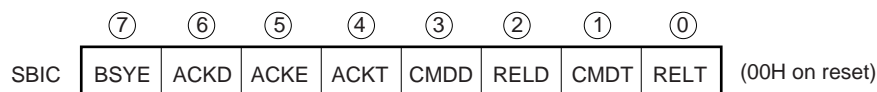
```
CLR1 CTXE
SET1 CRXE
```

**(2) Serial bus interface control register (SBIC)**

The SBIC is an 8-bit register composed of bits which control the serial bus status and flags which indicate various statuses of the input data from the serial bus.

SBIC can be read or written to with an 8-bit manipulation instruction or bit manipulation instruction, but read/write capability varies from bit to bit. The format of the SBIC is shown in Figure 17-12.

RESET input clears the value of SBIC to 00H.

**Figure 17-12 SBIC Format (1/2)****Bus Release Trigger Bit (W)**

RELT	The bus release signal (REL) trigger output control bit. The SO latch is set (1) by setting this bit, after which the RELT bit is automatically cleared (0).
------	--

**Command Trigger Bit (W)**

CMDT	The command signal (CMD) trigger output control bit. The SO latch is cleared (0) by setting this bit, after which the CMDT bit is automatically cleared (0).
------	--

**Bus Release Detection Flag (R)**

RELD	Clearing Conditions (RELD = 0)	Setting Condition (RELD = 1)
	<1> When a transfer start instruction is executed	When the bus release signal (REL) is detected
	<2> When reset is input	
	<3> When CTXE = CRXE = 0	

**Command Detection Flag (R)**

CMDD	Clearing Conditions (CMDD = 0)	Setting Condition (CMDD = 1)
	<1> When a transfer start instruction is executed	When the command signal (CMD) is detected
	<2> When the bus release signal (REL) is detected	
	<3> When reset is input	
	<4> When CTXE = CRXE = 0	

Figure 17-12 SBIC Format (2/2)

**Acknowledge Trigger Bit (W)**

ACKT	When ACKT is set after the end of a transfer, $\overline{\text{ACK}}$ is output in synchronization with the next $\overline{\text{SCK0}}$ . After the $\overline{\text{ACK}}$ signal is output, ACKT is automatically cleared (0).		
	<b>Cautions</b>		
	1. Do not set (1) before the end of the serial transfer.		
	2. ACKT cannot be cleared by software.		
	3. When ACKT is set, ACKE must be set to 0.		

**Acknowledge Enable Bit (R/W)**

ACHE	0	Disables automatic output of the acknowledge signal	
	1	Before completion of transfer	$\overline{\text{ACK}}$ is output in synchronization with the 9th $\overline{\text{SCK0}}$ clock cycle.
		After completion of transfer	$\overline{\text{ACK}}$ is output in synchronization with $\overline{\text{SCK0}}$ immediately after execution of the setting instruction.

**Acknowledge Detection Flag (R)**

ACKD	Clearing Conditions ( $\text{ACKD} = 0$ )		Setting Condition ( $\text{ACKD} = 1$ )
	<1>	On first fall of $\overline{\text{SCK0}}$ after transfer start instruction execution and busy release	When the acknowledge signal ( $\overline{\text{ACK}}$ ) is detected
	<2>	When reset is input	
	<3>	When $\text{CTXE} = \text{CRXE} = 0$	
	<4>	When bus release is detected (slave mode only)	

**Busy Enable Bit (R/W)**

BSYE	0	<1> Disabling of automatic busy signal output <2> Busy signal output is stopped in synchronization with the fall of $\overline{\text{SCK0}}$ immediately after execution of the clearing instruction.
	1	The busy signal is output in synchronization with the fall of $\overline{\text{SCK0}}$ following the acknowledge signal.

**Remark** (R) : Read-only  
(W) : Write-only  
(R/W): Read/write

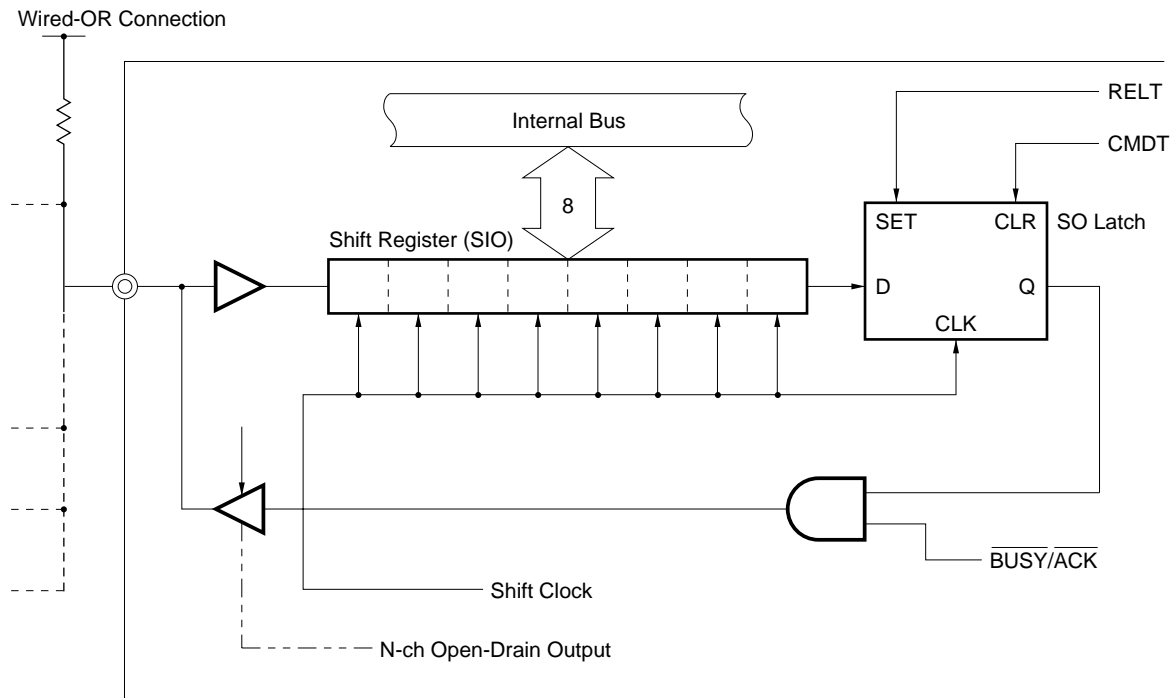


**(3) Shift register (SIO)**

SIO is an 8-bit shift register that performs parallel-to-serial conversion.

Data written to SIO is output to the serial data bus, and data is read from the serial data bus into SIO. The configuration around the SIO is shown in Figure 17-13.

**Figure 17-13 Configuration Around Shift Register (SIO)**



In the SBI data bus configuration, the input pin and output pin have a dual function. The output pin has an N-ch open-drain configuration, and a wired-OR configuration is achieved by means of an external pull-up resistor. Therefore, a device which is about to perform reception must set FFH in the SIO or set the transmission disabled state.

**17.6 SBI COMMUNICATION OPERATIONS AND VARIOUS SIGNALS**

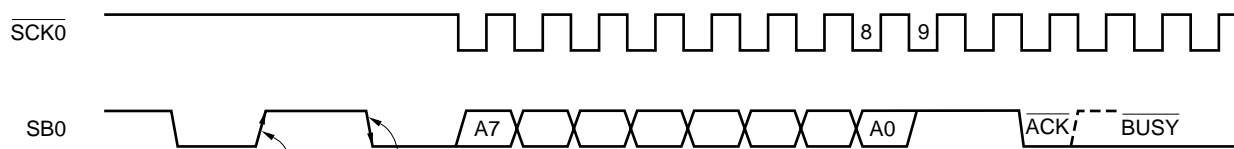
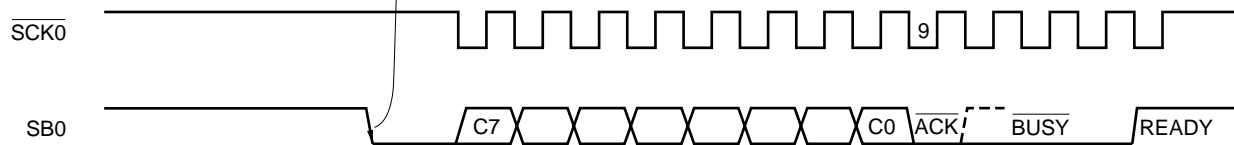
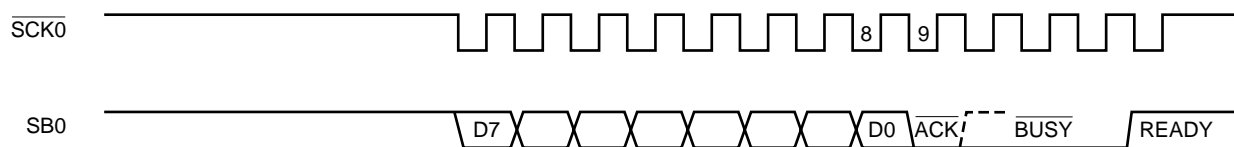
This section describes the SBI serial data format and explains the meaning of the signals used.

Serial data transferred via the SBI is identified as an address, a command, or data. One frame of serial data is made up of the following:

(Bus release signal) + (command signal) + 8-bit data +  $\overline{\text{ACK}}$  +  $\overline{\text{BUSY}}$

The address, command and data transfer timing is shown in Figure 17-14.

**Figure 17-14 SBI Transfer Timing**

**Address Transfer****Command Transfer****Data Transfer**

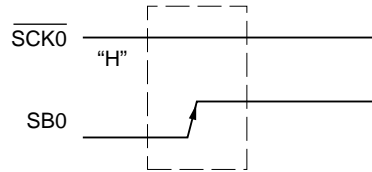
The bus release signal and command signal are output by the master.  $\overline{\text{BUSY}}$  is output by a slave.  $\overline{\text{ACK}}$  can be output by either the master or a slave (normally, it is output by the 8-bit data receiver).

The serial clock is output continuously by the master from the start of an 8-bit data transfer until  $\overline{\text{BUSY}}$  is released.

### 17.6.1 Bus Release Signal (REL)

The bus release signal is indicated by a low-to-high transition of the SB0 pin when  $\overline{\text{SCK0}}$  is high (when the serial clock is not being output). This signal is output by the master.

**Figure 17-15 Bus Release Signal**

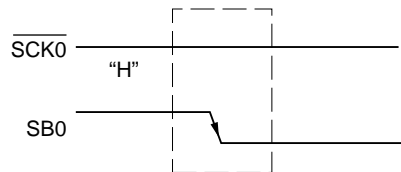


The bus release signal indicates that the master is going to send an address to a slave. The slaves incorporate hardware that detects the bus release signal.

### 17.6.2 Command Signal (CMD)

The command signal is indicated by a high-to-low transition of the SB0 pin when  $\overline{\text{SCK0}}$  is high (when the serial clock is not being output). This signal is output by the master.

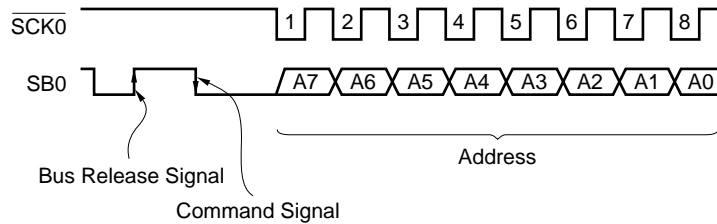
**Figure 17-16 Command Signal**



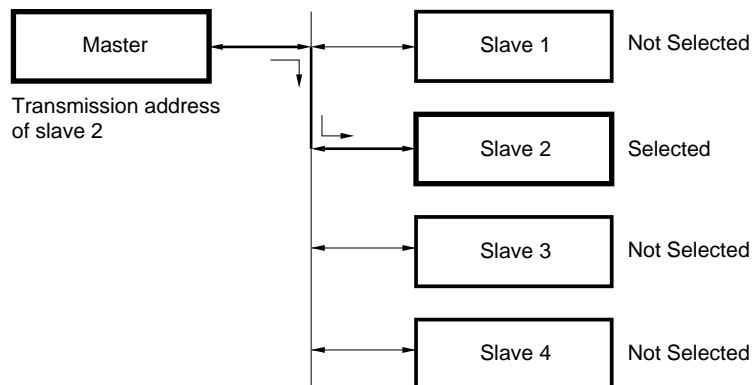
The slaves incorporate hardware that detects the command signal.

**17.6.3 Address**

An address is 8-bit data that is output by the master to select a specific slave from among the slaves connected to the bus lines.

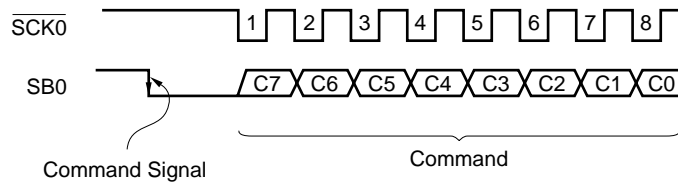
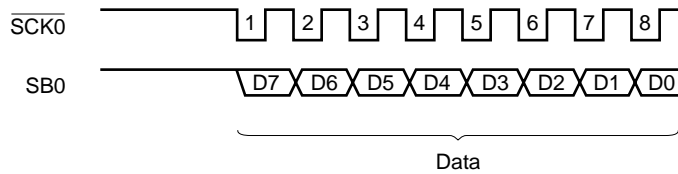
**Figure 17-17 Address**

The 8-bit data following a bus release signal and command signal is defined as an address. A slave detects this condition in hardware, and checks whether the 8-bit data matches its own specification number (slave address) in software or hardware. If the 8-bit data matches the slave address, that slave is selected and thenceforth communicates with the master until the master orders disconnection.

**Figure 17-18 Slave Selection by Address**

**17.6.4 Command and Data**

The master performs command transmission or data transmission to the slave selected by the address transmission.

**Figure 17-19 Command****Figure 17-20 Data**

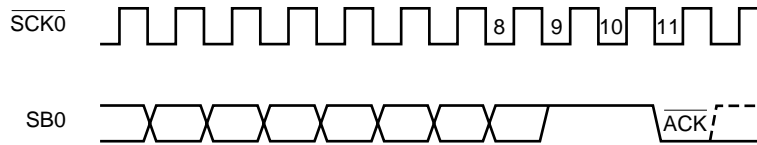
The 8-bit data following a command signal is defined as a command, and 8-bit data without a command signal is defined as data. The method of using commands and data can be decided freely according to the communication specifications.

**17.6.5 Acknowledge Signal ( $\overline{\text{ACK}}$ )**

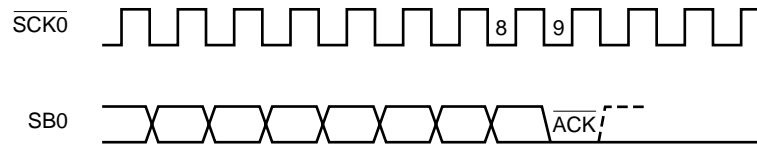
The acknowledge signal is used to confirm data reception between the sender and receiver.

**Figure 17-21 Acknowledge Signal**

[When output is synchronized with 11th  $\overline{\text{SCK0}}$ ]



[When output is synchronized with 9th  $\overline{\text{SCK0}}$ ]



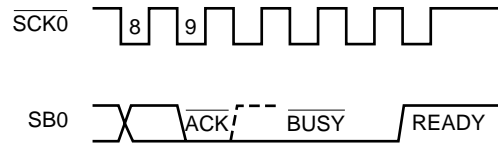
The acknowledge signal is a one-shot pulse synchronized with the fall of  $\overline{\text{SCK0}}$  after an 8-bit data transfer, and its position can be synchronized with any  $\overline{\text{SCK0}}$  clock pulse as required.

After transmitting 8-bit data, the sender checks whether the receiver has sent back an acknowledge signal. If an acknowledge signal has not been sent back within a given time after data transmission, it can be inferred that reception has not been performed correctly.

**17.6.6 Busy Signal ( $\overline{\text{BUSY}}$ ) and Ready Signal ( $\text{READY}$ )**

The busy signal is used to notify the master that the slave is preparing for data transmission/reception.

The ready signal is used to notify the master that the slave is ready to perform data transmission/reception.

**Figure 17-22 Busy Signal and Ready Signal**

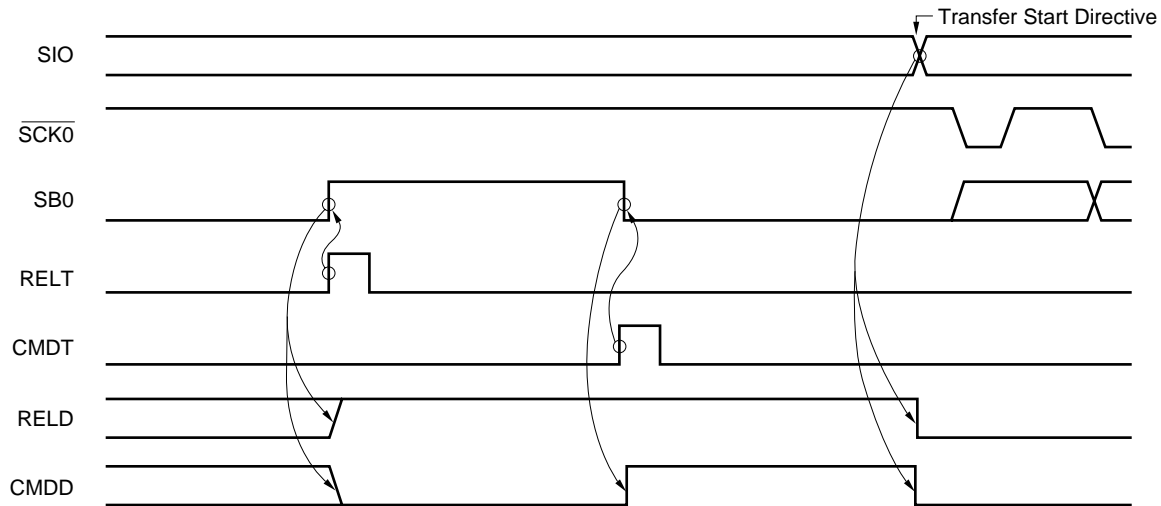
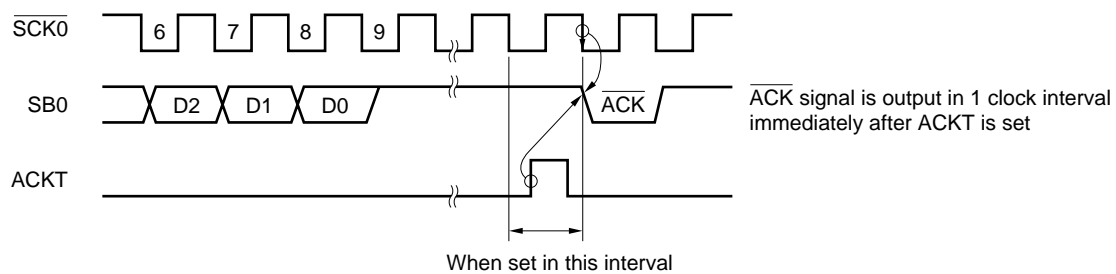
With the SBI, the slave notifies the master of the busy status by driving the  $\text{SB0}$  pin low.

The busy signal is output following an acknowledge signal output by the master or slave. The busy signal is set/released in synchronization with the fall of  $\overline{\text{SCK0}}$ . When the busy signal is released, the master terminates output of the serial clock,  $\overline{\text{SCK0}}$ .

When the busy signal is released and the ready signal state is established, the master is able to start the next transfer.

**17.6.7 Various Signals**

The operation of various signals in SBI mode and flags in serial bus interface control register (SBIC) is shown in Figures 17-23 to 17-27. Signals in SBI mode are shown in Table 17-1.

**Figure 17-23 RELT, CMDT, RELD, CMDD Operation****Figure 17-24 ACKT Operation**

**Caution** ACKT must not be set before the end of a transfer.



Figure 17-25 ACKE Operation

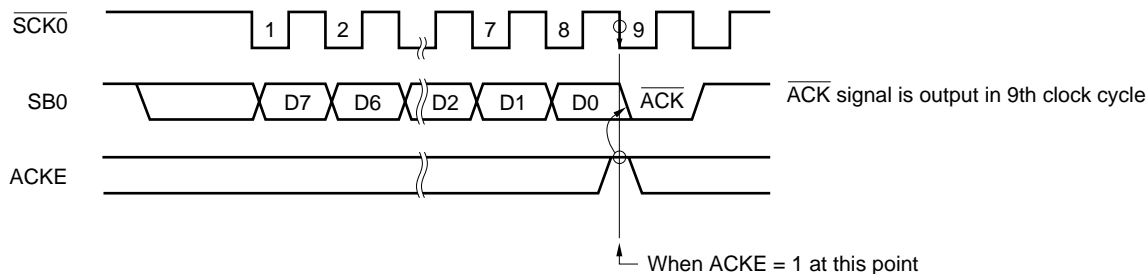
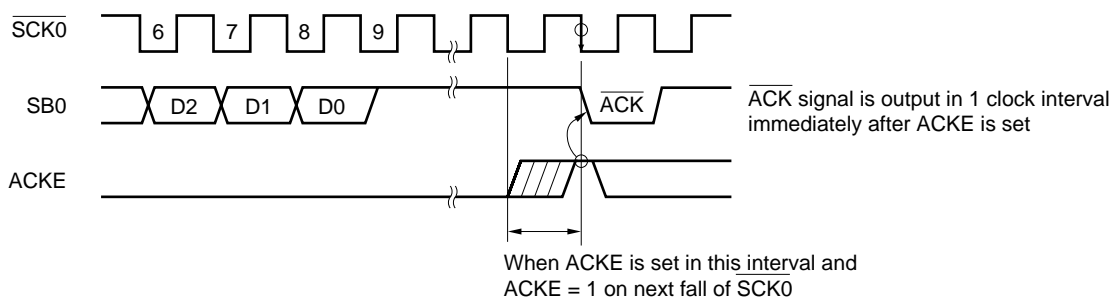
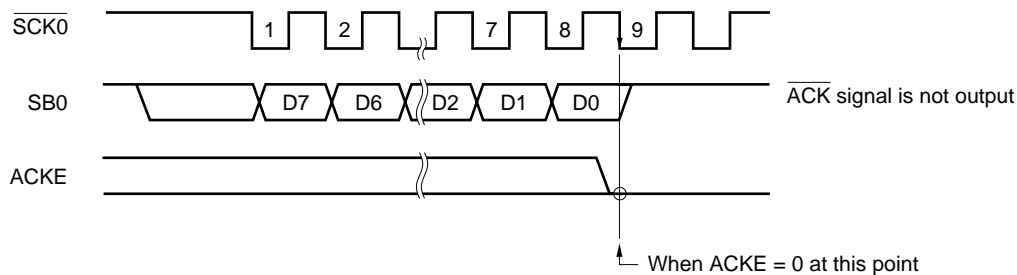
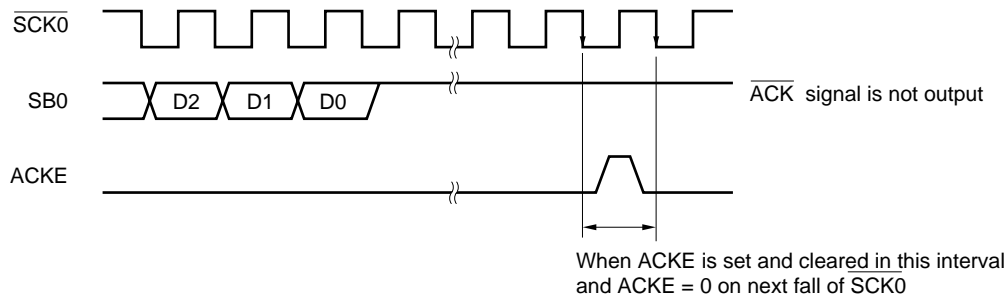
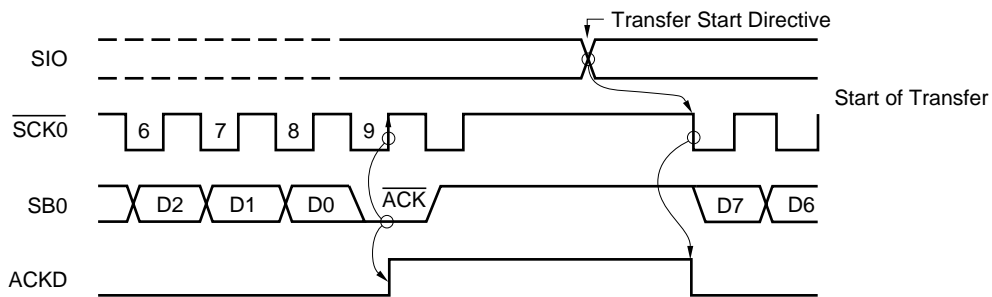
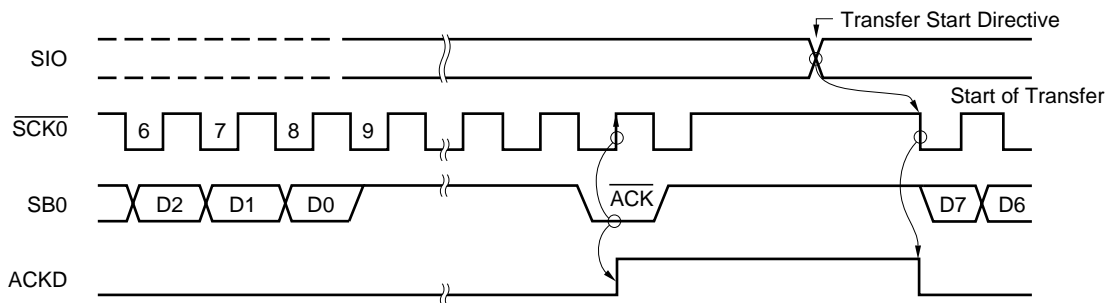
**(a) When ACKE = 1 on completion of transfer****(b) When ACKE is set after completion of transfer****(c) When ACKE = 0 on completion of transfer****(d) When ACKE = 1 interval is short**

Figure 17-26 ACKD Operation

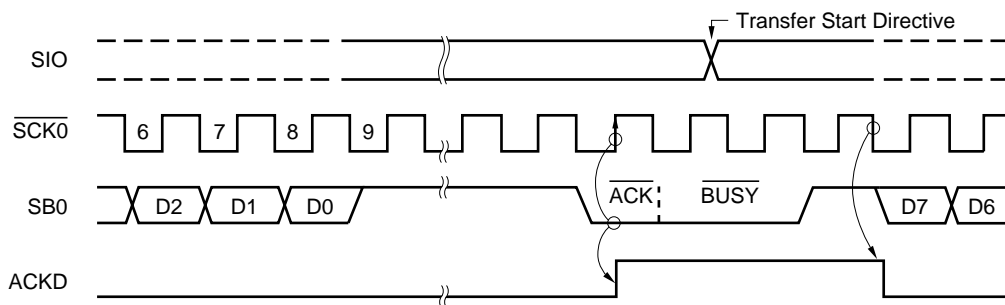
(a) When  $\overline{\text{ACK}}$  signal is output in 9th  $\text{SCK0}$  clock interval



(b) When  $\overline{\text{ACK}}$  signal is output after 9th  $\text{SCK0}$  clock interval



(c) Clearing timing when transfer start directive is given during BUSY state



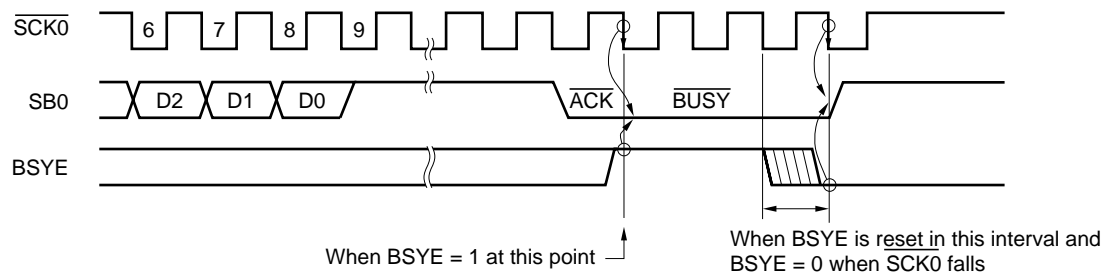
**Figure 17-27 BSYE Operation**

Table 17-1 Signals in SBI Mode (1/3)

Signal Name	Outputting Device	Definition	Timing Chart	Output Conditions	Effect on Flags	Meaning of Signal
Bus release signal (REL)	Master	SB0 rising edge when $\overline{\text{SCK0}} = 1$		• RELT set	• RELD set • CMDD cleared	Outputs CMD signal next and indicates transmit data is address.
Command signal (CMD)	Master	SB0 falling edge when $\overline{\text{SCK0}} = 1$		• CMDT set	• CMDD set	(1) Transmit data after REL signal output is address. (2) Send data with no REL signal output is command.

Table 17-1 Signals in SBI Mode (2/3)

Signal Name	Outputting Device	Definition	Timing Chart	Output Conditions	Effect on Flags	Meaning of Signal
Acknowledge signal (ACK)	Master/slave	Low-level signal output to SB0 in 1 SCK0 clock interval after serial receive completion		<1> ACKE = 1 <2> ACKT set	• ACKD set	Receive completed
Busy signal (BUSY)	Slave	Low-level signal output to SB0 after acknowledge signal		• BSYE = 1	—	Serial transmission/reception not possible because processing is in progress.
Ready signal (READY)	Slave	High-level signal output to SB0 before start or after completion of serial transfer.		<1> BSYE = 0 <2> Data write to SIO when CTXE = 1 (serial transfer start directive)* <3> Execution of instruction to read data from SIO when CTXE = 0 and CRXE = 1 <4> Change of CRXE bit from 0 to 1	—	Serial transmission/reception possible

\* For data transmission/reception, when the  $\overline{\text{BUSY}}$  state is set, the transfer starts after transition to the READY state.

Table 17-1 Signals in SBI Mode (3/3)

Signal Name	Outputting Device	Definition	Timing Chart	Output Conditions	Effect on Flags	Meaning of Signal
Serial Clock ( $\overline{\text{SCK0}}$ )	Master	Synchronization clock for output of address/command/data, $\overline{\text{ACK}}$ signal, synchronous $\overline{\text{BUSY}}$ signal, etc. Address/command/data is transferred in first 8 cycles.		<p>&lt;1&gt; Execution of instruction to write data to SIO when CTXE = 1 (serial transfer start directive)*<sup>1</sup></p> <p>&lt;2&gt; Execution of instruction to read data from SIO when CTXE = 0</p>	CSIF set (rise of 8th clock cycle)* <sup>2</sup>	Timing of signal output to serial data bus
Address (A7 to A0)	Master	8-bit data transferred in synchronization with $\overline{\text{SCK0}}$ after REL signal and CMD signal output		and CRXE = 1 <3> Change of CRXE bit from 0 to 1		Address value of slave device on serial bus
Command (C7 to C0)	Master	8-bit data transferred in synchronization with $\overline{\text{SCK0}}$ after CMD signal only is output without output of REL signal				Directive or message to slave device
Data (D7 to D0)	Master/slave	8-bit data transferred in synchronization with $\overline{\text{SCK0}}$ without output of either REL signal or CMD signal				Data to be processed by slave or master

\* 1. For data transmission/reception, when the  $\overline{\text{BUSY}}$  state is set, the transfer starts after transition to the READY state.

2. When WUP = 0, CSIF is always set on the 8th rise of  $\overline{\text{SCK0}}$ .

When WUP = 1, CSIF is set on the 8th rise of  $\overline{\text{SCK0}}$  only when an address is received.

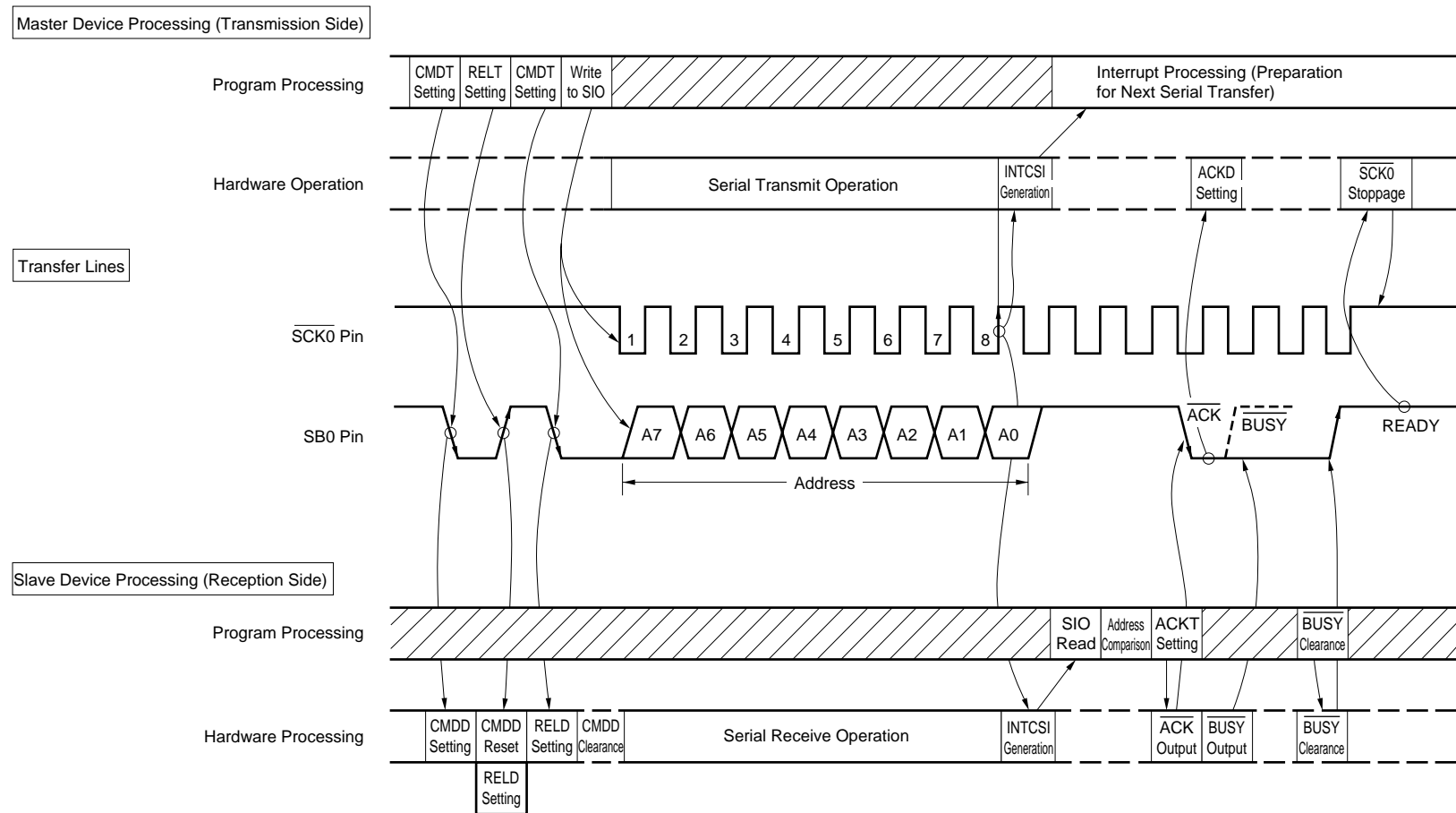
### 17.6.8 Communication Operations

With the SBI, the master selects the slave device it wishes to communicate with from among the devices connected to the serial bus by outputting an address on the serial bus.

After the communication target device has been decided, serial communication is implemented by the transmission and reception of commands and data between the master device and slave device.

Timing charts for the communication of each type of data are shown in Figures 17-28 to 17-31.

Figure 17-28 Address Transmission from Master Device to Slave Device

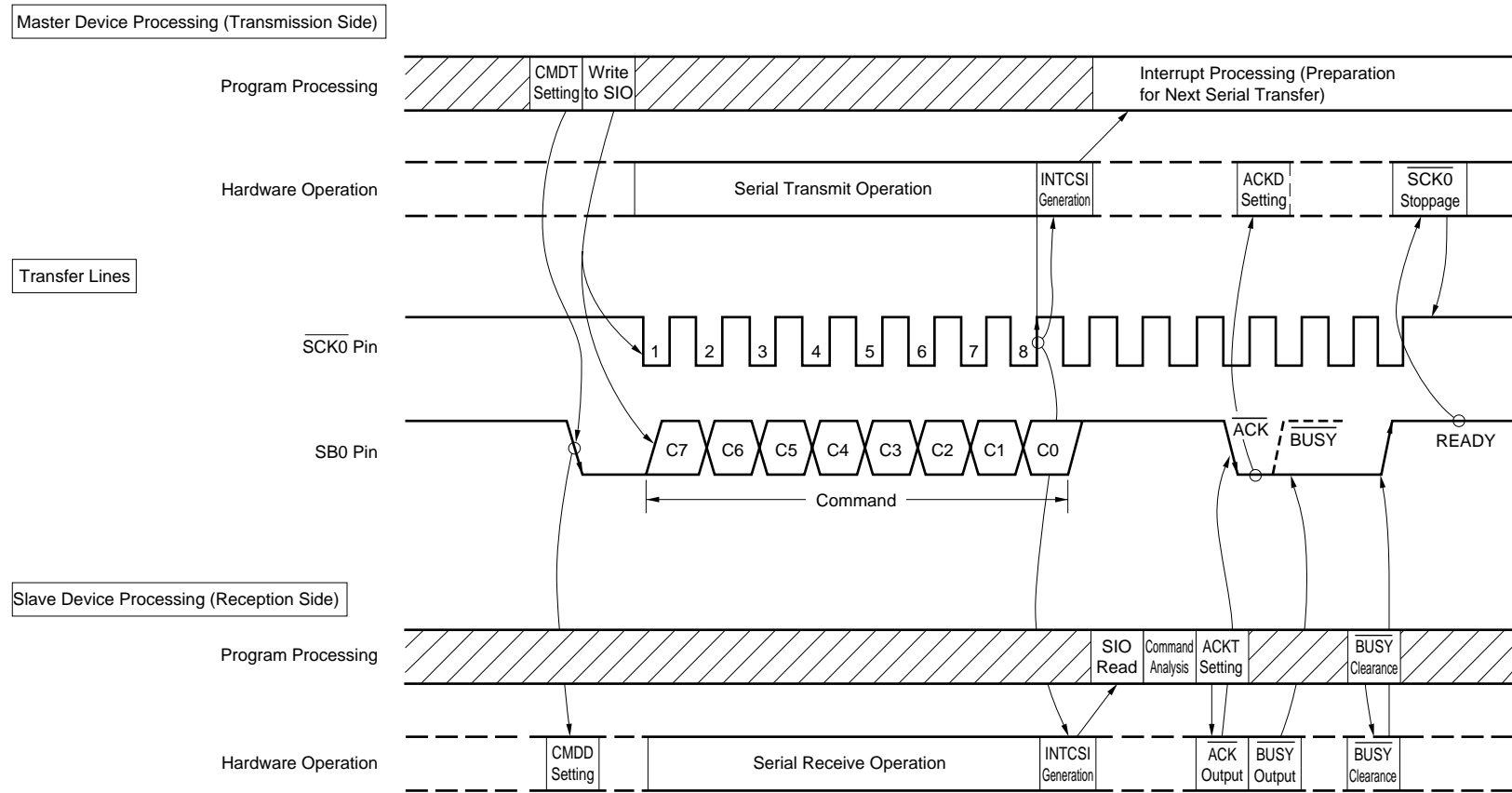


**Remark** This timing applies to the following conditions:

- Master: transmission only enabled
- Slave : reception only enabled, ACKE = 0, BSYE = 1



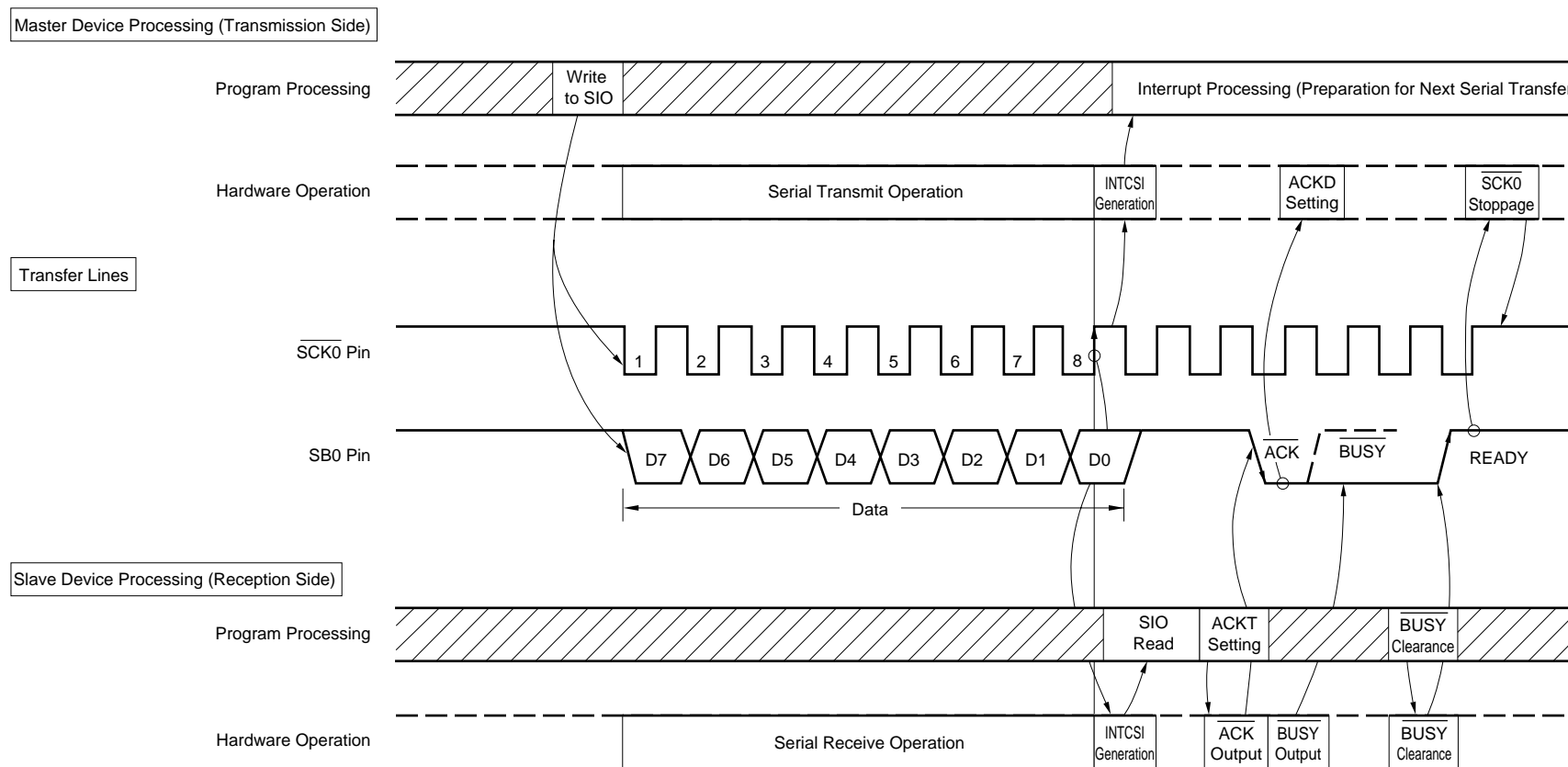
Figure 17-29 Command Transmission from Master Device to Slave Device



**Remark** This timing applies to the following conditions:

- Master: transmission only enabled
- Slave : reception only enabled, ACKE = 0, BSYE = 1

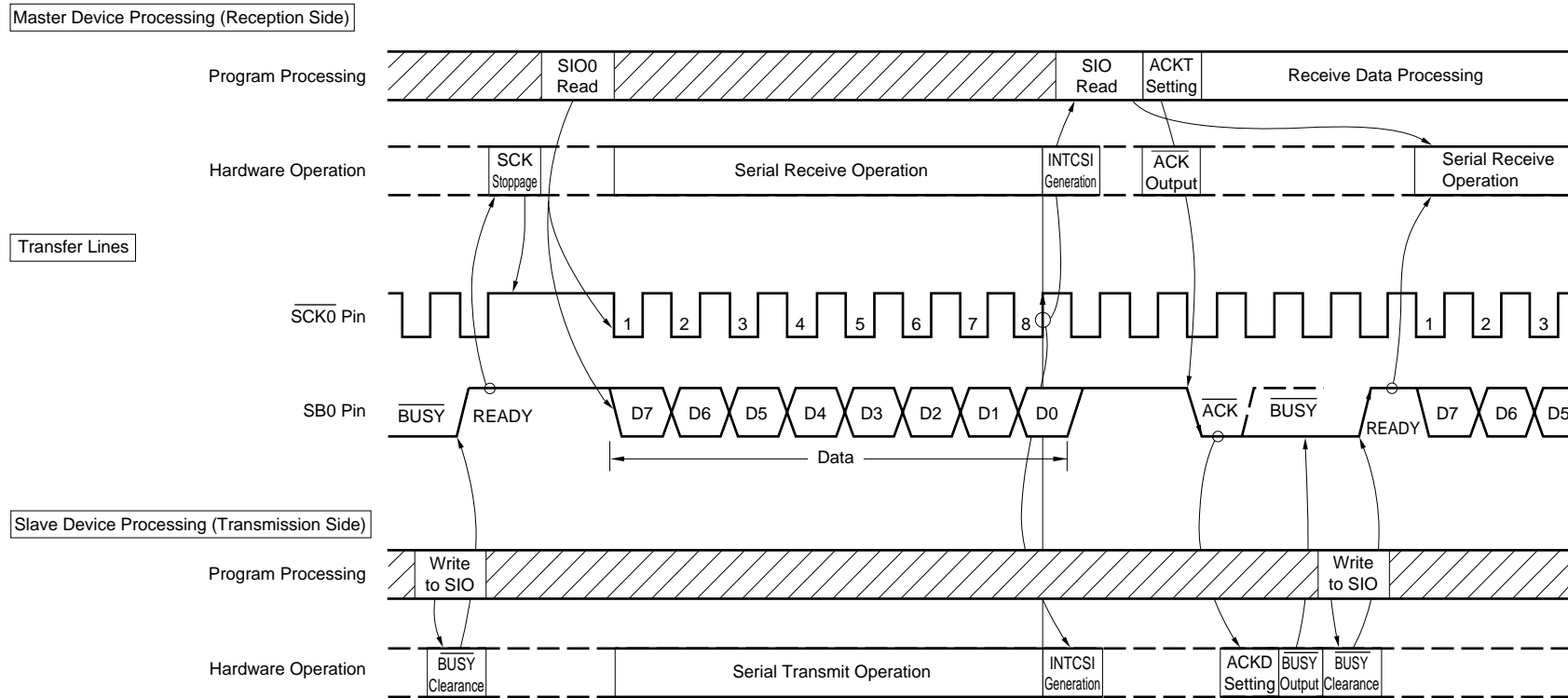
Figure 17-30 Data Transmission from Master Device to Slave Device



**Remark** This timing applies to the following conditions:

- Master: transmission only enabled
- Slave : reception only enabled, ACKE = 0, BSYE = 1

Figure 17-31 Data Transmission from Slave Device to Master Device



**Remark** This timing applies to the following conditions:

- Master: reception only enabled, ACKE = 0
- Slave : transmission only enabled, BSYE = 1

**17.6.9 Busy Release**

In consideration of high-speed transfers that also use the SBI macro service, busy release conditions vary according to the transmission/reception enable status.  $\overline{\text{BUSY}}$  release conditions are shown in Table 17-2.

**Table 17-2  $\overline{\text{BUSY}}$  Release Conditions**

Transmission/Reception Enable Status		$\overline{\text{BUSY}}$ Release Conditions
CTXE	CRXE	
0	0	None
0	1	BSYE $\leftarrow$ 0 or SIO read access
1	0	BSYE $\leftarrow$ 0 or SIO write access*
1	1	

\* If the next operation is a receive operation, FFH is used as the value written to shift register (SIO).

**17.6.10 Wake-Up Setting Operation**

Writing 1 to WUP in the busy state will set the wake-up state as soon as the READY state is established.

When the wake-up state is established, an interrupt (INTCSI) is generated only when an address is received. Also, acknowledge ( $\overline{\text{ACK}}$ ) signal detection is not performed in the wake-up state.

**17.6.11 Starting Transmit/Receive Operation**

The same method is used to start a transmit/receive operation as for busy release. If a transmit/receive operation start command is received while the slave is outputting a busy ( $\overline{\text{BUSY}}$ ) signal, the transmit/receive operation is held pending and is started when the busy state is released.

## 17.7 CAUTIONS

- (1) When changing from “CTXE = 0, CRXE = 1” to “CTXE = 1, CRXE = 0”, and when changing from “CTXE = 1, CRXE = 0” to “CTXE = 0, CRXE = 1”, ensure that this is not done with a single instruction, as this will result in misoperation of the serial clock counter, and the first communication after the change will finish in fewer than 8 bits. Instead, two instructions should be used as shown below.

**Example** To change “CTXE = 1, CRXE = 0” to “CTXE = 0, CRXE = 1”

```
CLR1 CTXE  
SET1 CRXE
```

- (2) When master/slave exchange processing is performed, since serial clock line ( $\overline{\text{SCK}}$ ) input/output switching is performed asynchronously between master and slave, a pull-up resistor is also required for the serial clock line ( $\overline{\text{SCK}}$ ).
- (3) ACKT must not be set before the end of a transfer.

[MEMO]

## CHAPTER 18 CLOCK OUTPUT FUNCTION

The  $\mu$ PD784026 has a clock function that outputs a signal scaled from the system clock.

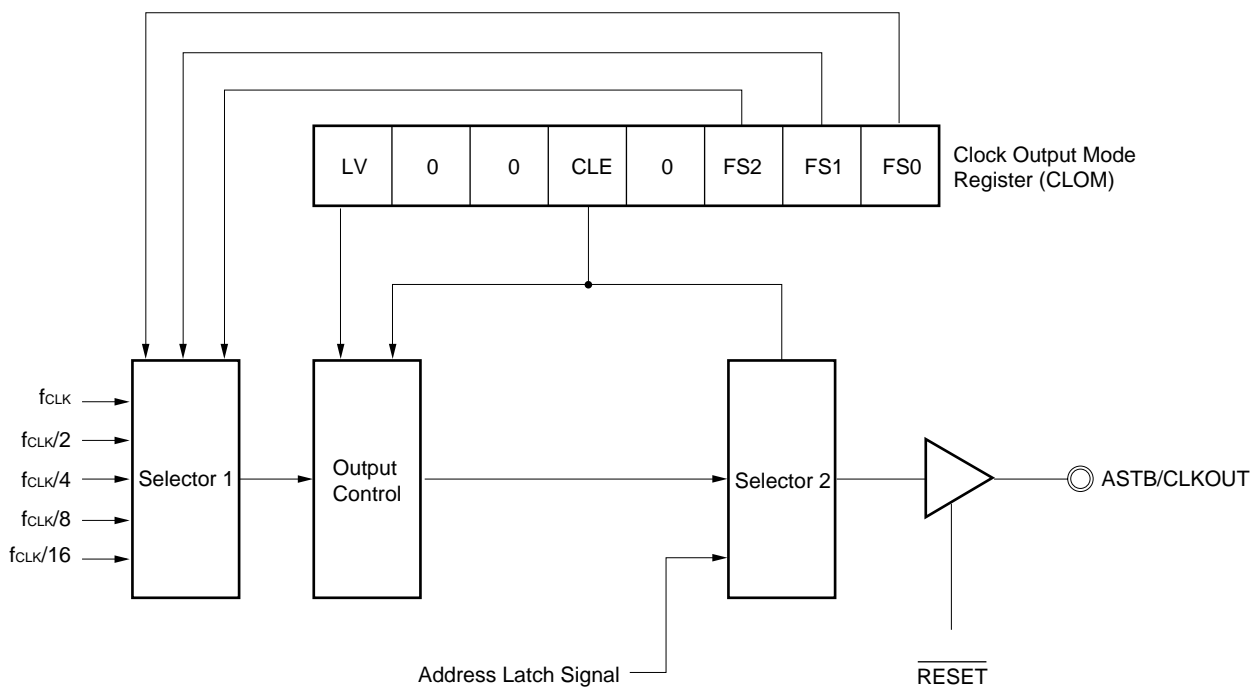
The clock output function can output the system clock directly, or a 1/2, 1/4, 1/8 or 1/16 system clock signal. In addition, it can be used as a 1-bit output port. The output pin has a dual function as the ASTB pin.

**Caution** This function cannot be used with the  $\mu$ PD784021, or when the external memory extension mode is used.

### 18.1 CONFIGURATION

The clock output function configuration is shown in Figure 18-1.

**Figure 18-1 Clock Output Function Configuration**



**(1) Clock output mode register (CLOM)**

Register that controls the operation of the clock output function.

**(2) Selector 1**

Selector that selects the frequency of the clock to be output.

**(3) Output control**

Controls the output signal in accordance with the contents of the clock output mode register (CLOM).

**(4) Selector 2**

Selects either the ASTB signal or the CLKOUT signal as the signal to be output to the ASTB/CLKOUT pin.

**(5) ASTB/CLKOUT pin**

Pin that outputs the signal selected by selector 2. While the  $\overline{\text{RESET}}$  input is low, the ASTB/CLKOUT pin is in the Hi-Z state, and when the  $\overline{\text{RESET}}$  input becomes high it outputs a low-level signal, and then outputs a signal according to the set function.



**18.2 CLOCK OUTPUT MODE REGISTER (CLOM)**

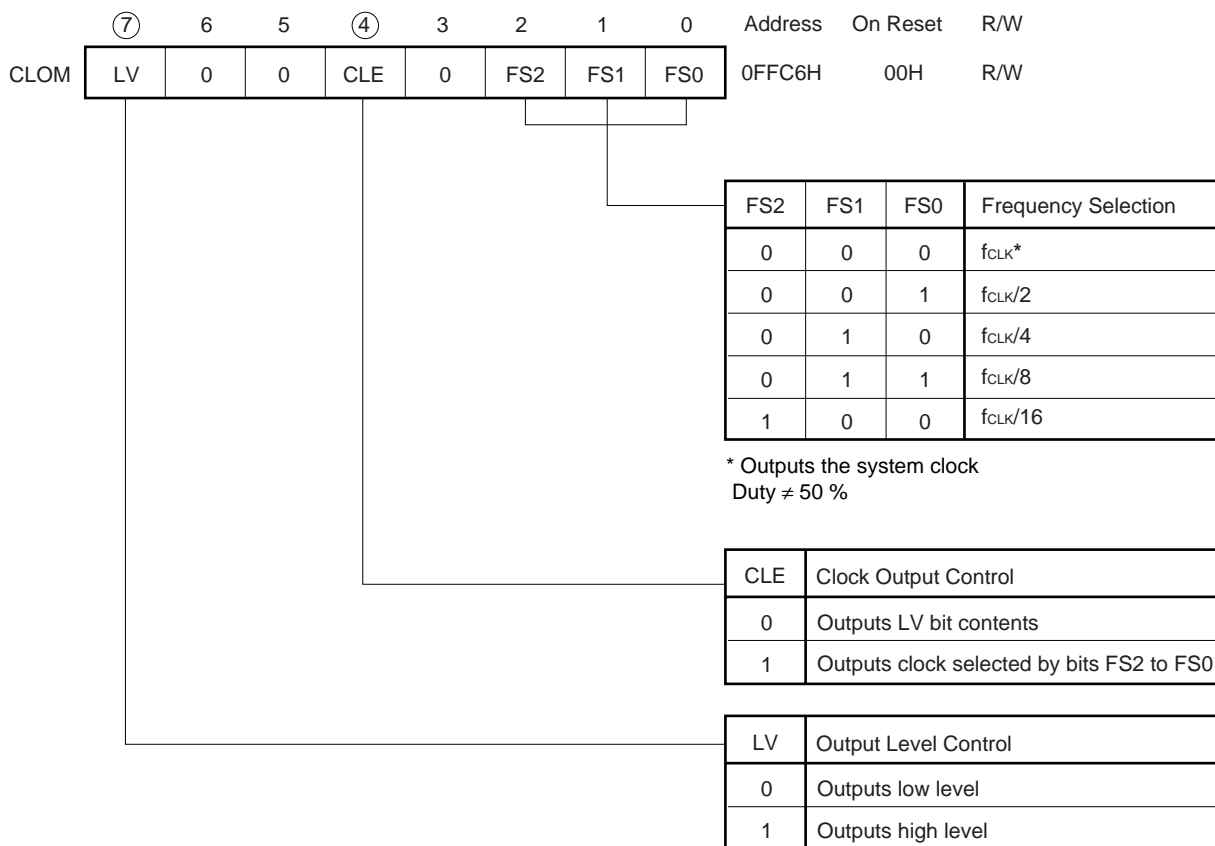
The CLOM controls the clock output function.

CLOM can be read or written to with an 8-bit manipulation instruction or bit manipulation instruction.

The CLOM format is shown in Figure 18-2.

RESET input clears the CLOM register to 00H.

**Figure 18-2 Clock Output Mode Register (CLOM) Format**



- Cautions**
1. With the  $\mu$ PD784021, and when the external memory extension mode is used, the clock output mode register (CLOM) should be set to 00H (value after RESET release).
  2. The other bits (FS0 to FS2 and LV) must not be changed while the CLE bit is set (1).
  3. The other bits (FS0 to FS2 and LV) must not be changed at the same time when the CLE bit is changed.

## 18.3 OPERATION

### 18.3.1 Clock Output

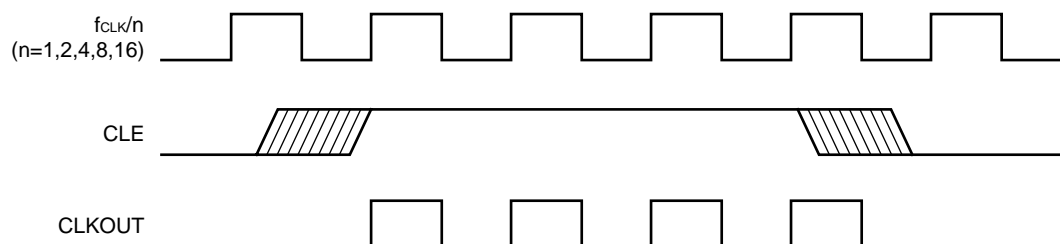
A signal with the clock output frequency selected by bits FS0 to FS2 is selected by selector 1 and output.

The output signal has the same level as the LV bit when the CLE bit is cleared (0), and is output from the clock signal immediately after the CLE bit is set (1).

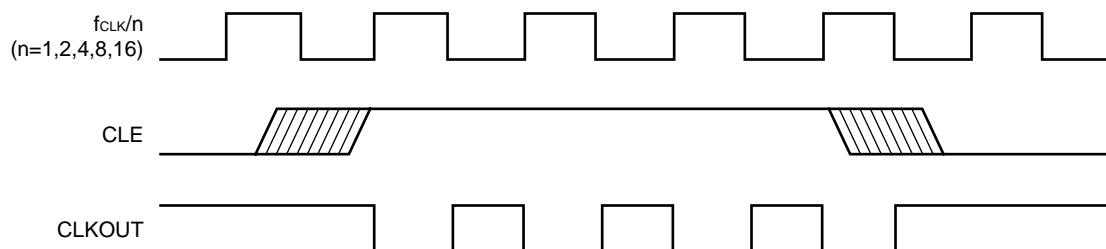
When the CLE bit is cleared (0), the contents of the LV bit are output in synchronization with the clock signal, and further output operations are stopped.

**Figure 18-3 Clock Output Operation Timing**

**(a) LV = 0**



**(b) LV = 1**



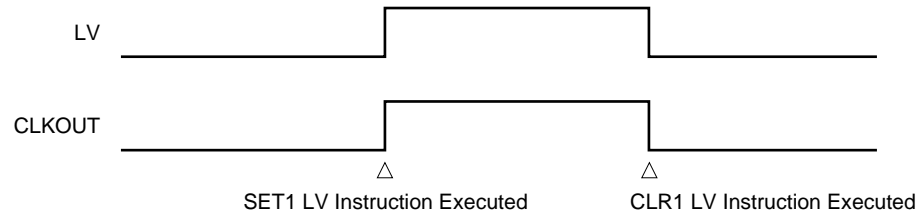
Setting of bits FS0 to FS2 and the LV bit should only be performed when CLE = 0 (bits FS0 to FS2 and the LV bit should not be changed within the same instruction that changes the CLE bit contents).

**<Operation Example>**

```
MOV CLOM, #82H ; CLKOUT pin: high level, clock output:  $f_{CLK}/4$ 
SET1 CLE       ; Starts clock output
...
CLR1 CLE       ; Stops clock output, CLKOUT pin: high level
```

**18.3.2 One-Bit Output Port**

When the CLE bit is cleared (0), the contents of the LV bit are output from the CLKOUT pin. The CLKOUT pin changes as soon as the contents of the LV bit change.

**Figure 18-4 One-Bit Output Port Operation****18.3.3 Operation in Standby Mode****(1) HALT mode**

The state prior to setting of the HALT mode is maintained. That is, if, during clock output, clock output has been performed continuously, and clock output has been disabled, the LV bit contents set before the HALT mode setting are output unchanged.

**(2) STOP mode and IDLE mode**

Clock output must be disabled before setting the STOP mode or IDLE mode (this must be done in software). The CLKOUT pin level output is the level before the STOP mode or IDLE mode was set (the contents of the LV bit).

**18.4 CAUTIONS**

- (1) This function cannot be used with the  $\mu$ PD784021, or when the external memory extension mode is used.
- (2) With the  $\mu$ PD784021, and when the external memory extension mode is used, the clock output mode register (CLOM) should be set to 00H (value after  $\overline{\text{RESET}}$  release).
- (3) The other bits (FS0 to FS2 and LV) must not be changed while the CLE bit is set (1).
- (4) The other bits (FS0 to FS2 and LV) must not be changed at the same time when the CLE bit is changed.

[MEMO]

## CHAPTER 19 EDGE DETECTION FUNCTION

P20 to P26 have an edge detection function that allows a rising edge/falling edge to be set programmably, and the detected edge is sent to internal hardware. The relation between pins P20 to P26 and the use of the detected edge is shown in Table 19-1.

**Table 19-1 Pins P20 to P26 and Use of Detected Edge**

Pin	Use	Detected Edge Specification Register
P20	NMI, standby circuit control	INTM0
P21	INTP0, timer/counter 1 capture signal timer/counter 1 count clock signal Real-time output port trigger signal	
P22	INTP1, timer/counter 2 CR22 capture signal	
P23	INTP2, CI (timer/counter 2 count clock signal), timer/counter 2 CR21 capture signal	
P24	INTP3, timer/counter 0 capture signal timer/counter 0 count clock signal	INTM1
P25	INTP4, standby circuit control	
P26	INTP5, A/D converter conversion start signal, standby circuit control	

The edge detection function operates at all times except in STOP mode and IDLE mode (although the edge detection function for pins P20, P25 and P26 also operates in STOP mode and IDLE mode).

For the P21/INTP0 pin, the noise elimination time when edge detection is performed can be selected by software.

### 19.1 EDGE DETECTION FUNCTION CONTROL REGISTERS

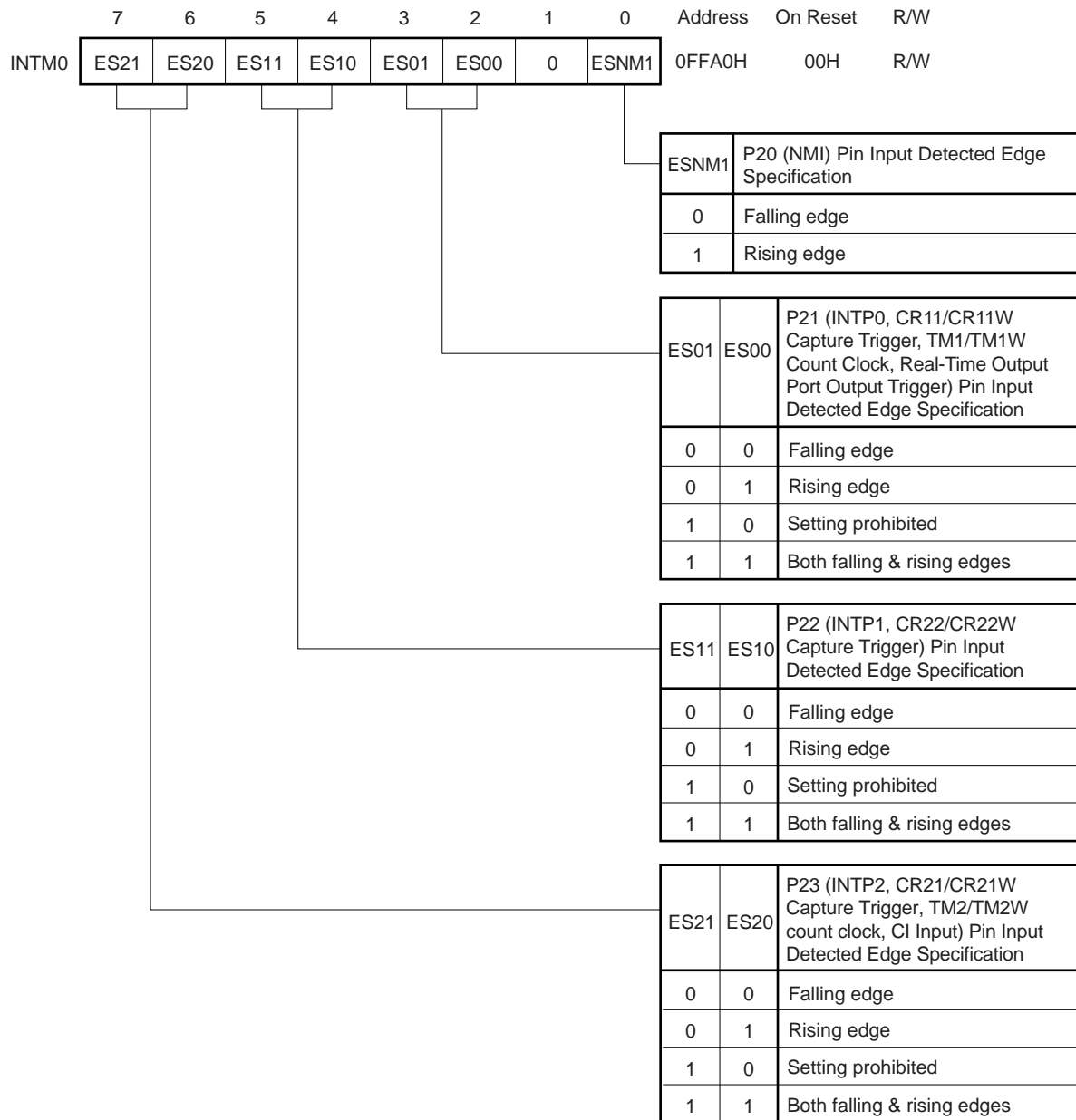
#### 19.1.1 External Interrupt Mode Registers (INTM0, INTM1)

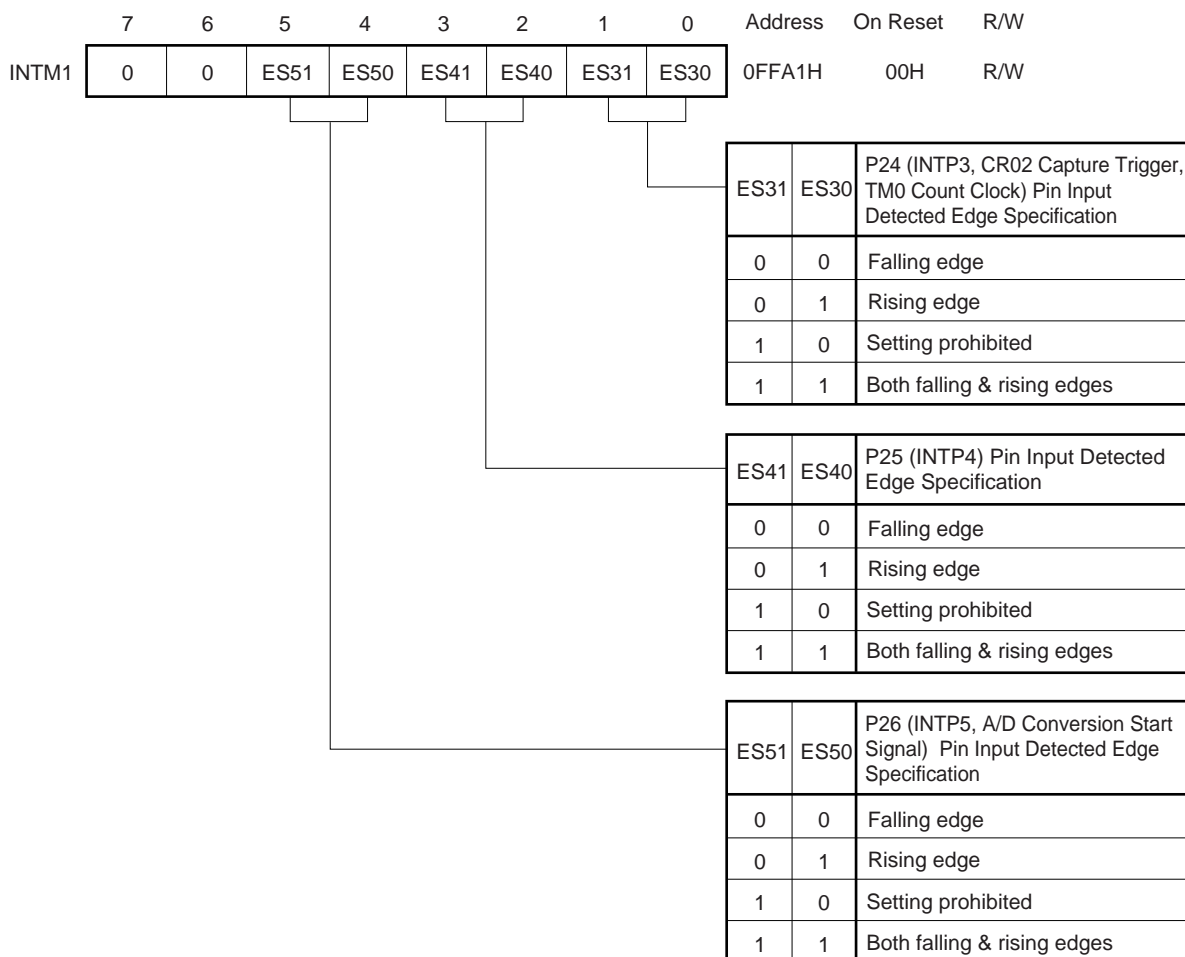
The INTM<sub>n</sub> (n = 0, 1) specify the valid edge to be detected on pins P20 to P26. The INTM0 specifies the valid edge for pins P20 to P23, and the INTM1 specifies the valid edge for pins P24 to P26.

The INTM<sub>n</sub> can be read or written to with an 8-bit manipulation instruction or bit manipulation instruction.

The format of INTM0 and INTM1 are shown in Figures 19-1 and 19-2 respectively.

RESET input clears these registers to 00H.

**Figure 19-1 External Interrupt Mode Register 0 (INTM0) Format**

**Figure 19-2 External Interrupt Mode Register 1 (INTM1) Format**

**Caution** Valid edge detection cannot be performed when the valid edge is changed by a write to the external interrupt mode register (INTMn: n = 0, 1). Also, if an edge is input during a change of the valid edge, that edge may or may not be judged to be a valid edge.

**19.1.2 Sampling Clock Selection Register (SCS0)**

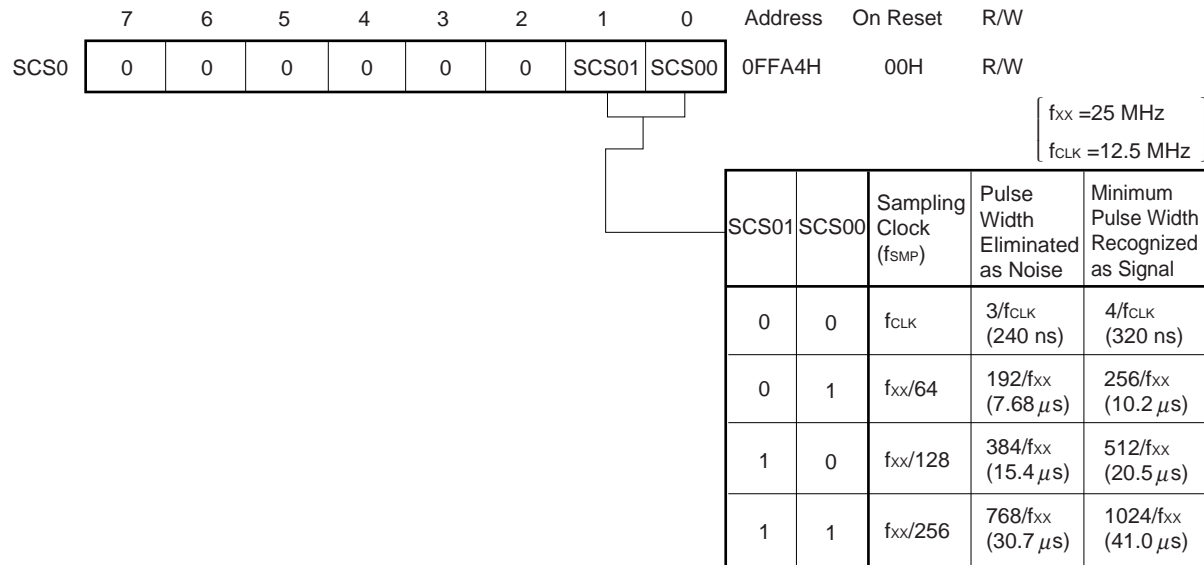
The SCS0 specifies the sampling clock ( $f_{SMP}$ ) for digital noise elimination performed on pin P21.

The SCS0 can be read or written to with an 8-bit manipulation instruction. The format of SCS0 is shown in Figure 19-

3.

$\overline{\text{RESET}}$  input clears the SCS0 register to 00H.

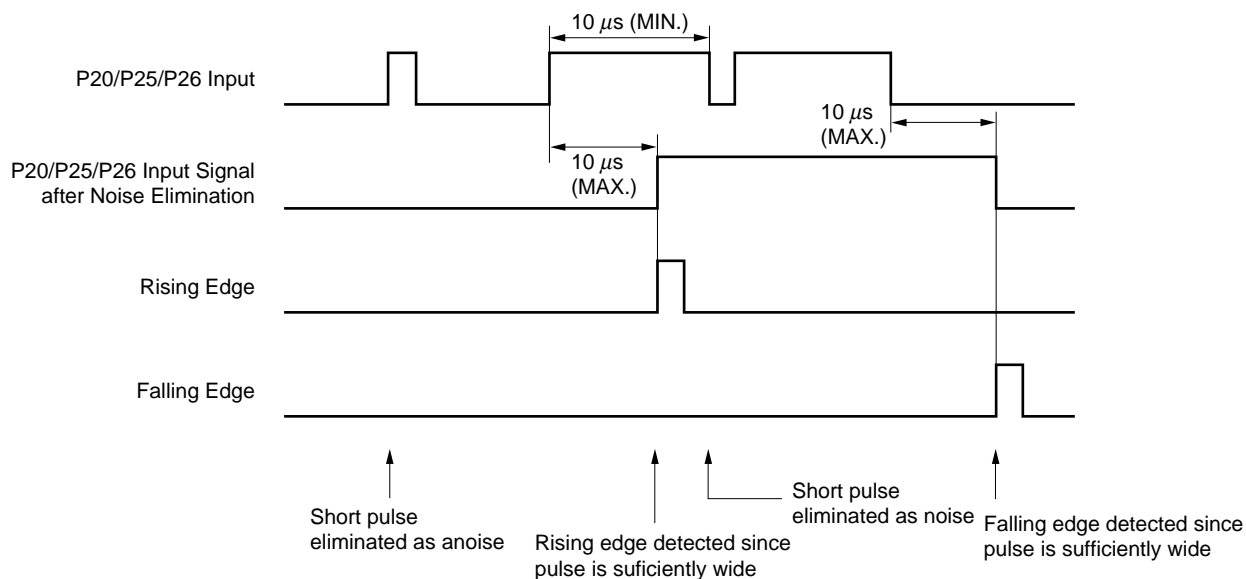
**Figure 19-3 Sampling Clock Selection Register (SCS0) Format**





**19.2 EDGE DETECTION FOR PINS P20, P25 AND P26**

On pins P20, P25 and P26, noise elimination is performed by means of analog delay before edge detection. Therefore, an edge cannot be detected unless the pulse width is a given time ( $10\ \mu\text{s}$ ) or longer.

**Figure 19-4 Edge Detection for Pins P20, P25 and P26**

**Caution** Since analog delay noise elimination is performed on pins P20, P25 and P26, an edge is detected up to  $10\ \mu\text{s}$  after it is actually input. Also, unlike pins P21 to P24, the delay before an edge is detected is not a specific value, because of differences in the characteristics of various devices.

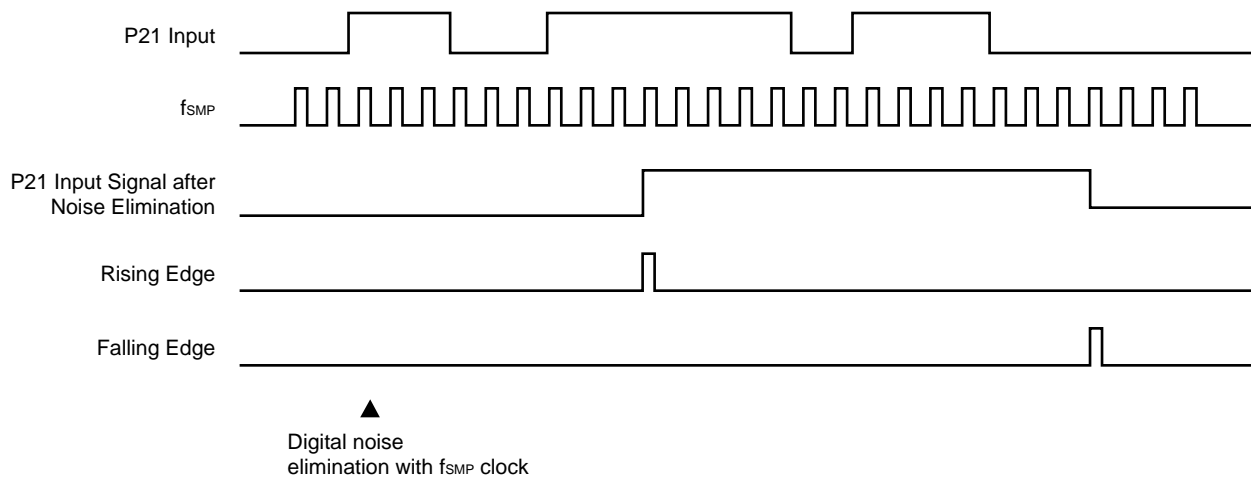
## 19.3 P21 PIN EDGE DETECTION

In P21 edge detection, digital noise elimination is performed using the clock ( $f_{SMP}$ ) specified by the sampling clock selection register (SCS0). In digital noise elimination, input is sampled using the  $f_{SMP}$  clock, and if the input level is not the same at least 4 times in succession (if it is the same only 3 or fewer times in succession), it is eliminated as noise. Therefore, the level must be maintained for at least 4  $f_{SMP}$  clock cycles in order to be recognized as a valid edge.

**Remark** When the pulse width of a signal with a comparatively long pulse width and a lot of noise, such as an infrared remote count reception signal, is measured, or when a signal is input in which oscillation occurs when an edge occurs, as with switch input chattering, for instance, it is better to set the sampling clock to low speed with the sampling clock selection register (SCS0). If the sampling clock is high-speed, there will be a reaction to the short-pulse noise components as well, and the program will frequently have to check whether the input is noise or a signal. However, by slowing down the sampling clock, reaction to short pulse width noise is eliminated and thus the program does not have to check so frequently, and can thus be simplified.

★

Figure 19-5 P21 Pin Edge Detection

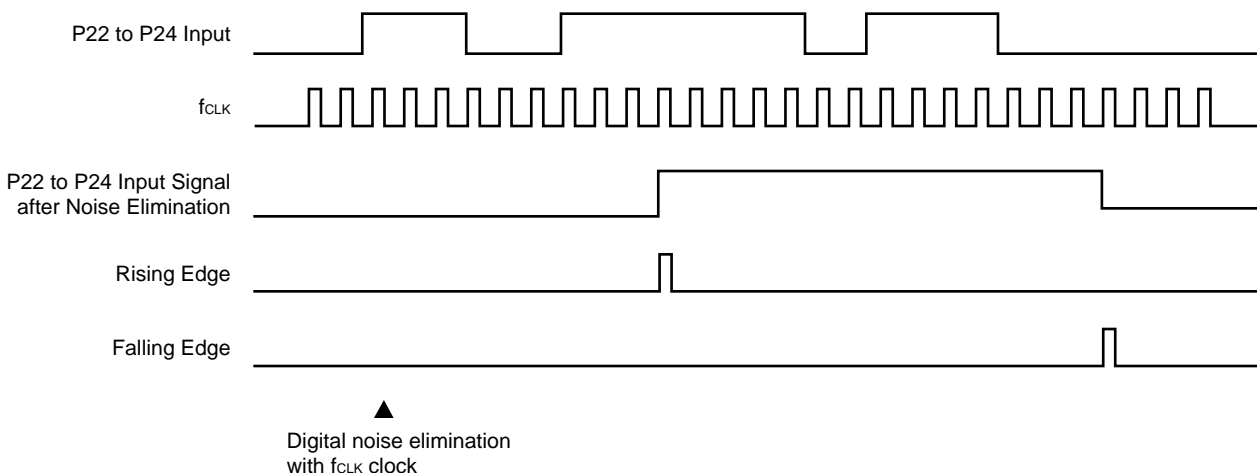


- Cautions**
1. Since digital noise elimination is performed with the  $f_{SMP}$  clock, there is a delay of 3 to 4  $f_{SMP}$  clocks between input of an edge to the pin and the point at which the edge is actually detected.
  2. If the input pulse width is 3 to 4  $f_{SMP}$  clocks, it is uncertain whether a valid edge will be detected. Therefore, to ensure reliable operation, the level should be held for at least 4 clocks.
  3. If noise input to the pin is synchronized with the  $f_{SMP}$  clock in the  $\mu$ PD784026, it may not be recognized as noise. If there is a possibility of such noise being input, noise should be eliminated by adding a filter to the input pin.

**19.4 PIN EDGE DETECTION FOR PINS P22 TO P24**

Edge detection for pins P22 to P24 is performed after digital noise elimination by means of clock sampling. Unlike the P21 pin,  $f_{CLK}$  is used as the sampling clock.

In digital noise elimination, input is sampled using the  $f_{CLK}$  clock, and if the input level is not the same at least 4 times in succession (if it is the same only 2 or fewer times in succession), it is eliminated as noise. Therefore, the level must be maintained for at least 4  $f_{CLK}$  clock cycles ( $0.32 \mu s$ :  $f_{CLK} = 12.5 \text{ MHz}$ ,  $f_{CLK} = 1/2 f_{XX}$ ,  $f_{XX} = 25 \text{ MHz}$ ) in order to be recognized as a valid edge.

**Figure 19-6 Edge Detection for Pins P22 to P24**

- Cautions**
1. Since digital noise elimination is performed with the  $f_{CLK}$  clock, there is a delay of 3 to 4  $f_{CLK}$  clocks between input of an edge to the pin and the point at which the edge is actually detected.
  2. If the input pulse width is 3 to 4  $f_{CLK}$  clocks, it is uncertain whether a valid edge will be detected. Therefore, to ensure reliable operation, the level should be held for at least 4 clocks.
  3. If noise input to a pin is synchronized with the  $f_{CLK}$  clock in the  $\mu PD784026$ , it may not be recognized as noise. If there is a possibility of such noise being input, noise should be eliminated by adding a filter to the input pins.

### 19.5 CAUTIONS

- (1) Valid edge detection cannot be performed when the valid edge is changed by a write to the external interrupt mode register (INTMn:  $n = 0, 1$ ). Also, if an edge is input during a change of the valid edge, that edge may or may not be judged to be a valid edge.
- (2) Since analog delay noise elimination is performed on pins P20, P25 and P26, an edge is detected up to 10  $\mu\text{s}$  after it is actually input. Also, unlike pins P21 to P24, the delay before an edge is detected is not a specific value, because of differences in the characteristics of various devices.
- (3) Since digital noise elimination is performed on the P21 pin with the  $f_{\text{SMP}}$  clock, there is a delay of 3 to 4  $f_{\text{SMP}}$  clocks between input of an edge to the pin and the point at which the edge is actually detected.
- (4) If the input pulse width on the P21 pin is 3 to 4  $f_{\text{SMP}}$  clocks, it is uncertain whether a valid edge will be detected. Therefore, to ensure reliable operation, the level should be held for at least 4 clocks.
- (5) If noise input of the P21 pin is synchronized with the  $f_{\text{SMP}}$  clock in the  $\mu\text{PD784026}$ , it may not be recognized as noise. If there is a possibility of such noise being input, noise should be eliminated by adding a filter to the input pins.
- (6) Since digital noise elimination is performed on pins P22 to P24 with the  $f_{\text{CLK}}$  clock, there is a delay of 3 to 4  $f_{\text{CLK}}$  clocks between input of an edge to the pin and the point at which the edge is actually detected.
- (7) If the input pulse width on pins P22 to P24 is 3 to 4  $f_{\text{CLK}}$  clocks, it is uncertain whether a valid edge will be detected. Therefore, to ensure reliable operation, the level should be held for at least 4 clocks.
- (8) If noise input to pins P22 to P24 is synchronized with the  $f_{\text{CLK}}$  clock in the  $\mu\text{PD784026}$ , it may not be recognized as noise. If there is a possibility of such noise being input, noise should be eliminated by adding a filter to the input pins.

## CHAPTER 20 INTERRUPT FUNCTIONS

The  $\mu$ PD784026 is provided with three interrupt request processing modes, vectored interrupt, context switching, and macro service (see **Table 20-1**). These three processing modes can be set as required in the program. However interrupt processing by macro service can only be selected for interrupt request sources provided with the macro service processing mode shown in Table 20-2. Context switching cannot be selected for non-maskable interrupts or operand error interrupts.

Multiple-interrupt control using 4 priority levels can easily be performed for maskable vectored interrupts.

**Table 20-1 Interrupt Request Service Modes**

Interrupt Request Processing Mode	Processing Performed	PC & PSW Contents	Service
Vectored interrupts	Software	Saving to & restoration from stack	Executed by branching to service program at address* specified by vector table
Context switching		Saving to & restoration from fixed area in register bank	Executed by automatic switching to register bank specified by vector table and branching to service program at address* specified by fixed area in register bank
Macro service	Hardware (firmware)	Retained	Execution of pre-set processing such as data transfers between memory and I/O

- \* The start addresses of all interrupt service programs must be in the base area. If the body of a service program cannot be located in the base area, a branch instruction to the service program should be written in the base area.

**20.1 INTERRUPT REQUEST SOURCES**

The  $\mu$ PD784026 has the 25 interrupt request sources shown in Table 20-2, with a vector table allocated to each.

**Table 20-2 Interrupt Request Sources (1/2)**

Type of Interrupt Request	Default Priority	Interrupt Request Generating Source	Generating Unit	Interrupt Control Register Name	Context Switching	Macro Service	Macro Service Control Word Address	Vector Table Address
Software	None	BRK instruction execution	—	—	Not possible	Not possible	—	3EH
		BRKCS instruction execution	—	—	Possible	Not possible	—	—
Operand error	None	Invalid operand in MOV STBC, #byte instruction or MOV WDM, #byte instruction, and LOCATION instruction	—	—	Not possible	Not possible	—	3CH
Non-maskable	None	NMI (pin input edge detection)	Edge detection	—	Not possible	Not possible	—	2H
		INTWDT (watchdog timer overflow)	Watchdog timer	—	Not possible	Not possible	—	4H

Table 20-2 Interrupt Request Sources (2/2)

Type of Interrupt Request	Default Priority	Interrupt Request Generating Source	Generating Unit	Interrupt Control Register Name	Context Switching	Macro Service	Macro Service Control Word Address	Vector Table Address
Maskable	0	INTP0 (pin input edge detection)	Edge detection	PIC0	Possible	Possible	0FE06H	6H
	1	INTP1 (pin input edge detection)		PIC1			0FE08H	8H
	2	INTP2 (pin input edge detection)		PIC2			0FE0AH	0AH
	3	INTP3 (pin input edge detection)		PIC3			0FE0CH	0CH
	4	INTC00 (TM0-CR00 match signal generation)	Timer/counter 0	CIC00			0FE0EH	0EH
	5	INTC01 (TM0-CR01 match signal generation)		CIC01			0FE10H	10H
	6	INTC10 (TM1-CR10 or TM1W-CR10W match signal generation)	Timer/counter 1	CIC10			0FE12H	12H
	7	INTC11 (TM1-CR11 or TM1W-CR11W match signal generation)		CIC11			0FE14H	14H
	8	INTC20 (TM2-CR20 or TM2W-CR20W match signal generation)	Timer/counter 2	CIC20			0FE16H	16H
	9	INTC21 (TM2-CR21 or TM2W-CR22W match signal generation)		CIC21			0FE18H	18H
	10	INTC30 (TM3-CR30 or TM3W-CR30W match signal generation)	Timer 3	CIC30			0FE1AH	1AH
	11	INTP4 (pin input edge detection)	Edge detection	PIC4			0FE1CH	1CH
	12	INTP5 (pin input edge detection)		PIC5			0FE1EH	1EH
	13	INTAD (A/D conversion end)	A/D converter	ADIC			0FE20H	20H
	14	INTSER (asynchronous serial interface receive error)	Asynchronous	SERIC		Not possible	0FE22H	22H
	15	INTSR (asynchronous serial interface reception end)	serial interface/ clocked serial	SRIC		Possible	0FE24H	24H
		INTCSI1 (clocked serial interface transfer end)		CSIIC1				
	16	INTST (asynchronous serial interface transmission end)	interface 1	STIC			0FE26H	26H
	17	INTCSI (clocked serial interface transfer end)	Clocked serial interface	CSIIC			0FE28H	28H
	18	INTSER2 (asynchronous serial interface 2 receive error)	Asynchronous	SERIC2		Not possible	0FE2AH	2AH
	19	INTSR2 (asynchronous serial interface 2 reception end)	serial interface 2/ clocked serial	SRIC2		Possible	0FE2CH	2CH
		INTCSI2 (clocked serial interface 2 transfer end)		CSIIC2				
	20	INTST2 (asynchronous serial interface 2 transmission end)	interface 2	STIC2			0FE2EH	2EH

- Remarks**
1. The default priority is a fixed number. This indicates the order of priority when interrupt requests specified as having the same priority are generated simultaneously.
  2. The INTSR and INTCSI1 interrupts are generated by the same hardware (they cannot both be used simultaneously). Therefore, although the same hardware is used for the interrupts, two names are provided, for use in each of the two modes. The same applies to INTSR2 and INTCSI2.

### 20.1.1 Software Interrupts

Interrupts by software consist of the BRK instruction which generates a vectored interrupt and the BRKCS instruction which performs context switching.

Software interrupts are acknowledged even in the interrupt disabled state, and are not subject to priority control.

### 20.1.2 Operand Error Interrupts

These interrupts are generated if there is an illegal operand in an MOV STBC, #byte instruction or MOV WDMC, #byte instruction, and LOCATION instruction.

Operand error interrupts are acknowledged even in the interrupt disabled state, and are not subject to priority control.

### 20.1.3 Non-Maskable Interrupts

A non-maskable interrupt is generated by NMI pin input or the watchdog timer.

Non-maskable interrupts are acknowledged unconditionally\*, even in the interrupt disabled state. They are not subject to interrupt priority control, and are of higher priority than any other interrupt.

- \* Except during execution of the service program for the same non-maskable interrupt, and during execution of the service program for a higher-priority non-maskable interrupt

### 20.1.4 Maskable Interrupts

A maskable interrupt is one subject to masking control according to the setting of an interrupt mask flag. In addition, acknowledgment enabling/disabling can be specified for all maskable interrupts by means of the IE flag in the program status word (PSW).

In addition to normal vectored interrupt, maskable interrupts can be acknowledged by context switching and macro service (though some interrupts cannot use macro service: see **Table 20-2**).

The priority order for maskable interrupt requests when interrupt requests of the same priority are generated simultaneously is predetermined (default priority) as shown in Table 20-2. Also, multiprocessing control can be performed with interrupt priorities divided into 4 levels. However, macro service requests are acknowledged without regard to priority control or the IE flag.



## 20.2 INTERRUPT PROCESSING MODES

There are three  $\mu$ PD784026 interrupt processing modes, as follows:

- Vectored interrupt processing
- Macro service
- Context switching

### 20.2.1 Vectored Interrupt Processing

When an interrupt is acknowledged, the program counter (PC) and program status word (PSW) are automatically saved to the stack, a branch is made to the address indicated by the data stored in the vector table, and the interrupt processing routine is executed.

### 20.2.2 Macro Service

When an interrupt is acknowledged, CPU execution is temporarily suspended and a data transfer is performed by hardware. Since macro service is performed without the intermediation of the CPU, it is not necessary to save or restore CPU statuses such as the program counter (PC) and program status word (PSW) contents. This is therefore very effective in improving the CPU service time (See **20.8 Macro Service Function**).

### 20.2.3 Context Switching

When an interrupt is acknowledged, the prescribed register bank is selected by hardware, a branch is made to a pre-set vector address in the register bank, and at the same time the current program counter (PC) and program status word (PSW) are saved in the register bank (see **20.4.2 BRKCS Instruction Software Interrupt (Software Context Switching) Acknowledgment Operation** and **20.7.2 Context Switching**).

**Remark** “Context” refers to the CPU registers that can be accessed by a program while that program is being executed. These registers include general registers, the program counter (PC), program status word (PSW), and stack pointer (SP).

**20.3 INTERRUPT PROCESSING CONTROL REGISTERS**

$\mu$ PD784026 interrupt processing is controlled for each interrupt request by various control registers that perform interrupt processing specification. The interrupt control registers are listed in Table 20-3.

**Table 20-3 Control Registers**

Register Name	Symbol	Function
Interrupt control registers	PIC0 PIC1 PIC2 PIC3 CIC00 CIC01 CIC10 CIC11 CIC20 CIC21 CIC30 PIC4 PIC5 ADIC SERIC SRIC CSIC1 STIC CSIC SERIC2 SRIC2 CSIC2 STIC2	Registers that perform each interrupt request generation recording, mask control, vectored interrupt processing or macro service processing specification, context switching function enabling/disabling, and priority specification.
Interrupt mask registers	MK0 MK1L	Maskable interrupt request mask control Linked to mask control flags in interrupt control registers Word accesses or byte accesses possible
In-service priority register	ISPR	Records priority of interrupt request currently being acknowledged
Interrupt mode control register	IMC	Controls nesting of maskable interrupts for which lowest priority level (level 3) is specified
Watchdog timer mode register	WDM	Specifies priority of interrupts due to NMI pin input and interrupts due to watchdog timer overflow
Program status word	PSW	Specifies enabling/disabling of maskable interrupt acknowledgment

An interrupt control register is allocated to each interrupt source. The flags of each register perform control of the contents corresponding to the relevant bit position in the register. The interrupt control register flag names corresponding to each interrupt request signal are shown in Table 20-4.

**Table 20-4 Interrupt Control Register Flags Corresponding to Interrupt Requests**

Default Priority	Interrupt Request Signal	Interrupt Control Registers					
			Interrupt Request Flag	Interrupt Mask Flag	Macro Service Enable Flag	Priority Specification Flag	Context Switching Enable Flag
0	INTP0	PIC0	PIF0	PMK0	PISM0	PPR00 PPR01	PCSE0
1	INTP1	PIC1	PIF1	PMK1	PISM1	PPR10 PPR11	PCSE1
2	INTP2	PIC2	PIF2	PMK2	PISM2	PPR20 PPR21	PCSE2
3	INTP3	PIC3	PIF3	PMK3	PISM3	PPR30 PPR31	PCSE3
4	INTC00	CIC00	CIF00	CMK00	CISM00	CPR000 CPR001	CCSE00
5	INTC01	CIC01	CIF01	CMK01	CISM01	CPR010 CPR011	CCSE01
6	INTC10	CIC10	CIF10	CMK10	CISM10	CPR100 CPR101	CCSE10
7	INTC11	CIC11	CIF11	CMK11	CISM11	CPR110 CPR111	CCSE11
8	INTC20	CIC20	CIF20	CMK20	CISM20	CPR200 CPR201	CCSE20
9	INTC21	CIC21	CIF21	CMK21	CISM21	CPR210 CPR211	CCSE21
10	INTC30	CIC30	CIF30	CMK30	CISM30	CPR300 CPR301	CCSE30
11	INTP4	PIC4	PIF4	PMK4	PISM4	PPR40 PPR41	PCSE4
12	INTP5	PIC5	PIF5	PMK5	PISM5	PPR50 PPR51	PCSE5
13	INTAD	ADIC	ADIF	ADMK	ADISM	ADPR0 ADPR1	ADCSE
14	INTSER	SERIC	SERIF	SERMK	—	SERPR0 SERPR1	SERCSE
15	INTSR	SRIC	SRIF	SRMK	SRISM	SRPR0 SRPR1	SRCSE
	INTCSI1	CSIIC1	CSIIF1	CSIMK1	CSIISM1	CSIPR10 CSIPR11	CSICSE1
16	INTST	STIC	STIF	STMK	STISM	STPR0 STPR1	STCSE
17	INTCSI	CSIIC	CSIIF	CSIMK	CSIISM	CSIPR0 CSIPR1	CSICSE
18	INTSER2	SERIC2	SERIF2	SERMK2	—	SERPR20 SERPR21	SERCSE2
19	INTSR2	SRIC2	SRIF2	SRMK2	SRISM2	SRPR20 SRPR21	SRCSE2
	INTCSI2	CSIIC2	CSIIF2	CSIMK2	CSIISM2	CSIPR20 CSIPR21	CSICSE2
20	INTST2	STIC2	STIF2	STMK2	STISM2	STPR20 STPR21	STCSE2

### 20.3.1 Interrupt Control Registers

An interrupt control register is allocated to each interrupt source, and performs priority control, mask control, etc. for the corresponding interrupt request. The interrupt control register format is shown in Figure 20-1.

#### (1) Priority specification flags (××PR1/××PR0)

The priority specification flags specify the priority on an individual interrupt source basis for the 21 maskable interrupts. Up to 4 priority levels can be specified, and a number of interrupt sources can be specified at the same level. Among maskable interrupt sources, level 0 is the highest priority.

If multiple interrupt requests are generated simultaneously among interrupt source of the same priority level, they are acknowledged in default priority order.

These flags can be manipulated bit-wise in software.

$\overline{\text{RESET}}$  input sets all bits to “1”.

#### (2) Context switching enable flag (××CSE)

The context switching enable flag specifies that a maskable interrupt request is to be processed by context switching. In context switching, the register bank specified beforehand is selected by hardware, a branch is made to a vector address stored beforehand in the register bank, and at the same time the current contents of the program counter (PC) and program status word (PSW) are saved in the register bank.

Context switching is suitable for real-time processing, since execution of interrupt processing can be started faster than with normal vectored interrupt processing.

This flag can be manipulated bit-wise in software.

$\overline{\text{RESET}}$  input sets all bits to “0”.

#### (3) Macro service enable flag (××ISM)

The macro service enable flag specifies whether an interrupt request corresponding to that flag is to be handled by vectored interrupt or context switching, or by macro service.

When macro service processing is selected, at the end of the macro service (when the macro service counter reaches 0) the macro service enable flag is automatically cleared (0) by hardware (vectored interrupt processing/context switching processing).

This flag can be manipulated bit-wise in software.

$\overline{\text{RESET}}$  input sets all bits to “0”.

**(4) Interrupt mask flag (××MK)**

An interrupt mask flag specifies enabling/disabling of vectored interrupt processing and macro service processing for the interrupt request corresponding to that flag.

The interrupt mask contents are not changed by the start of interrupt processing, etc., and are the same as the interrupt mask register contents (see **20.3.2 Interrupt Mask Registers (MK0/MK1L)**).

Macro service processing requests are also subject to mask control, and macro service requests can also be masked with this flag.

This flag can be manipulated in software.

$\overline{\text{RESET}}$  input sets all bits to “1”.

**(5) Interrupt request flag (××IF)**

An interrupt request flag is set (1) by generation of the interrupt request that corresponds to that flag. When the interrupt is acknowledged, the flag is automatically cleared (0) by hardware.

This flag can be manipulated in software.

$\overline{\text{RESET}}$  input sets all bits to “0”.

**Figure 20-1 Interrupt Control Registers (××ICn) (1/3)**

	⑦	⑥	⑤	④	3	2	1	0	Address	On Reset	R/W
PIC0	PIF0	PMK0	PISM0	PCSE0	0	0	PPR01	PPR00	0FFE0H	43H	R/W
PIC1	PIF1	PMK1	PISM1	PCSE1	0	0	PPR11	PPR10	0FFE1H	43H	R/W
PIC2	PIF2	PMK2	PISM2	PCSE2	0	0	PPR21	PPR20	0FFE2H	43H	R/W
PIC3	PIF3	PMK3	PISM3	PCSE3	0	0	PPR31	PPR30	0FFE3H	43H	R/W
CIC00	CIF00	CMK00	CISM00	CCSE00	0	0	CPR001	CPR000	0FFE4H	43H	R/W
CIC01	CIF01	CMK01	CISM01	CCSE01	0	0	CPR011	CPR010	0FFE5H	43H	R/W
CIC10	CIF10	CMK10	CISM10	CCSE10	0	0	CPR101	CPR100	0FFE6H	43H	R/W
CIC11	CIF11	CMK11	CISM11	CCSE11	0	0	CPR111	CPR110	0FFE7H	43H	R/W

××PRn1 (Bit 1)	××PRn0 (Bit 0)	Interrupt Request Priority Specification
0	0	Priority 0 (highest priority)
0	1	Priority 1
1	0	Priority 2
1	1	Priority 3

××CSEn (Bit 4)	Context Switching Processing Specification
0	Processed by vectored interrupt
1	Processed by context switching

××ISMn (Bit 5)	Interrupt Processing Mode Specification
0	Vectored interrupt processing/ context switching processing
1	Macro service processing

××MKn (Bit 6)	Interrupt Processing Enabling/Disabling
0	Interrupt processing enabled
1	Interrupt processing disabled

××IFn (Bit 7)	Interrupt Request Generation Presence/Absence
0	No interrupt request (interrupt signal not being generated)
1	Interrupt request state (interrupt signal being generated)

**Figure 20-1 Interrupt Control Registers (××ICn) (2/3)**

	⑦	⑥	⑤	④	3	2	1	0	Address	On Reset	R/W
CIC20	CIF20	CMK20	CISM20	CCSE20	0	0	CPR201	CPR200	0FFE8H	43H	R/W
CIC21	CIF21	CMK21	CISM21	CCSE21	0	0	CPR211	CPR210	0FFE9H	43H	R/W
CIC30	CIF30	CMK30	CISM30	CCSE30	0	0	CPR301	CPR300	0FFEAH	43H	R/W
PIC4	PIF4	PMK4	PISM4	PCSE4	0	0	PPR41	PPR40	0FEBH	43H	R/W
PIC5	PIF5	PMK5	PISM5	PCSE5	0	0	PPR51	PPR50	0FECH	43H	R/W
ADIC	ADIF	ADMK	ADISM	ADCSE	0	0	ADPR1	ADPR0	0FFEDH	43H	R/W
SERIC	SERIF	SERMK	0	SERCSE	0	0	SERPR1	SERPR0	0FEEH	43H	R/W
SRIC	SRIF	SRMK	SRISM	SRCSE	0	0	SRPR1	SRPR0	0FEFH	43H	R/W

××PRn1 (Bit 1)	××PRn0 (Bit 0)	Interrupt Request Priority Specification
0	0	Priority 0 (highest priority)
0	1	Priority 1
1	0	Priority 2
1	1	Priority 3

××CSEn (Bit 4)	Context Switching Processing Specification
0	Processed by vectored interrupt
1	Processed by context switching

××ISMn (Bit 5)	Interrupt Processing Mode Specification
0	Vectored interrupt processing/ context switching processing
1	Macro service processing

××MKn (Bit 6)	Interrupt Processing Enabling/Disabling
0	Interrupt processing enabled
1	Interrupt processing disabled

××IFn (Bit 7)	Interrupt Request Generation Presence/Absence
0	No interrupt request (interrupt signal not being generated)
1	Interrupt request state (interrupt signal being generated)



**Figure 20-1 Interrupt Control Registers (××ICn) (3/3)**

	⑦	⑥	⑤	④	3	2	1	0	Address	On Reset	R/W
CSIIC1	CSIF1	CSIMK1	CSIISM1	CSICSE1	0	0	CSIPR11	CSIPR10	0FFEFDH	43H	R/W
STIC	STIF	STMK	STISM	STCSE	0	0	STPR1	STPR0	0FFF0H	43H	R/W
CSIIC	CSIF	CSIMK	CSIISM	CSICSE	0	0	CSIPR1	CSIPR0	0FFF1H	43H	R/W
SERIC2	SERIF2	SERMK2	0	SERCSE2	0	0	SERPR21	SERPR20	0FFF2H	43H	R/W
SRIC2	SRIF2	SRMK2	SRISM2	SRCSE2	0	0	SRPR21	SRPR20	0FFF3H	43H	R/W
CSIIC2	CSIF2	CSIMK2	CSIISM2	CSICSE2	0	0	CSIPR21	CSIPR20	0FFF3H	43H	R/W
STIC2	STIF2	STMK2	STISM2	STCSE2	0	0	STPR21	STPR20	0FFF4H	43H	R/W

××PRn1 (Bit 1)	××PRn0 (Bit 0)	Interrupt Request Priority Specification
0	0	Priority 0 (highest priority)
0	1	Priority 1
1	0	Priority 2
1	1	Priority 3

××CSEn (Bit 4)	Context Switching Processing Specification
0	Processed by vectored interrupt
1	Processed by context switching

××ISMn (Bit 5)	Interrupt Processing Mode Specification
0	Vectored interrupt processing/ context switching processing
1	Macro service processing

××MKn (Bit 6)	Interrupt Processing Enabling/Disabling
0	Interrupt processing enabled
1	Interrupt processing disabled

××IFn (Bit 7)	Interrupt Request Generation Presence/Absence
0	No interrupt request (interrupt signal not being generated)
1	Interrupt request state (interrupt signal being generated)

**20.3.2 Interrupt Mask Registers (MK0/MK1L)**

The MK0 and MK1L are composed of interrupt mask flags. MK0 is a 16-bit register which can be manipulated as 8-bit units, MK0L and MK0H, as well as being manipulated as a 16-bit unit.

MK1L is an 8-bit register that can be manipulated as an 8-bit unit. In addition, each bit of the MK0 and MK1L can be manipulated individually with a bit manipulation instruction. Each interrupt mask flag controls enabling/disabling of the corresponding interrupt request.

When an interrupt mask flag is set (1), acknowledgment of the corresponding interrupt request is disabled.

When an interrupt mask flag is cleared (0), the corresponding interrupt request can be acknowledged as a vectored interrupt or macro service request.

Each interrupt mask flag in the MK0 and MK1L is the same flag as the interrupt mask flag in the interrupt control register. The MK0 and MK1L are provided for en bloc control of interrupt masking.

After RESET input, the MK0 is set to FFFFH, the MK1L is set to FFH, and all maskable interrupts are disabled.

**Figure 20-2 Interrupt Mask Register (MK0, MK1L) Format (1/2)**

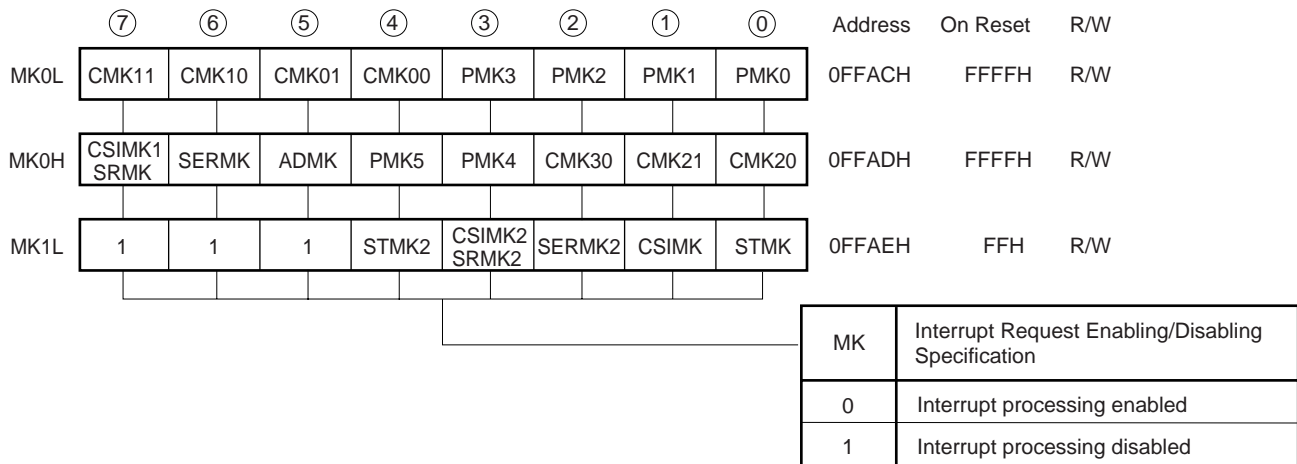
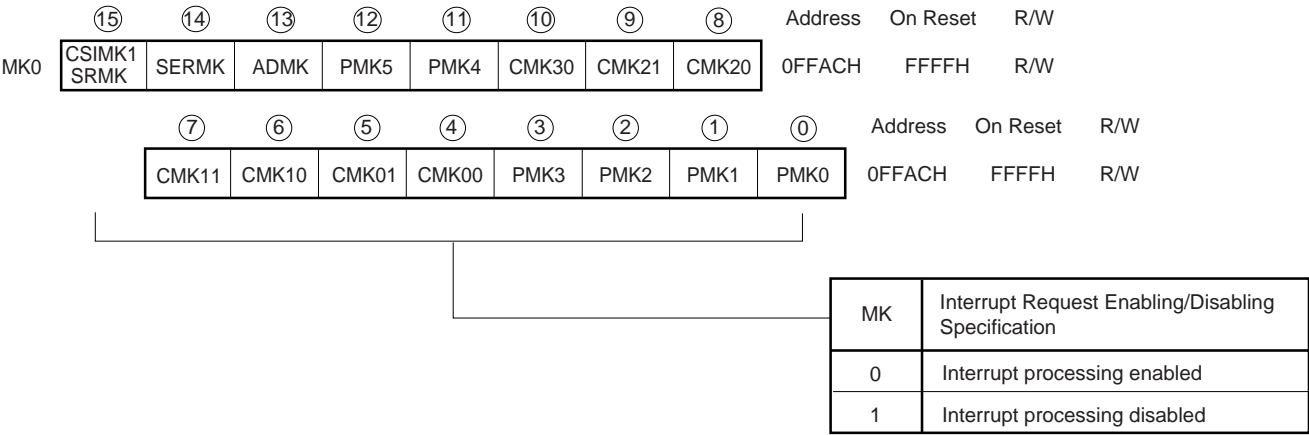
**(1) Byte Accesses**

Figure 20-2 Interrupt Mask Register (MK0, MK1L) Format (2/2)

(2) Word Accesses



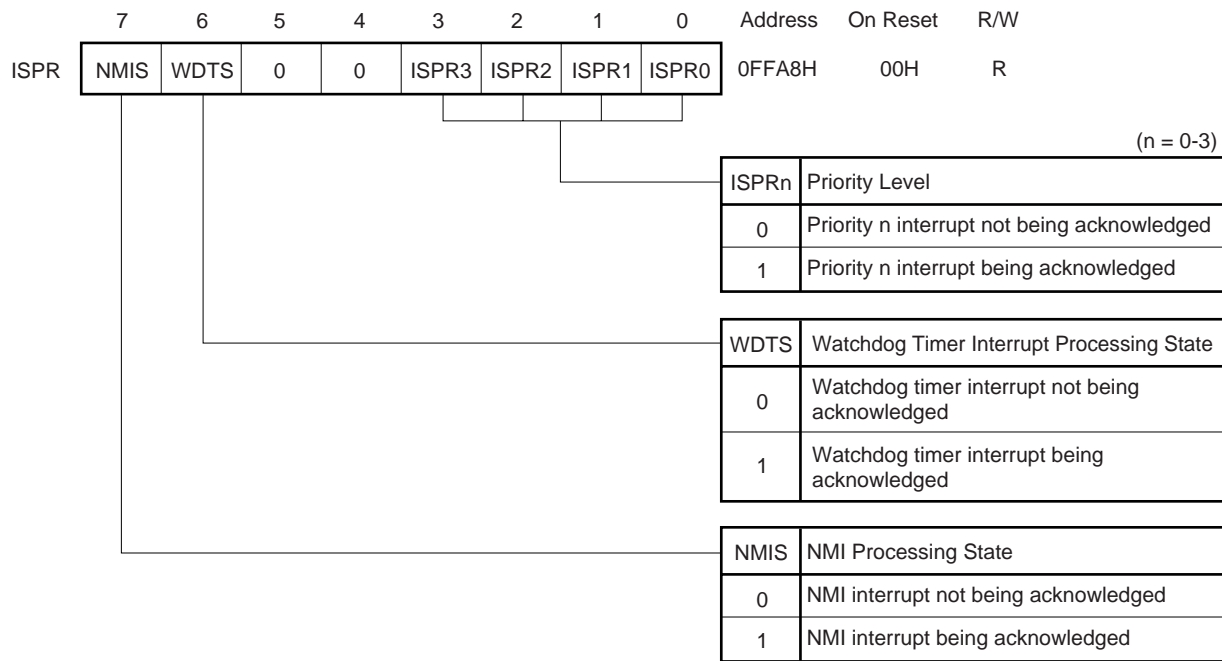
**20.3.3 In-Service Priority Register (ISPR)**

The ISPR shows the priority level of the maskable interrupt currently being processed and the non-maskable interrupt being processed. When a maskable interrupt request is acknowledged, the bit corresponding to the priority of that interrupt request is set (1), and remains set until the service program ends. When a non-maskable interrupt is acknowledged, the bit corresponding to the priority of that non-maskable interrupt is set (1), and remains set until the service program ends.

When an RETI instruction or RETCS instruction is executed, the bit, among those set (1) in the ISPR, that corresponds to the highest-priority interrupt request is automatically cleared (0) by hardware.

The contents of the ISPR are not changed by execution of an RETB or RETCSB instruction.

RESET input clears the ISPR register to 00H.

**Figure 20-3 In-Service Priority Register (ISPR) Format**

**Caution** In-service priority register (ISPR) is a read-only register. There is a risk of misoperation if a write is performed on this register.

**20.3.4 Interrupt Mode Control Register (IMC)**

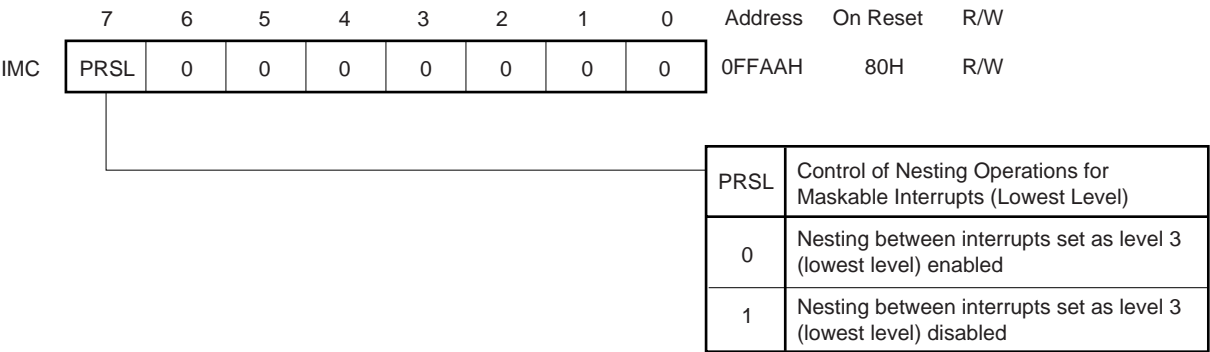
The IMC contains the PRSL flag. The PRSL flag specifies enabling/disabling of nesting of maskable interrupts for which the lowest priority level (level 3) is specified.

When the IMC is manipulated, the interrupt disabled state (DI state) should be set first to prevent misoperation.

The IMC can be read or written to with an 8-bit manipulation instruction or bit manipulation instruction.

$\overline{\text{RESET}}$  input sets the IMC register to 80H.

**Figure 20-4 Interrupt Mode Control Register (IMC) Format**



**20.3.5 Watchdog Timer Mode Register (WDM)**

The PRC bit of the WDM specifies the priority of NMI pin input non-maskable interrupts and watchdog timer overflow non-maskable interrupts.

The WDM can be written to only by a dedicated instruction. This dedicated instruction, MOV WDM, #byte, has a special code configuration (4 bytes), and a write is not performed unless the 3rd and 4th bytes of the operation code are mutual

★ 1's complements.

★ If the 3rd and 4th bytes of the operation code are not 1's complements, a write is not performed and an operand error interrupt is generated. In this case, the return address saved in the stack area is the address of the instruction that was the source of the error, and thus the address that was the source of the error can be identified from the return address saved in the stack area.

If recovery from an operand error is simply performed by means of an RETB instruction, an endless loop will result.

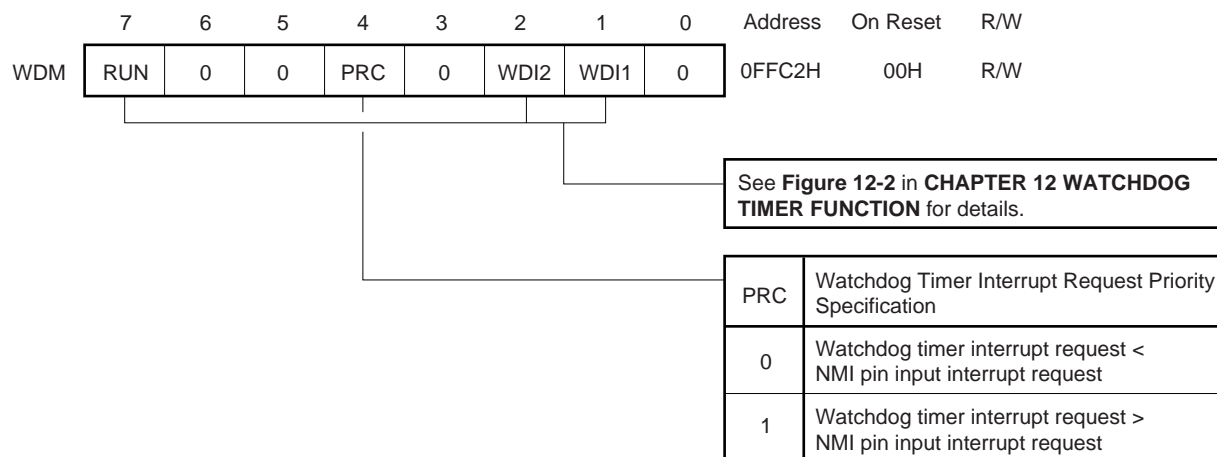
As an operand error interrupt is only generated in the event of an inadvertent program loop (with the NEC assembler, RA78K4, only the correct dedicated instruction is generated when MOV WDM, #byte is written), system initialization should be performed by the program.

Other write instructions (MOV WDM, A; AND WDM, #byte; SET1 WDM.7, etc.) are ignored and do not perform any operation. That is, a write is not performed to the WDM, and an interrupt such as an operand error interrupt is not generated.

The WDM can be read at any time by a data transfer instruction.

RESET input clears the WDM register to 00H.

**Figure 20-5 Watchdog Timer Mode Register (WDM) Format**



**Caution** The watchdog timer mode register (WDM) can only be written to with a dedicated instruction (MOV WDM, #byte).

**20.3.6 Program Status Word (PSW)**

The PSW is a register that holds the current status regarding instruction execution results and interrupt requests. The IE flag that sets enabling/disabling of maskable interrupts is mapped in the low-order 8 bits of the PSW (PSWL).

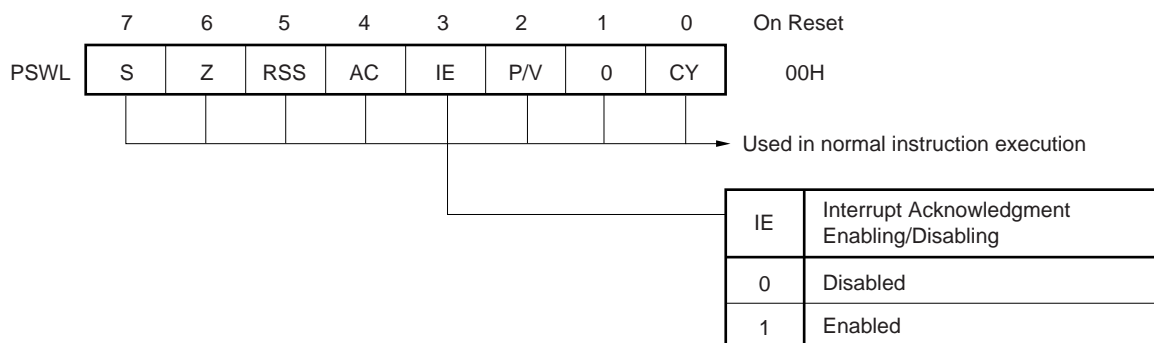
PSWL can be read or written to with an 8-bit manipulation instruction, and can also be manipulated with a bit manipulation instruction or dedicated instruction (EI/DI).

When a vectored interrupt is acknowledged or a BRK instruction is executed, PSWL is saved to the stack and the IE flag is cleared (0). PSWL is also saved to the stack by the PUSH PSW instruction, and is restored from the stack by the RETI, RETB and POP PSW instructions.

When context switching or a BRKCS instruction is executed, PSWL is saved to a fixed area in the register bank, and the IE flag is cleared (0). PSWL is restored from the fixed area in the register bank by an RETCSI or RETCSB instruction.

RESET input clears PSWL to 80H.

**Figure 20-6 Program Status Word (PSWL) Format**

**20.4 SOFTWARE INTERRUPT ACKNOWLEDGMENT OPERATIONS**

A software interrupt is acknowledged in response to execution of a BRK or BRKCS instruction. Software interrupts cannot be disabled.

**20.4.1 BRK Instruction Software Interrupt Acknowledgment Operation**

When a BRK instruction is executed, the program status word (PSW), program counter (PC) are saved in that order to the stack, the IE flag is cleared (0), the vector table (003EH/003FH) contents are loaded into the low-order 16 bits of the PC, and 0000B into the high-order 4 bits, and a branch is performed (the start of the service program must be in the base area).

The RETB instruction must be used to return from a BRK instruction software interrupt.

**Caution** The RETI instruction must not be used to return from a BRK instruction software interrupt.

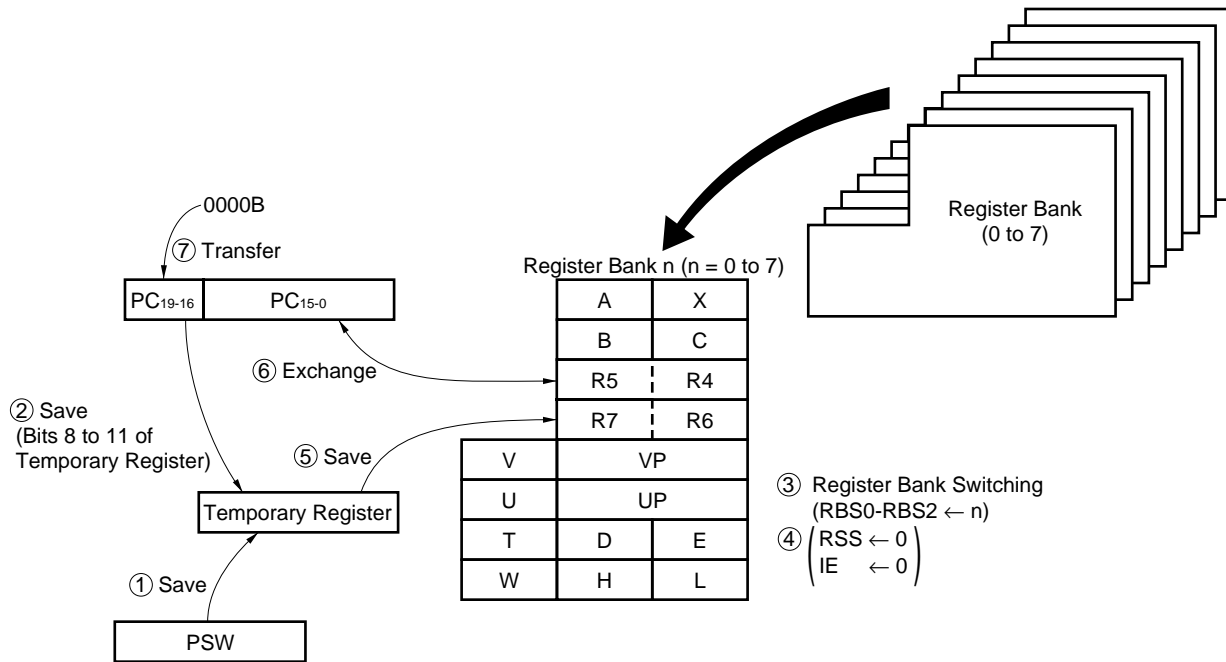
### 20.4.2 BRKCS Instruction Software Interrupt (Software Context Switching) Acknowledgment Operation

The context switching function can be initiated by executing a BRKCS instruction.

The register bank to be used after context switching is specified by the BRKCS instruction operand.

When a BRKCS instruction is executed, the program branches to the start address of the interrupt service program (which must be in the base area) stored beforehand in the specified register bank, and the contents of the program status word (PSW) and program counter (PC) are saved in the register bank.

**Figure 20-7 Context Switching Operation by Execution of a BRKCS Instruction**

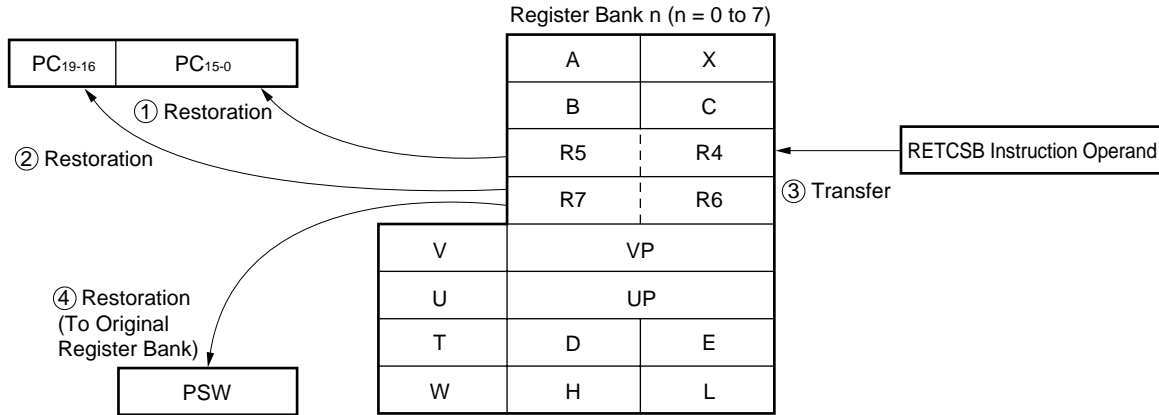


The RETCSB instruction is used to return from a software interrupt due to a BRKCS instruction. The RETCSB instruction must specify the start address of the interrupt service program for the next time context switching is performed by a BRKCS instruction. This interrupt service program start address must be in the base area.

**Caution** The RETCS instruction must not be used to return from a BRKCS instruction software interrupt.



**Figure 20-8 Return from BRKCS Instruction Software Interrupt (RETCSB Instruction Operation)**



## 20.5 OPERAND ERROR INTERRUPT ACKNOWLEDGMENT OPERATION

An operand error interrupt is generated when the data obtained by inverting all the bits of the 3rd byte of the operand of an MOV STBC, #byte instruction or LOCATION instruction or an MOV WDM, #byte instruction does not match the 4th byte of the operand. Operand error interrupts cannot be disabled.

When an operand error interrupt is generated, the program status word (PSW) and the start address of the instruction that caused the error are saved to the stack, the IE flag is cleared (0), the vector table value is loaded into the program counter (PC), and a branch is performed (within the base area only).

As the address saved to the stack is the start address of the instruction in which the error occurred, simply writing an RETB instruction at the end of the operand error interrupt service program will result in generation of another operand error interrupt. You should therefore either process the address in the stack or initialize the program by referring to **20.12 RESTORING INTERRUPT FUNCTION TO INITIAL STATE**.

## 20.6 NON-MASKABLE INTERRUPT ACKNOWLEDGMENT OPERATION

Non-maskable interrupts are acknowledged even in the interrupt disabled state. Non-maskable interrupts can be acknowledged at all times except during execution of the service program for an identical non-maskable interrupt or a non-maskable interrupt of higher priority.

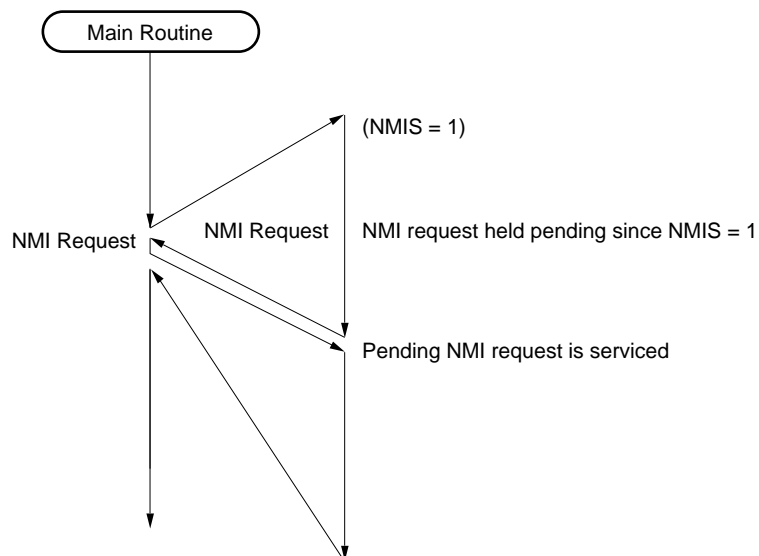
The relative priorities of non-maskable interrupts are set by the PRC bit of the watchdog timer mode register (WDM) (see **20.3.5 Watchdog Timer Mode register (WDM)**).

Except in the cases described in **20.9 WHEN INTERRUPT REQUESTS AND MACRO SERVICE ARE TEMPORARILY HELD PENDING**, a non-maskable interrupt request is acknowledged immediately. When a non-maskable interrupt request is acknowledged, the program status word (PSW) and program counter (PC) are saved in that order to the stack, the IE flag is cleared (0), the in-service priority register (ISPR) bit corresponding to the acknowledged non-maskable interrupt is set (1), the vector table contents are loaded into the PC, and a branch is performed. The ISPR bit that is set (1) is the NMIS bit in the case of a non-maskable interrupt due to edge input to the NMI pin, and the WDTS bit in the case of watchdog timer overflow.

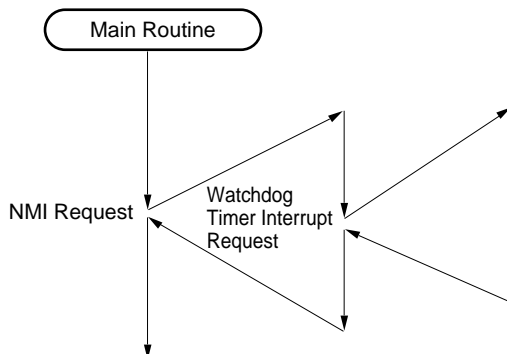
When the non-maskable interrupt service program is executed, non-maskable interrupt requests of the same priority as the non-maskable interrupt currently being executed and non-maskable interrupts of lower priority than the non-maskable interrupt currently being executed are held pending. A pending non-maskable interrupt is acknowledge after completion of the non-maskable interrupt service program currently being executed (after execution of the RETI instruction). However, even if the same non-maskable interrupt request is generated more than once during execution of the non-maskable interrupt service program, only one non-maskable interrupt is acknowledged after completion of the non-maskable interrupt service program.

**Figure 20-9 Non-Maskable Interrupt Request Acknowledgment Operations (1/2)**

- (a) When a new NMI request is generated during NMI service program execution

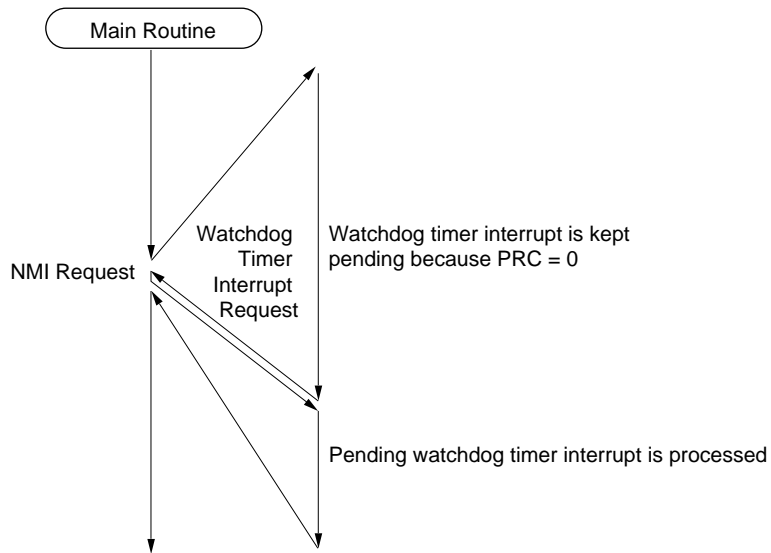


- (b) When a watchdog timer interrupt request is generated during NMI service program execution (when the watchdog timer interrupt priority is higher (when  $PRC$  in the  $WDM = 1$ ))

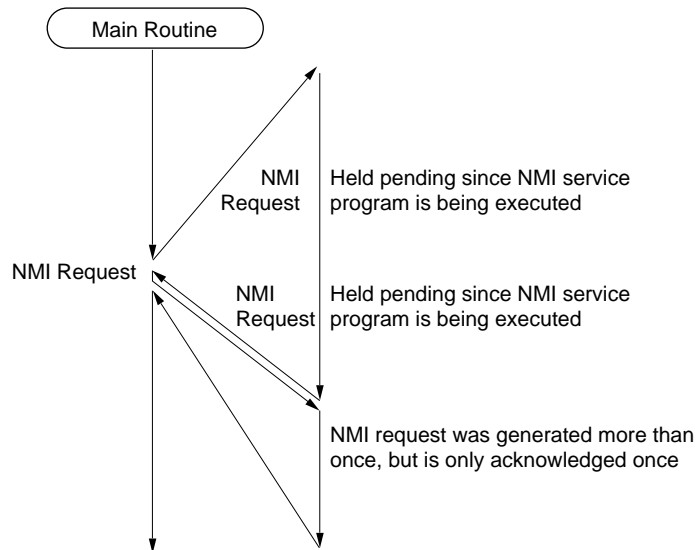


**Figure 20-9 Non-Maskable Interrupt Request Acknowledgment Operations (2/2)**

- (c) When a watchdog timer interrupt request is generated during NMI service program execution (when the NMI interrupt priority is higher (when PRC in the WDM = 0))



- (d) When an NMI request is generated twice during NMI service program execution



- Cautions**
1. Macro service requests are acknowledged and processed even during execution of a non-maskable interrupt service program. If you do not want macro service processing to be performed during a non-maskable interrupt service program, you should manipulate the interrupt mask register in the non-maskable interrupt service program to prevent macro service generation.
  2. The RETI instruction must be used to return from a non-maskable interrupt. Subsequent interrupt acknowledgment will not be performed normally if a different instruction is used. For details on how to resume program execution from the initial status after the non-maskable interrupt has been acknowledged, see 20.12 RESTORING INTERRUPT FUNCTION TO INITIAL STATE.
  3. Non-maskable interrupts are always acknowledged, except during non-maskable interrupt service program execution (except when a high non-maskable interrupt request is generated during execution of a low-priority non-maskable interrupt service program) and for a certain period after execution of the special instructions shown in 20.9. Therefore, a non-maskable interrupt will be acknowledged even when the stack pointer (SP) value is undefined, in particular after reset release, etc. In this case, depending on the value of the SP, it may happen that the program counter (PC) and program status word (PSW) are written to the address of a write-inhibited special function register (SFR) (see Table 3-5 in 3.9 Special Function Registers (SFR)), and the CPU becomes deadlocked, or an unexpected signal is output from a pin, or the PC and PSW are written to an address in which RAM is not mounted, with the result that the return from the non-maskable interrupt service program is not performed normally and an inadvertent program loop occurs. Therefore, the program following RESET release must be as shown below.

```
CSEG AT 0
DW  STRT
CSEG BASE
STRT:
LOCATION 0FH; or LOCATION 0
MOVG SP, #imm24
```

## 20.7 MASKABLE INTERRUPT ACKNOWLEDGMENT OPERATION

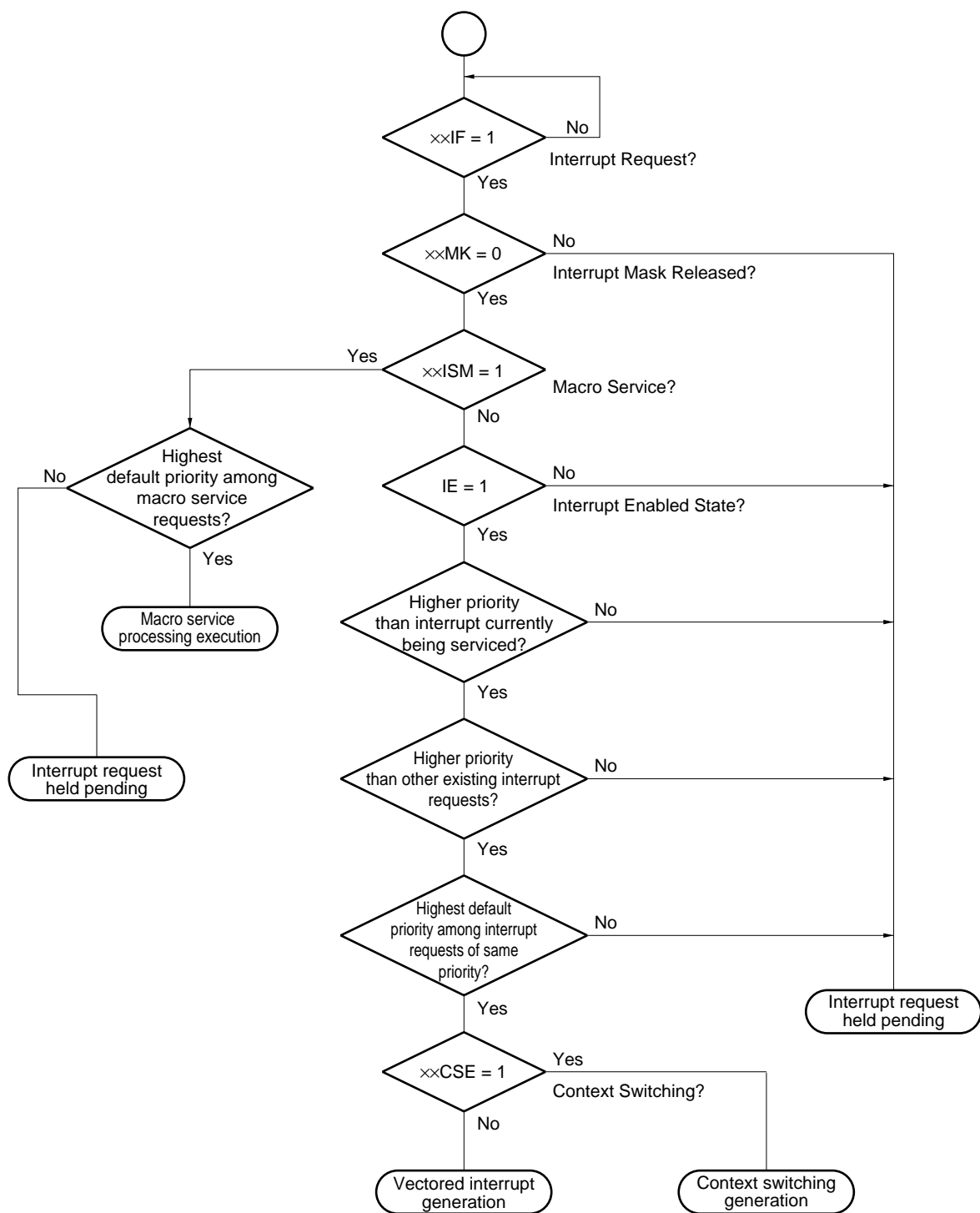
A maskable interrupt can be acknowledged when the interrupt request flag is set (1) and the mask flag for that interrupt is cleared (0). When processing is performed by macro service, the interrupt is acknowledged and processed by macro service immediately. In the case of vectored interrupt and context switching, an interrupt is acknowledged in the interrupt enabled state (when the IE flag is set (1)) if the priority of that interrupt is one for which acknowledgment is permitted.

If maskable interrupt requests are generated simultaneously, the interrupt for which the highest priority is specified by the priority specification flag is acknowledged. If the interrupts have the same priority specified, they are acknowledged in accordance with their default priorities.

A pending interrupt is acknowledged when a state in which it can be acknowledged is established.

The interrupt acknowledgment algorithm is shown in Figure 20-10.

Figure 20-10 Interrupt Acknowledgment Processing Algorithm



### 20.7.1 Vectored Interrupt

When a vectored interrupt maskable interrupt request is acknowledged, the program status word (PSW) and program counter (PC) are saved in that order to the stack, the IE flag is cleared (0) (the interrupt disabled state is set), and the in-service priority register (ISPR) bit corresponding to the priority of the acknowledged interrupt is set (1). Also, data in the vector table predetermined for each interrupt request is loaded into the PC, and a branch is performed. The return from a vectored interrupt is performed by means of the RETI instruction.

**Caution** When a maskable interrupt is acknowledged by vectored interrupt, the RETI instruction must be used to return from the interrupt. Subsequent interrupt acknowledgment will not be performed normally if a different instruction is used.

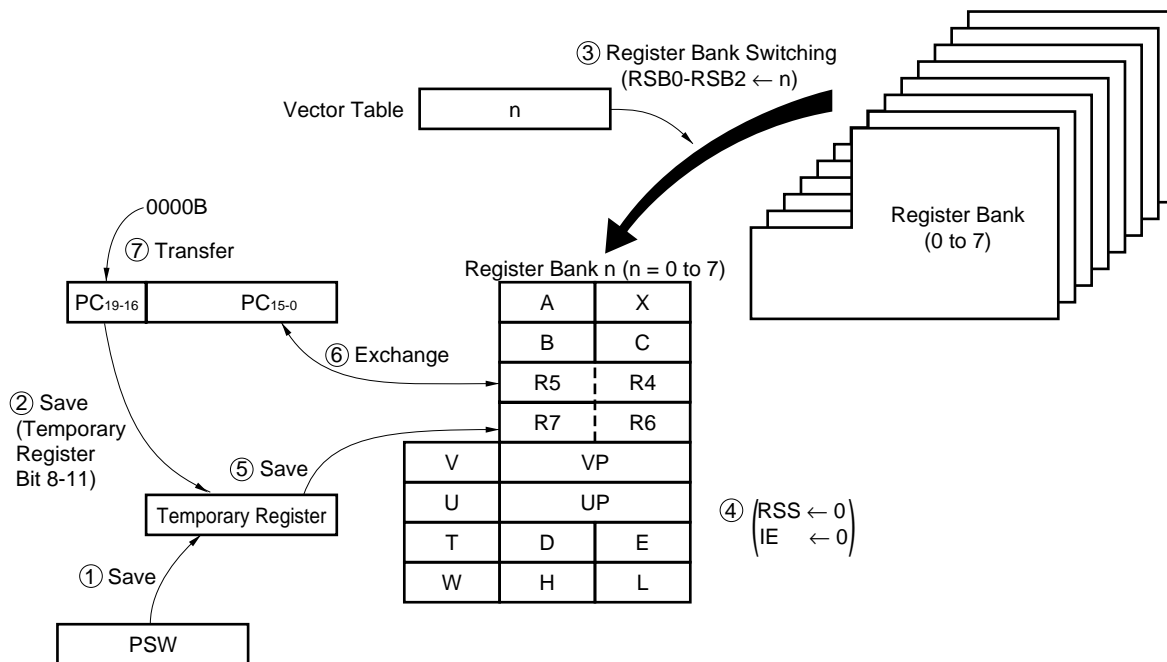
### 20.7.2 Context Switching

Initiation of the context switching function is enabled by setting (1) the context switching enable flag of the interrupt control register.

When an interrupt request for which the context switching function is enabled is acknowledged, the register bank specified by 3 bits of the low-order address (even address) of the corresponding vector table address is selected.

The vector address stored beforehand in the selected register bank is transferred to the program counter (PC), and at the same time the contents of the PC and program status word (PSW) up to that time are saved in the register bank and a branch is made to the interrupt service program.

Figure 20-11 Context Switching Operation by Generation of an Interrupt Request

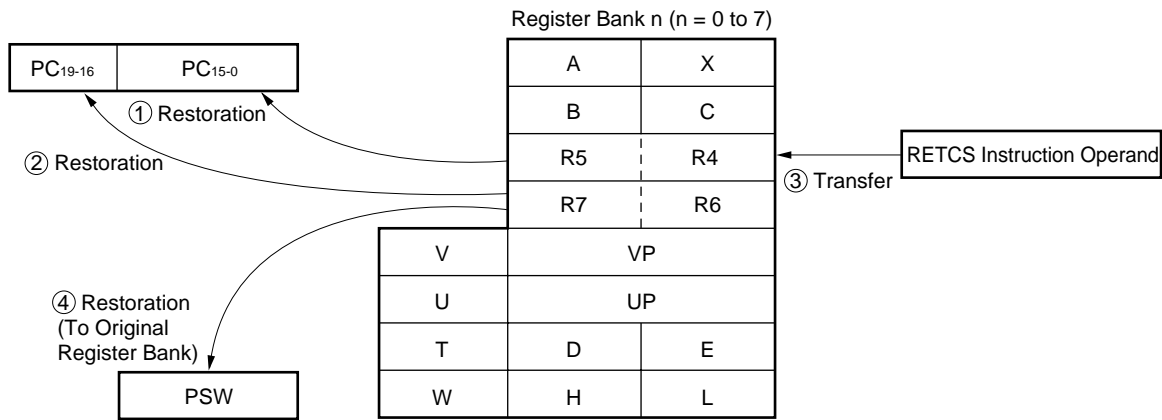




The RETCS instruction is used to return from an interrupt that uses the context switching function. The RETCS instruction must specify the start address of the interrupt service program to be executed when that interrupt is acknowledged next. This interrupt service program start address must be in the base area.

**Caution** The RETCS instruction must be used to return from an interrupt processed by context switching. Subsequent interrupt acknowledgment will not be performed normally if a different instruction is used.

**Figure 20-12 Return from Interrupt that Uses Context Switching by Means of RETCS Instruction**



**20.7.3 Maskable Interrupt Priority Levels**

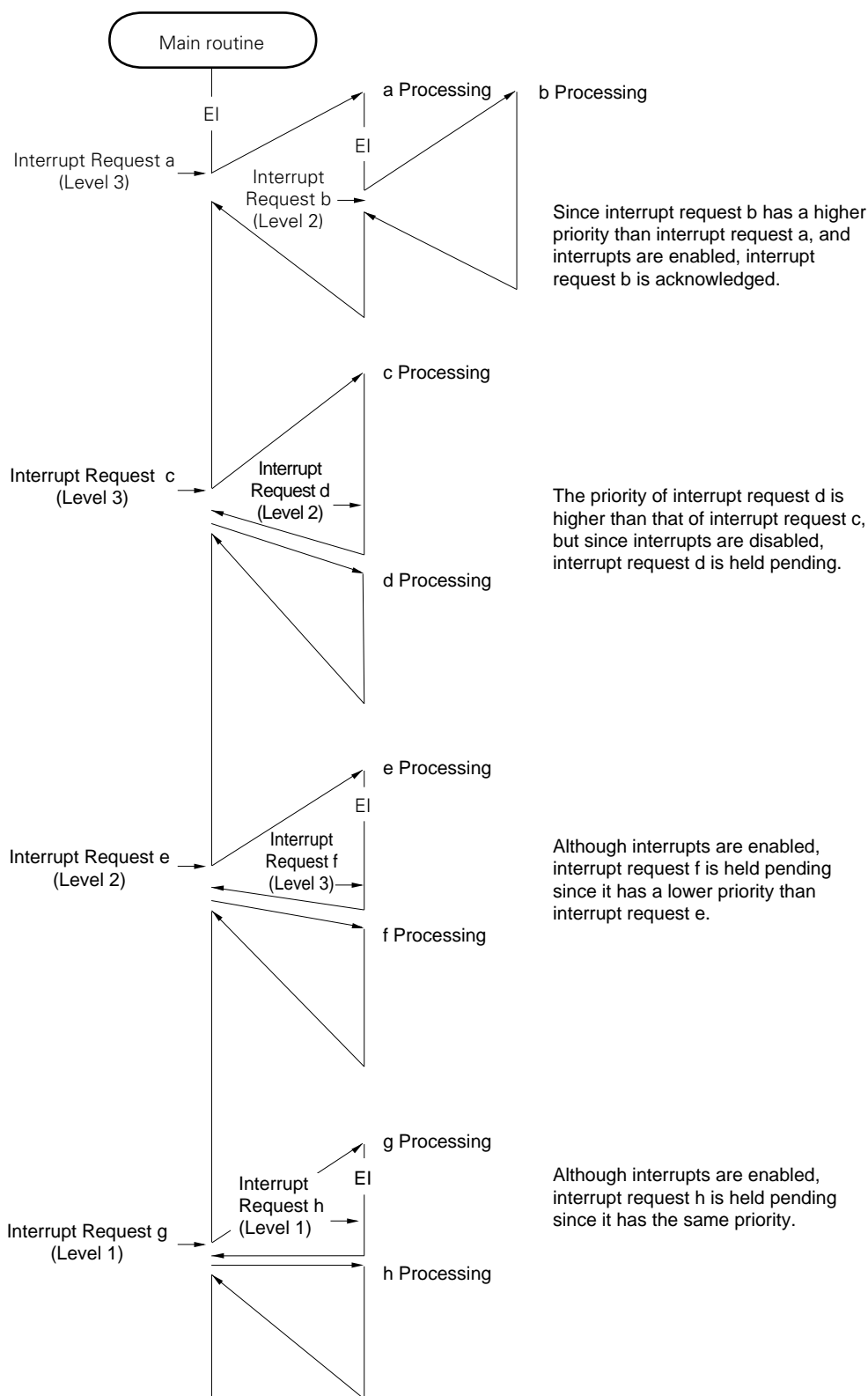
The  $\mu$ PD784026 performs nesting interrupt processing in which an interrupt is acknowledged during processing of another interrupt. Nesting interrupts can be controlled by priority levels.

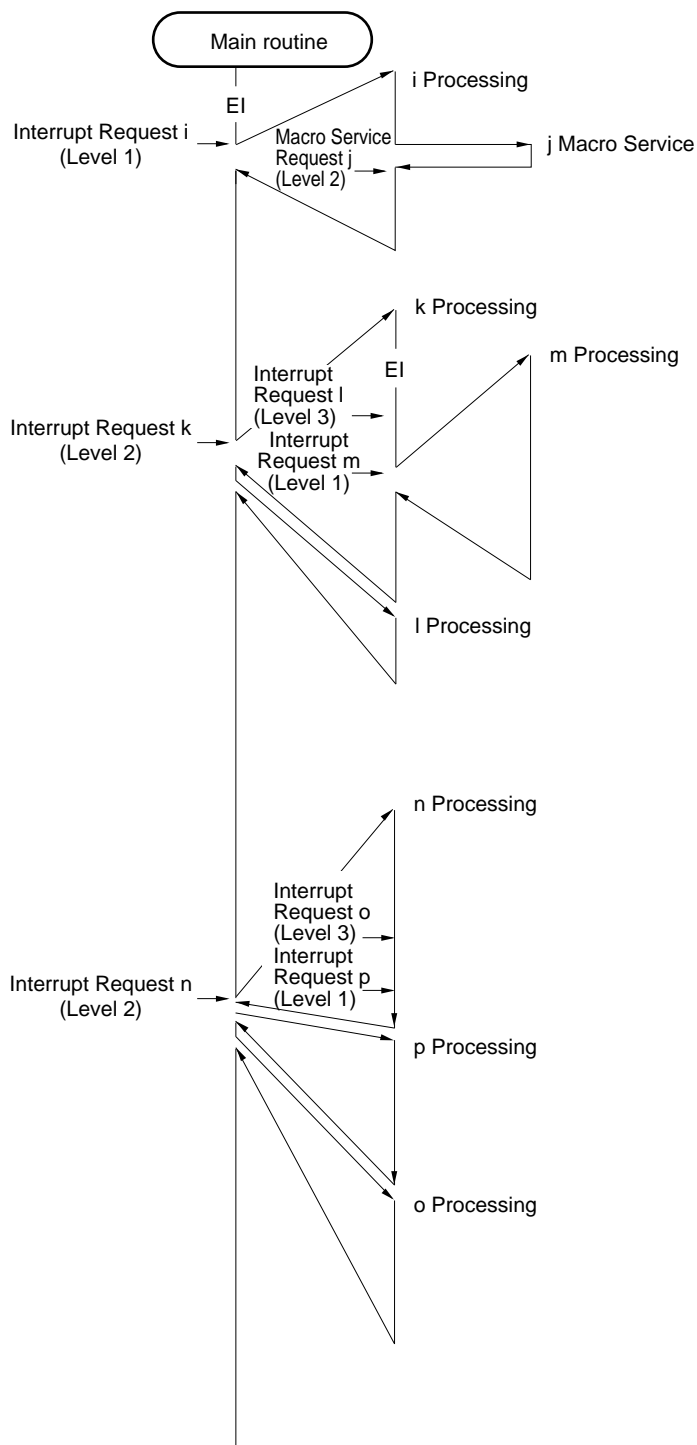
There are two kinds of priority control, control by default priority and programmable priority control in accordance with the setting of the priority specification flag. In priority control by means of default priority, interrupt processing is performed in accordance with the priority preassigned to each interrupt request (default priority) (see **Table 20-2**). In programmable priority control, interrupt requests are divided into four levels according to the setting of the priority specification flag. Interrupt requests for which nesting interrupt is permitted are shown in Table 20-5.

Since the IE flag is cleared (0) automatically when an interrupt is acknowledged, when nesting interrupt is used, the IE flag should be set (1) to enable interrupts by executing an EI instruction in the interrupt service program, etc.

**Table 20-5 Nesting Interrupt**

Priority of Interrupt Currently Being Acknowledged	ISPR Value	IE Flag in PSW	PRSL in IMC Register	Acknowledgeable Maskable Interrupts
No interrupt being acknowledged	00000000	0	×	• All macro service only
		1	×	• All maskable interrupts
3	00001000	0	×	• All macro service only
		1	0	• All maskable interrupts
		1	1	• All macro service • Maskable interrupts specified as priority 0/1/2
2	0000×100	0	×	• All macro service only
		1	×	• All macro service • Maskable interrupts specified as priority 0/1
1	0000××10	0	×	• All macro service only
		1	×	• All macro service • Maskable interrupts specified as priority 0
0	0000×××1	×	×	• All macro service only
Non-maskable interrupts	1000×××× 0100×××× 1100××××	×	×	• All macro service only

**Figure 20-13 Examples of Processing When Another Interrupt Request is Generated During Interrupt Processing (1/3)**

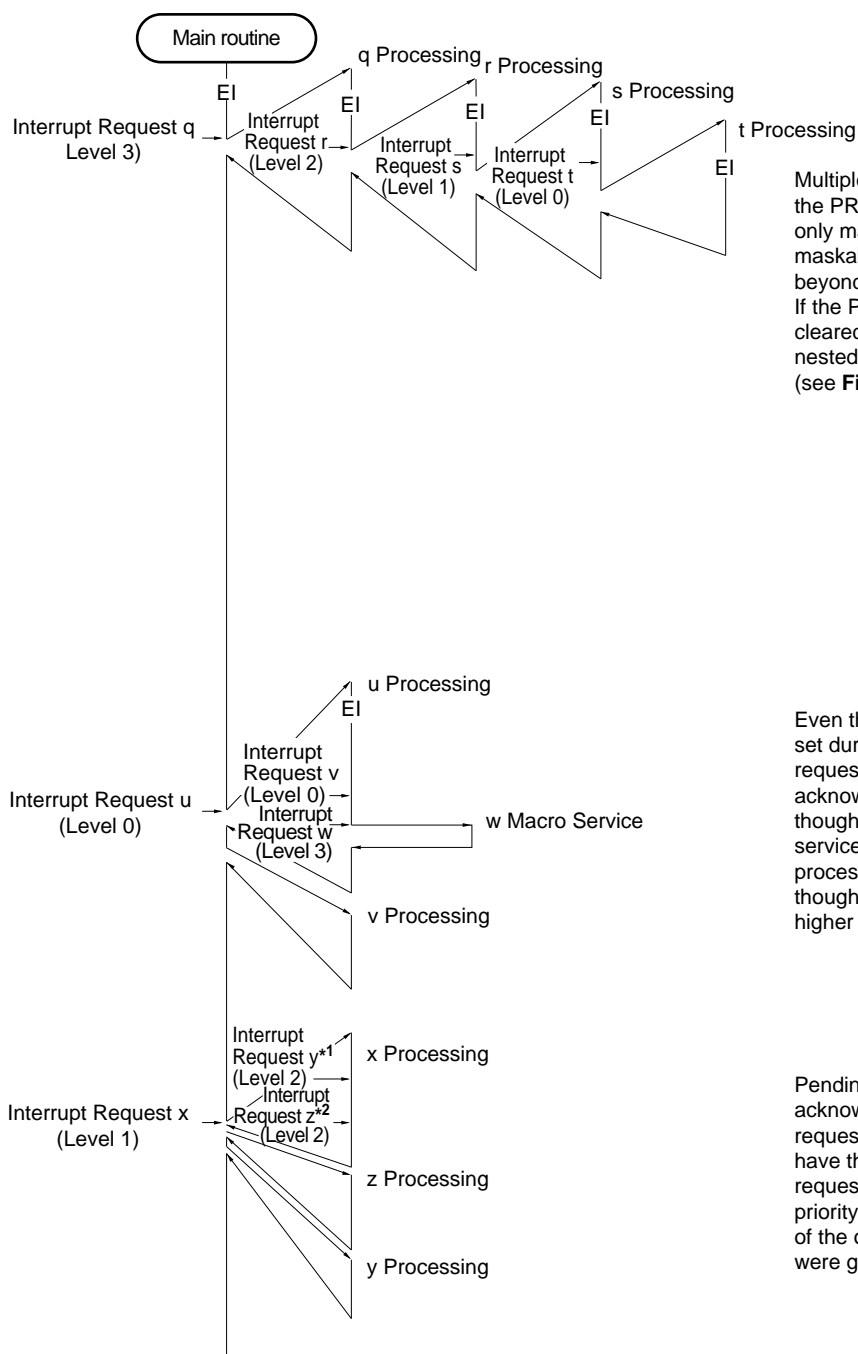
**Figure 20-13 Examples of Processing When Another Interrupt Request is Generated During Interrupt Processing (2/3)**

The macro service request is processed irrespective of interrupt enabling/disabling and priority.

The interrupt request is held pending since it has a lower priority than interrupt request k. Interrupt request m generated after interrupt request l has a higher priority, and is therefore acknowledged first.

Since processing of interrupt request n performed in the interrupt disabled state, interrupt requests o and p are held pending. After interrupt request n processing, the pending interrupt requests are acknowledged. Although interrupt request o was generated first, interrupt request p has a higher priority and is therefore acknowledged first.

Figure 20-13 Examples of Processing When Another Interrupt Request is Generated During Interrupt Processing (3/3)



Multiple acknowledgment of levels 3 to 0. If the PRSL bit of the IMC register is set (1), only macro service requests and non-maskable interrupts generate nesting beyond this.

If the PRSL bit of the IMC register is cleared (0), level 3 interrupts can also be nested during level 3 interrupt processing (see Figure 20-15).

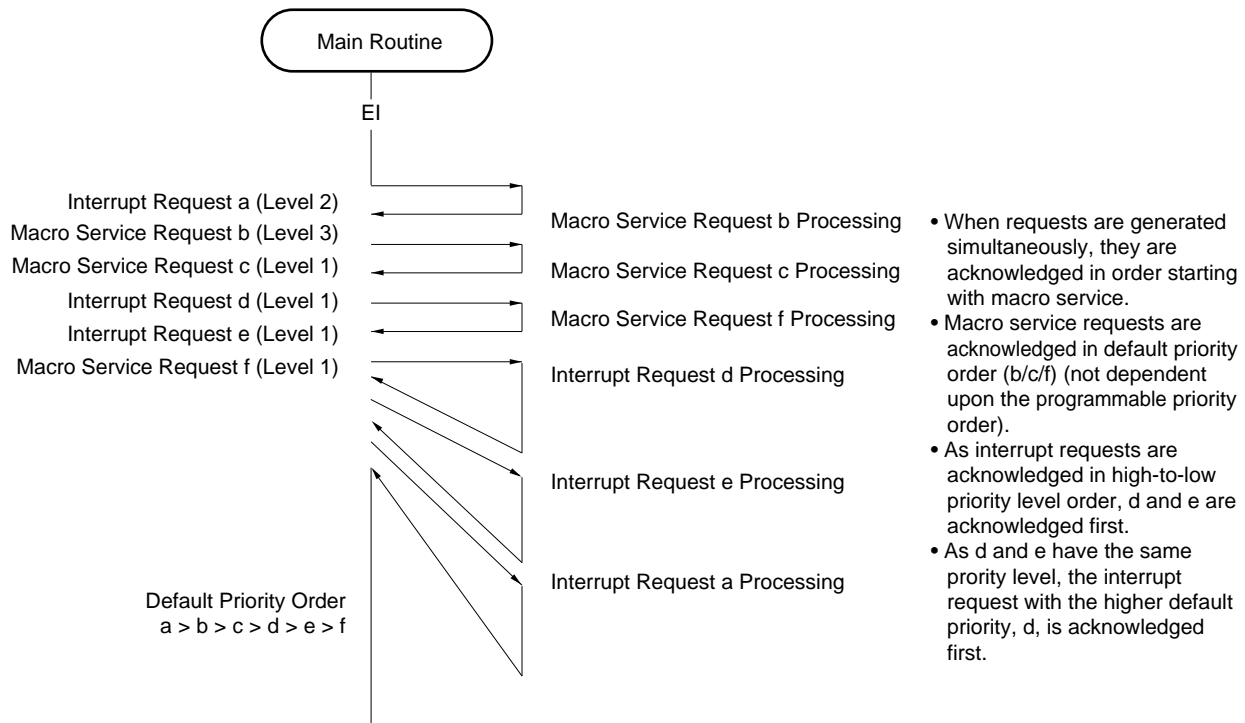
Even though the interrupt enabled state is set during processing of level 0 interrupt request u, the interrupt request is not acknowledged but held pending even though its priority is 0. However, the macro service request is acknowledged and processed irrespective of its level and even though there is a pending interrupt with a higher priority level.

Pending interrupt requests y and z are acknowledged after processing of interrupt request x. As interrupt requests y and z have the same priority level, interrupt request z which has the higher default priority is acknowledged first, irrespective of the order in which the interrupt requests were generated.

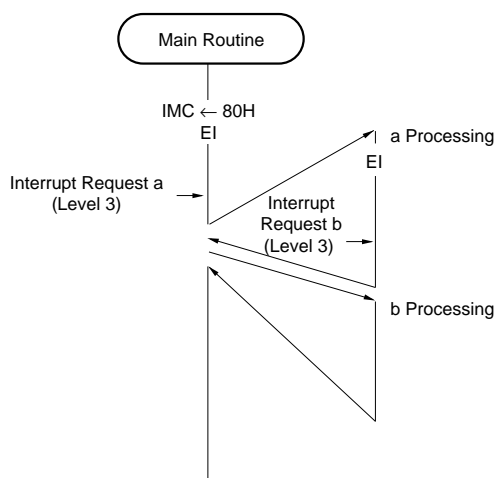
- \* 1. Low default priority
- 2. High default priority

**Remarks 1.** "a" to "z" in the figure are arbitrary names used to differentiate between the interrupt requests and macro service requests.

- 2. High/low default priorities in the figure indicate the relative priority levels of the two interrupt requests.

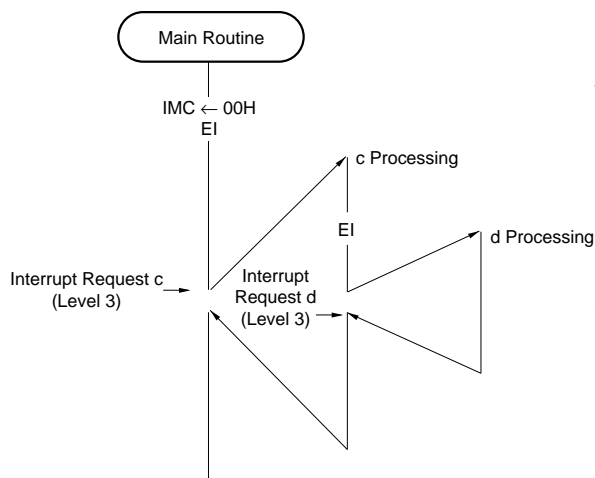
**Figure 20-14 Examples of Processing of Simultaneously Generated Interrupts**

**Remark** “a” to “f” in the figure are arbitrary names used to differentiate between the interrupt requests and macro service requests.

**Figure 20-15 Differences in Level 3 Interrupt Acknowledgment According to IMC Register Setting**

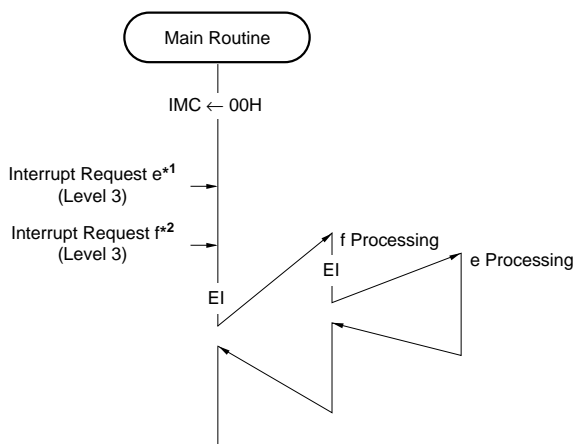
The PRSL bit of the IMC is set to 1, and nesting between level 3 interrupts is disabled.

Even though interrupts are enabled, interrupt request b is held pending since it has the same priority as interrupt request a.



The PRSL bit of the IMC is set to 0, so that a level 3 interrupt is acknowledged even during level 3 interrupt processing (nesting is possible).

Since level 3 interrupt request c is being serviced in the interrupt enabled state and PRSL = 0, interrupt request d, which is also level 3, is acknowledged.



As interrupt request 3 and f are both of the same level, the one with the higher default priority, f, is acknowledged first. When the interrupt enabled state is set during processing of interrupt request f, pending interrupt request e is acknowledged since PRSL = 0.

- \* 1. Low default priority
- 2. High default priority

**Remarks** 1. "a" to "f" in the figure are arbitrary names used to differentiate between the interrupt requests and macro service requests.

2. High/low default priorities in the figure indicate the relative priority levels of the two interrupt requests.

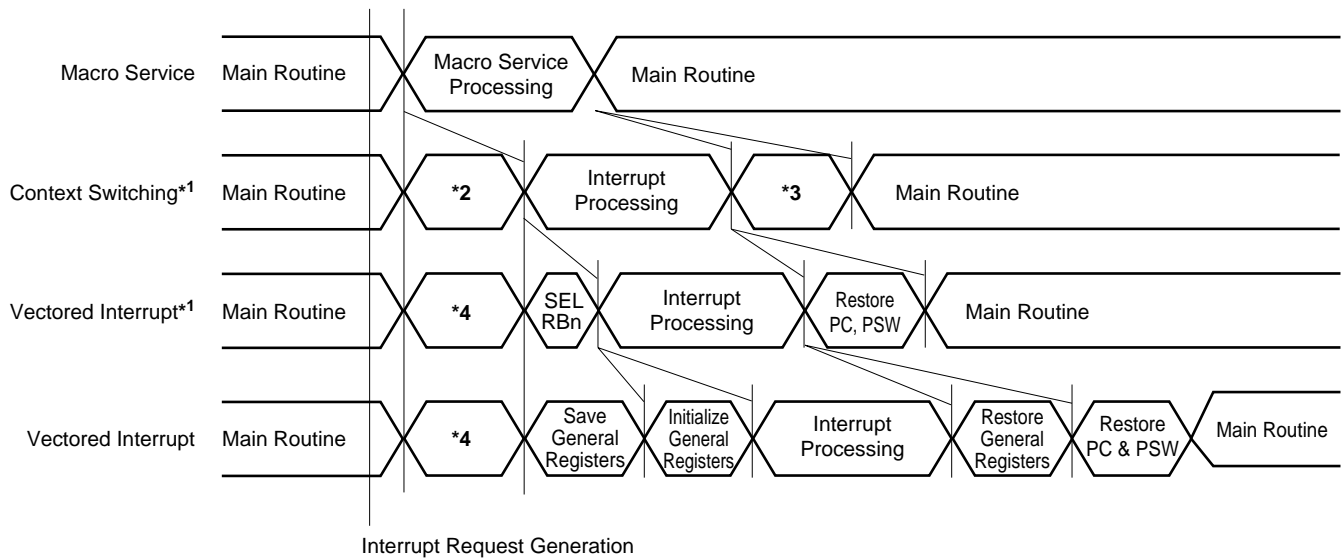
## 20.8 MACRO SERVICE FUNCTION

### 20.8.1 Outline of Macro Service Function

Macro service is one method of processing interrupts. With a normal interrupt, the program counter (PC) and program status word (PSW) are saved, and the start address of the interrupt service program is loaded into the PC, but with macro service, different processing (mainly data transfers) is performed instead of this processing. This enables interrupt requests to be responded to quickly, and moreover, since transfer processing is faster than processing by a program, the processing time can also be reduced.

Also, since a vectored interrupt is generated after processing has been performed the specified number of times, another advantage is that vectored interrupt programs can be simplified.

**Figure 20-16 Differences between Vectored Interrupt and Macro Service Processing**



- \* 1. When register bank switching is used, and an initial value has been set in the register beforehand
- 2. Register bank switching by context switching, saving of PC and PSW
- 3. Register bank, PC and PSW restoration by context switching
- 4. PC and PSW saved to the stack, vector address loaded into PC

### 20.8.2 Types of Macro Service

Macro service can be used with the 19 kinds of interrupt shown in Table 20-6 (17 of which can be used simultaneously). There are four kinds of operation, which can be used to suit the application.



**Table 20-6 Interrupts for Which Macro Service Can be Used**

Default Priority	Interrupt Request Generation Source	Generating Unit	Macro Service Control Word Address
0	INTP0 (pin input edge detection)	Edge detection	0FE06H
1	INTP1 (pin input edge detection)		0FE08H
2	INTP2 (pin input edge detection)		0FE0AH
3	INTP3 (pin input edge detection)		0FE0CH
4	INTC00 (TM0-CR00 match signal generation)	Timer/counter 0	0FE0EH
5	INTC01 (TM0-CR01 match signal generation)		0FE10H
6	INTC10 (TM1-CR10 or TM1W-CR10W match signal generation)	Timer/counter 1	0FE12H
7	INTC11 (TM1-CR11 or TM1W-CR11W match signal generation)		0FE14H
8	INTC20 (TM2-CR20 or TM2W-CR20W match signal generation)	Timer/counter 2	0FE16H
9	INTC21 (TM2-CR21 or TM2W-CR21W match signal generation)		0FE18H
10	INTC30 (TM3-CR30 or TM3W-CR30W match signal generation)	Timer 3	0FE1AH
11	INTP4 (pin input edge detection)	Edge detection	0FE1CH
12	INTP5 (pin input edge detection)		0FE1EH
13	INTAD (A/D conversion end)	A/D converter	0FE20H
14	INTSR (asynchronous serial interface reception end)	Asynchronous serial interface/ clocked serial interface 1	0FE24H
	INTCSI1 (clocked serial interface transfer end)		0FE26H
15	INTST (asynchronous serial interface transmission end)		
16	INTCSI (clocked serial interface transfer end)	Clocked serial interface	0FE28H
17	INTSR2 (asynchronous serial interface 2 reception end)	Asynchronous serial interface 2/ clocked serial interface 2	0FE2CH
	INTCSI2 (clocked serial interface 2 transfer end)		0FE2EH
18	INTST2 (asynchronous serial interface 2 transmission end)		

- Remarks**
1. The default priority is a fixed number. This indicates the order of priority when macro service requests are generated simultaneously,
  2. The INTSR and INTCSI1 interrupts are generated by the same hardware (they cannot both be used simultaneously). Therefore, although the same hardware is used for the interrupts, two names are provided, for use in each of the two modes. The same applies to INTSR2 and INTCSI2.

There are four kinds of macro service, as shown below.

**(1) Type A**

One byte or one word of data is transferred between a special function register (SFR) and memory each time an interrupt request is generated, and a vectored interrupt request is generated when the specified number of transfers have been performed.

Memory that can be used in the transfers is limited to internal RAM addresses 0FE00H to 0FEFFH when the LOCATION 0 instruction is executed, and addresses 0FFE00H to 0FFEFFH when the LOCATION 0FH instruction is executed. The specification method is simple and is suitable for low-volume, high-speed data transfers.

**(2) Type B**

As with type A, one byte or one word of data is transferred between a special function register (SFR) and memory each time an interrupt request is generated, and a vectored interrupt request is generated when the specified number of transfers have been performed.

The SFR and memory to be used in the transfers is specified by the macro service channel (the entire 1M-byte memory space can be used).

This is a general version of type A, suitable for large volumes of transfer data.

**(3) Type C**

Data is transferred from memory to two special function registers (SFR) each time an interrupt request is generated, and a vectored interrupt request is generated when the specified number of transfers have been performed.

With type C macro service, not only are data transfers performed to two locations in response to a single interrupt request, but it is also possible to add output data ring control and a function that automatically adds data to a compare register. The entire 1M-byte memory space can be used.

Type C is mainly used with the INTC10 and INTC11 interrupts, and is used for stepping motor control, etc., by macro service, with P0L or P0H and CR10, CR10W, CR11 and CR11W used as the SFRs to which data is transferred.

**(4) Counter mode**

This mode is to decrement the macro service counter (MSC) when an interrupt occurs and is used to count the division operation of an interrupt and interrupt generation circuit.

When MSC is 0, a vectored interrupt can be generated.

To restart the macro service, MSC must be set again.

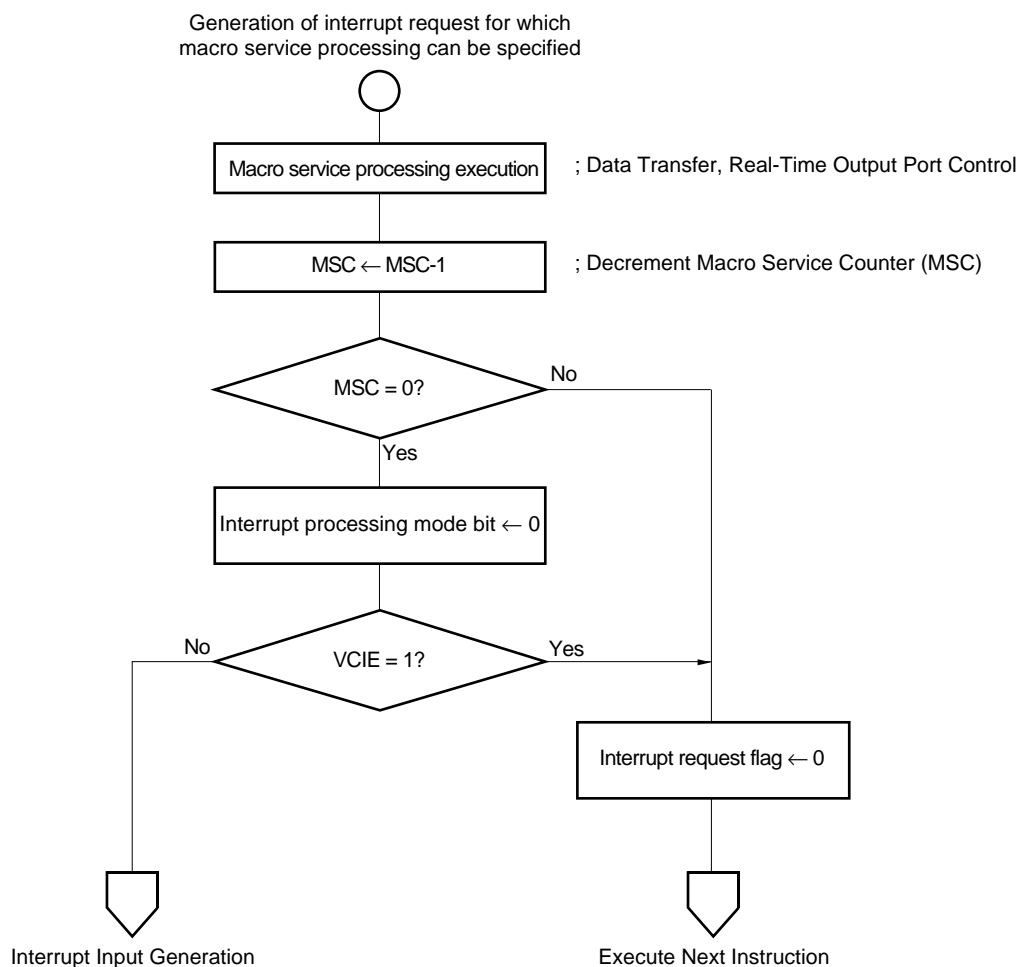
MSC is fixed to 16 bits and cannot be used as an 8-bit counter.

### 20.8.3 Basic Macro Service Operation

Interrupt requests for which the macro service processing generated by the algorithm shown in Figure 20-10 can be specified are basically processed in the sequence shown in Figure 20-17.

Interrupt requests for which macro service processing can be specified are not affected by the status of the IE flag, but are disabled by setting (1) an interrupt mask flag in the interrupt mask register (MK0). Macro service processing can be executed in the interrupt disabled state and during execution of an interrupt service program.

**Figure 20-17 Macro Service Processing Sequence**



The macro service type and transfer direction are determined by the value set in the macro service control word mode register. Transfer processing is then performed using the macro service channel specified by the channel pointer according to the macro service type.

The macro service channel is memory which contains the macro service counter which records the number of transfers, the transfer destination and transfer source pointers, and data buffers, and can be located at any address in the range FE00H to FEFH when the LOCATION 0 instruction is executed, or FFE00H to FFEFFH when the LOCATION 0FH instruction is executed.

#### 20.8.4 Operation at End of Macro Service

In macro service, processing is performed the number of times specified during execution of another program. Macro service ends when the processing has been performed the specified number of times (when the macro service counter (MSC) reaches 0). Either of two operations may be performed at this point, as specified by the VCIE bit (bit 7) of the macro service mode register for each macro service.

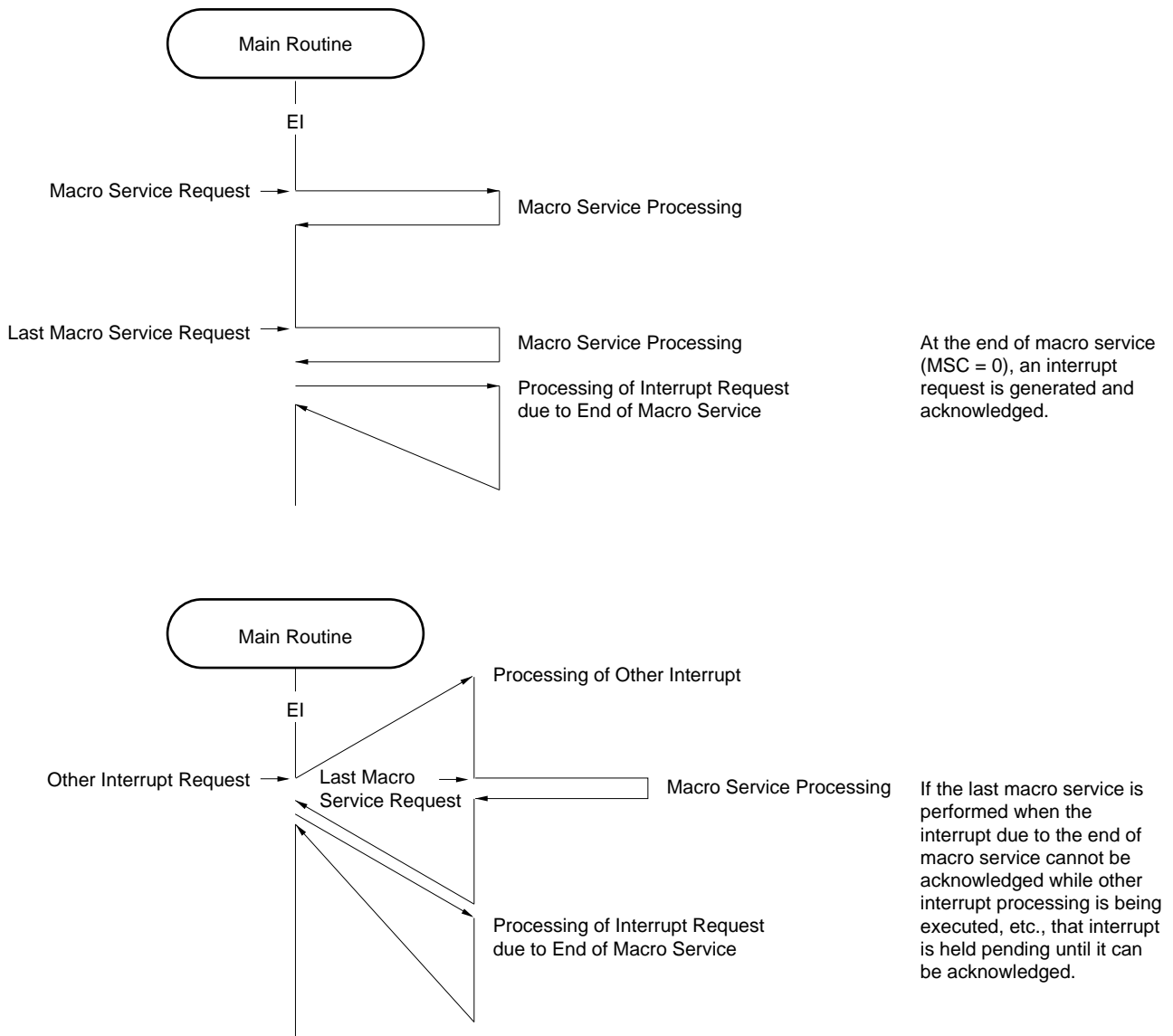
##### (1) When VCIE bit is 0

In this mode, an interrupt is generated as soon as the macro service ends. Figure 20-18 shows an example of macro service and interrupt acknowledgment operations when the VCIE bit is 0.

This mode is used when a series of operations end with the last macro service processing performed, for instance. It is mainly used in the following cases:

- Asynchronous serial interface receive data buffering (INTSR/INTSR2)
- A/D conversion result fetch (INTAD)
- Compare register update as the result of a match between a timer register and the compare register (INTC00/INTC01/INTC10/INTC11/INTC20/INTC21/INTC30)
- Timer/counter capture register read due to edge input to the INTP<sub>n</sub> pin (INTP0/INTP1/INTP2/INTP3)

**Figure 20-18 Operation at End of Macro Service When VCIE = 0**



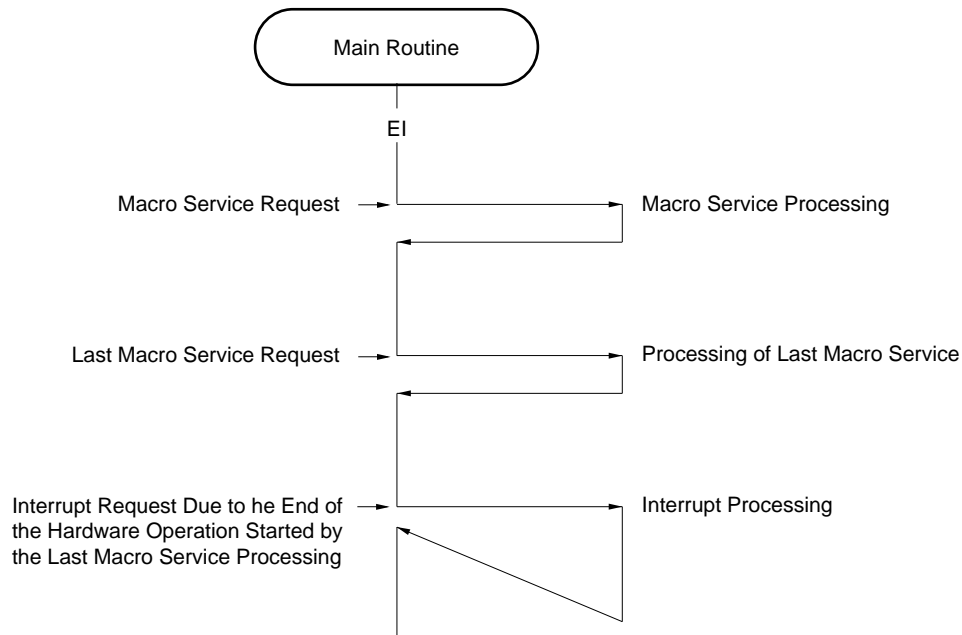
**(2) When VCIE bit is 1**

In this mode, an interrupt is not generated after macro service ends. Figure 20-19 shows an example of macro service and interrupt acknowledgment operations when the VCIE bit is 1.

This mode is used when the final operation is to be started by the last macro service processing performed, for instance. It is mainly used in the following cases:

- Clocked serial interface receive data transfers (INTCSI/INTCSI1/INTCSI2)
- Asynchronous serial interface data transfers (INTST/INTST2)
- To stop a stepping motor in the case (INTC10/INTC11) of stepping motor control by means of macro service type C using the real-time output port and timer/counter.

**Figure 20-19 Operation at End of Macro Service When VCIE = 1**



**20.8.5 Macro Service Control Registers****(1) Macro service control word**

The  $\mu$ PD784026's macro service function is controlled by the macro service control mode register and macro service channel pointer. The macro service processing mode is set by means of the macro service mode register, and the macro service channel address is indicated by the macro service channel pointer.

The macro service mode register and macro service channel pointer are mapped onto the part of the internal RAM shown in Figure 20-20 for each macro service as the macro service control word.

When macro service processing is performed, the macro service mode register and channel pointer values corresponding to the interrupt requests for which macro service processing can be specified must be set beforehand.

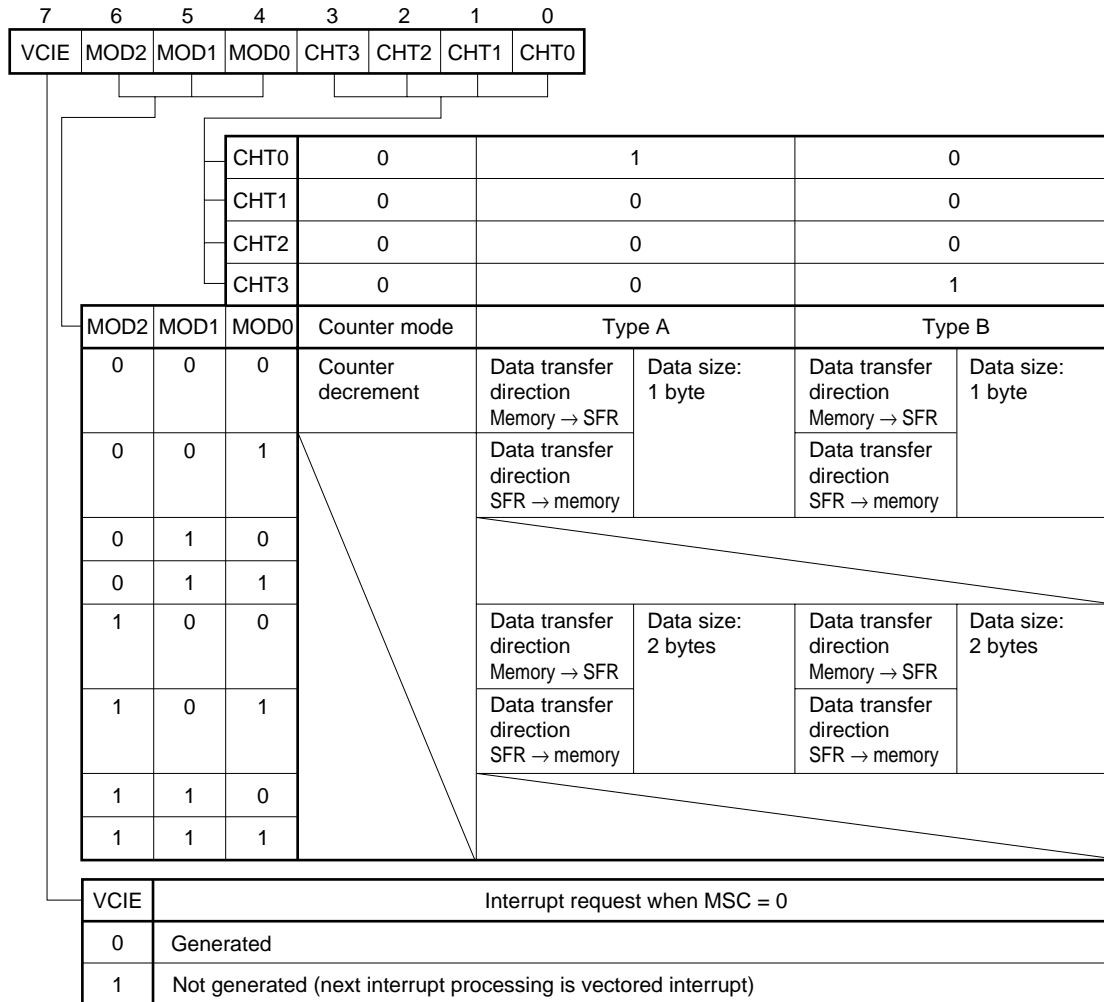
**Figure 20-20 Macro Service Control Word Format**

Reserved Word	Address		Source
STCHP2	0FE2F	Channel Pointer	} INTST2
STMMD2	0FE2E	Mode Register	
SRCHP2/CSICHP2	0FE2D	Channel Pointer	} INTSR2/INTCSI2
SRMMD2/CSIMMD2	0FE2C	Mode Register	
CSICHP	0FE29	Channel Pointer	} INTCSI
CSIMMD	0FE28	Mode Register	
STCHP	0FE27	Channel Pointer	} INTST
STMMD	0FE26	Mode Register	
SRCHP/CSICHP1	0FE25	Channel Pointer	} INTSR/INTCSI1
SRMMD/CSIMMD1	0FE24	Mode Register	
ADCHP	0FE21	Channel Pointer	} INTAD
ADMMD	0FE20	Mode Register	
PCHP5	0FE1F	Channel Pointer	} INTP5
PMMD5	0FE1E	Mode Register	
PCHP4	0FE1D	Channel Pointer	} INTP4
PMMD4	0FE1C	Mode Register	
CCHP30	0FE1B	Channel Pointer	} INTC30
CMMD30	0FE1A	Mode Register	
CCHP21	0FE19	Channel Pointer	} INTC21
CMMD21	0FE18	Mode Register	
CCHP20	0FE17	Channel Pointer	} INTC20
CMMD20	0FE16	Mode Register	
CCHP11	0FE15	Channel Pointer	} INTC11
CMMD11	0FE14	Mode Register	
CCHP10	0FE13	Channel Pointer	} INTC10
CMMD10	0FE12	Mode Register	
CCHP01	0FE11	Channel Pointer	} INTC01
CMMD01	0FE10	Mode Register	
CCHP00	0FE0F	Channel Pointer	} INTC00
CMMD00	0FE0E	Mode Register	
PCHP3	0FE0D	Channel Pointer	} INTP3
PMMD3	0FE0C	Mode Register	
PCHP2	0FE0B	Channel Pointer	} INTP2
PMMD2	0FE0A	Mode Register	
PCHP1	0FE09	Channel Pointer	} INTP1
PMMD1	0FE08	Mode Register	
PCHP0	0FE07	Channel Pointer	} INTP0
PMMD0	0FE06	Mode Register	

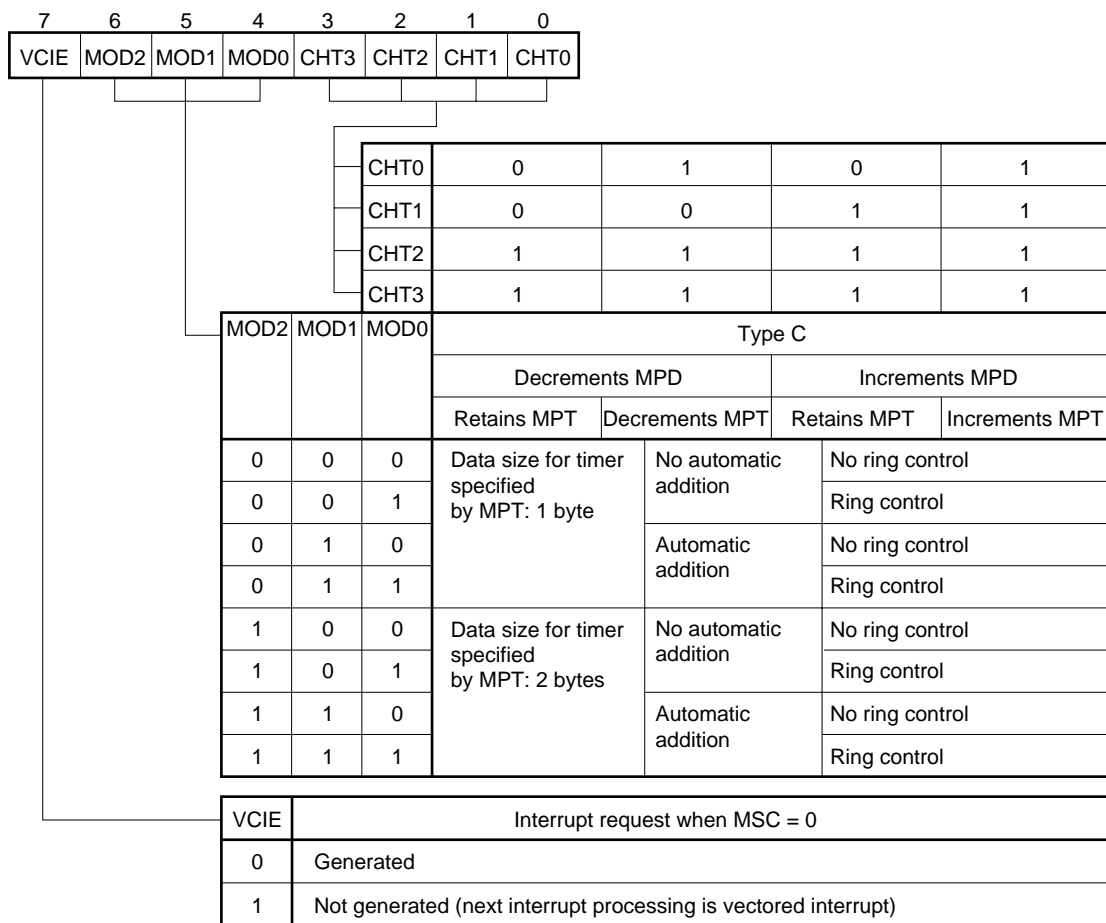
**(2) Macro service mode register**

The macro service mode register is an 8-bit register that specifies the macro service operation. This register is written in internal RAM as part of the macro service control word (see **Figure 20-20**).

The format of the macro service mode register is shown in Figure 20-21.

**Figure 20-21 Macro Service Mode Register Format (1/2)**



**Figure 20-21 Macro Service Mode Register Format (2/2)****(3) Macro service channel pointer**

The macro service channel pointer specifies the macro service channel address. The macro service channel can be located in the 256-byte space from FE00H to FEFFH when the LOCATION 0 instruction is executed, or FFE00H to FFEFFH when the LOCATION 0FH instruction is executed, and the high-order 16 bits of the address are fixed. Therefore, the low-order 8 bits of the data stored to the highest address of the macro service channel are set in the macro service channel pointer.

### 20.8.6 Macro Service Type A

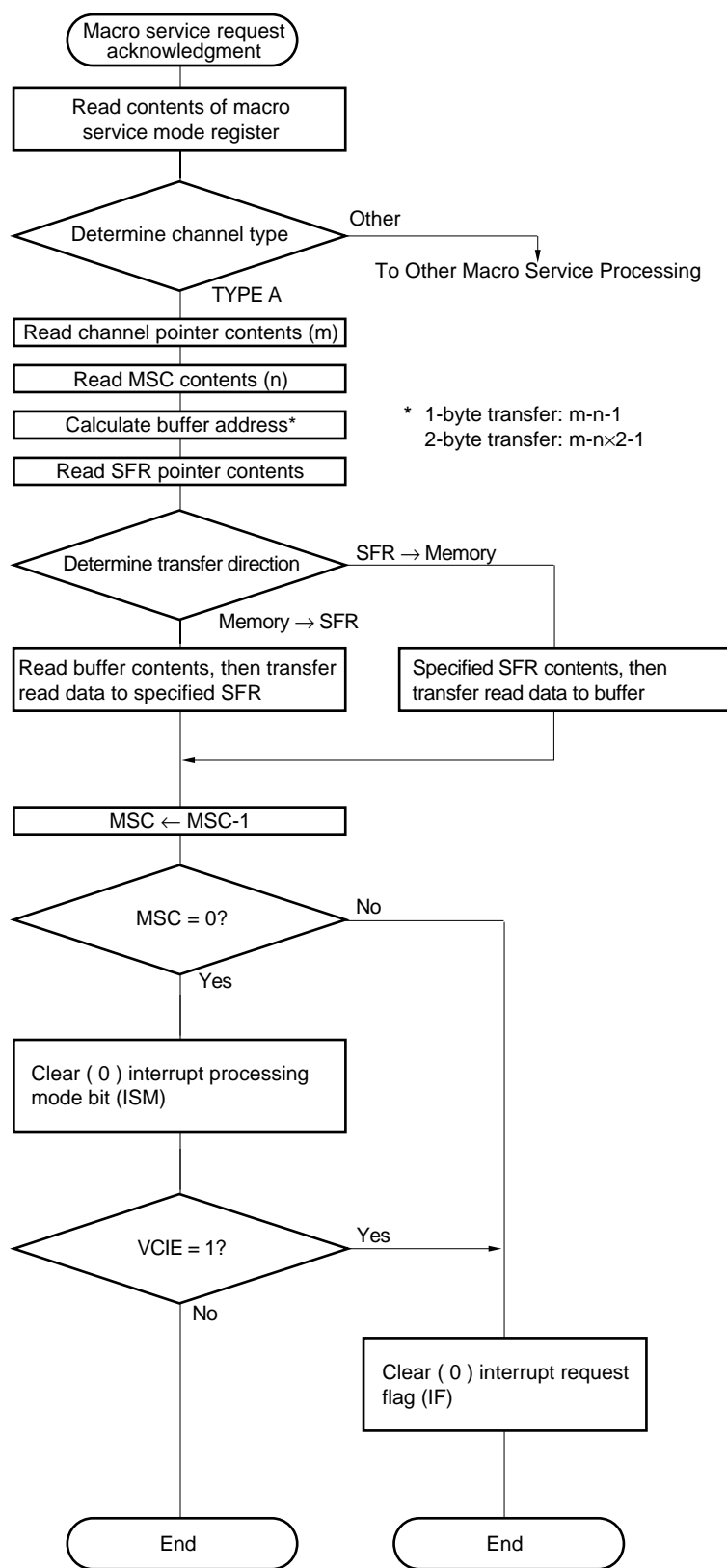
#### (1) Operation

Data transfers are performed between buffer memory in the macro service channel and an SFR specified in the macro service channel.

With type A, the data transfer direction can be selected as memory-to-SFR or SFR-to-memory.

Data transfers are performed the number of times set beforehand in the macro service counter. One macro service processing transfers 8-bit or 16-bit data.

Type A macro service is useful when the amount of data to be transferred is small, as transfers can be performed at high speed.

**Figure 20-22 Macro Service Data Transfer Processing Flow (Type A)**

(Vectored Interrupt Request Generation)

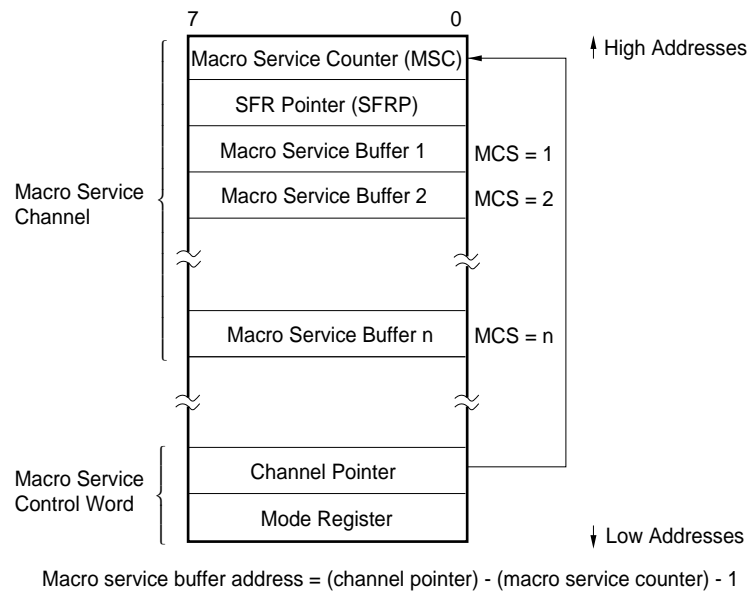
**(2) Macro service channel configuration**

The channel pointer and 8-bit macro service counter (MSC) indicate the buffer address in internal RAM (FE00H to FFEFFH when the LOCATION 0 instruction is executed, or FFE00H to FFEFFH when the LOCATION 0FH instruction is executed) which is the transfer source or transfer destination (see **Figure 20-23**). In the channel pointer, the low-order 8 bits of the address are written to the macro service counter in the macro service channel.

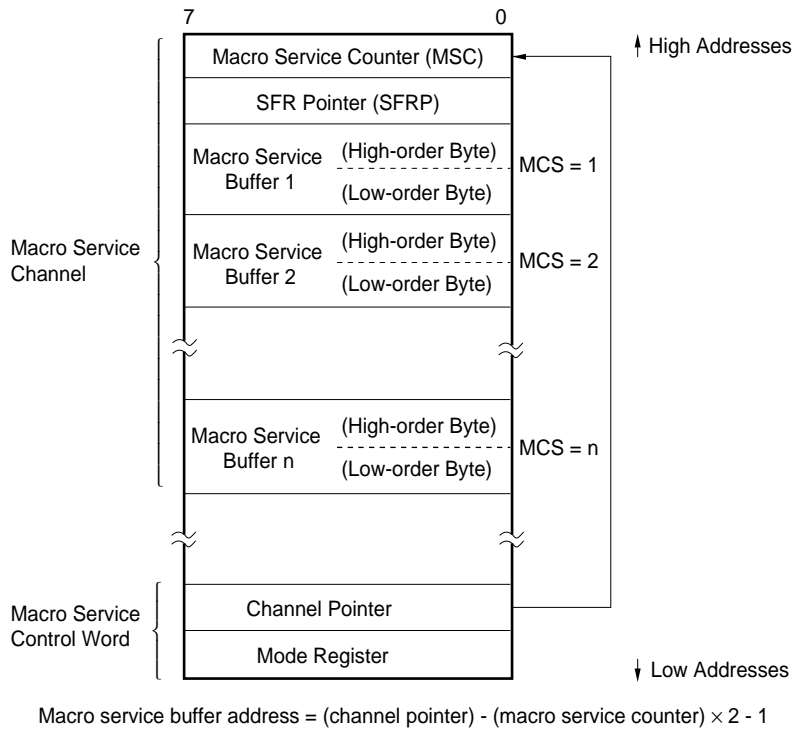
The SFR involved with the access is specified by the SFR pointer (SFRP). The low-order 8 bits of the SFR address are written to the SFRP.

Figure 20-23 Type A Macro Service Channel

(a) 1-byte transfers



(b) 2-byte transfers





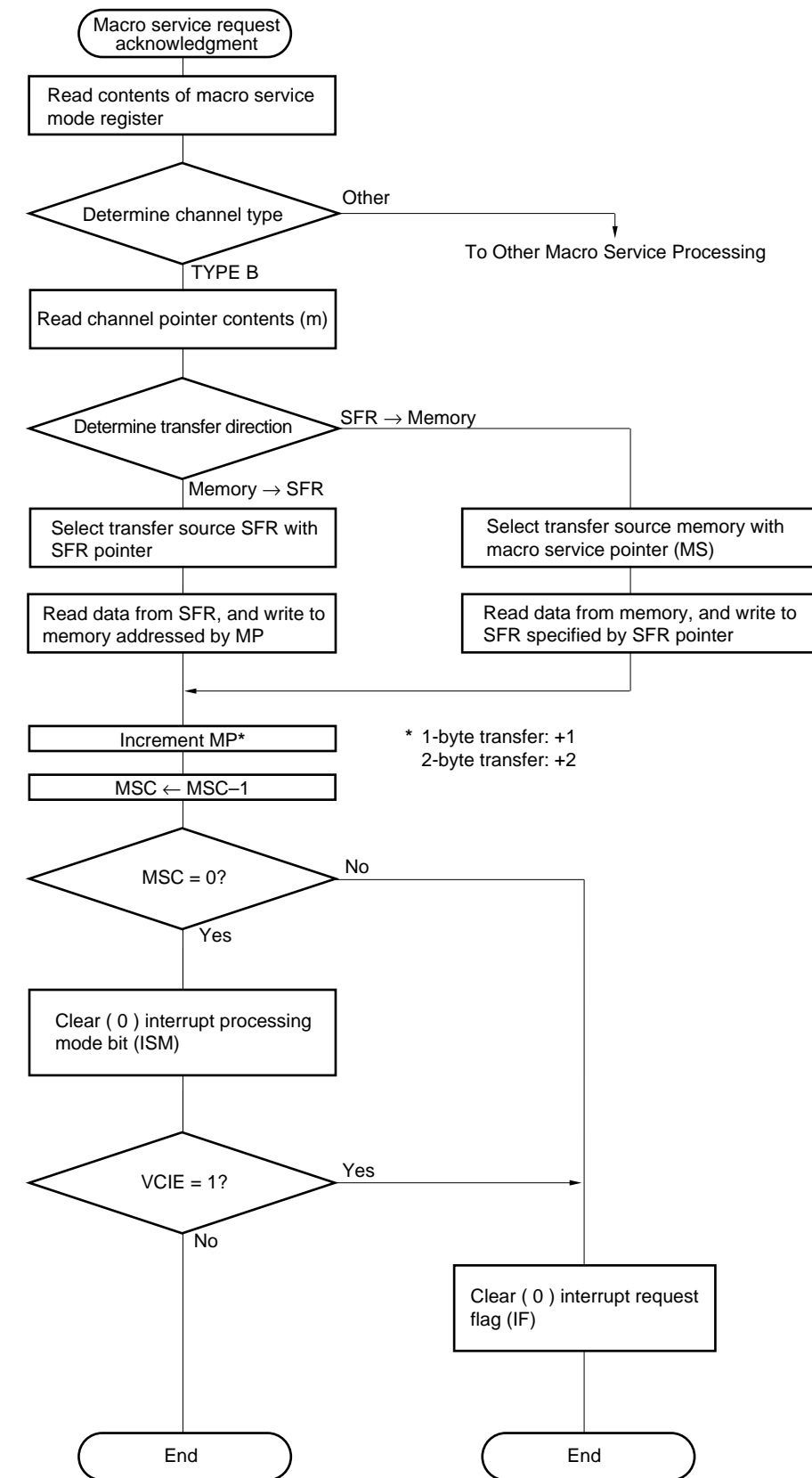
### 20.8.7 Macro Service Type B

#### (1) Operation

Data transfers are performed between a data area in memory and an SFR specified by the macro service channel. With type B, the data transfer direction can be selected as memory-to-SFR or SFR-to-memory.

Data transfers are performed the number of times set beforehand in the macro service counter. One macro service processing transfers 8-bit or 16-bit data.

This type of macro service is macro service type A for general purposes and is ideal for processing a large amount of data because up to 64 K bytes of data buffer area when 8-bit data is transferred or 1 M bytes of data buffer area when 16-bit data is transferred can be set in any address space.

**Figure 20-25 Macro Service Data Transfer Processing Flow (Type B)**

(Vectored Interrupt Request Generation)



**(2) Macro service channel configuration**

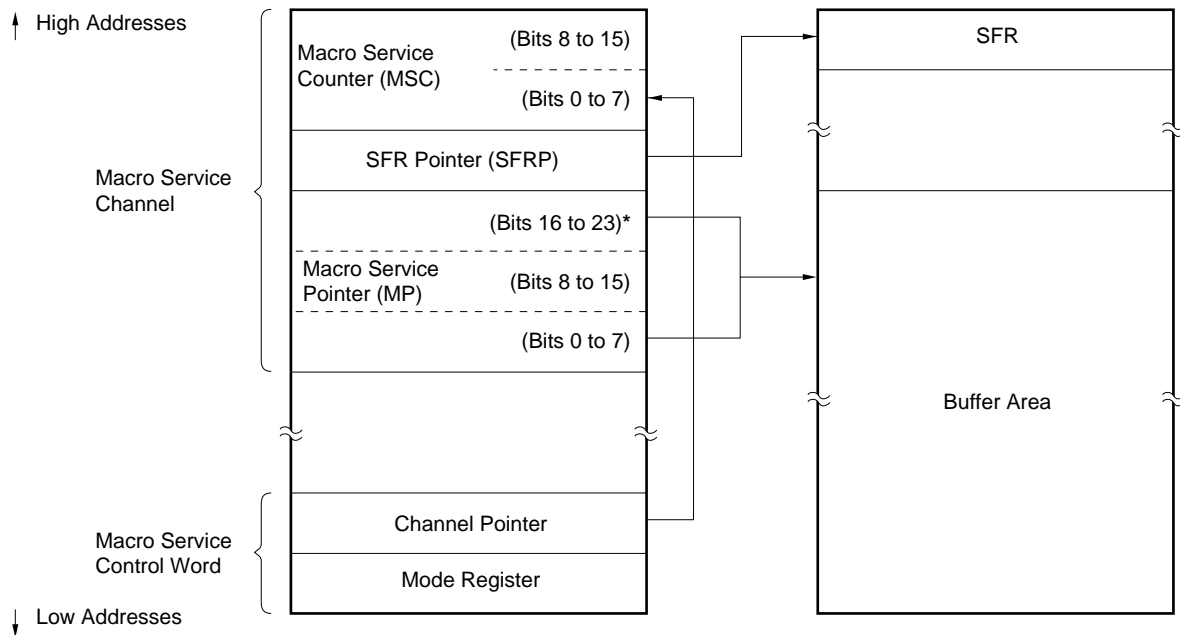
The macro service pointer (MP) indicates the data buffer area in the 1 M-byte memory space that is the transfer destination or transfer source.

The low-order 8 bits of the SFR that is the transfer destination or transfer source is written to the SFR pointer (SFRP).

The macro service counter (MSC) is a 16-bit counter that specifies the number of data transfers.

The macro service channel that stores the MP, SFRP and MSC is located in internal RAM space addresses 0FE00H to 0FEFFH when the LOCATION 0 instruction is executed, or 0FFE00H to 0FFEFFH when the LOCATION 0FH instruction is executed.

The macro service channel is indicated by the channel pointer as shown in Figure 20-26. In the channel pointer, the low-order 8 bits of the address are written to the macro service counter in the macro service channel.

**Figure 20-26 Type B Macro Service Channel**

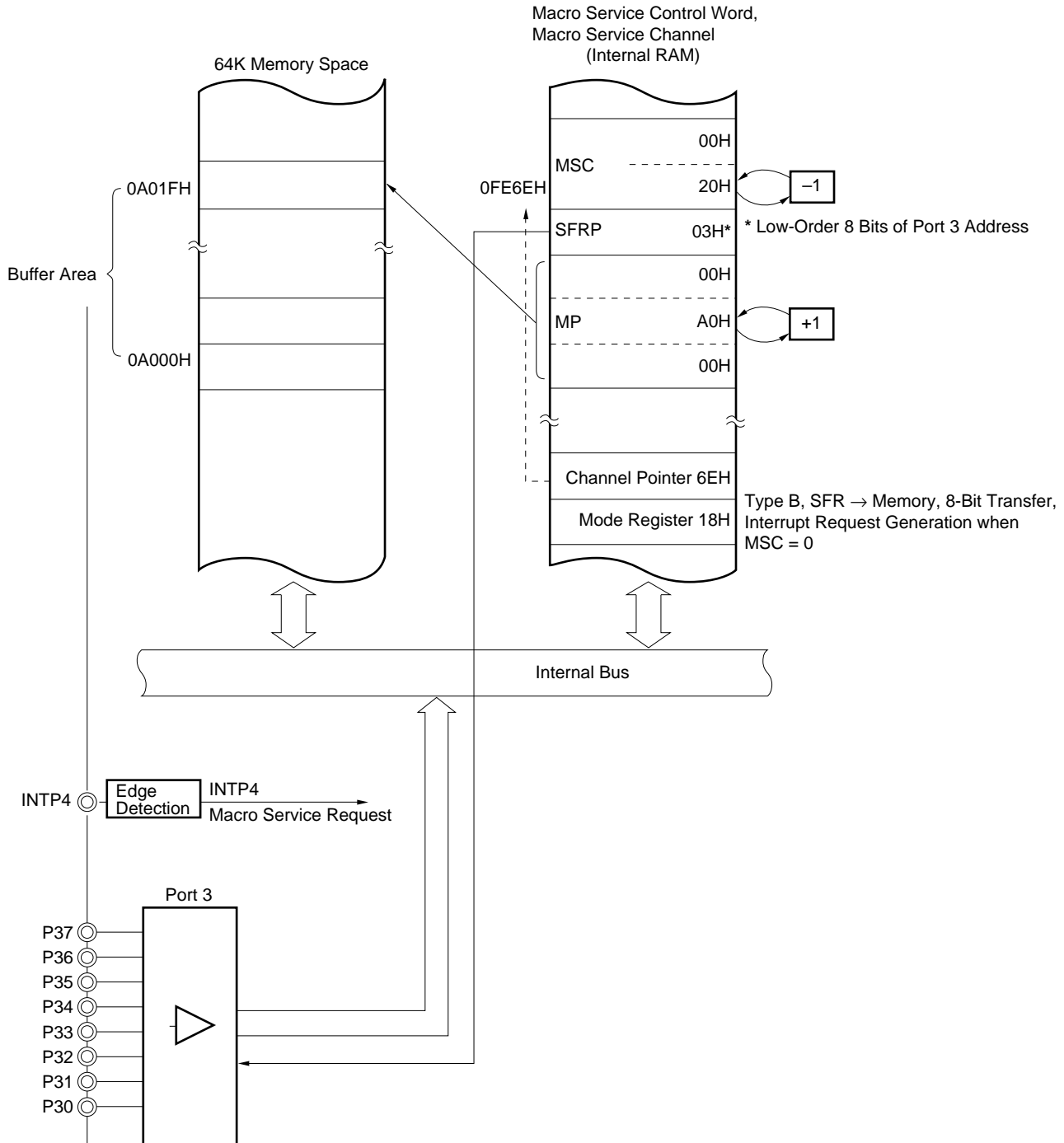
Macro service buffer address = macro service pointer

\* Bits 20 to 23 must be set to 0.

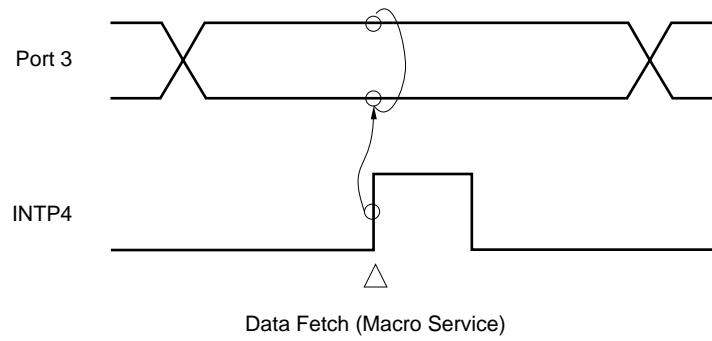
**(3) Example of use of type B**

An example is shown below in which parallel data is input from port 3 in synchronization with an external signal. The INTP4 external interrupt pin is used for synchronization with the external signal.

**Figure 20-27 Parallel Data Input Synchronized with External Interrupts**



**Remark** Macro service channel addresses in the figure are the values when the LOCATION 0 instruction is executed. When the LOCATION 0FH instruction is executed, 0F0000H should be added to the values in the figure.

**Figure 20-28 Parallel Data Input Timing**

### 20.8.8 Macro Service Type C

#### (1) Operation

In type C macro service, data in the memory specified by the macro service channel is transferred to two SFRs, for timer use and data use, specified by the macro service channel in response to a single interrupt request (the SFRs can be freely selected). An 8-bit or 16-bit timer SFR can be selected.

In addition to the basic data transfers described above, type C macro service, the following functions can be added to type C macro service to reduce the size of the buffer area and alleviate the burden on software.

These specifications are made by using the mode register of the macro service control word.

#### (a) Updating of timer macro service pointer

It is possible to choose whether the timer macro service pointer (MPT) is to be kept as it is or incremented/decremented. The MPT is incremented or decremented in the same direction as the macro service pointer (MPD) for data.

#### (b) Updating of data macro service pointer

It is possible to choose whether the data macro service pointer (MPD) is to be incremented or decremented.

#### (c) Automatic addition

The current compare register value is added to the data addressed by the timer macro service pointer (MPT), and the result is transferred to the compare register. If automatic addition is not specified, the data addressed by the MPT is simply transferred to the compare register.

#### (d) Ring control

An output data pattern of the length specified beforehand is automatically output repeatedly.

Figure 20-29 Macro Service Data Transfer Processing Flow (Type C) (1/2)

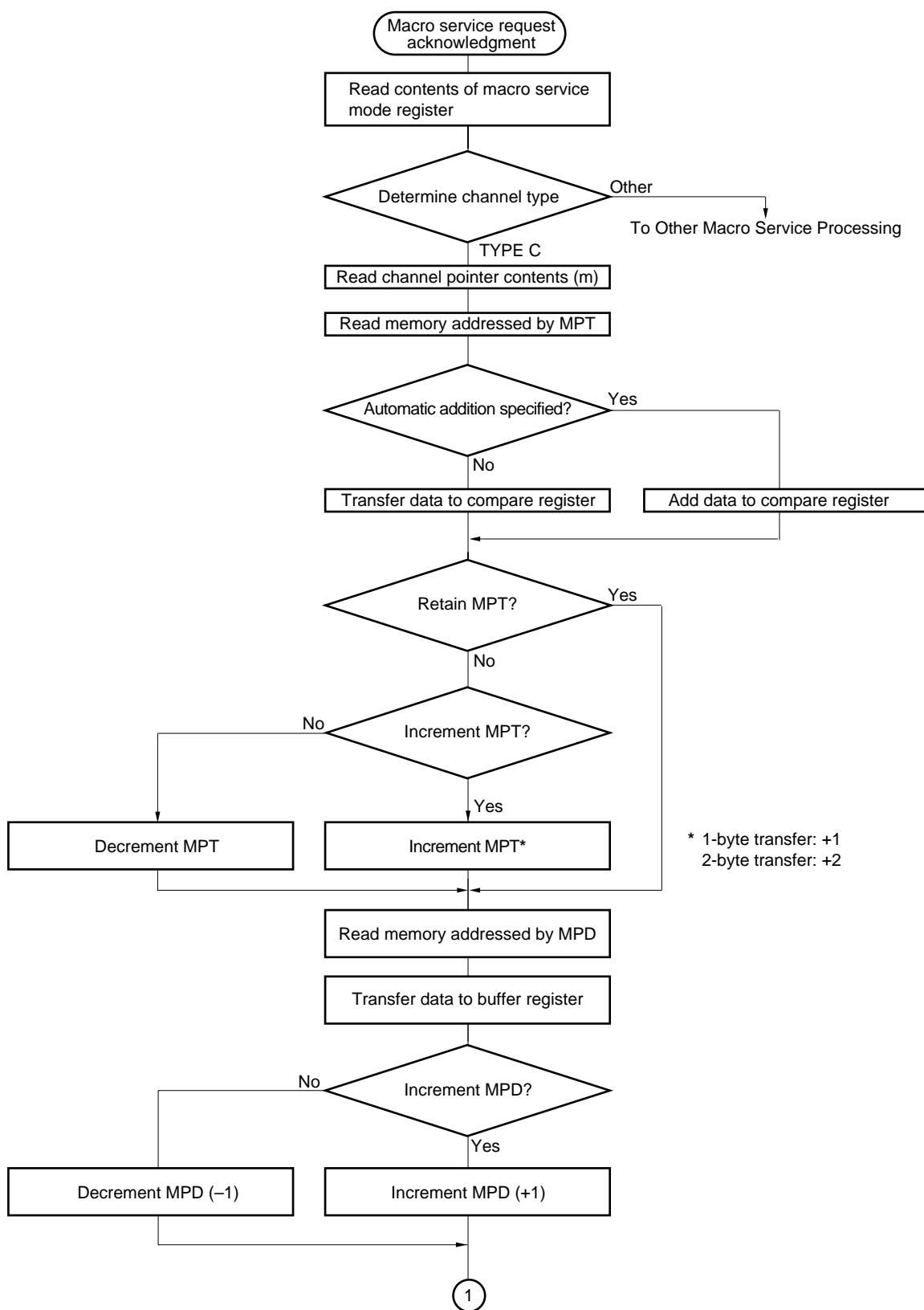
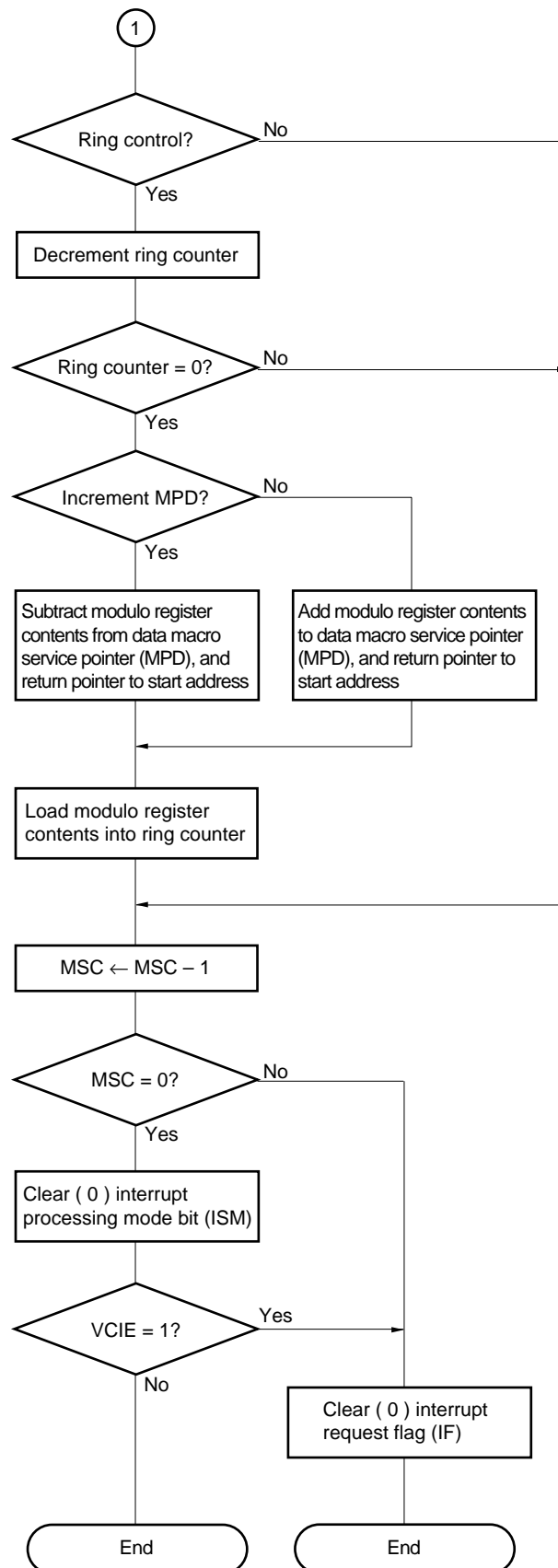


Figure 20-29 Macro Service Data Transfer Processing Flow (Type C) (2/2)



(Vectored Interrupt Request Generation)

**(2) Macro service channel configuration**

There are two kinds of type C macro service channel, as shown in Figure 20-30.

The timer macro service pointer (MPT) mainly indicates the data buffer area in the 1M-byte memory space to be transferred or added to the timer/counter compare register.

The data macro service pointer (MPD) indicates the data buffer area in the 1M-byte memory space to be transferred to the real-time output port.

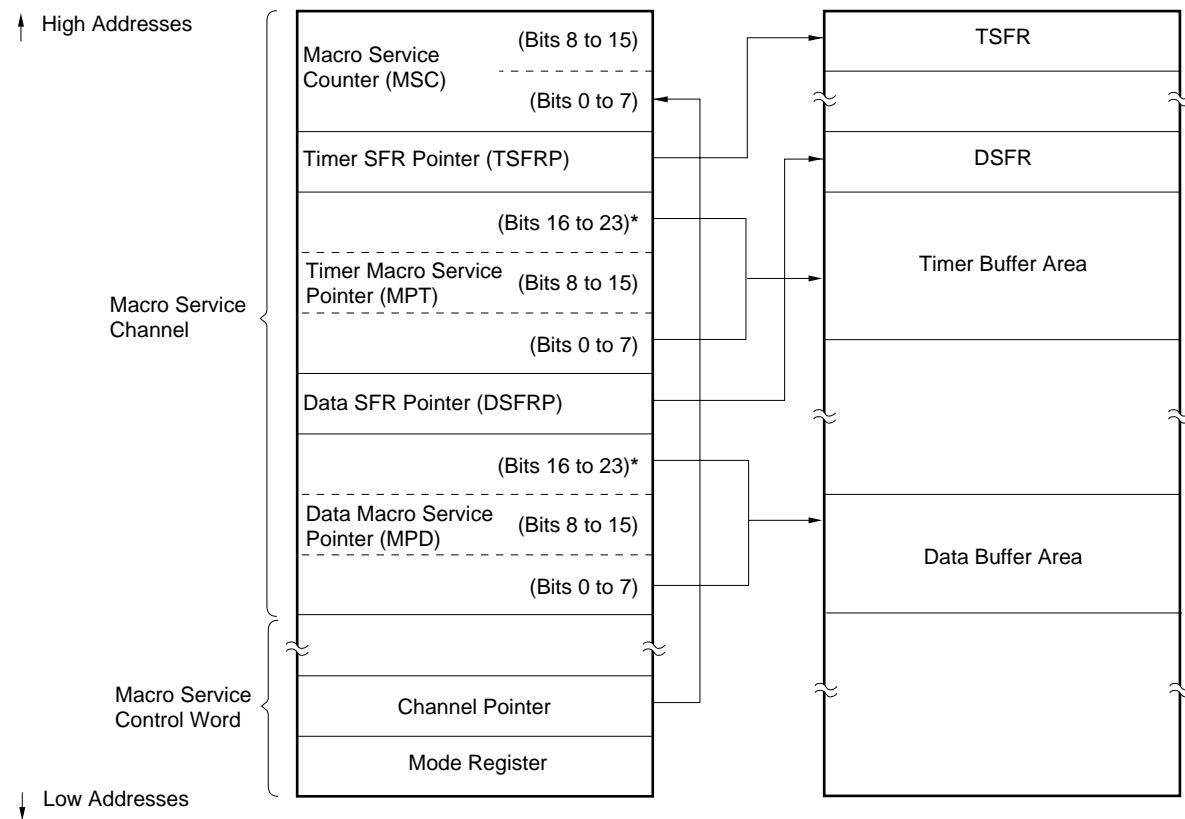
The modulo register (MR) specifies the number of repeat patterns when ring control is used.

The ring counter (RC) holds the step in the pattern when ring control is used. When initialization is performed, the same value as in the MR is normally set in this counter.

The macro service counter (MSC) is a 16-bit counter that specifies the number of data transfers.

The low-order 8 bits of the SFR that is the transfer destination is written to the timer SFR pointer (TSFRP) and data SFR pointer (DSFRP).

The macro service channel that stores these pointers and counters is located in internal RAM space addresses 0FE00H to 0FEFFH when the LOCATION 0 instruction is executed, or 0FFE00H to 0FFEFFH when the LOCATION 0FH instruction is executed. The macro service channel is indicated by the channel pointer as shown in Figure 20-30. In the channel pointer, the low-order 8 bits of the address are written to the macro service counter in the macro service channel.

**Figure 20-30 Type C Macro Service Channel (1/2)****(a) No ring control**

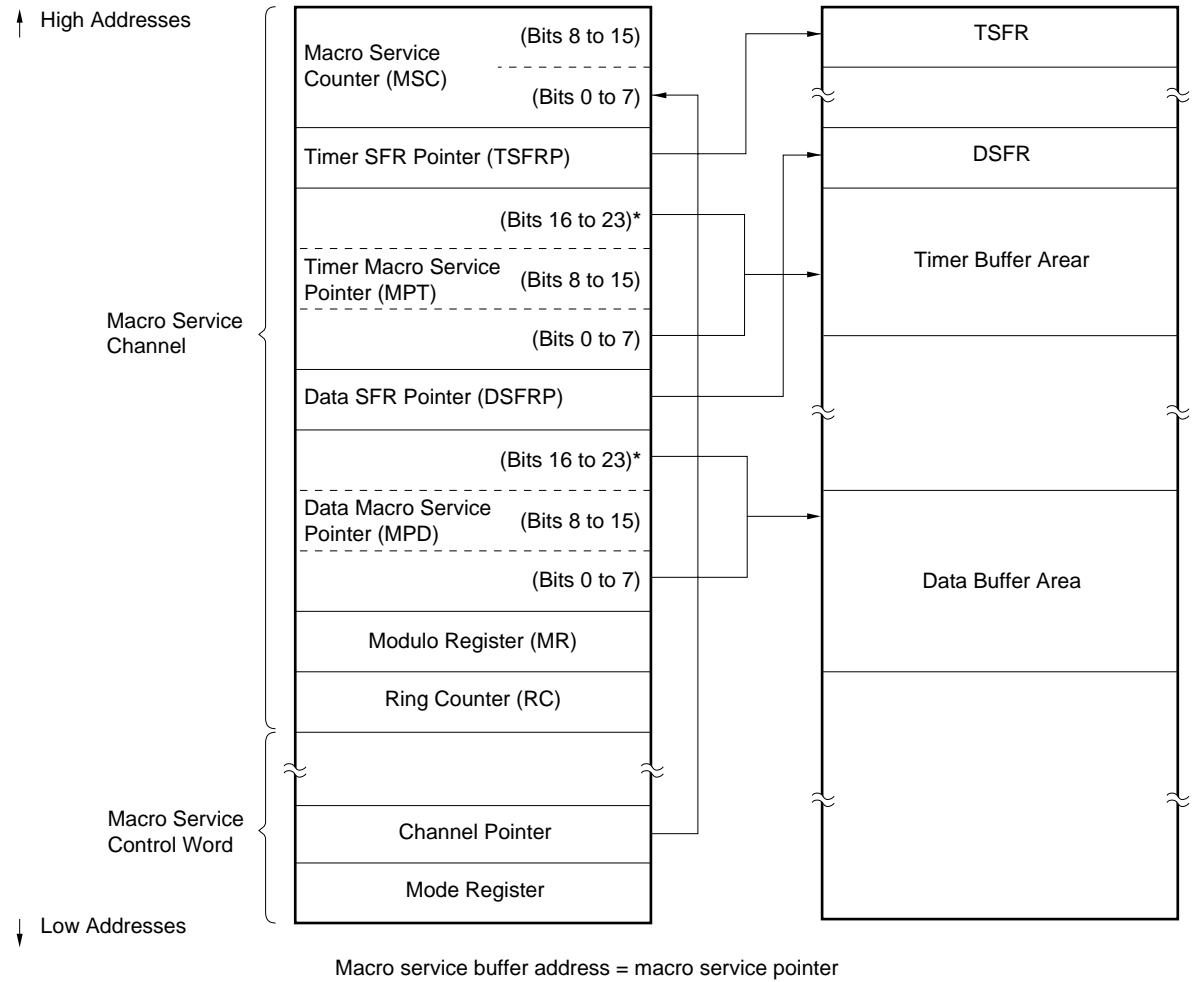
Macro service buffer address = macro service pointer

\* Bits 20 to 23 must be set to 0.



Figure 20-30 Type C Macro Service Channel (2/2)

**(b) With ring control**



\* Bits 20 to 23 must be set to 0.

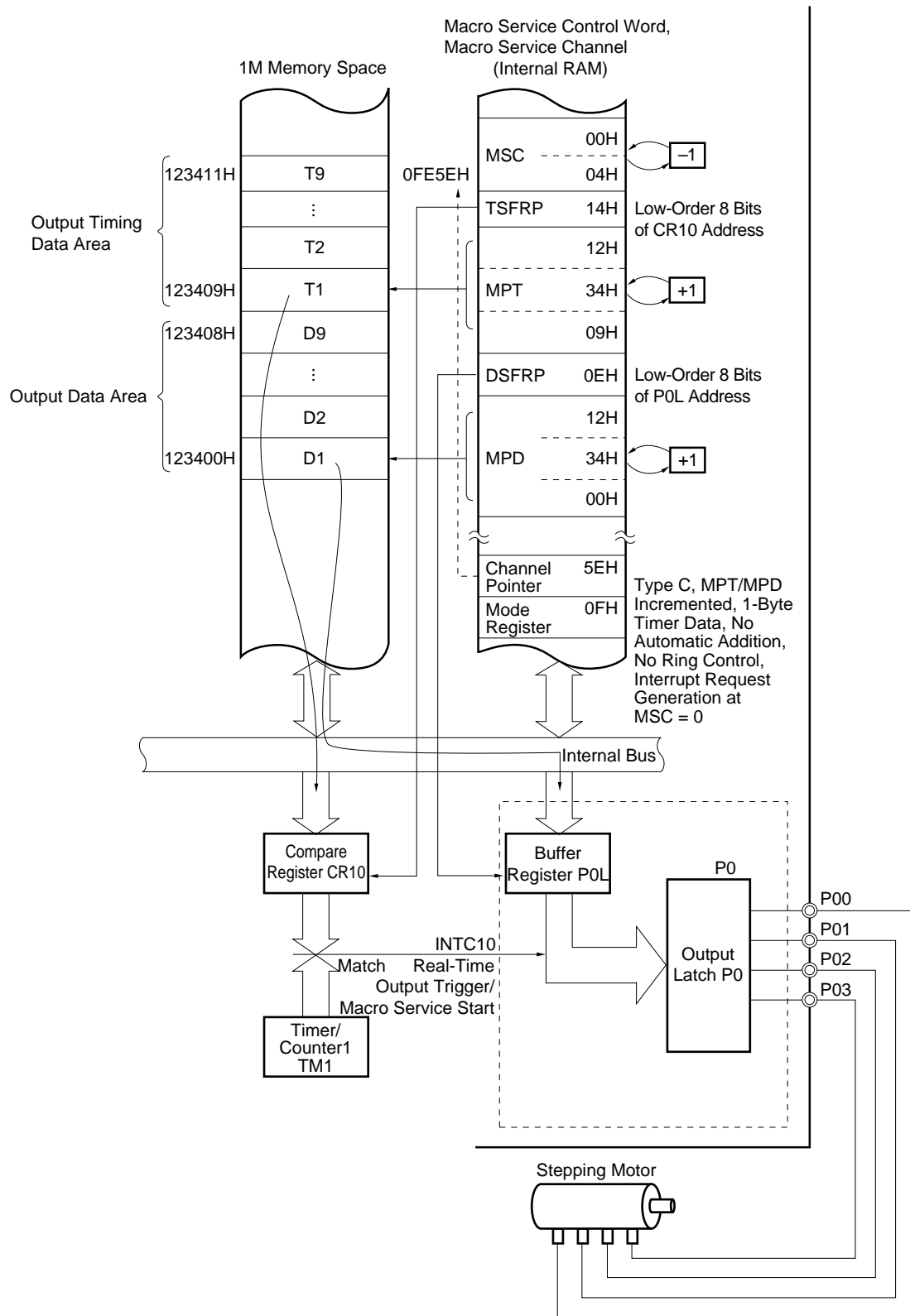
**(3) Examples of use of type C**

**(a) Basic operation**

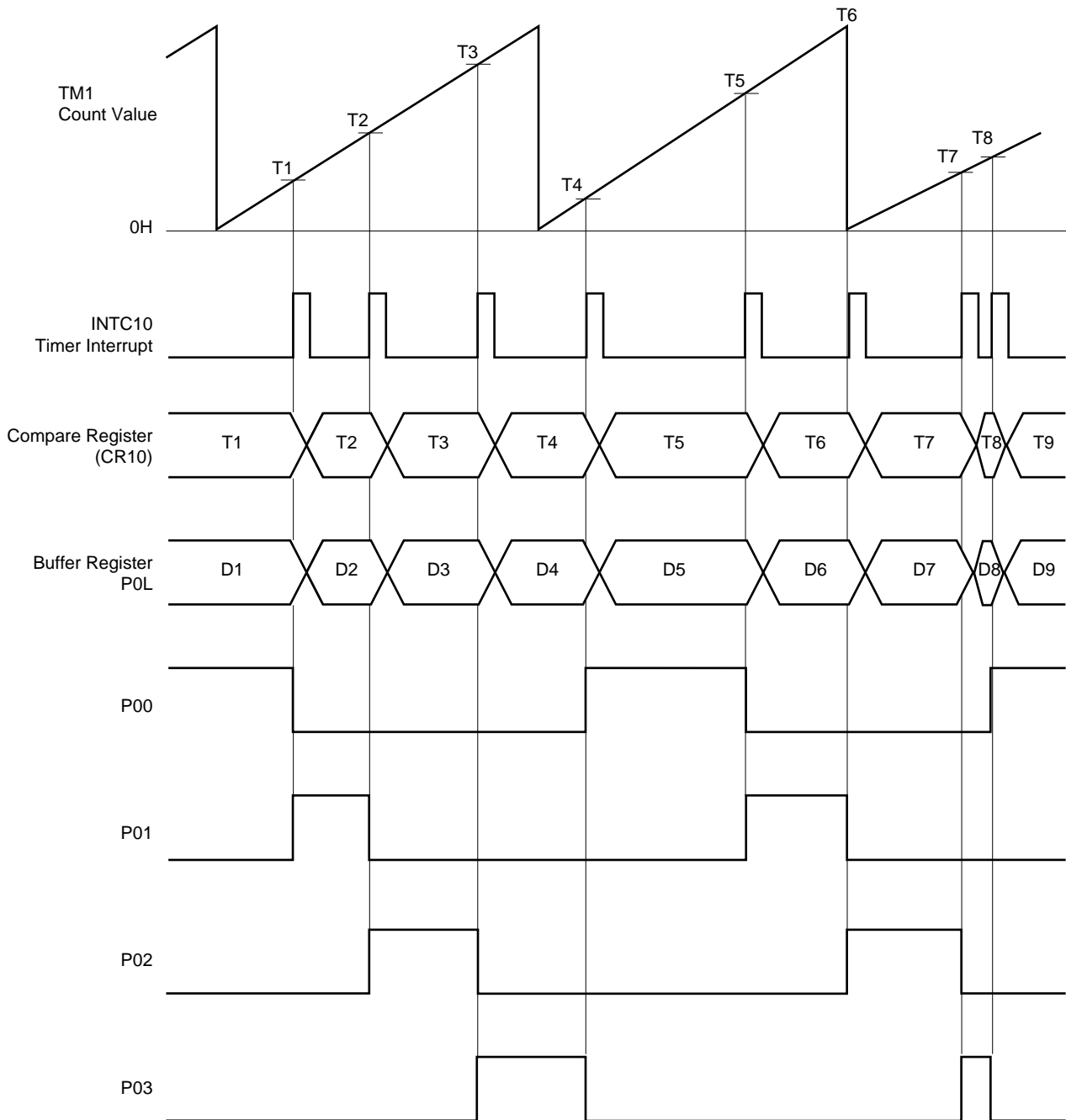
An example is shown below in which the output pattern to the real-time output port and the output interval are directly controlled.

Update data is transferred from the two data storage areas set in the 1 M-byte space beforehand to the real-time output function buffer register (POL) and the compare register (CR10).

Figure 20-31 Stepping Motor Open Loop Control by Real-Time Output Port



**Remark** Internal RAM addresses in the figure are the values when the LOCATION 0 instruction is executed. When the LOCATION 0FH instruction is executed, 0F0000H should be added to the values in the figure.

**Figure 20-32 Data Transfer Control Timing**

**(b) Examples of use of automatic addition control and ring control****(i) Automatic addition control**

The output timing data ( $\Delta t$ ) specified by the macro service pointer (MPT) is added to the contents of the compare register, and the result is written back to the compare register.

Use of this automatic addition control eliminates the need to calculate the compare register setting value in the program each time.

**(ii) Ring control**

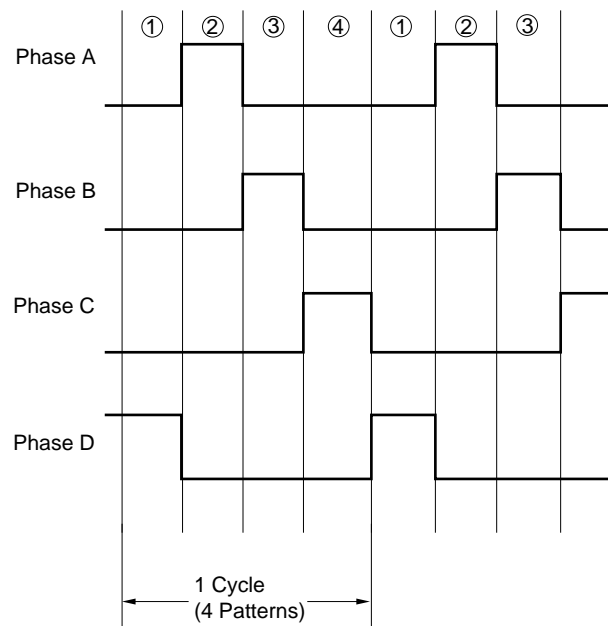
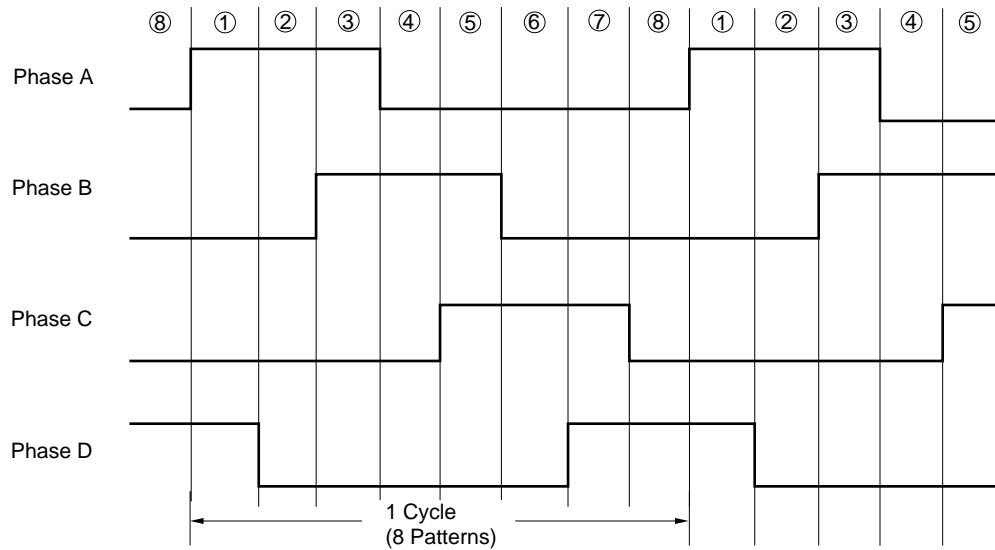
With ring control, the predetermined output patterns is prepared for one cycle only, and the one-cycle data patterns are output repeatedly in order in ring form.

When ring control is used, only the output patterns for one cycle need be prepared, allowing the size of the data ROM area to be reduced.

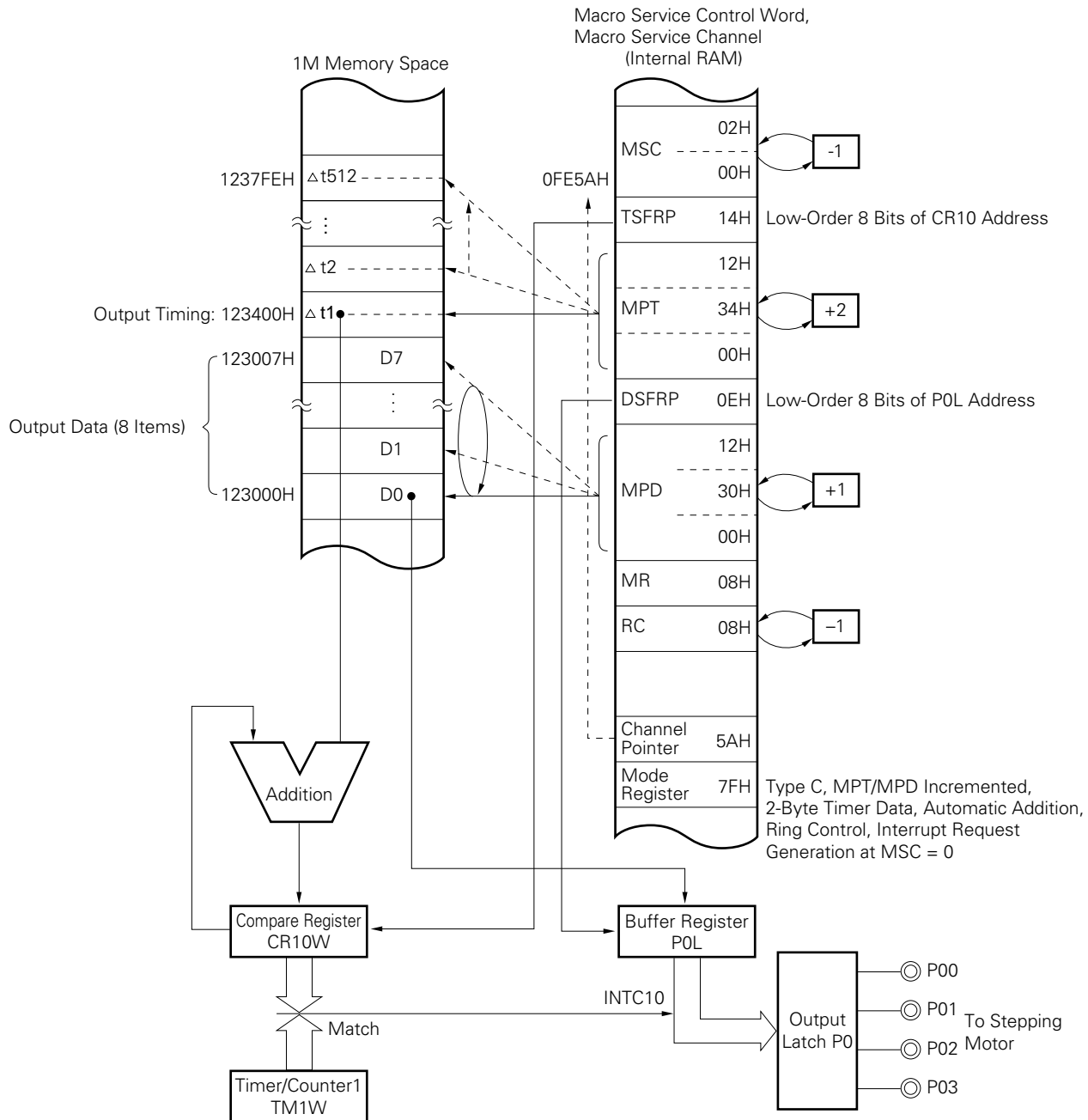
The macro service counter (MSC) is decremented each time a data transfer is performed.

With ring control, too, an interrupt request is generated when  $MSC = 0$ .

When controlling a stepping motor, for example, the output patterns will vary depending on the configuration of the stepping motor concerned, and the phase excitation method (single-phase excitation, two-phase excitation, etc.), but repeat patterns are used in all cases. Examples of single-phase excitation and 1-2-phase excitation of a 4-phase stepping motor are shown in Figures 20-33 and 20-34.

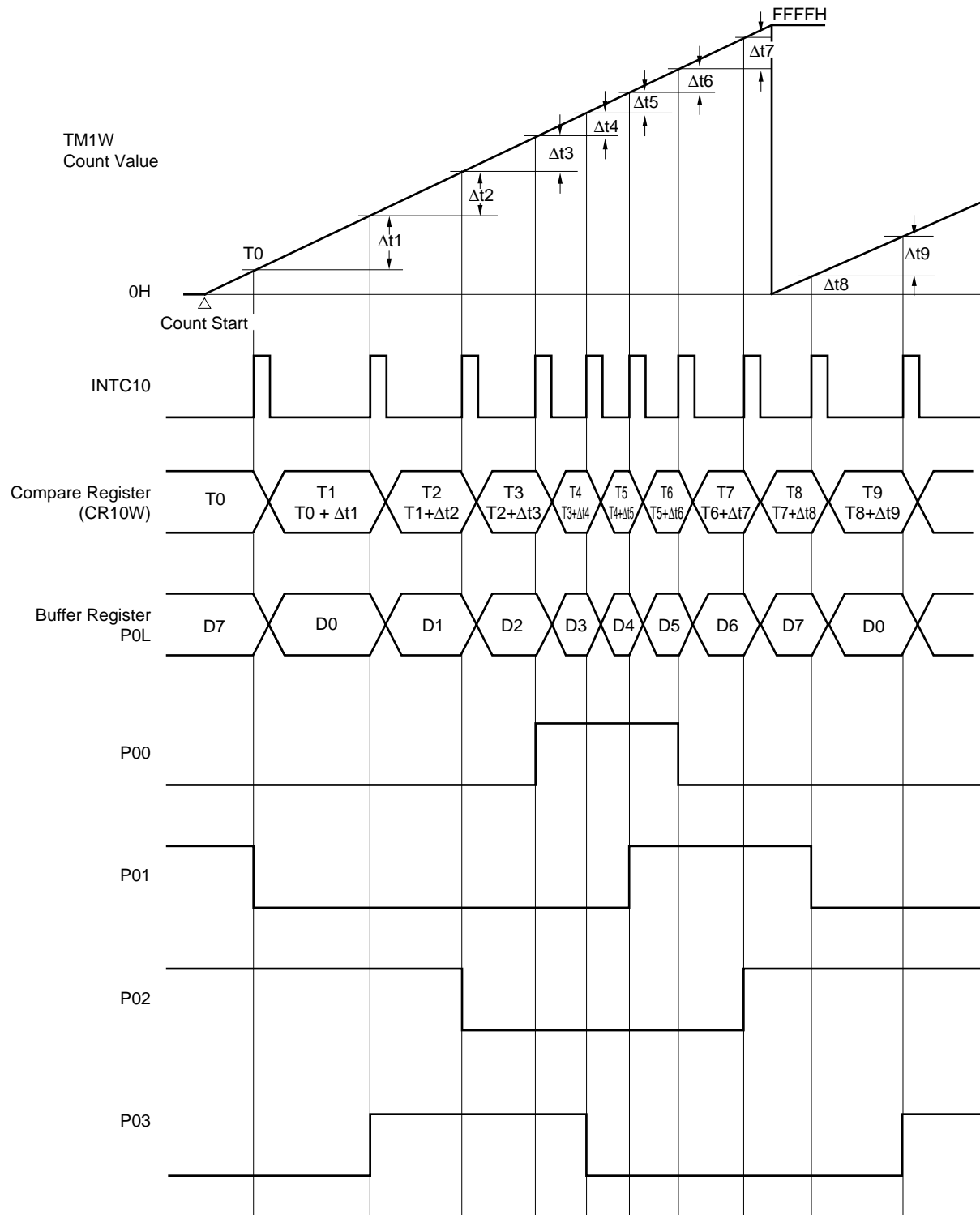
**Figure 20-33 Single-Phase Excitation of 4-Phase Stepping Motor****Figure 20-34 1-2-Phase Excitation of 4-Phase Stepping Motor**

**Figure 20-35 Automatic Addition Control + Ring Control Block Diagram 1**  
**(When Output Timing Varies with 1-2-Phase Excitation)**

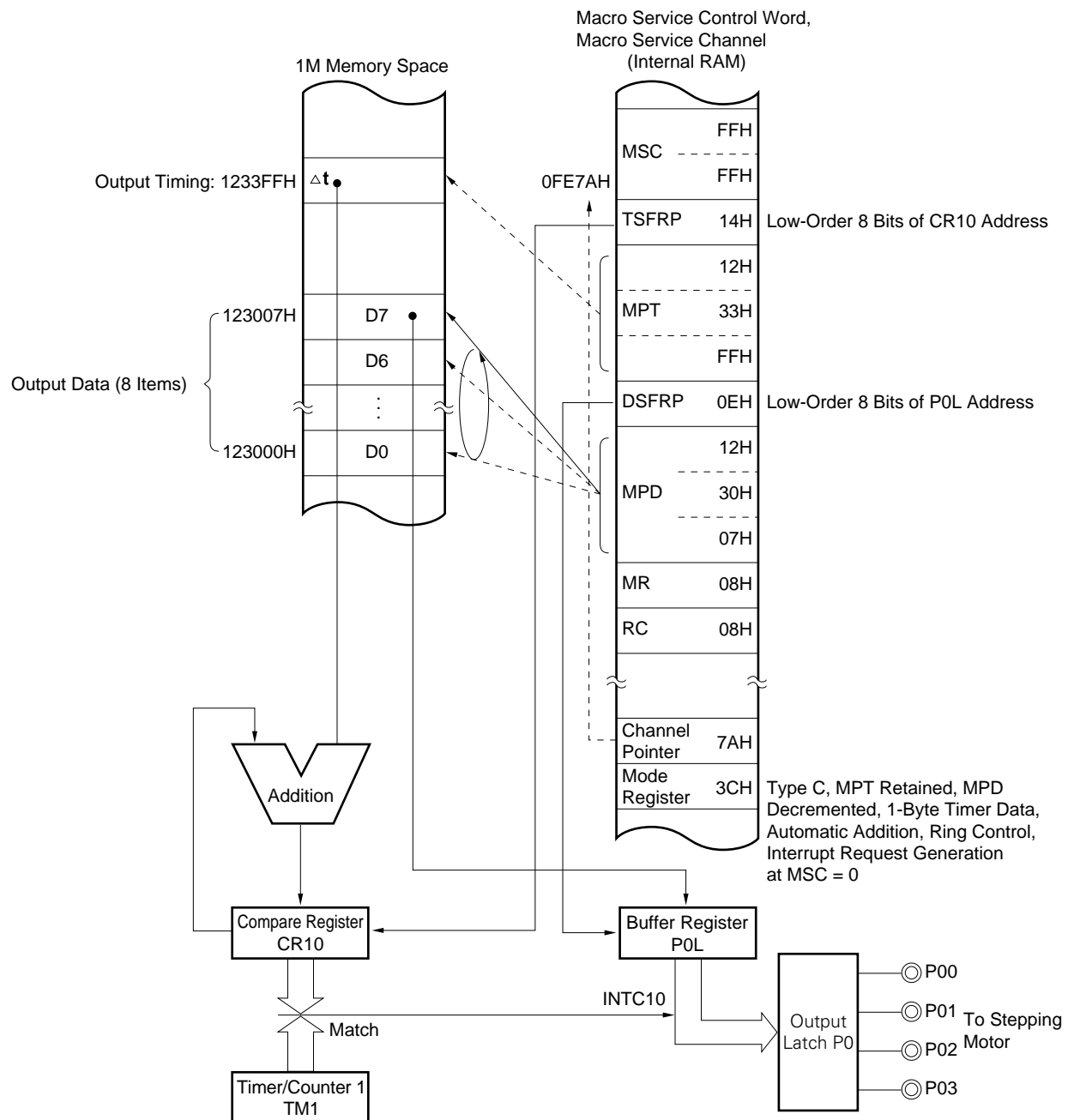


**Remark** Internal RAM addresses in the figure are the values when the LOCATION 0 instruction is executed. When the LOCATION 0FH instruction is executed, 0F0000H should be added to the values in the figure.

**Figure 20-36 Automatic Addition Control + Ring Control Timing Diagram 1**  
**(When Output Timing Varies with 1-2-Phase Excitation)**



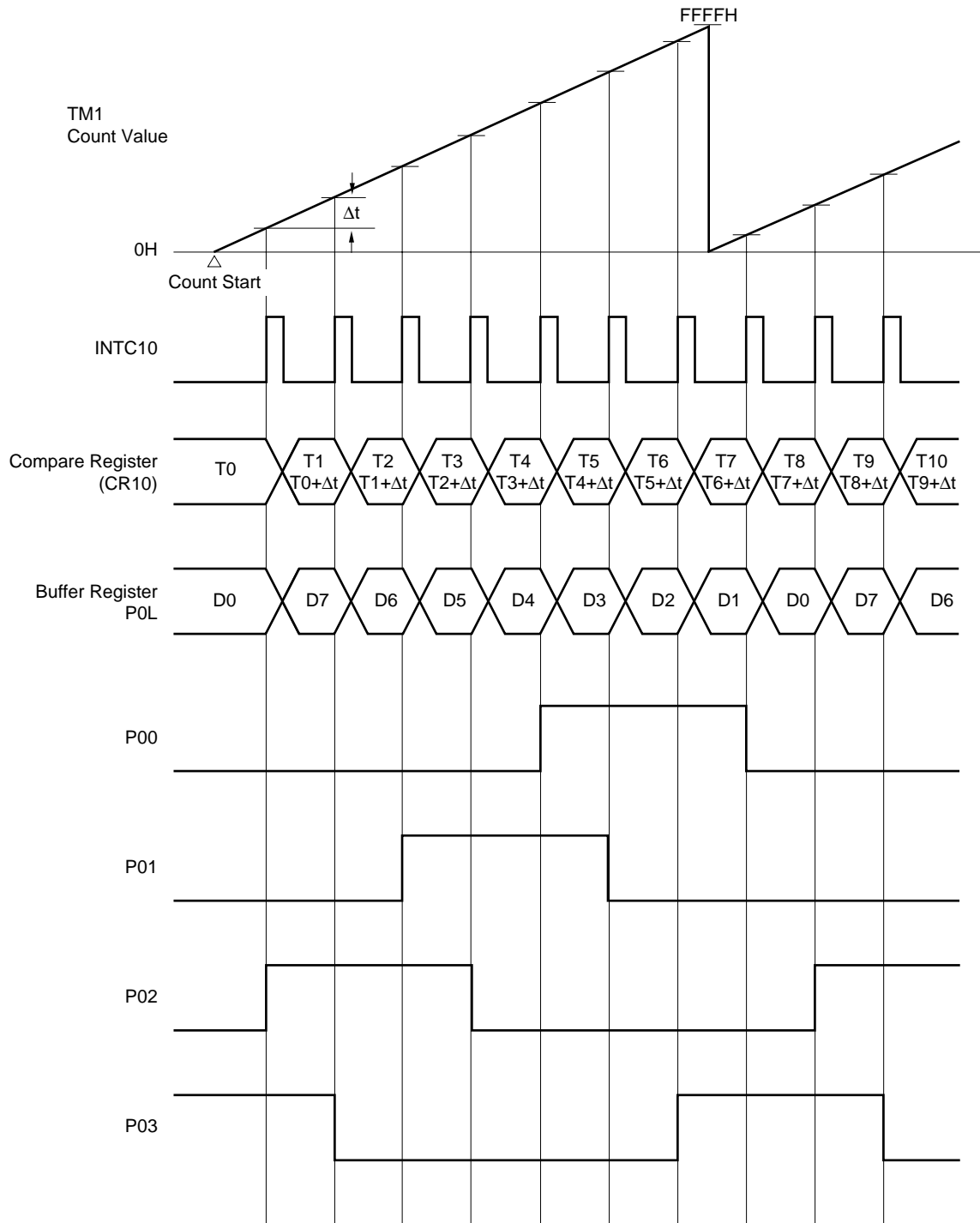
**Figure 20-37 Automatic Addition Control + Ring Control Block Diagram 2**  
**(1-2-Phase Excitation Constant-Velocity Operation)**



**Remark** Internal RAM addresses in the figure are the values when the LOCATION 0 instruction is executed. When the LOCATION 0FH instruction is executed, 0F0000H should be added to the values in the figure.



**Figure 20-38 Automatic Addition Control + Ring Control Timing Diagram 2**  
**(1-2-Phase Excitation Constant-Velocity Operation)**

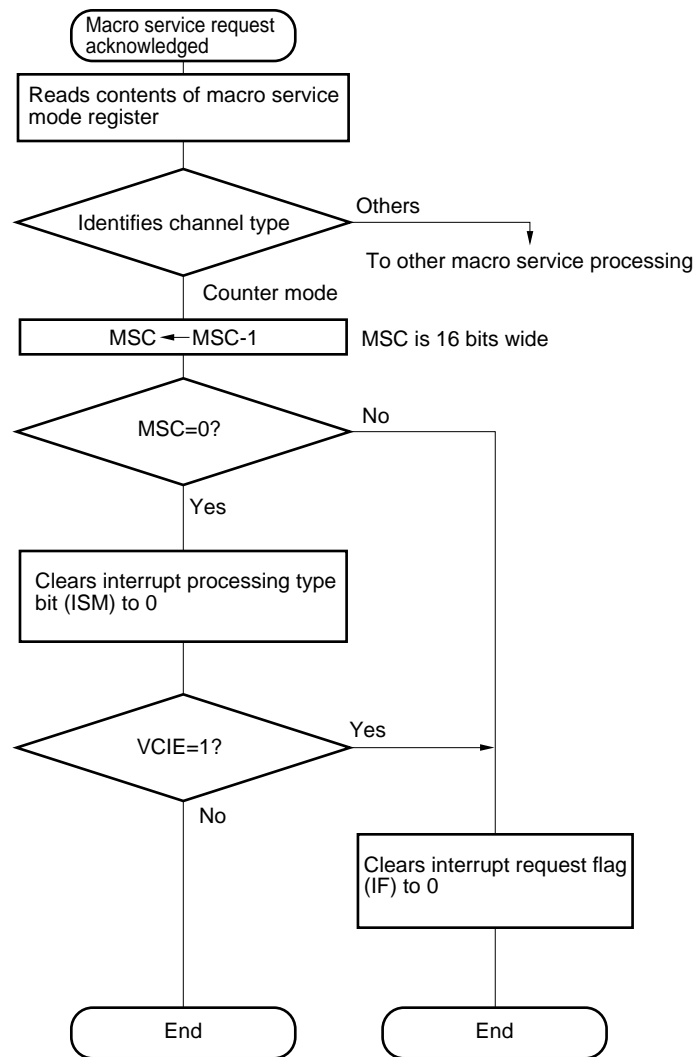


**20.8.9 Counter Mode****(1) Operation**

MSC is decremented the number of times set in advance to the macro service counter (MSC).

Because the number of times an interrupt occurs can be counted, this function can be used as an event counter where the interrupt generation cycle is long.

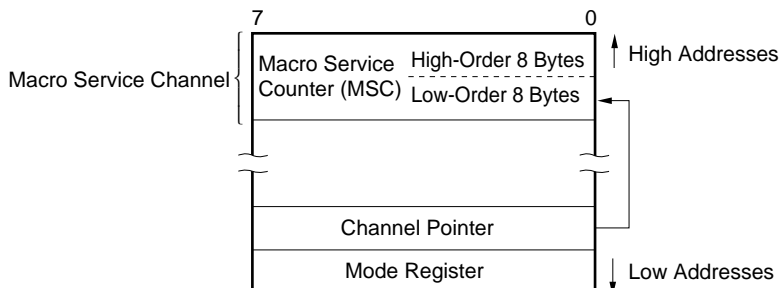
**Figure 20-39 Macro Service Data Transfer Processing Flow (Counter Mode)**



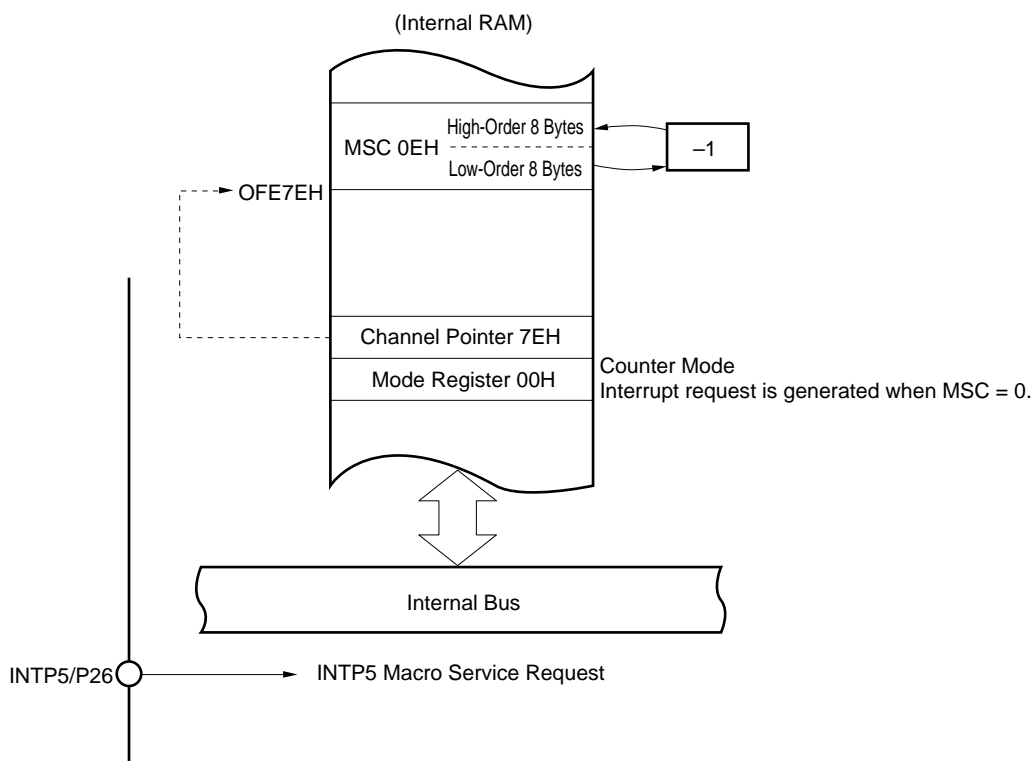
(Vectored interrupt request is generated)

**(2) Configuration of macro service channel**

The macro service channel consists of only a 16-bit macro service counter. The low-order 8 bits of the address of the MSC are written to the channel pointer.

**Figure 20-40 Counter Mode****(3) Example of using counter mode**

Here is an example of counting the number of edges input to external interrupt pin INTP5.

**Figure 20-41 Counting Number of Edges**

**Remark** The internal RAM address in the figure above is the value when the LOCATION 0 instruction is executed. When the LOCATION 0FH instruction is executed, add 0F0000H to this value.

## ★ 20.9 WHEN INTERRUPT REQUESTS AND MACRO SERVICE ARE TEMPORARILY HELD PENDING

When the following instructions are executed, interrupt acknowledgment and macro service processing is deferred for 8 system clock cycles. However, software interrupts are not deferred.

EI  
 DI  
 BRK  
 BRKCS  
 RETCS  
 RETCSB !addr16  
 RETI  
 RETB  
 LOCATION 0H or LOCATION 0FH  
 POP PSW  
 POP post  
 MOV PSWL, A  
 MOV PSWL, #byte  
 MOVG SP, #imm24

Write instruction and bit manipulation instruction for an interrupt control register\*, or the MK0, MK1L, IMC or ISPR register  
 (Excluding the BT and BF instructions)

PSW bit manipulation instruction

(Excluding the BT PSWL, bit, \$addr20, BF PSWL, bit, \$addr20, BT PSWH.bit, \$addr20, BF PSWH.bit, \$addr20, SET1 CY, NOT1 CY, and CLR1 CY instructions)

\* Interrupt control registers: PIC0, PIC1, PIC2, PIC3, PIC4, PIC5, CIC00, CIC01, CIC10, CIC11, CIC20, CIC21, CIC30, ADIC, SERIC, SRIC, CSIIC1, STIC, CSIIC, SERIC2, SRIC2, CSIIC2, STIC2

- Cautions 1.** When an interrupt related register is polled using a BF instruction, etc., the branch destination of that BR instruction, etc., should not be that instruction. If a program is written in which a branch is made to that instruction itself, all interrupts and macro service requests will be held pending until a condition whereby a branch is not made by that instruction arises.

Bad Example

```

:
:
LOOP : BF PIC0.7, $LOOP
:
:
xxx

```

All interrupts and macro service requests are held pending until PIC0.7 is 1.

← Interrupts and macro service requests are not processed until after execution of the instruction following the BF instruction.

Good Example (1)

```

:
:
LOOP : NOP
:
:
BF PIC0.7, $LOOP
:
:

```

← Interrupts and macro service requests are processed after execution of the NOP instruction, so that interrupts are never held pending for a long period.

Good Example (2)

```

:
:
LOOP : BT PIC0.7, $NEXT
:
:
BR $LOOP
:
NEXT :
:

```

Using a BTCLR instruction instead of a BT instruction has the advantage that the flag is cleared (0) automatically.

← Interrupts and macro service requests are processed after execution of the BR instruction, so that interrupts are never held pending for a long period.

- 2.** For a similar reason, if problems are caused by a long pending period for interrupts and macro service when instructions to which the above applies are used in succession, a time at which interrupts and macro service requests can be acknowledged should be provided by inserting an NOP instruction, etc., in the series of instructions.

**20.10 INSTRUCTIONS WHOSE EXECUTION IS TEMPORARILY SUSPENDED BY AN INTERRUPT OR MACRO SERVICE**

Execution of the following instructions is temporarily suspended by an acknowledgeable interrupt request or macro service request, and the interrupt or macro service request is acknowledged. The suspended instruction is resumed after completion of the interrupt service program or macro service processing.

Temporarily suspended instructions:

MOVM, XCHM, MOVBK, XCHBK  
 CMPME, CMPMNE, CMPMC, CMPMNC  
 CMPBKE, CMPBKNE, CMPBKC, CMPBKNC  
 SACW

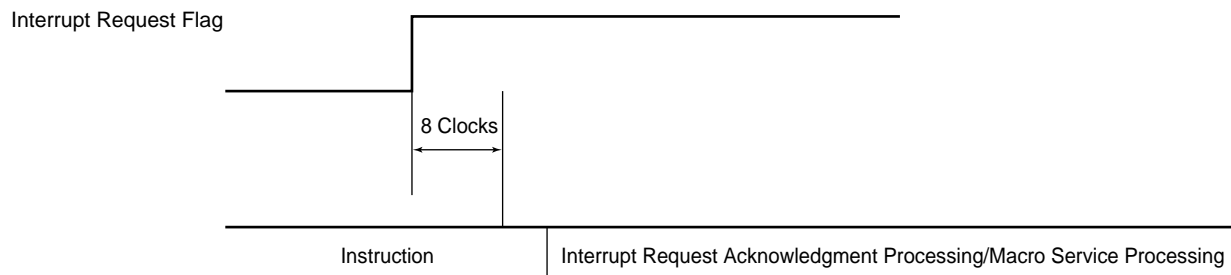
**20.11 INTERRUPT AND MACRO SERVICE OPERATION TIMING**

Interrupt requests are generated by hardware. The generated interrupt request sets (1) an interrupt request flag.

When the interrupt request flag is set (1), a time of 8 clocks ( $0.64 \mu\text{s}$ :  $f_{\text{CLK}} = 12.5 \text{ MHz}$ ) is taken to determine the priority, etc.

Following this, if acknowledgment of that interrupt or macro service is enabled, interrupt request acknowledgment processing is performed when the instruction being executed ends. If the instruction being executed is one which temporarily defers interrupts and macro service, the interrupt request is acknowledged after the following instruction (see **20.9 WHEN INTERRUPT REQUESTS AND MACRO SERVICE ARE TEMPORARILY HELD PENDING** for deferred instructions).

**Figure 20-42 Interrupt Request Generation and Acknowledgment (Unit: Clocks =  $1/f_{\text{CLK}}$ )**



**20.11.1 Interrupt Acknowledge Processing Time**

The time shown in Table 20-7 is required to acknowledge an interrupt request. After the time shown in this table has elapsed, execution of the interrupt processing program is started.

**Table 20-7 Interrupt Acknowledge Processing Time**(Unit: Clock =  $1/f_{CLK}$ )

Vector Table	IROM						EMEM					
Branch Destination	IROM, PRAM			EMEM			PRAM			EMEM		
Stack	IRAM	PRAM	EMEM	IRAM	PRAM	EMEM	IRAM	PRAM	EMEM	IRAM	PRAM	EMEM
Vectored Interrupts	26	29	$37 + 4n$	27	30	$38 + 4n$	30	33	$41 + 4n$	31	34	$42 + 4n$
Context Switching	22	–	–	23	–	–	22	–	–	23	–	–

**Remarks 1.** IROM : internal ROM (with high-speed fetch specified)

PRAM : peripheral RAM of internal RAM (only when LOCATION 0 instruction is executed in the case of branch destination)

IRAM : internal high-speed RAM

EMEM : internal ROM when external memory and high-speed fetch are not specified

- n is the number of wait states per byte necessary for writing data to the stack (the number of wait states is the sum of the number of address wait states and the number of access wait states).
- If the vector table is EMEM, and if wait states are inserted in reading the vector table, add 2 m to the value of the vectored interrupt in the above table, and add m to the value of context switching, where m is the number of wait states per byte necessary for reading the vector table.
- If the branch destination is EMEM and if wait states are inserted in reading the instruction at the branch destination, add that number of wait states.
- If the stack is occupied by PRAM and if the value of the stack pointer (SP) is odd, add 4 to the value in the above table.
- The number of wait states is the sum of the number of address wait states and the number of access wait states.

## 20.11.2 Processing Time of Macro Service

Macro service processing time differs depending on the type of the macro service, as shown in Table 20-8.

Table 20-8 Macro Service Processing Time

(Units: Clock =  $f_{CLK}$ )

Processing Type of Macro Service			Data Area	
			IRAM	Others
Type A	SFR → memory	1 byte	24	–
		2 bytes	25	–
	Memory → SFR	1 byte	24	–
		2 bytes	26	–
Type B	SFR → memory		33	35
	Memory → SFR		34	36
Type C			49	53
Counter mode	MSC ≠ 0		17	–
	MSC = 0		25	–

★

- Remarks**
1. IRAM: internal high-speed RAM
  2. In the following cases in the other data areas, add the number of clocks specified below.
    - If the data size is 2 bytes with IROM or IRAM, and the data is located at an odd address: 4 clocks
    - If the data size is 1 byte with EMEM: number of wait states for data access
    - If the data size is 2 bytes with EMEM:  $4 + 2n$  (where  $n$  is the number of wait states per byte)
  3. If MSC = 0 with type A, B, or C, add 1 clock.
  4. With type C, add the following value depending on the function to be used and the status at that time.
    - Ring control: 4 clocks. Adds 7 more clocks if the ring counter is 0 during ring control.



**20.12 RESTORING INTERRUPT FUNCTION TO INITIAL STATE**

If an inadvertent program loop or system error is detected by means of an operand error interrupt, the watchdog timer, NMI pin input, etc., the entire system must be restored to its initial state. In the  $\mu$ PD784026, interrupt acknowledgment related priority control is performed by hardware. This interrupt acknowledgment related hardware must also be restored to its initial state, otherwise subsequent interrupt acknowledgment control may not be performed normally.

A method of initializing interrupt acknowledgment related hardware in the program is shown below. The only way of performing initialization by hardware is by RESET input.

**Example**

```

MOVW MK0, #0FFFFH ; Mask all maskable interrupts
MOV  MK1L, #0FFH
IRESL :
CMP  ISPR, #0      ; No interrupt service programs running?
BZ   $NEXT
MOVG SP, #RETVL    ; Forcibly change SP location
RETI                               ; Forcibly terminate running interrupt service program, return
                                address = IRESL

RETVL :
DW   LOWW (IRESL)  ; Stack data to return to IRESL with RETI instruction
DB   0
DB   HIGHW (IRESL) ; LOWW & HIGHW are assembler operators for calculating low-order
                                16 bits & high-order 16 bits respectively of symbol NEXT

NEXT :

```

- It is necessary to ensure that a non-maskable interrupt request is not generated via the NMI pin during execution of this program.
- After this, internal peripheral hardware initialization and interrupt control register initialization are performed.
- When interrupt control register initialization is performed, the interrupt request flags must be cleared (0).

**20.13 CAUTIONS**

- (1) The in-service priority register (ISPR) is read-only. Writing to this register may result in misoperation.
- (2) The watchdog timer mode register (WDM) can only be written to with a dedicated instruction (MOV WDM, #byte).
- (3) The RETI instruction must not be used to return from a software interrupt caused by a BRK instruction. The RETB instruction must be used.
- (4) The RETCS instruction must not be used to return from a software interrupt caused by a BRKCS instruction. The RETCSB instruction must be used.
- (5) When a maskable interrupt is acknowledged by vectored interrupt, the RETI instruction must be used to return from the interrupt. Subsequent interrupt related operations will not be performed normally if a different instruction is used.
- (6) The RETCS instruction must be used to return from a context switching interrupt. Subsequent interrupt related operations will not be performed normally if a different instruction is used.
- (7) Macro service requests are acknowledged and processed even during execution of a non-maskable interrupt service program. If you do not want macro service processing to be performed during a non-maskable interrupt service program, you should manipulate the interrupt mask register in the non-maskable interrupt service program to prevent macro service generation.
- (8) The RETI instruction must be used to return from a non-maskable interrupt. Subsequent interrupt acknowledgment will not be performed normally if a different instruction is used. For details on how to resume program execution from the initial status after the non-maskable interrupt has been acknowledged, see **20.12 RESTORING INTERRUPT FUNCTION TO INITIAL STATE**.
- (9) Non-maskable interrupts are always acknowledged, except during non-maskable interrupt service program execution (except when a high non-maskable interrupt request is generated during execution of a low-priority non-maskable interrupt service program) and for a certain period after execution of the special instructions shown in **20.9**. Therefore, a non-maskable interrupt will be acknowledged even when the stack pointer (SP) value is undefined, in particular after reset release, etc. In this case, depending on the value of the SP, it may happen that the program counter (PC) and program status word (PSW) are written to the address of a write-inhibited special function register (SFR) (see **Table 3-5 in 3.9 SPECIAL FUNCTION REGISTERS (SFR)**), and the CPU becomes deadlocked, or the PC and PSW are written to an unexpected signal is output from a pin, or an address is which RAM is not mounted, with the result that the return from the non-maskable interrupt service program is not performed normally and an inadvertent program loop occurs.

Therefore, the program following  $\overline{\text{RESET}}$  release must be as follows.

```

CSEG AT 0
DW STRT
CSEG BASE
STRT:
LOCATION 0FH; or LOCATION 0
MOVG SP, #imm24

```

- (10) When an interrupt related register is polled using a BF instruction, etc., the branch destination of that BR instruction, etc., should not be that instruction. If a program is written in which a branch is made to that instruction itself, all interrupts and macro service requests will be held pending until a condition whereby a branch is not made by that instruction arises.

## Bad Example

```

      .
      .
LOOP: BF PIC0.7, $LOOP

```

All interrupts and macro service requests are held pending until PIC0.7 is 1.

```

      .
      .
      .

```

← Interrupts and macro service requests are not processed until after execution of the instruction following the BF instruction.

## Good Example (1)

```

      .
      .
LOOP: NOP
      BF PIC0.7, $LOOP
      .
      .

```

← Interrupts and macro processed requests are processed after execution of the NOP instruction, so that interrupts are never held pending for a long period.

## Good Example (2)

```

      .
      .
LOOP: BT PIC0.7, $NEXT
      .
      .
      BR $LOOP
      .
      .
NEXT:

```

Using a BTCLR instruction instead of a BT instruction has the advantage that the flag is cleared (0) automatically.

← Interrupts and macro service requests are processed after execution of the BR instruction, so that interrupts are never held pending for a long period.

- (11) For a similar reason to that given in (10), if problems are caused by a long pending period for interrupts and macro service when instructions to which the above applies are used in succession, a time at which interrupts and macro service requests can be acknowledged should be provided by inserting an NOP instruction, etc., in the series of instructions.

[MEMO]

## CHAPTER 21 LOCAL BUS INTERFACE FUNCTION

The local bus interface function is provided for the connection of external memory (ROM and RAM) and I/Os.

External memory (ROM and RAM) and I/Os are accessed using the  $\overline{RD}$ ,  $\overline{WR}$  and ASTB pin signals, with pins AD0 to AD7 used as the multiplexed address/data bus and pins A8 to A19 as the address bus.

The basic bus interface timing is shown in Figures 21-5 and 21-6.

Also provided are a wait function for interfacing with low-speed memory, a refresh signal output function for refreshing pseudo-static RAM, and a bus hold function for connecting devices that have a bus master function, such as a DMA controller.

### 21.1 MEMORY EXTENSION FUNCTION

With the  $\mu$ PD784026, external memory and I/O extension can be performed by setting the memory extension mode register (MM).

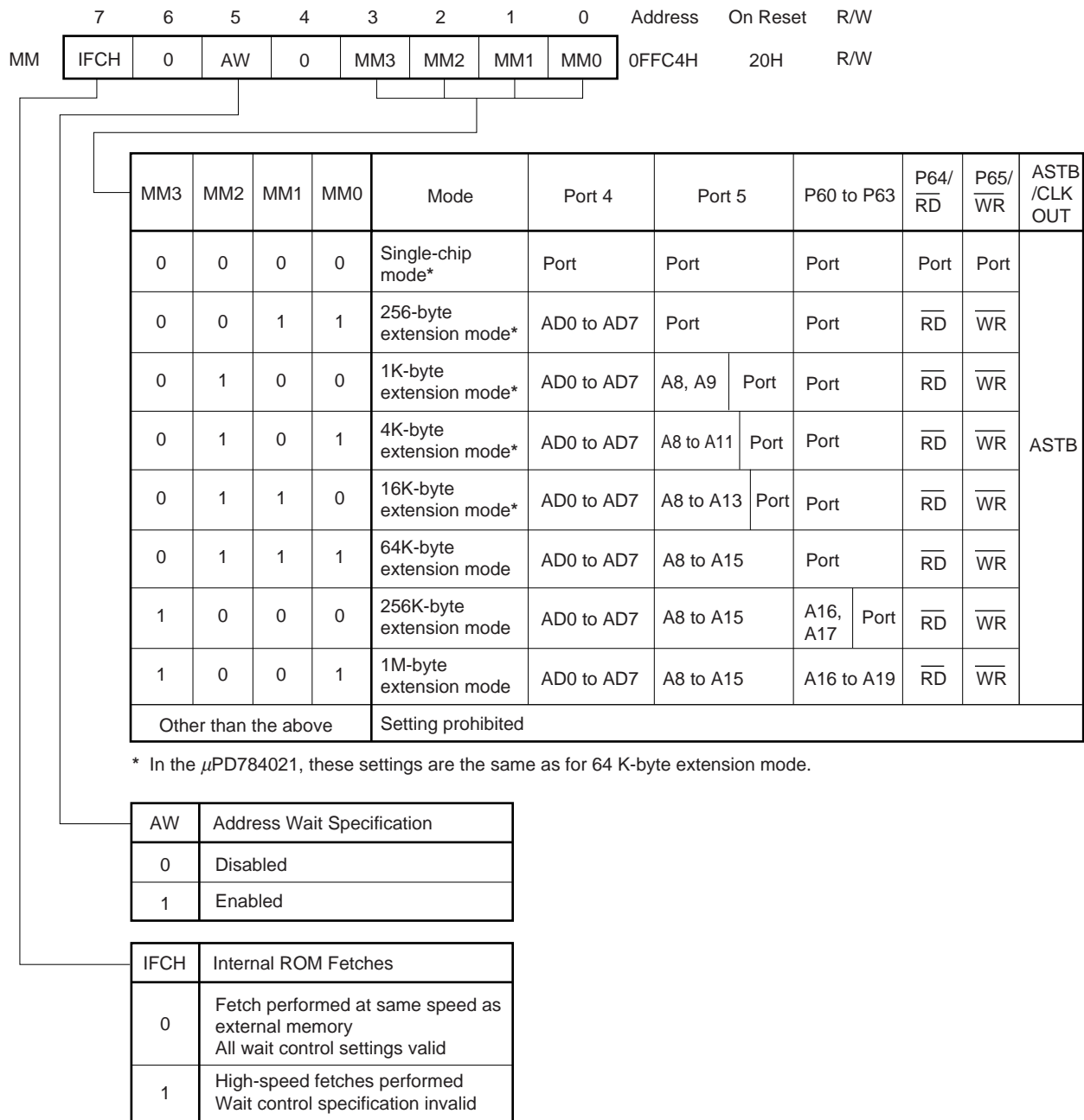
The  $\mu$ PD784021 can access 64 K-byte external memory from the initial state, and the accessible memory space can be extended by setting the MM.

#### 21.1.1 Memory Extension Mode Register (MM)

The MM is an 8-bit register that performs external extension memory control, address wait number specification, and internal fetch cycle control.

The MM register can be read or written to with an 8-bit manipulation instruction or bit manipulation instruction. The MM format is shown in Figure 21-1.

$\overline{RESET}$  input sets the MM register to 20H.

**Figure 21-1 Memory Extension Mode Register (MM) Format**

### 21.1.2 Memory Map with External Memory Extension

The memory map when memory extension is used is shown in Figures 21-2 to 21-5. External devices at the same addresses as the internal ROM area, internal RAM area and SFR area (excluding the external SFR area (0FFD0H to 0FFDFH)) cannot be accessed. If an access is made to these addresses, the memory or SFR in the  $\mu$ PD784026 has access priority and no  $\overline{\text{ASTB}}$  signal,  $\overline{\text{RD}}$  signal or  $\overline{\text{WR}}$  signal is output (these pins remain at the inactive level). The address bus output level remains at the level output prior to this, and the address/data bus output becomes high-impedance.

Except in 1M-byte extension mode, the address output externally is output with the upper part of the address specified by the program masked.

#### **Example 1:**

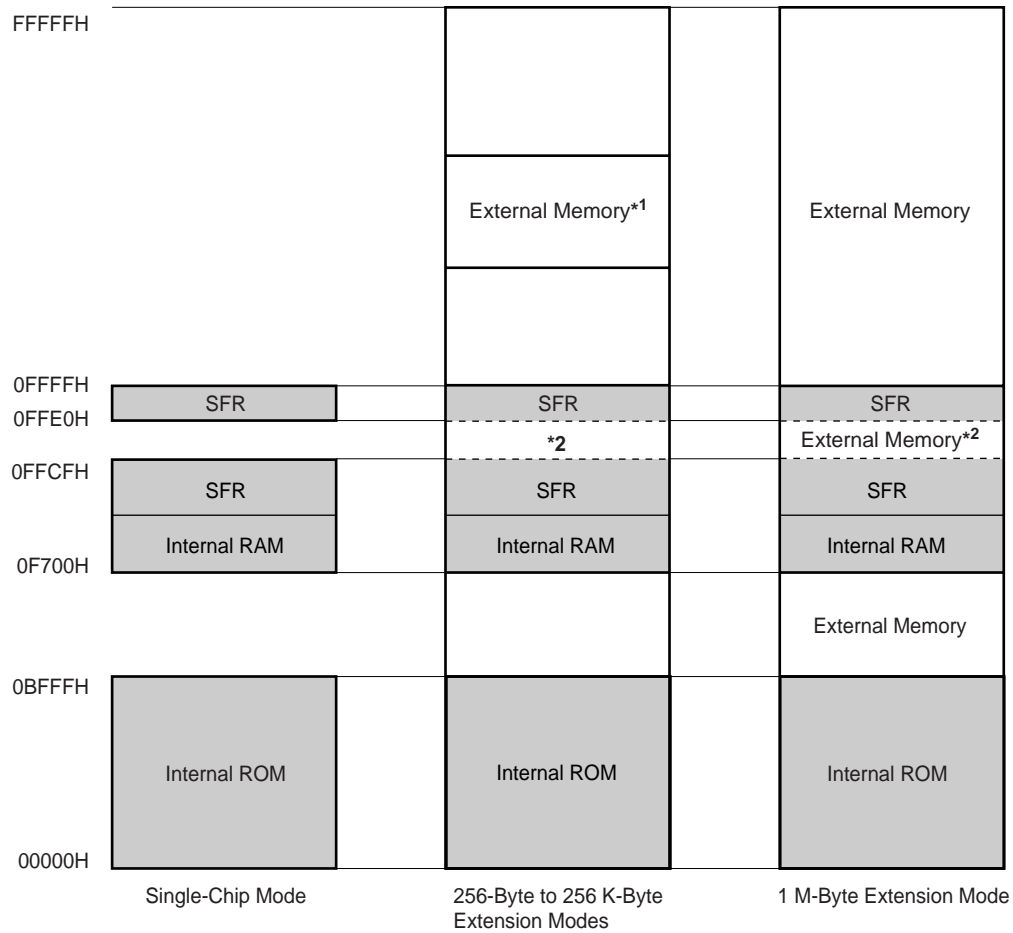
In 256-byte extension mode, when address 54321H is accessed by the program, the output address is 21H.

#### **Example 2:**

In 256-byte extension mode, when address 67821H is accessed by the program, the output address is 21H.

Figure 21-2  $\mu$ PD784025 Memory Map (1/2)

(a) When LOCATION 0 instruction is executed



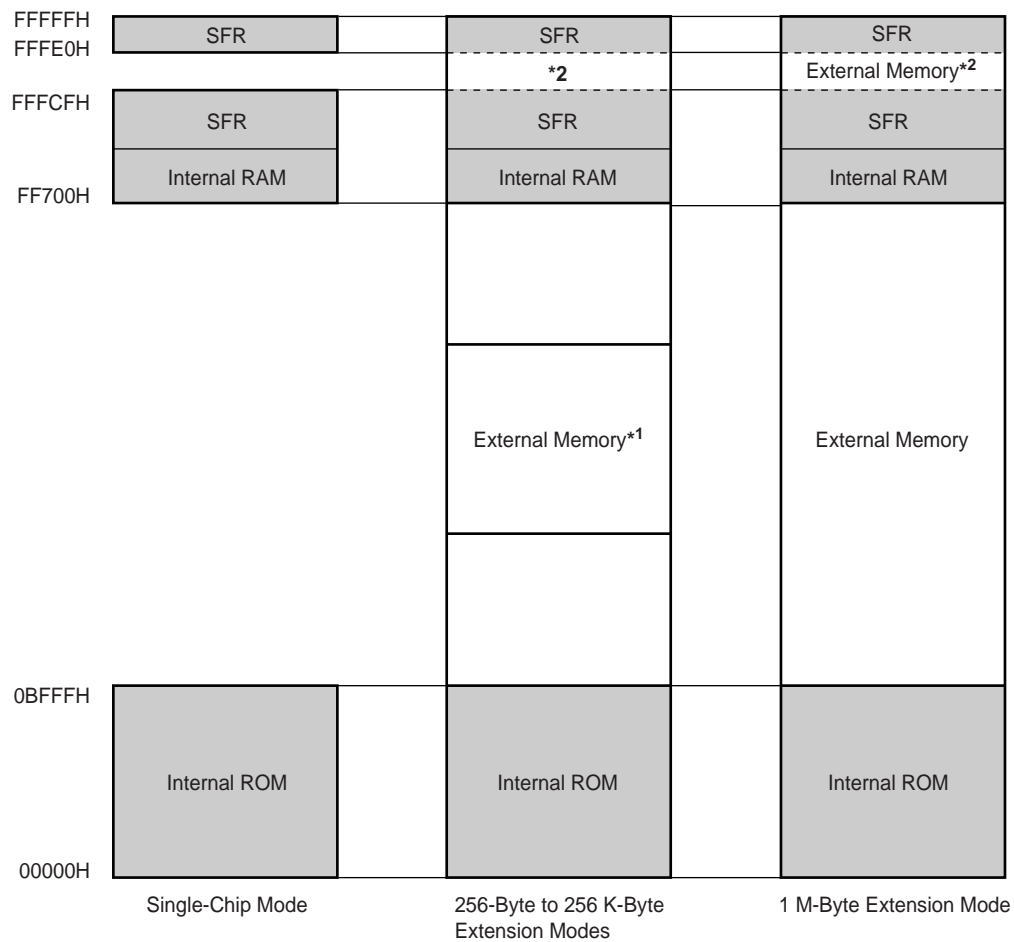
\* 1. Any extension size area in unshaded part

2. External SFR area



Figure 21-2  $\mu$ PD784025 Memory Map (2/2)

(b) When LOCATION 0FH instruction is executed

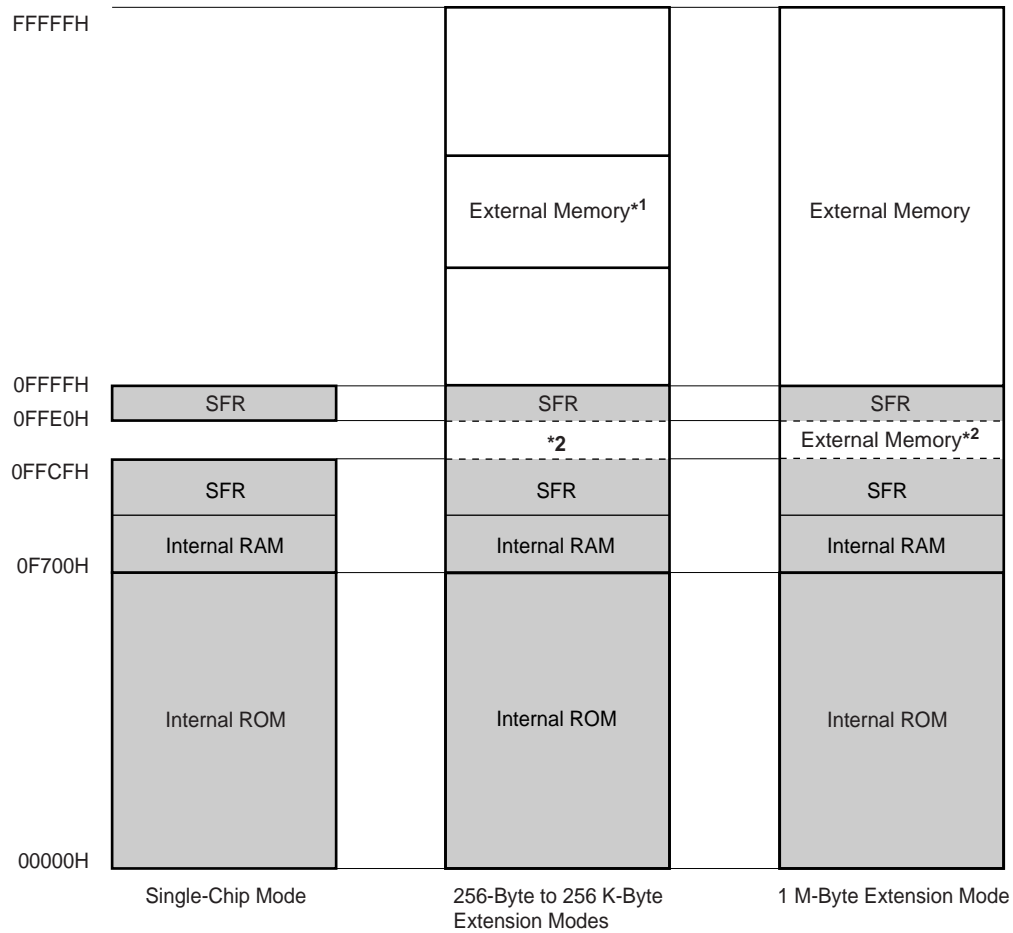


\* 1. Any extension size area in unshaded part

2. External SFR area

Figure 21-3  $\mu$ PD784026 Memory Map (1/2)

(a) When LOCATION 0 instruction is executed

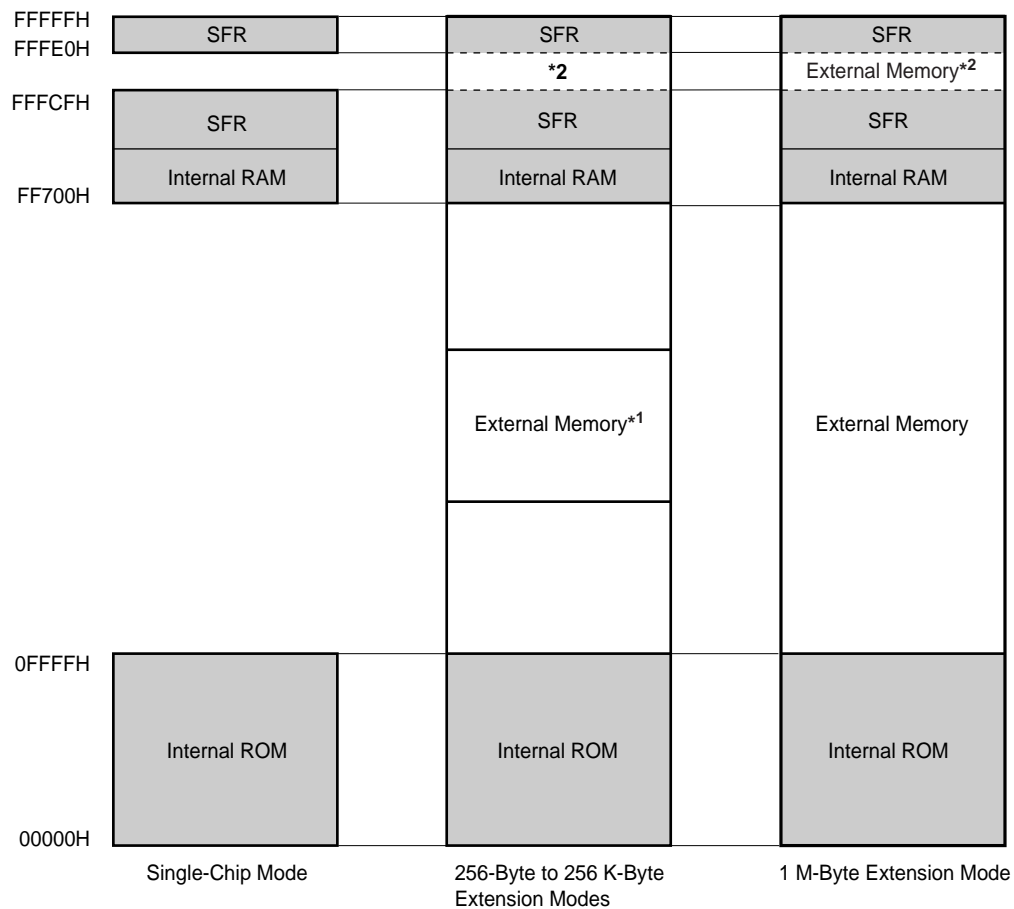


\* 1. Any extension size area in unshaded part

2. External SFR area

Figure 21-3  $\mu$ PD784026 Memory Map (2/2)

(b) When LOCATION 0FH instruction is executed

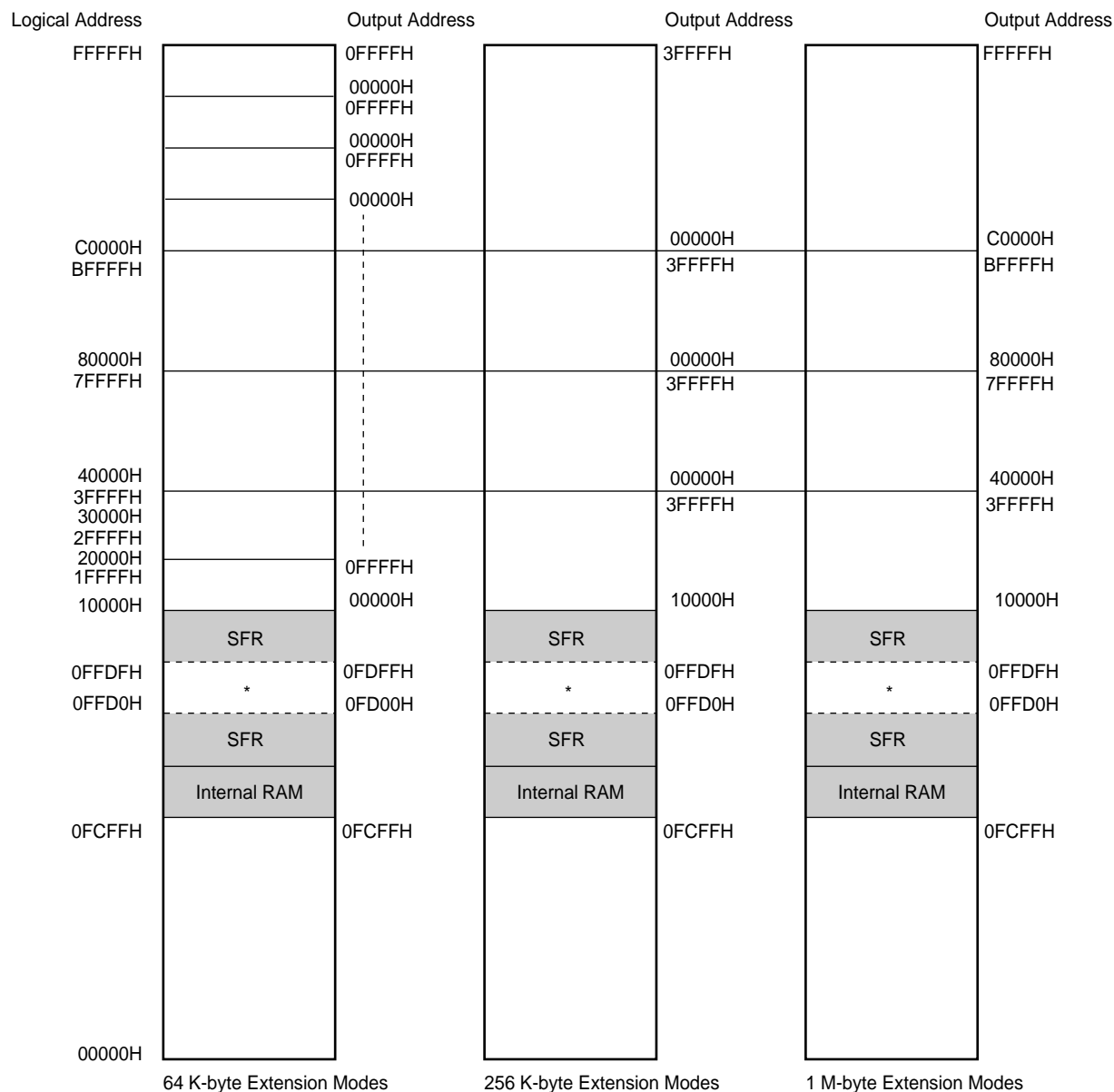


\* 1. Any extension size area in unshaded part

2. External SFR area

Figure 21-4  $\mu$ PD784020 Memory Map (1/2)

(a) When LOCATION 0 instruction is executed



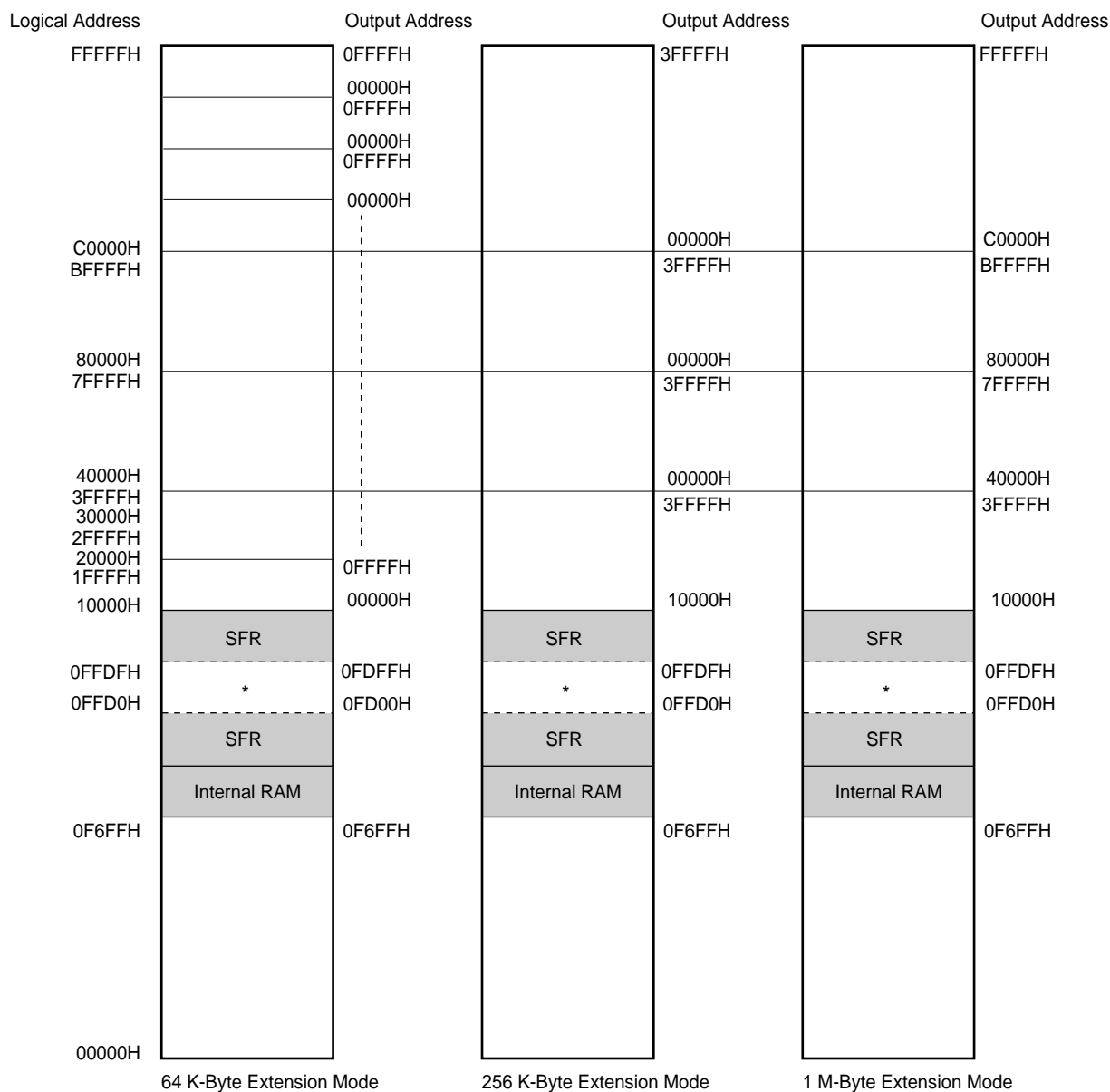
\* External SFR space

**(b) When LOCATION 0FH instruction is executed**

\* External SFR space

Figure 21-5  $\mu$ PD784021 Memory Map (1/2)

(a) When LOCATION 0 instruction is executed



\* External SFR space

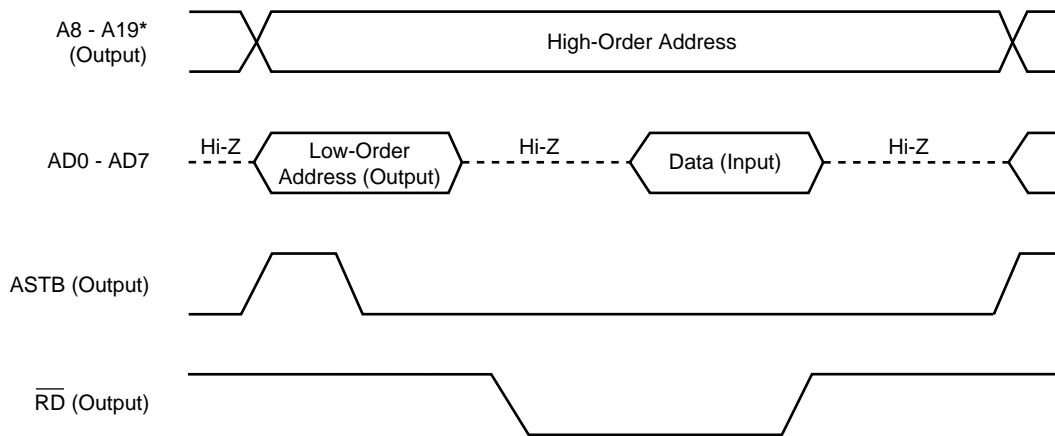
**(b) When LOCATION 0FH instruction is executed**

\* External SFR space

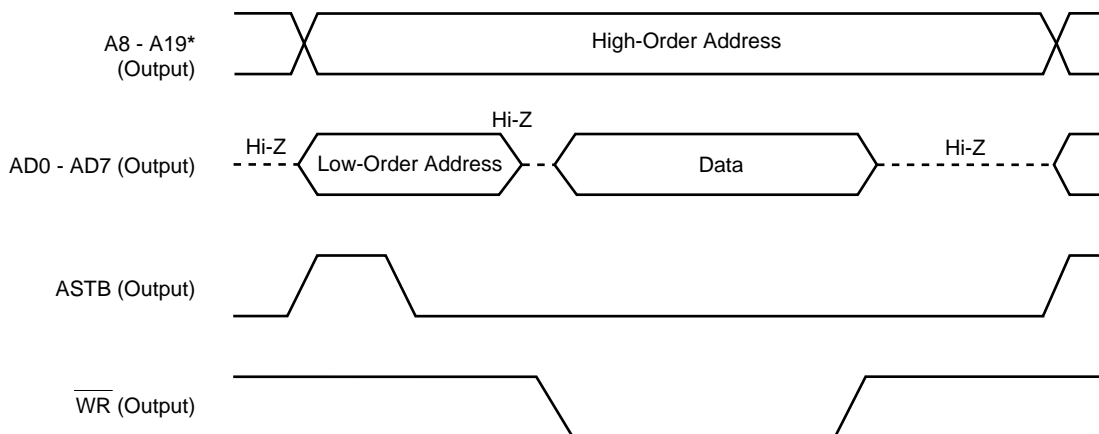
**21.1.3 Basic Operation of Local Bus Interface**

The local bus interface accesses external memory using  $\overline{ASTB}$ ,  $\overline{RD}$ ,  $\overline{WR}$ , an address/data bus (AD0 to AD7) and address bus (A8 to A19). When the local bus interface is used, P64, P65 and port 4 automatically operate as  $\overline{RD}$ ,  $\overline{WR}$  and AD0 to AD7 (in the  $\mu\text{PD784021}$ , these operate only as  $\overline{RD}$ ,  $\overline{WR}$  and AD0 to AD7 at all times). On the address bus, only the pins that correspond to the extension memory size operate as address bus pins.

An outline of the memory access timing is shown in Figures 21-6 and 21-7.

**Figure 21-6 Read Timing**

\* The number of address bus pins used depends on the extension mode size.

**Figure 21-7 Write Timing**

\* The number of address bus pins used depends on the extension mode size.



## 21.2 WAIT FUNCTION

When a low-speed memory or I/O is connected externally to the  $\mu$ PD784026, waits can be inserted in the external memory access cycle.

There are two kinds of wait cycle, an address wait for securing the address decoding time, and an access wait for securing the access time.

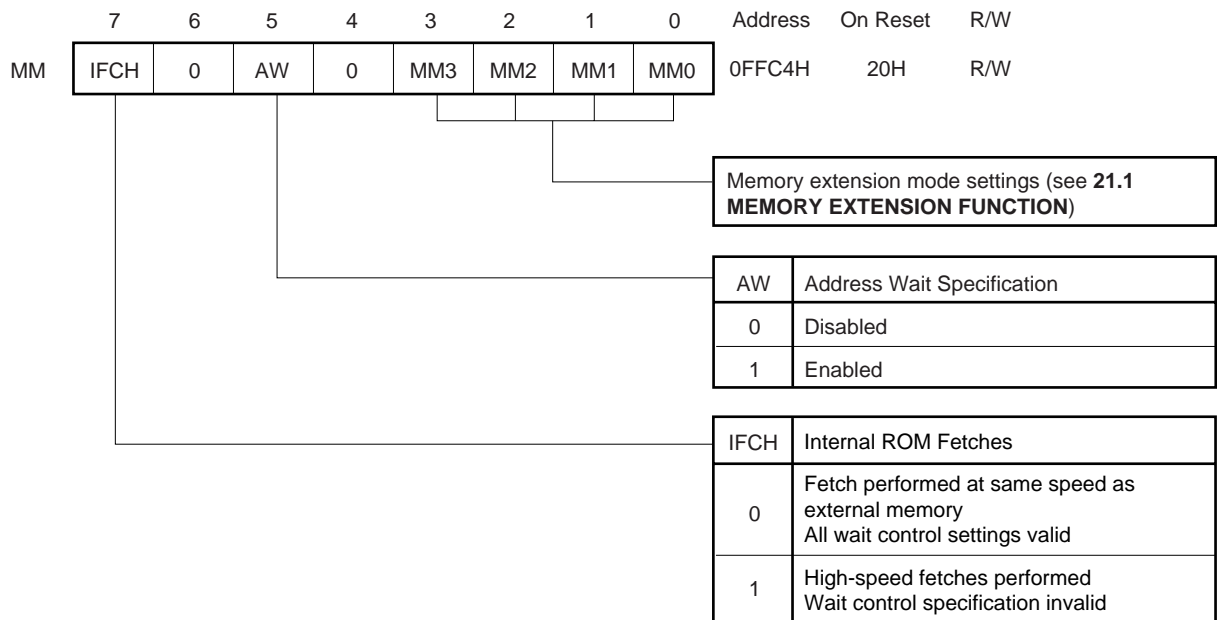
### 21.2.1 Wait Function Control Registers

#### (1) Memory extension mode register (MM)

The IFCH bit of the MM performs wait control setting for internal ROM accesses, and the AW bit performs address wait setting.

The MM can be read or written to with an 8-bit manipulation instruction. The MM format is shown in Figure 21-8. When  $\overline{\text{RESET}}$  is input, the MM register is set to 20H, the same cycle as for external memory is used for internal ROM accesses, and the address wait function is validated.

**Figure 21-8 Memory Extension Mode Register (MM) Format**



**(2) Programmable wait control registers (PWC1/PWC2)**

The PWC1 and PWC2 specify the number of waits.

PWC1 is an 8-bit register that divides the space from 0 to FFFFH into four, and specifies wait control for each of these four spaces. PWC2 is a 16-bit register that divides the space from 10000H to FFFFH into four, and specifies wait control for each of these four spaces.

The PWC1 can be read or written to with an 8-bit manipulation instruction, and the PWC2 with a 16-bit manipulation instruction. The PWC1 and PWC2 formats are shown in Figure 21-9.

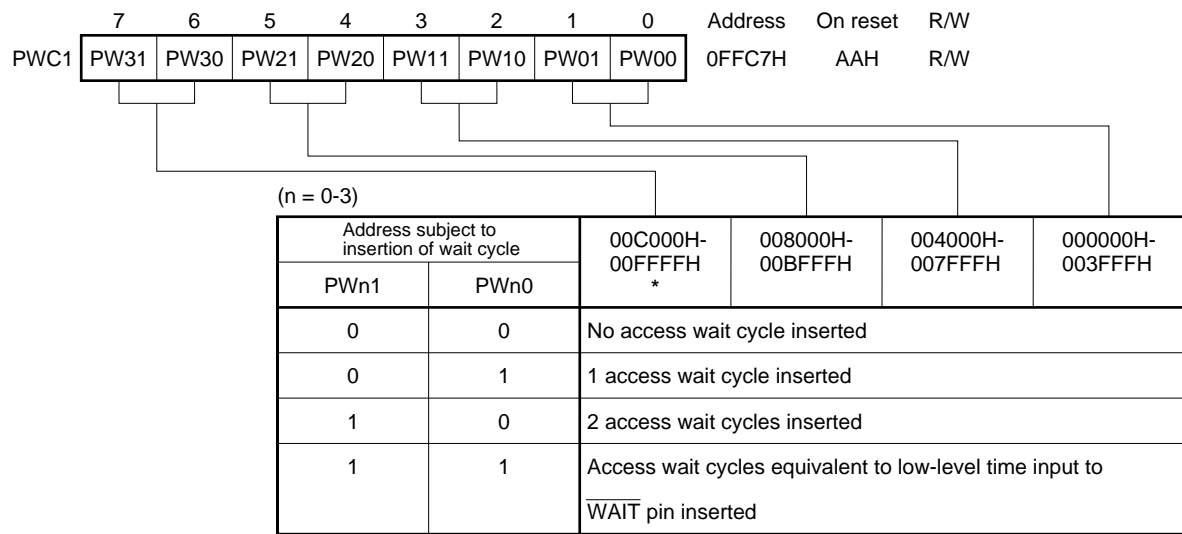
The high-order 8 bits of the PWC2 are fixed at AAH, and therefore ensure that the high-order 8 bits are set to AAH. When  $\overline{\text{RESET}}$  is input, the PWC1 is set to AAH, and the PWC2 to AAAAH, and 2-wait insertion is performed on the entire space.

- ★ **Caution** Do not set external wait for the internal ROM area. If external wait is set for the internal ROM area, the CPU is deadlocked. This deadlock status can be cleared only by reset input.

★

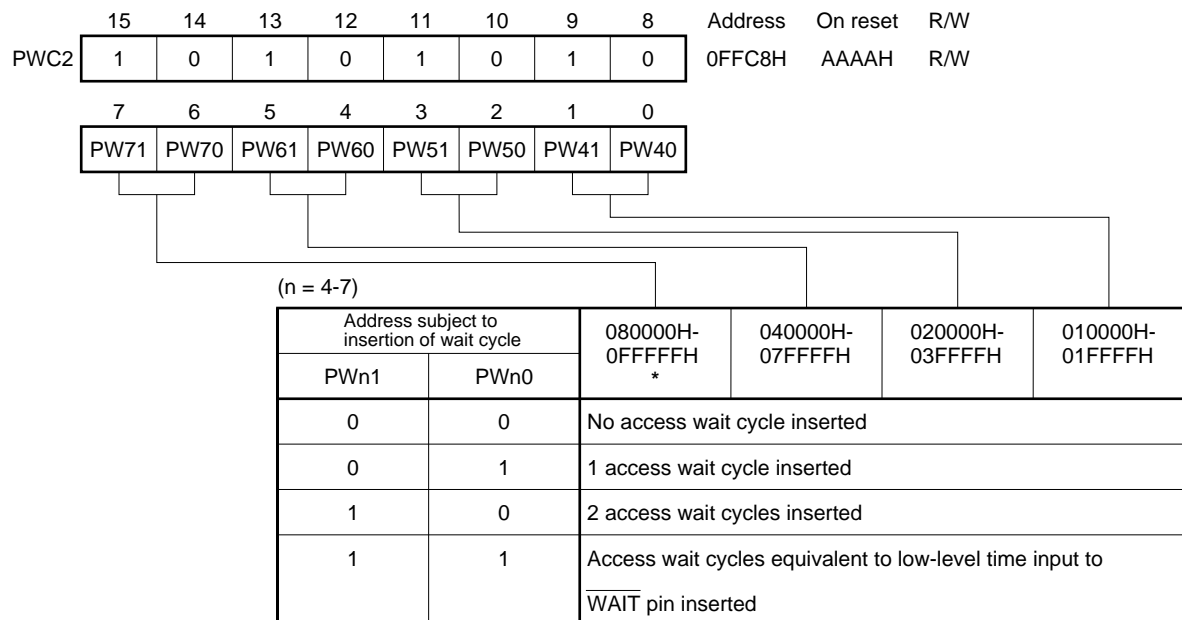
Figure 21-9 Programmable Wait Control Register (PWC1/PWC2) Format

## (a) Programmable wait control register 1 (PWC1)



\* Except portion overlapping the internal data area.

## (b) Programmable wait control register 2 (PWC2)



\* Except portion overlapping the internal data area.

**Caution** When the bus hold function is used, access wait control cannot be performed by means of the  $\overline{\text{WAIT}}$  pin, and 0, 1 or 2 waits must be selected for the entire space.

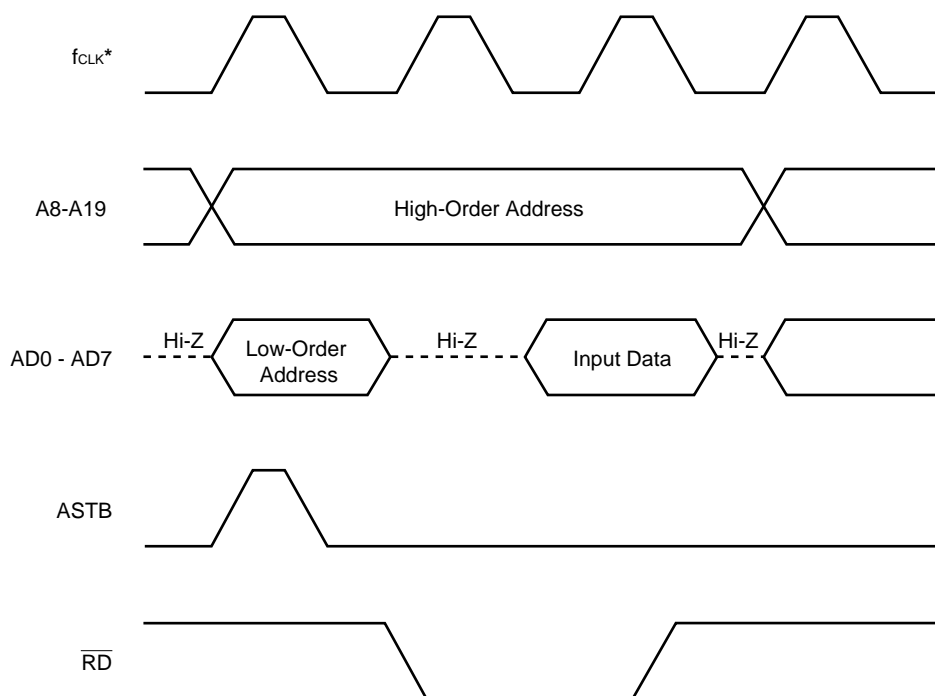
**21.2.2 Address Waits**

Address waits are used to secure the address decoding time. If the AW bit of the memory extension mode register (MM) is set (1), waits are inserted in every memory access\*. When an address wait is inserted, the high-level period of the ASTB signal is extended by one system clock cycle (80 ns:  $f_{CLK} = 12.5$  MHz).

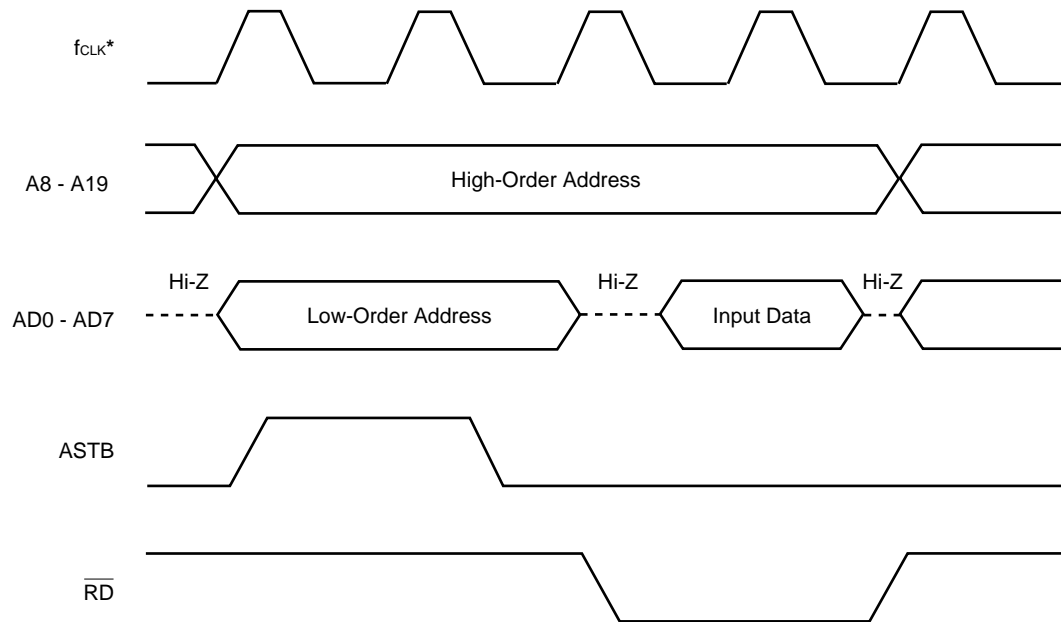
- \* Except for the internal RAM, internal SFRs, and internal ROM during high-speed fetch. If it is specified that the internal ROM is accessed in the same cycle as the external ROM, an address wait state is inserted even when the internal ROM is accessed.

**Figure 21-10 Address Wait Function Read/Write Timing (1/3)**

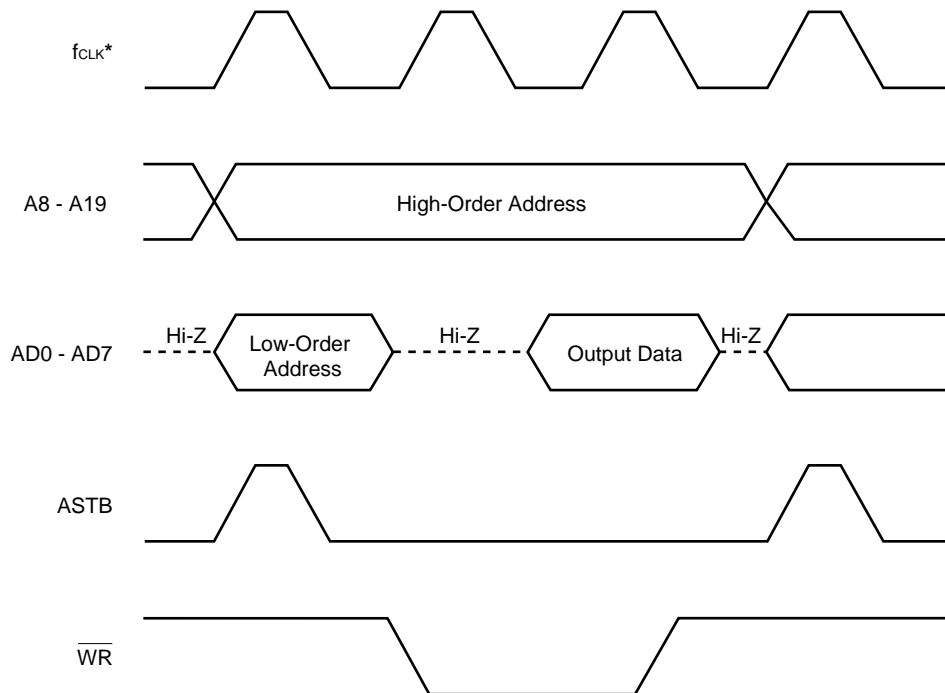
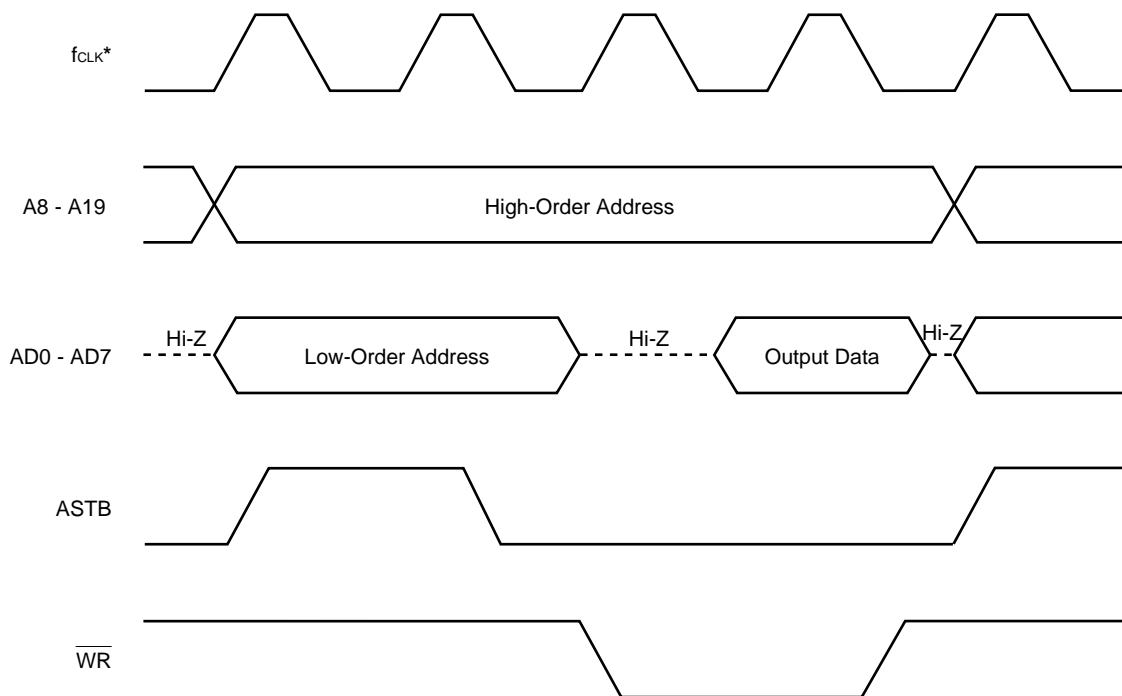
**(a) Read timing with no address wait insertion**



- \*  $f_{CLK}$ : Internal system clock frequency. This signal is present inside the  $\mu$ PD784026 only.

**Figure 21-10 Address Wait Function Read/Write Timing (2/3)****(b) Read timing with address wait insertion**

\* fCLK: Internal system clock frequency. This signal is present inside the  $\mu\text{PD784026}$  only.

**Figure 21-10 Address Wait Function Read/Write Timing (3/3)****(c) Write timing with no address wait insertion****(d) Write timing with address wait insertion**

\* fCLK: Internal system clock frequency. This signal is present inside the  $\mu$ PD784026 only.

### 21.2.3 Access Waits

Access waits are inserted in the  $\overline{RD}$  or  $\overline{WR}$  signal low-level period, and extend the low-level period by  $1/f_{CLK}$  (80 ns:  $f_{CLK}$  = 12.5 MHz) per cycle.

There are two wait insertion methods, using either the programmable wait function that automatically inserts the preset number of cycles, or the external wait function controlled by a wait signal from outside.

For wait cycle insertion control, the 1 M-byte memory space is divided into eight as shown in Figure 21-11, and control is specified for each space by means of the programmable wait control registers (PWC1/PWC2). Waits are not inserted in accesses to internal ROM or internal RAM using high-speed fetches. In accesses to internal SFRs, waits are inserted at the necessary times regardless of this specification.

If access operations are specified as being performed in the same number of cycles as for external ROM, waits are inserted also in internal ROM accesses in accordance with the PWC1 settings.

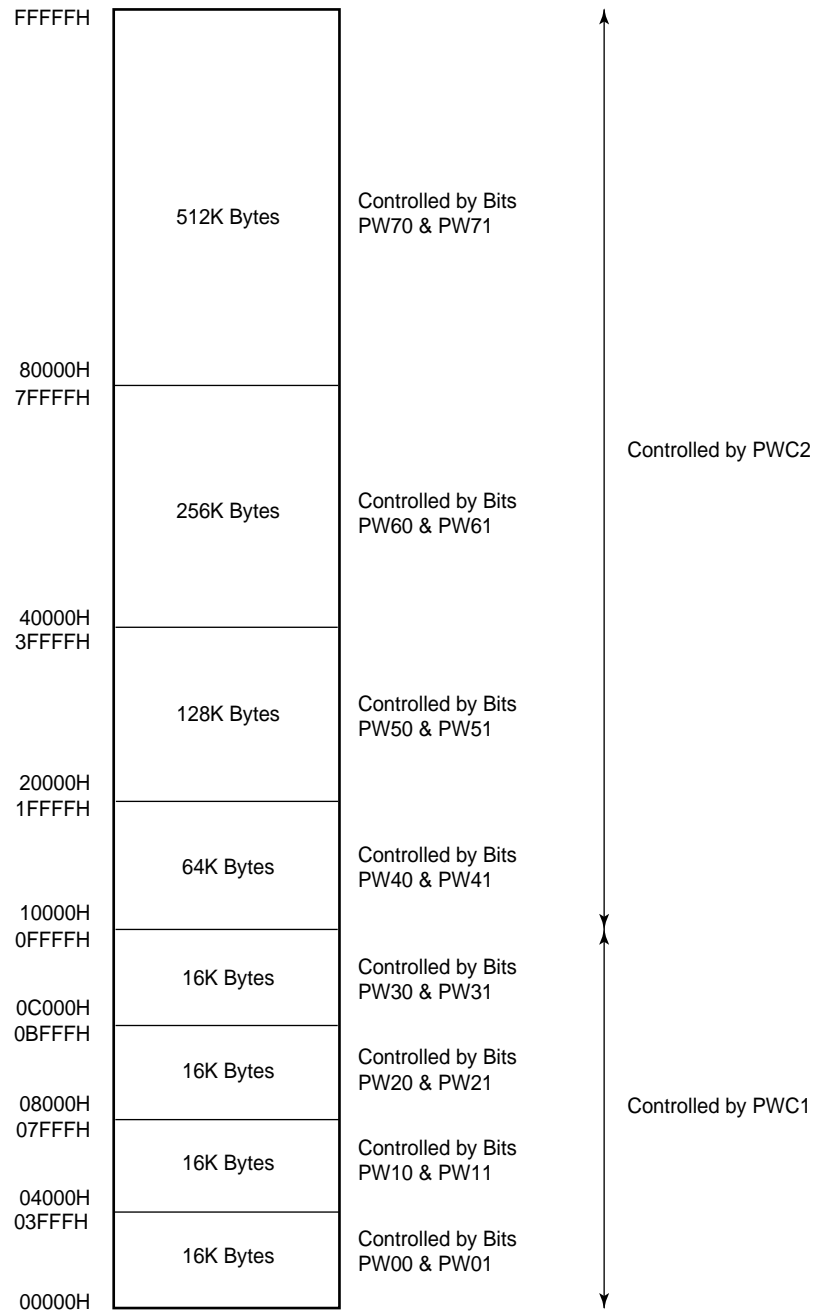
If there is a space for which control by a wait signal from outside has been selected by means of the PWC1/PWC2, the P66 pin operates as the  $\overline{WAIT}$  signal input pin. After  $\overline{RESET}$  input, the P66 pin operates as a general-purpose input/output port.

Bus timing in the case of access wait insertion is shown in Figures 21-12 to 21-14.

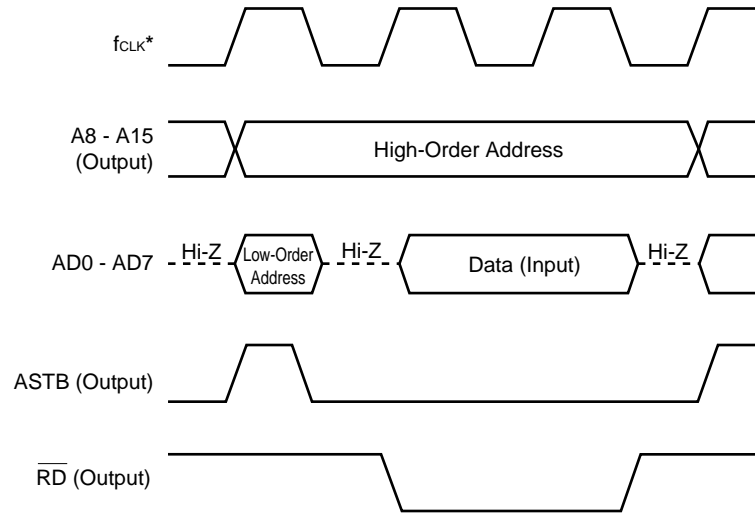
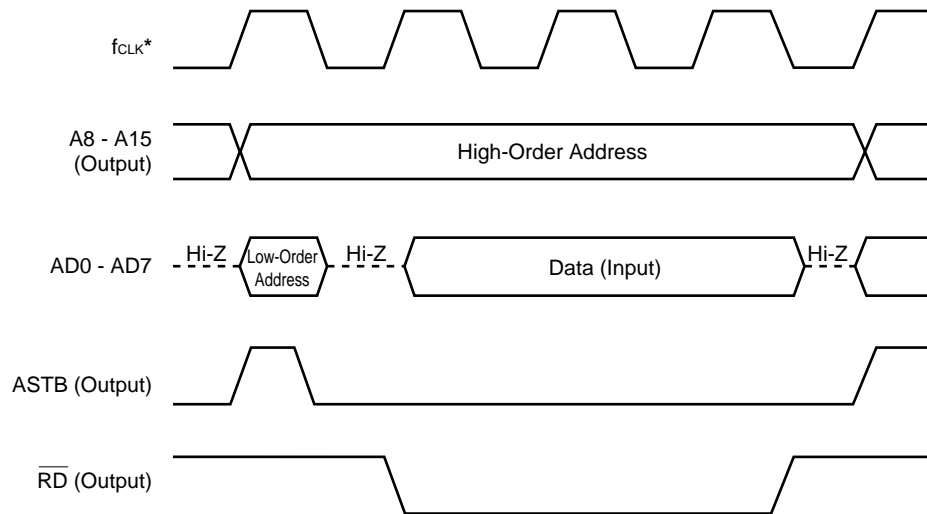
**Cautions** 1. The external wait function cannot be used when the bus hold function is used.

★

2. Do not set external wait for the internal ROM area. If external wait is set for the internal ROM area, the CPU is deadlocked. This deadlock status can be cleared only by reset input.

**Figure 21-11 Wait Control Spaces**

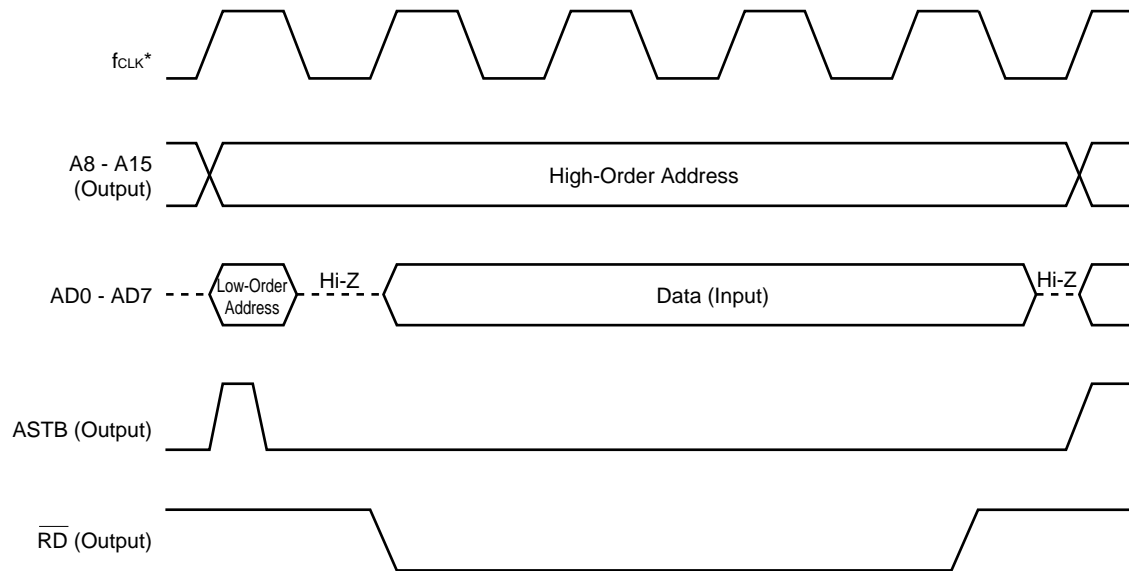


**Figure 21-12 Access Wait Function Read Timing (1/2)****(a) 0 wait cycles set****(b) 1 wait cycle set**

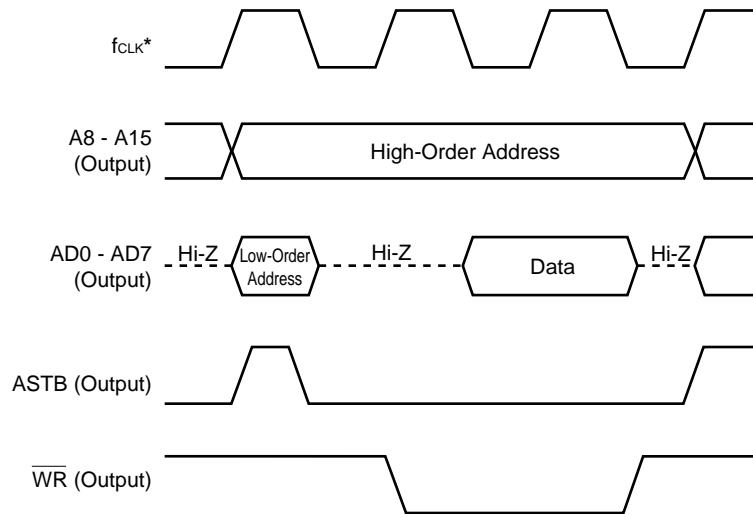
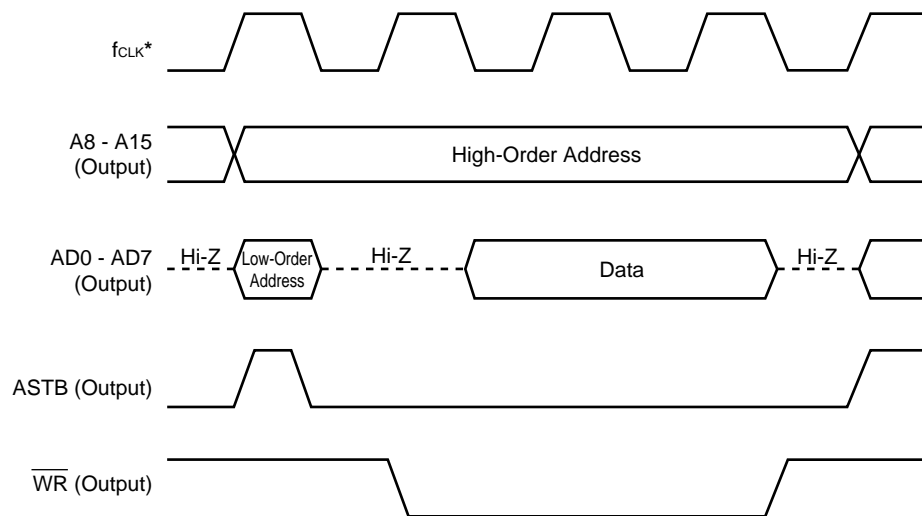
\*  $f_{CLK}$ : Internal system clock frequency. This signal is only present inside the  $\mu$ PD784026.

Figure 21-12 Access Wait Function Read Timing (2/2)

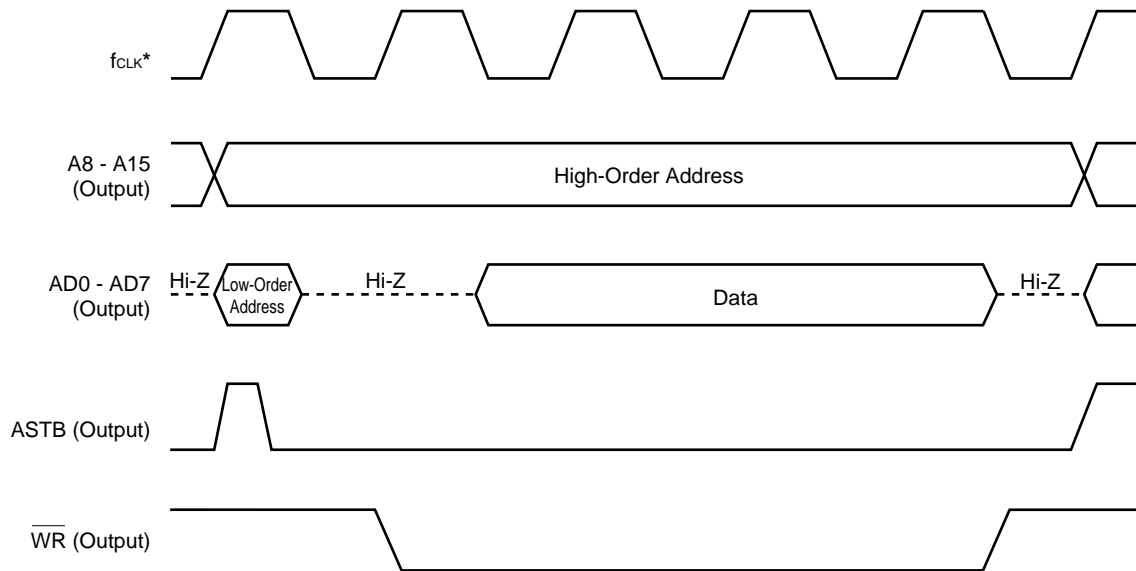
## (c) 2 wait cycles set



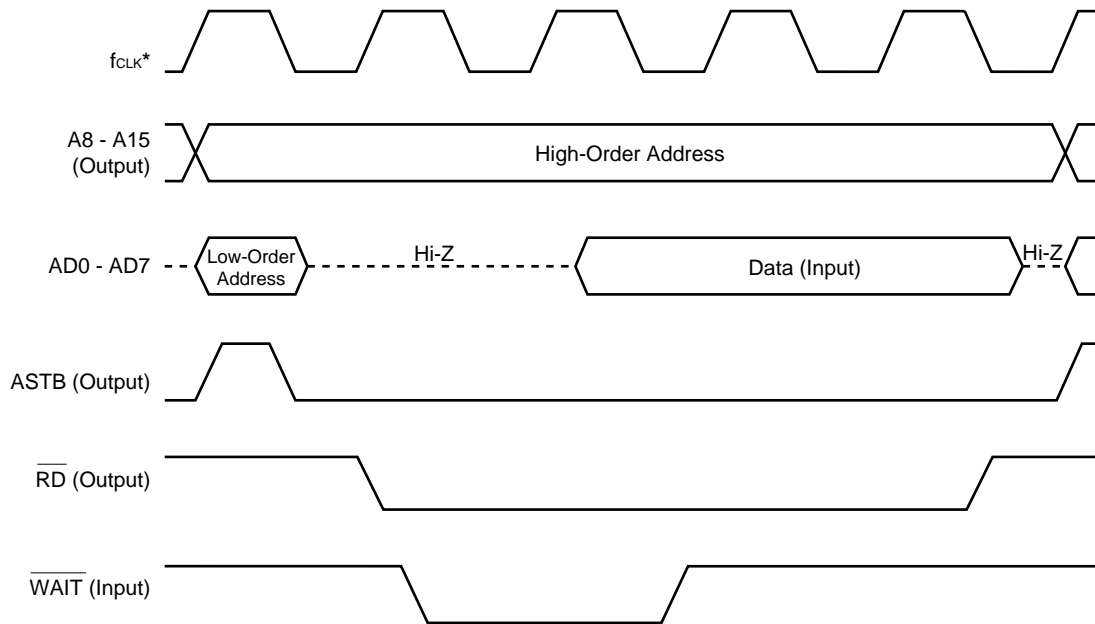
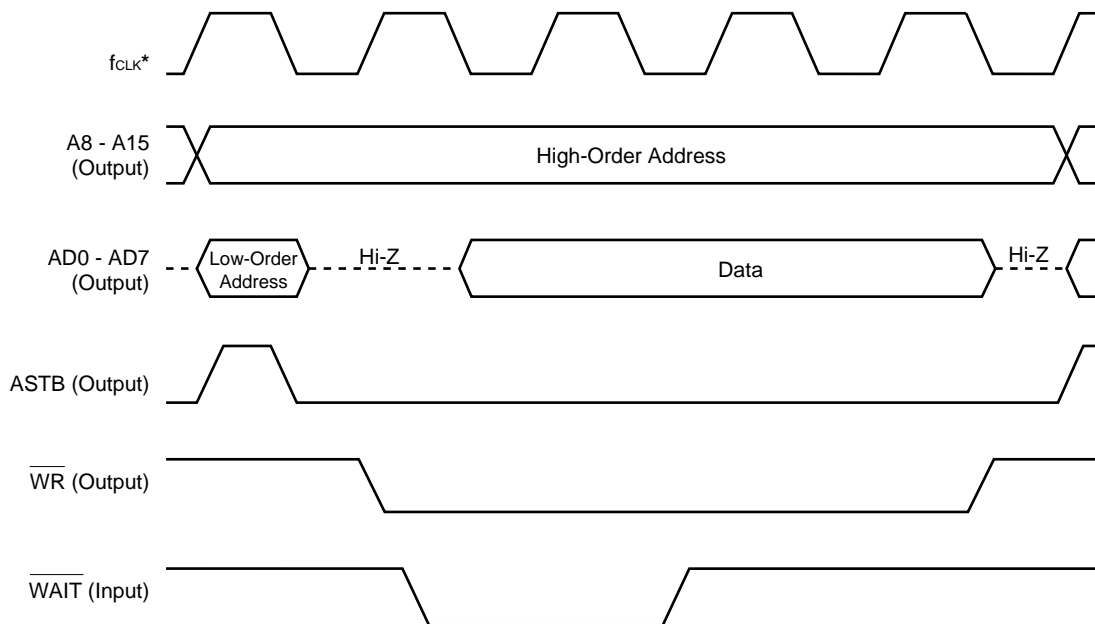
\* fclk: Internal system clock frequency. This signal is only present inside the  $\mu\text{PD784026}$ .

**Figure 21-13 Access Wait Function Write Timing (1/2)****(a) 0 wait cycles set****(b) 1 wait cycle set**

\* fCLK: Internal system clock frequency. This signal is only present inside the  $\mu$ PD784026.

**Figure 21-13 Access Wait Function Write Timing (2/2)****(c) 2 wait cycles set**

\* fCLK: Internal system clock frequency. This signal is only present inside the  $\mu$ PD784026.

**Figure 21-14 Timing with External Wait Signal****(a) Read timing****(b) Write timing**

\* fCLK: Internal system clock frequency. This signal is only present inside the  $\mu\text{PD784026}$ .

### 21.3 PSEUDO-STATIC RAM REFRESH FUNCTION

The  $\mu$ PD784026 incorporates a pseudo-static RAM refresh function for direct connection of pseudo-static RAM.

The pseudo-static RAM refresh function outputs refresh pulses at any desired intervals. The refresh pulse output interval is specified by the refresh mode register (RFM) setting.

The refresh area specification register (RFA) specifies the addresses on which refresh operations can be performed at the same time as memory access operations. This enables bus cycle insertions for refresh operations to be greatly decreased, thus minimizing the reduction in performance due to refresh operations.

The  $\mu$ PD784026 is provided with a function for supporting self-refresh operations that offers low power consumption by a pseudo-static RAM application system.

**Cautions 1. The refresh function cannot be used when the bus hold function is used.**

- ★ **2. The pseudo-static RAM refresh function does not operate normally with an in-circuit emulator. For details, refer to 21.5 CAUTIONS.**

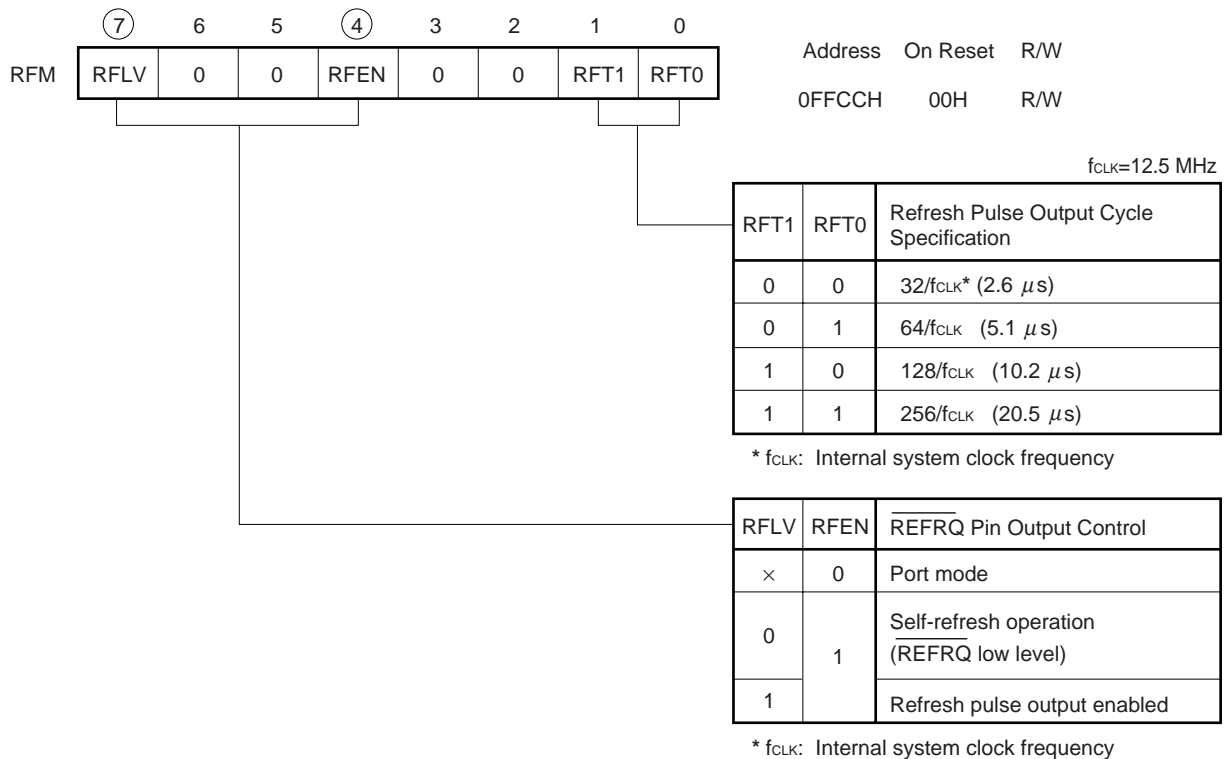
## 21.3.1 Control Registers

## (1) Refresh mode register (RFM)

The RFM is an 8-bit register that controls the pseudo-static RAM refresh cycle and switching to self-refresh operations. The RFM register can be read or written to with an 8-bit manipulation instruction or bit manipulation instruction. RFM format is shown in Figure 21-15.

$\overline{\text{RESET}}$  input clears the RFM register to 00H and sets the  $\overline{\text{REFRQ}}$  pin to port mode, so that it operates as the dual-function P67 pin.

Figure 21-15 Refresh Mode Register (RFM) Format



**Remark**  $\times$ : 0 or 1

**Caution** The refresh function cannot be used when the bus hold function is used. In this case, ensure that refreshing is specified as disabled.

**(2) Refresh area specification register (RFA)**

The RFA is an 8-bit register that specifies the areas on which refresh operations can be performed at the same time as memory access operations.

The RFA register can be read or written to with an 8-bit manipulation instruction and bit manipulation instruction. RFA format is shown in Figure 21-16.

$\overline{\text{RESET}}$  input clears the RFA register to 00H.

**Figure 21-16 Refresh Area Specification Register (RFA) Format**

	7	6	5	4	3	2	1	0	Address	On Reset	R/W
RFA	RFA7	RFA6	RFA5	RFA4	RFA3	RFA2	RFA1	RFA0	0FFCDH	00H	R/W
(n= 0 to 7)											
Refresh Specification Area RFA <sub>n</sub>	080000H to 0FFFFFFH	040000H to 07FFFFFFH	020000H to 03FFFFFFH	010000H to 01FFFFFFH	00C000H to 00FFFFFFH	008000H to 00BFFFFFFH	004000H to 007FFFFFFH	000000H to 003FFFFFFH			
0	Refreshing performed at same time as memory access operations in corresponding block										
1	Refreshing performed exclusively from memory access operations in corresponding block										

**21.3.2 Operations****(1) Pulse refresh operation**

To support the pulse refresh cycles of pseudo-static RAM, refresh pulses are output from the  $\overline{\text{REFRQ}}$  pin in synchronization with bus cycles.

The system clock frequency and bits 1 and 0 (RFT1/RFT0) of the refresh mode register (RFM) are adjusted so that 512 or more refresh pulses are generated in an 8ms period.

**Table 21-1 System Clock Frequency and Refresh Pulse Output Cycle When Pseudo-Static RAM is Used**

System Clock Frequency ( $f_{\text{CLK}}$ ) MHz	Refresh Pulse Output Cycle Specification	RFT1	RFT0
$8.192 < f_{\text{CLK}} \leq 12.5$	$128/f_{\text{CLK}}$	1	0
$4.096 < f_{\text{CLK}} \leq 8.192$	$64/f_{\text{CLK}}$	0	1
$2.048 < f_{\text{CLK}} \leq 4.096$	$32/f_{\text{CLK}}$	0	0

These pulse refresh operations are performed so that they do not overlap external memory access operations. During a refresh cycle, an external memory access cycle is held pending ( $\overline{\text{ASTB}}$ ,  $\overline{\text{RD}}$ ,  $\overline{\text{WR}}$ , etc. are inactive), and during an external memory access cycle, a refresh cycle is held pending.

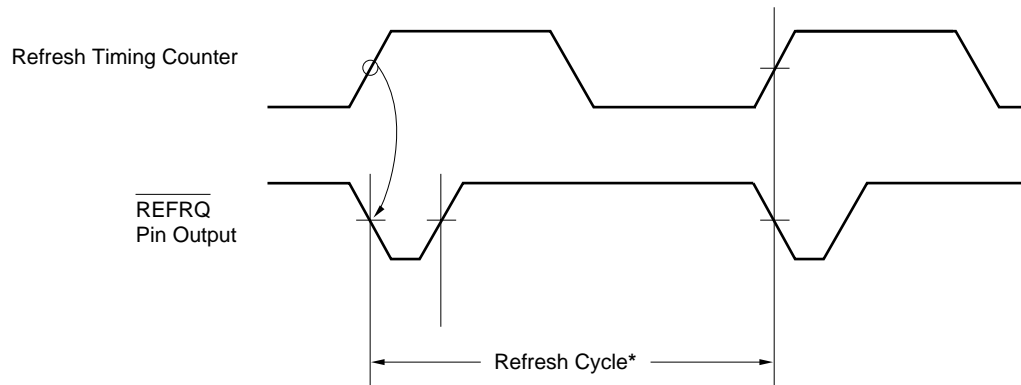
If there is no overlapping with an external memory access operation, the refresh cycle is performed without affecting CPU instruction execution.



**(a) Internal memory accesses**

In the case of internal memory accesses in which the external pseudo-static RAM is not accessed, also, refresh bus cycles are output at the intervals specified by the refresh mode (RFM) register so that the data stored in the pseudo-static RAM is retained. In this case, CPU instruction execution is not affected.

**Figure 21-17 Pulse Refresh Operation in Internal Memory Access**



\* Cycle specified by the RFT1 and RFT0 bits of the RFM

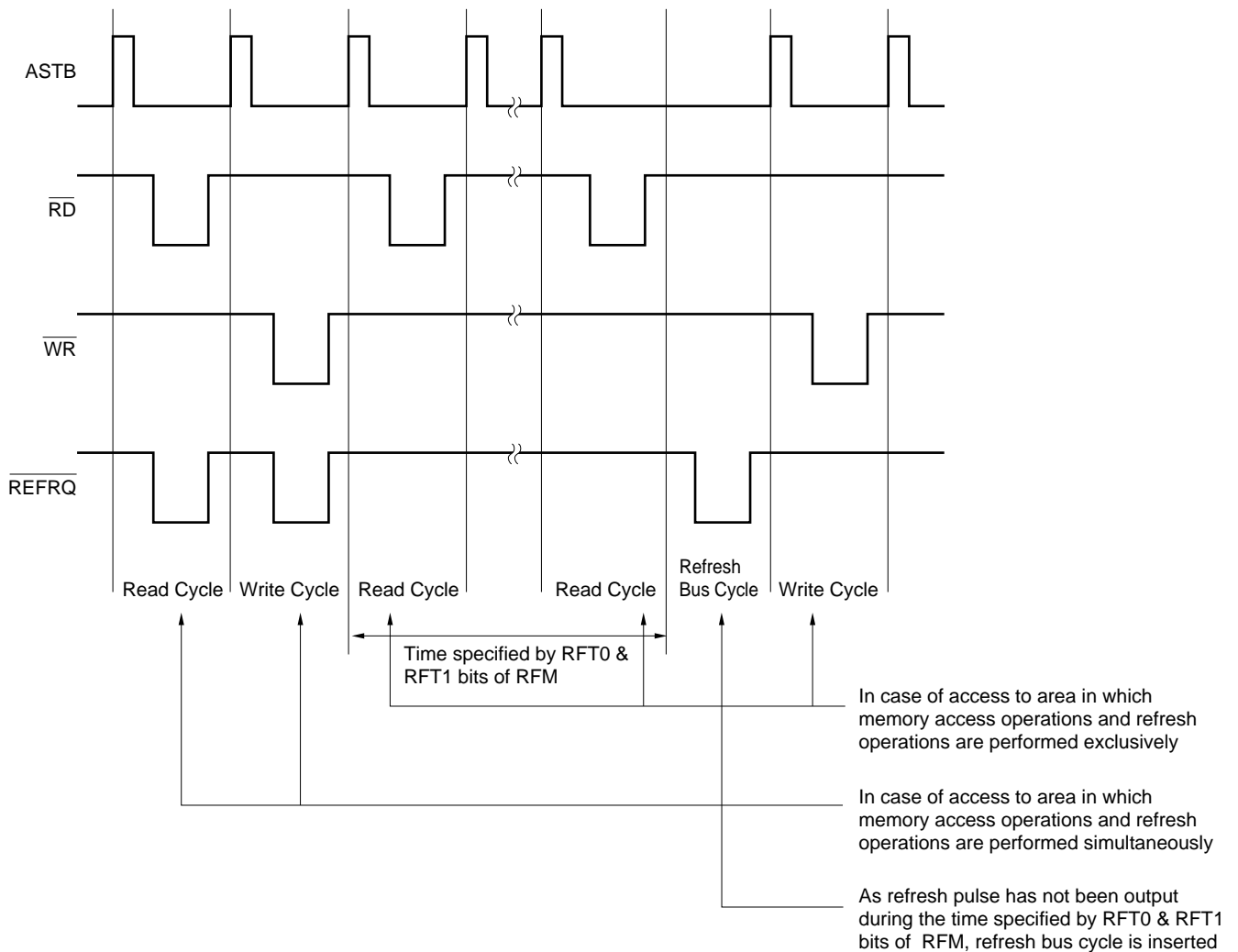
**(b) External memory accesses**

When an access is made to an address corresponding to a cleared (0) bit in the refresh area specification register (RFA), a refresh pulse is always output from the  $\overline{\text{REFRQ}}$  pin at the same time as the  $\overline{\text{RD}}$  signal or  $\overline{\text{WR}}$  signal, irrespective of the cycle specified by the refresh mode register (RFM).

After refresh pulse output, accesses to internal memory or accesses to addresses corresponding to a set (1) bit in the RFA continue, and after the time specified by the RFT0 and RFT1 bits of the RFM has elapsed, a refresh bus cycle is generated so as not to overlap a memory access cycle, and a refresh pulse is output.

In this way, refreshing can be performed while memory that does not need refreshing, such as PROM, is being accessed, refresh bus cycle insertions can be reduced, and instruction execution can be performed efficiently.

**Figure 21-18 Refresh Pulse Output Operation**



**(2) Self-refresh operation**

This mode is used to retain the contents of pseudo-static RAM in standby mode.

**(a) Self-refresh operating mode setting**

When bit 4 (RFEN) of the refresh mode (RFM) register is set to “1”, and bit 7 (RFLV) to “0”, a low level is output from the  $\overline{\text{REFRQ}}$  pin, and the self-refresh operating mode is specified for the pseudo-static RAM.

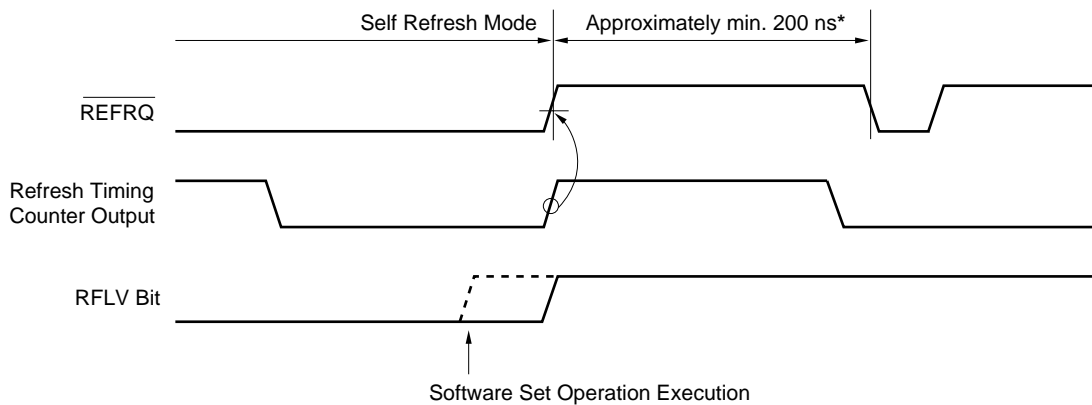
**(b) Return from self-refresh operation**

Refresh pulse output to the pseudo-static RAM is disabled approximately 200 ns\* after the  $\overline{\text{REFRQ}}$  pin output level changes from low to high. Therefore, the  $\mu\text{PD784026}$  arranges for refresh pulses not to be output during the disabled time by raising the  $\overline{\text{REFRQ}}$  pin in synchronization with the refresh timing counter.

To enable this low-to-high transition of the  $\overline{\text{REFRQ}}$  pin level to be recognized, the RFLV bit read level is set (1) when the  $\overline{\text{REFRQ}}$  pin level changes from low to high.

\* This time varies according to the speed rank, etc. of the pseudo-static RAM.

**Figure 21-19 Timing for Return from Self-Refresh Operation**



\* Refreshing disabled time

## 21.4 BUS HOLD FUNCTION

The bus hold function is provided for the connection of a device that functions as the bus master, such as a DMA controller. In response to a request from the bus master device, all local bus interface pins are set to high impedance (except HLDAK), and local bus interface mastership is passed to that device.

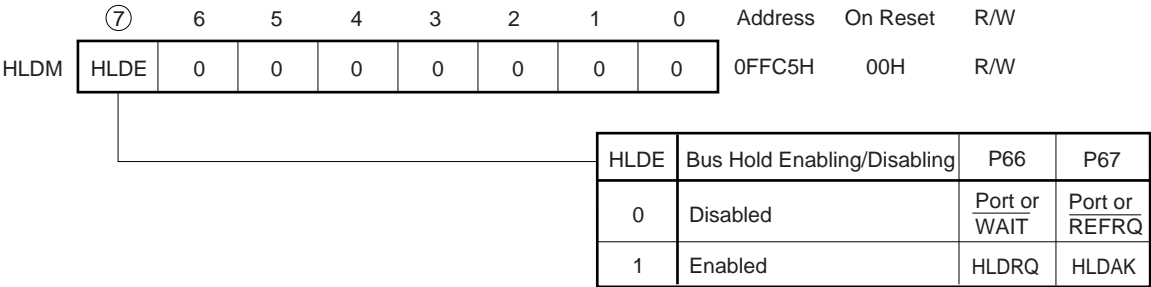
The bus hold function cannot be used when the external wait function or refresh function is used.

### 21.4.1 Hold Mode Register (HLDM)

The HLDM is an 8-bit register that specifies enabling/disabling of the bus hold function. HLDM format is shown in Figure 21-20.

When  $\overline{\text{RESET}}$  is input, the HLDM register is cleared to 00H, so that the bus hold function is disabled, and the HLDRQ and HLDAK pins are set to port mode and operate as the P66 and P67 pins.

★ **Figure 21-20 Hold Mode Register (HLDM) Format**



**Caution** The bus hold function must be disabled when the external wait function or refresh function is used.

**21.4.2 Operation**

When the HLDE bit of the hold mode register (HLDM) is set (1), the bus hold function is enabled. When the bus hold function is enabled, pins P66 and P67 operate as the HLDRQ and HLDAK pins respectively. The HLDRQ pin becomes high-impedance, and the HLDAK pin outputs a low-level signal.

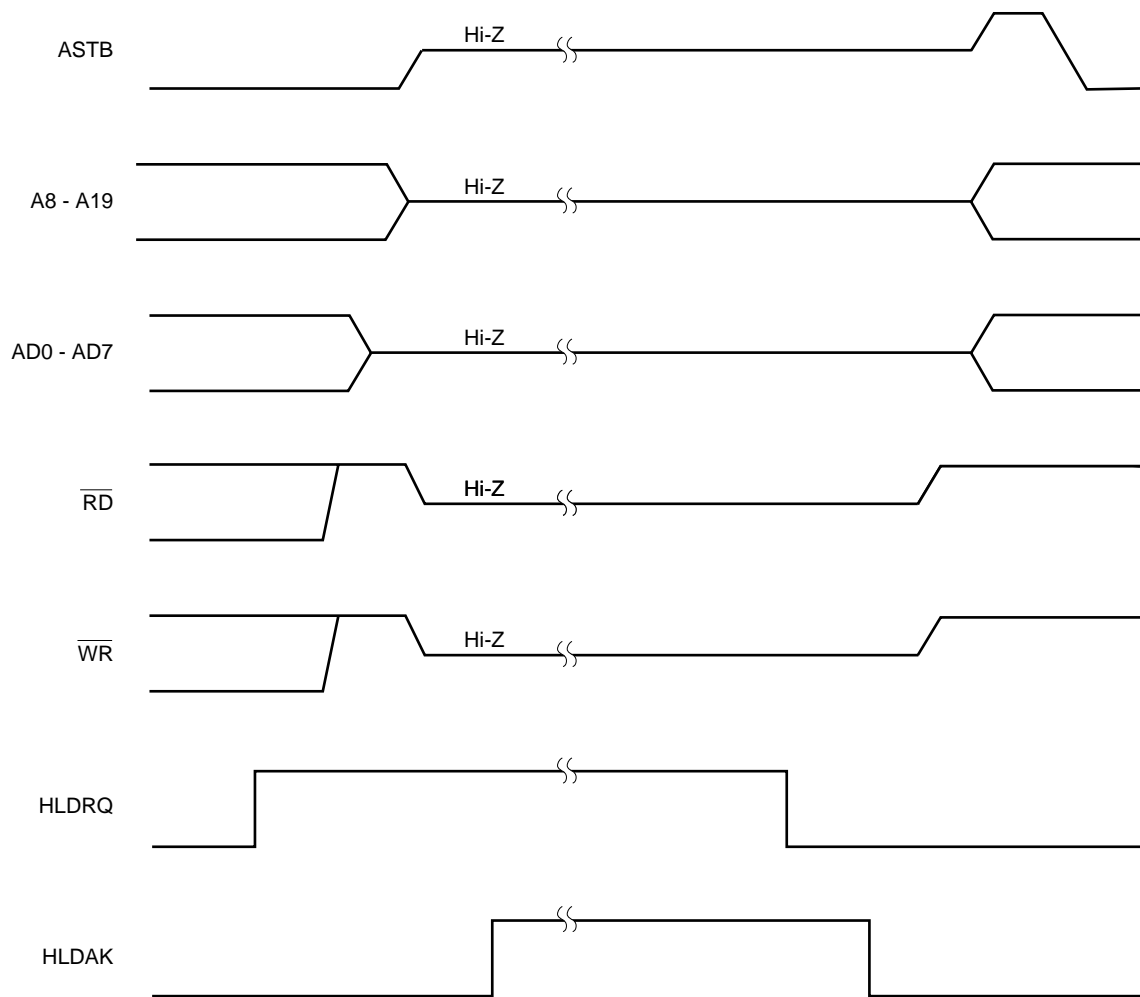
- ★ If a high-level signal is input to the HLDRQ pin when the bus hold function is enabled, at the end of the access operation being executed the address bus (A8 to A19), address/data bus (AD0 to AD7),  $\overline{RD}$ ,  $\overline{WR}$  and ASTB pins are all set to high-impedance, the HLDAK pin output level is driven high, and the hold mode is established. Because the address bus, address/data bus,  $\overline{RD}$ ,  $\overline{WR}$ , and ASTB pins go into a high-impedance state at this time, it is recommended to connect a pull-up resistor to the  $\overline{RD}$  and  $\overline{WR}$  pins, and a pull-down resistor to the ASTB pin.

While the HLDAK pin is high (in the hold mode) the  $\mu$ PD784026 does not use the local bus interface, and therefore an external DMA controller, etc. is free to access the memory.

When the HLDRQ pin input level changes from high to low, the hold mode is released, the HLDAK pin level changes from high to low, and then the  $\mu$ PD784026 resumes use of the local bus.

- A transition to the hold mode is performed between bus cycles, and the instruction being executed may be suspended.
- ★ When a program is fetched from the internal memory, the instruction can be executed until an instruction that uses the local bus interface appears. If the external memory is not accessed, therefore, execution of an instruction is not suspended.
- ★ If the hold mode is set while instructions that does not use the local bus interface are being executed when a program is fetched from the external memory, the  $\mu$ PD784026 continues executing the instructions already prefetched until an instruction that uses the local bus interface appears, and suspends execution when an instruction using the local bus interface appears or there are no more prefetched instructions. The suspended instruction is resumed from the point at which it was suspended when the hold mode is released.

When a program is fetched from the internal ROM or internal RAM, instruction execution is continued until an instruction that uses the local bus interface appears.

**Figure 21-21 Hold Mode Timing**

## 21.5 CAUTIONS

- (1) When the bus hold function is used, the external wait function (access wait control performed by means of  $\overline{\text{WAIT}}$  pin) cannot be used. 0, 1 or 2 waits must be selected for the entire space.
- (2) The external wait function cannot be used when the bus hold function is used.
- (3) The refresh function cannot be used when the bus hold function is used. In this case, ensure that refreshing is specified as disabled.
- ★ (4) Do not set external wait for the internal ROM area. If external wait is set for the internal ROM area, the CPU is deadlocked. This deadlock status can be cleared only by reset input.
- ★ (5) The pseudo-static RAM refresh function does not operate normally with an in-circuit emulator. Perform final check of the operations of a system using the pseudo-static RAM refresh function with the  $\mu\text{PD784026}$ .  
To debug the program logically, either map the pseudo-static RAM to the alternate memory of the in-circuit emulator, or replace the pseudo-static RAM On the target board with a static RAM.

[MEMO]



## CHAPTER 22 STANDBY FUNCTION

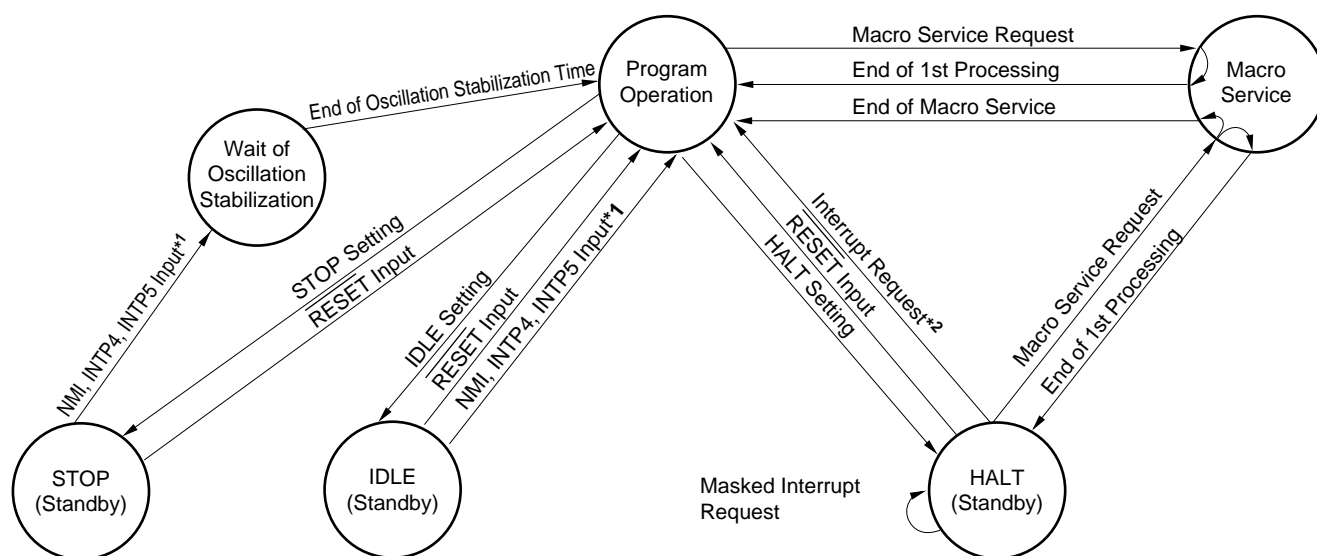
### 22.1 CONFIGURATION AND FUNCTION

The  $\mu$ PD784026 has a standby function that enables the system power consumption to be reduced. The standby function includes three modes as follows:

- **HALT mode** ..... In this mode the CPU operating clock is stopped. Intermittent operation in combination with the normal operating mode enables the total system power consumption to be reduced.
- **IDLE mode** ..... In this mode the oscillator continues operating while the entire remainder of the system is stopped. Normal program operation can be restored at a low power consumption close to that of the STOP mode and in a time equal to that of the HALT mode.
- **STOP mode** .... In this mode the oscillator is stopped and the entire system is stopped. Ultra-low power consumption can be achieved, consisting of leakage current only.

These modes are set by software. The standby mode (STOP/IDLE/HALT mode) transition diagram is shown in Figure 22-1, and the standby function block diagram in Figure 22-2.

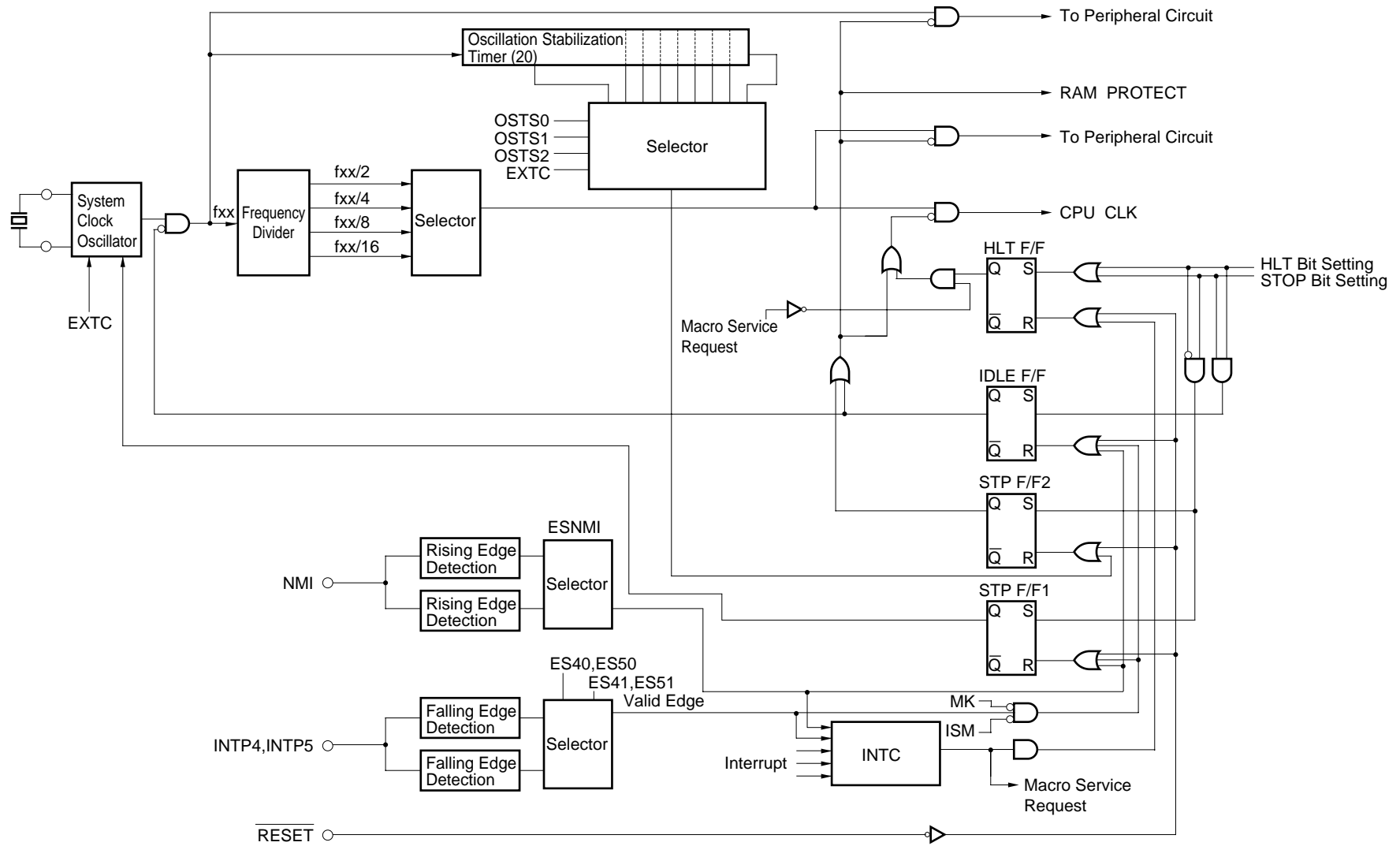
**Figure 22-1 Standby Mode Transition Diagram**



- \* 1. When INTP4 and INTP5 are not masked  
2. Unmasked interrupt request only

★ **Remark** Only external input is valid as NMI. The watchdog timer must not be used to release the standby mode (STOP/ HALT/IDLE mode).

Figure 22-2 Standby Function Block Diagram



## 22.2 CONTROL REGISTERS

### 22.2.1 Standby Control Register (STBC)

The STBC is used to select the STOP mode setting and the internal system clock.

To prevent entry into standby mode due to an inadvertent program loop, the STBC register can only be written to with a dedicated instruction. This dedicated instruction, MOV STBC, #byte, has a special code configuration (4 bytes), and a write is only performed if the 3rd and 4th bytes of the operation code are mutual 1's complements.

- ★ If the 3rd and 4th bytes of the operation code are not mutual 1's complements, a write is not performed and an operand error interrupt is generated. In this case, the return address saved in the stack area is the address of the instruction that was the source of the error, and thus the address that was the source of the error can be identified from the return address saved in the stack area.

If recovery from an operand error is simply performed by means of an RETB instruction, an endless loop will result.

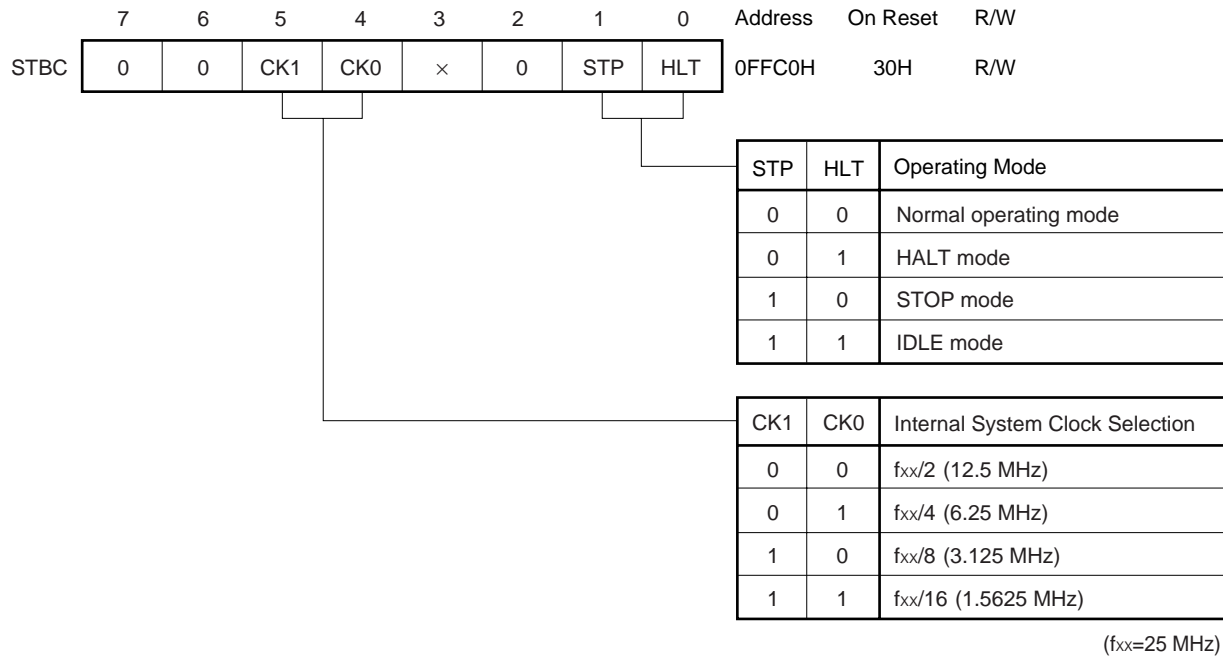
As an operand error interrupt is only generated in the event of an inadvertent program loop (with the NEC assembler, RA78K4, only the correct dedicated instruction is generated when MOV STBC, #byte is written), system initialization should be performed by the program.

Other write instructions (MOV STBC, A; AND STBC, #byte; SET1 STBC.7, etc.) are ignored and do not perform any operation. That is, a write is not performed to the STBC, and an interrupt such as an operand error interrupt is not generated.

The STBC can be read at any time by a data transfer instruction.

$\overline{\text{RESET}}$  input sets the STBC register to 30H.

The format of the STBC is shown in Figure 22-3.

**Figure 22-3 Standby Control Register (STBC) Format**

**Cautions** 1. If the STOP mode is used when using external clock input, the EXTC bit of the oscillation stabilization time specification register (OSTS) must be set (1) before setting STOP mode. If the STOP mode is used with the EXTC bit cleared (0) when using external clock input, the  $\mu$ PD784026 may suffer damage or reduced reliability.

When setting the EXTC bit of OSTS to 1, be sure to input a clock in phase reverse to that of the clock input to the X1 pin, to the X2 pin (refer to 4.3.1 Clock Oscillator).

- ★ 2. Execute a NOP instruction three times after the standby instruction (after the standby mode has been released). Otherwise, the standby instruction is not executed if execution of the standby instruction conflicts with an interrupt request, and the interrupt is acknowledged after several instructions that follow the standby instruction have been executed. The instruction that is executed before the interrupt is acknowledged is the instruction that is started to be executed within 6 clocks after the standby instruction has been executed.

**Example**   MOV STBC, #byte  
               NOP  
               NOP  
               NOP

### 22.2.2 Oscillation Stabilization Time Specification Register (OSTS)

The OSTS specifies the oscillator operation and the oscillation stabilization time when STOP mode is released. The EXTC bit of the OSTS specifies whether crystal/ceramic oscillation or an external clock is used. STOP mode can be set when external clock input is used only when the EXTC bit is set (1).

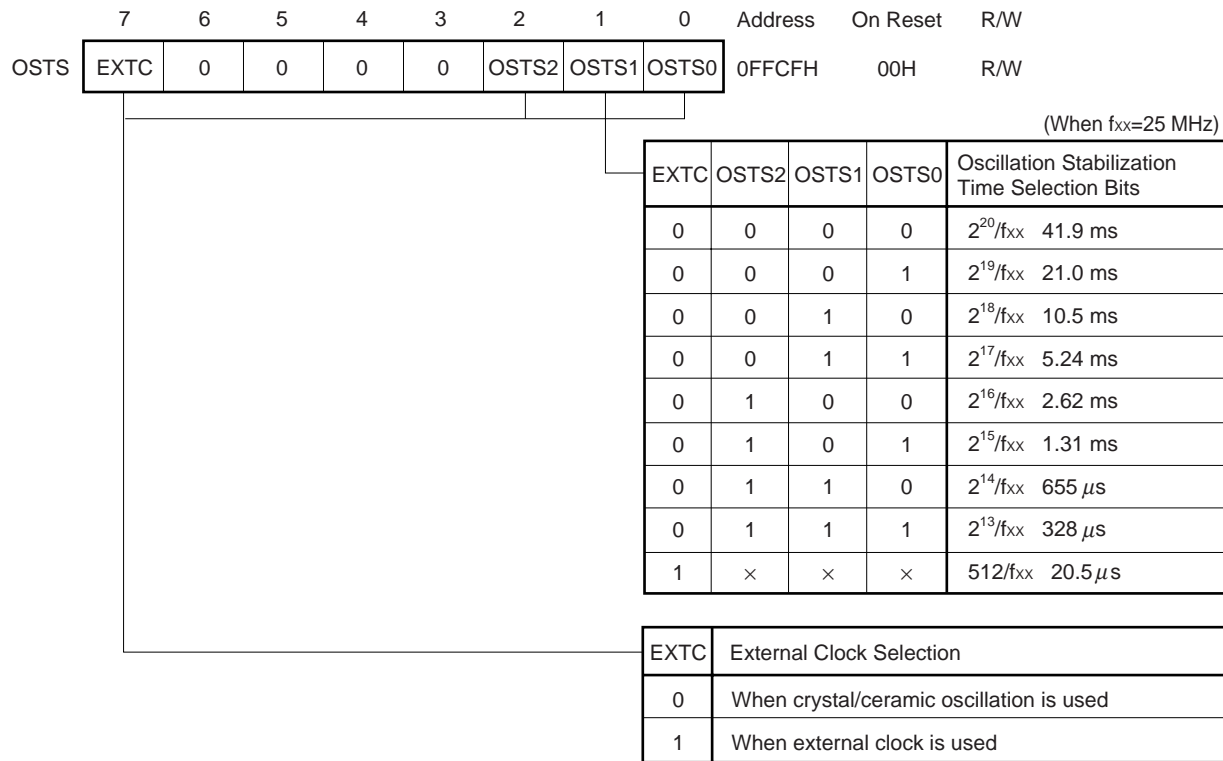
Bits OSTS0 to OSTS2 of the OSTS select the oscillation stabilization time when STOP mode is released. In general, an oscillation stabilization time of at least 40ms should be selected when a crystal resonator is used, and at least 4 ms when a ceramic oscillator is used.

The time taken for oscillation stabilization is affected by the crystal resonator or ceramic resonator used, and the capacitance of the connected capacitor. Therefore, if you want to set a short oscillation stabilization time, you should consult the crystal resonator or ceramic resonator manufacturer.

The OSTS can be written to only with an 8-bit transfer instruction.

$\overline{\text{RESET}}$  input clears the OSTS register to 00H.

The format of the OSTS is shown in Figure 22-4.

**Figure 22-4 Oscillation Stabilization Time Specification Register (OSTS) Format**

- Cautions**
1. When crystal/ceramic oscillation is used, the EXTC bit of the oscillation stabilization time specification register (OSTS) must be cleared (0) before use. If the EXTC bit is set (1), oscillation will stop.
  2. If the STOP mode is used when using external clock input, the EXTC bit must be set (1) before setting STOP mode. If the STOP mode is used with the EXTC bit cleared (0) the  $\mu$ PD784026 may suffer damage or reduced reliability.  
When setting the EXTC bit of OSTS to 1, be sure to input a clock in phase reverse to that of the clock input to the X1 pin, to the X2 pin (refer to 4.3.1 Clock Oscillator).

## 22.3 HALT MODE

### 22.3.1 HALT Mode Setting and Operating States

The HALT mode is selected by setting (1) the HLT bit of the standby control (STBC) register.

The only writes that can be performed on the STBC are 8-bit data writes by means of a dedicated instruction. HALT mode setting is therefore performed by means of the “MOV STBC, #byte” instruction.

- ★ Write an NOP instruction three times after the instruction that has set the HALT mode (or after the HALT mode has been released). Otherwise, several instructions may be executed after the HALT mode has been released and before the interrupt is acknowledged. This may change the sequential relation between interrupt processing and instruction execution, and cause a problem.

**Caution** If HALT mode setting is performed when a condition that releases HALT mode is in effect, HALT mode is not entered, and execution of the next instruction, or a branch to a vectored interrupt service program, is performed. To ensure that a definite HALT mode setting is made, interrupt requests should be cleared (0), etc. before entering HALT mode.

Table 22-1 Operating States in HALT Mode

Clock oscillator		Operating
Internal system clock		Operating
CPU		Operation stopped*
I/O lines		Retain state prior to HALT mode setting
Peripheral functions		Continue operating
Internal RAM		Retained
Bus lines	AD0 to AD7	High-impedance
	A8 to A19	Retained
$\overline{\text{RD}}$ , $\overline{\text{WR}}$ output		High level
ASTB output		Low level
$\overline{\text{REFRQ}}$ output		Continue operating
HLDRQ input		Continue operating (input)
HLDAK output		Continue operating

- \* Macro service processing is executed.

### 22.3.2 HALT Mode Release

HALT mode can be released by the following three sources.

- Non-maskable interrupt request (Only NMI pin input is possible.)
- Maskable interrupt request (vectored interrupt/context switching/macro service)
- $\overline{\text{RESET}}$  input

Release sources and an outline of operations after release are shown in Table 22-2.

The operations after releasing the HALT mode are shown in Table 22-5.



Table 22-2 HALT Mode Release and Operations after Release

Release Source	MK* <sup>1</sup>	IE* <sup>2</sup>	State on Release	Operation after Release
RESET input	×	×	—	Normal reset operation
Non-maskable interrupt request (Only NMI pin input possible/ except watchdog timer* <sup>6</sup> )	×	×	<ul style="list-style-type: none"> <li>Non-maskable interrupt service program not being executed</li> <li>Low-priority non-maskable interrupt service program being executed</li> </ul>	Interrupt request acknowledgment
			<ul style="list-style-type: none"> <li>Service program for same request being executed</li> <li>High-priority non-maskable interrupt service program being executed</li> </ul>	Execution of instruction after MOV STBC, #byte instruction (interrupt request that released HALT mode is held pending* <sup>3</sup> )
Maskable interrupt request (excluding macro service request)	0	1	<ul style="list-style-type: none"> <li>Interrupt service program not being executed</li> <li>Low-priority maskable interrupt service program being executed</li> <li>PRSL bit*<sup>4</sup> cleared (0) during execution of priority level 3 interrupt service program</li> </ul>	Interrupt request acknowledgment
			<ul style="list-style-type: none"> <li>Same-priority maskable interrupt service program being executed (If PRSL bit*<sup>4</sup> is cleared (0), excluding execution of priority level 3 interrupt service program)</li> <li>High-priority interrupt service program being executed</li> </ul>	Execution of instruction after MOV STBC, #byte instruction (interrupt request that released HALT mode is held pending* <sup>3</sup> )
	0	0	—	
	1	×	—	HALT mode retained
Macro service request	0	×	—	Macro service processing execution End condition not established → HALT mode again End condition established → If VCIE* <sup>5</sup> = 1: HALT mode again If VCIE* <sup>5</sup> = 0: Same as release by maskable interrupt request
	1	×	—	HALT mode retained

\* 1. Interrupt mask bit in individual interrupt request source

2. Interrupt enable flag in program status word (PSW)

3. Pending interrupt requests are acknowledged when acknowledgment becomes possible.

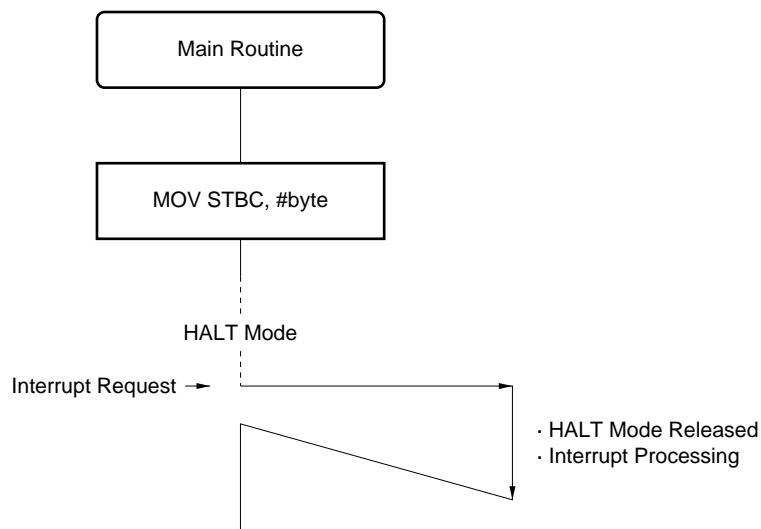
4. Bit in interrupt mode control register (IMC)

5. Bit in macro service mode register of macro service control word in individual macro service request source

6. The HALT mode cannot be released by the watchdog timer.

Figure 22-5. Operation After HALT Mode Has Been Released (1/4)

(1) If interrupt occurs after HALT mode has been set



(2) Reset after HALT mode has been set

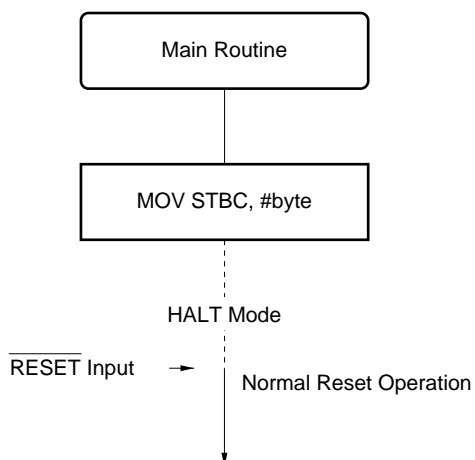
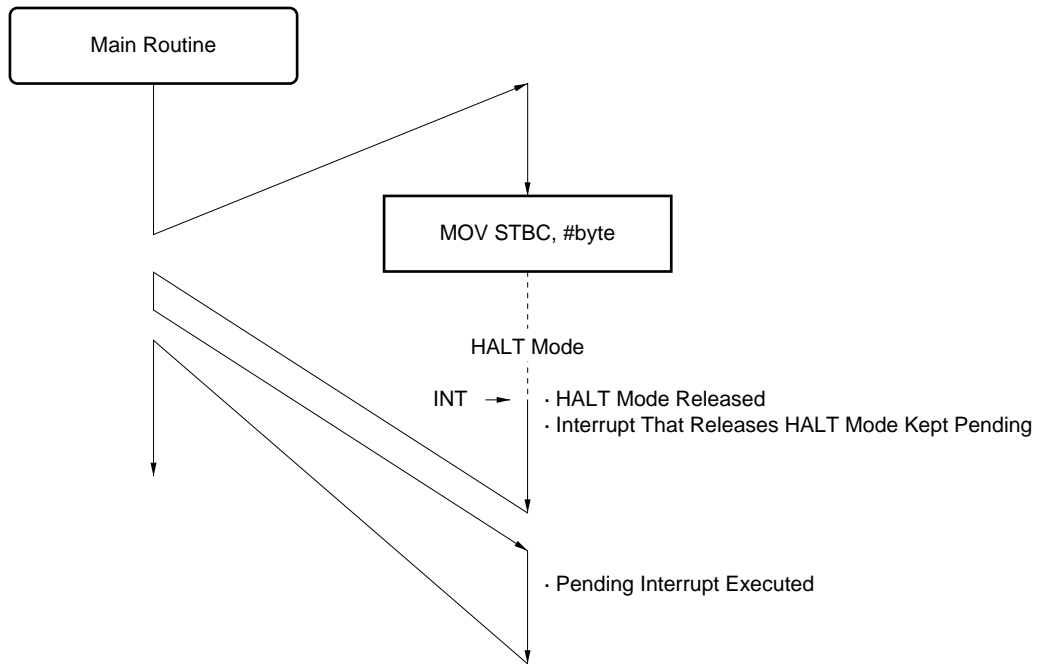


Figure 22-5. Operation After HALT Mode Has Been Released (2/4)

- (3) If HALT mode is set while interrupt routine of interrupt with priority higher than or same as the interrupt that releases the HALT mode is executed



- (4) If HALT mode is set while interrupt routine of interrupt with priority lower than interrupt that releases HALT mode is executed

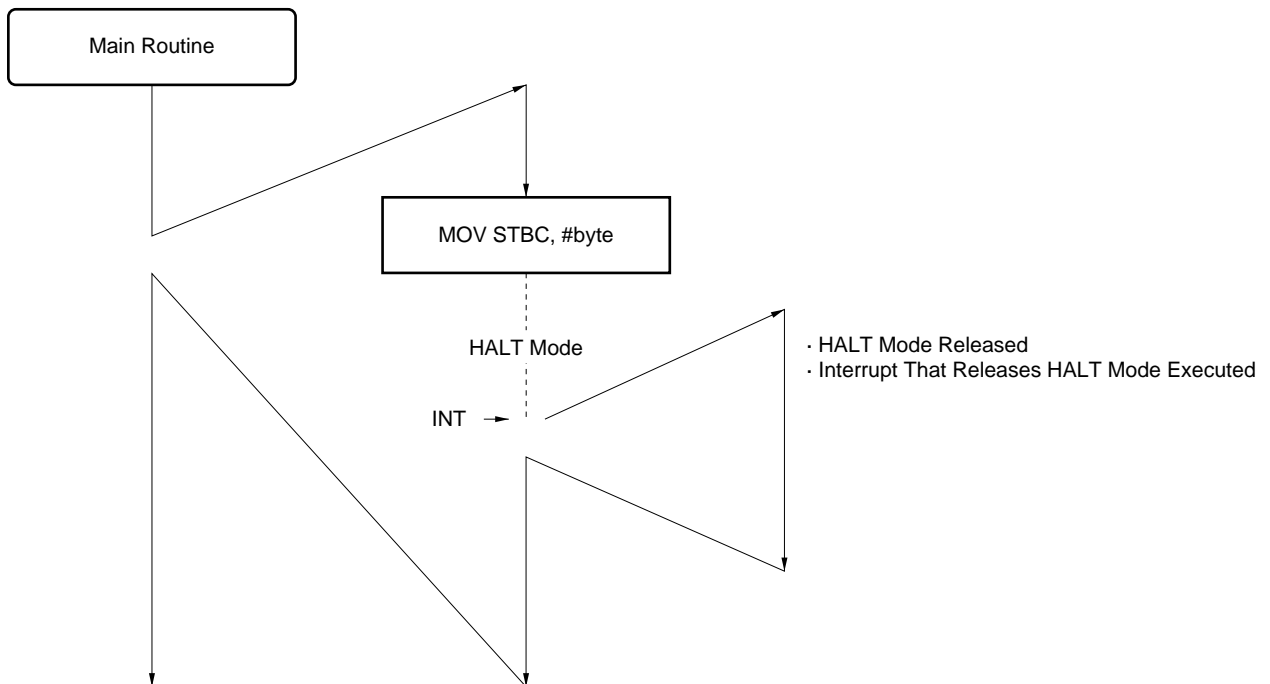
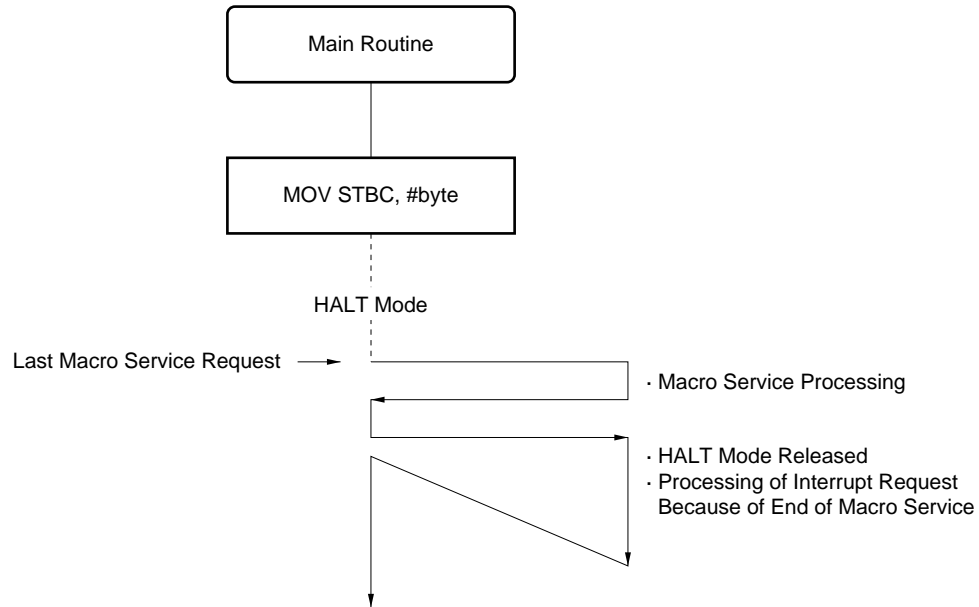


Figure 22-5. Operation After HALT Mode Has Been Released (3/4)

## (5) If macro service request is issued in HALT mode

- (a) When interrupt request is issued (VCIE = 0) immediately after the end condition of macro service has been satisfied



- (b) When interrupt request is not issued (VCIE = 1) if the end condition of macro service is not satisfied, or after the end condition of macro service has been satisfied

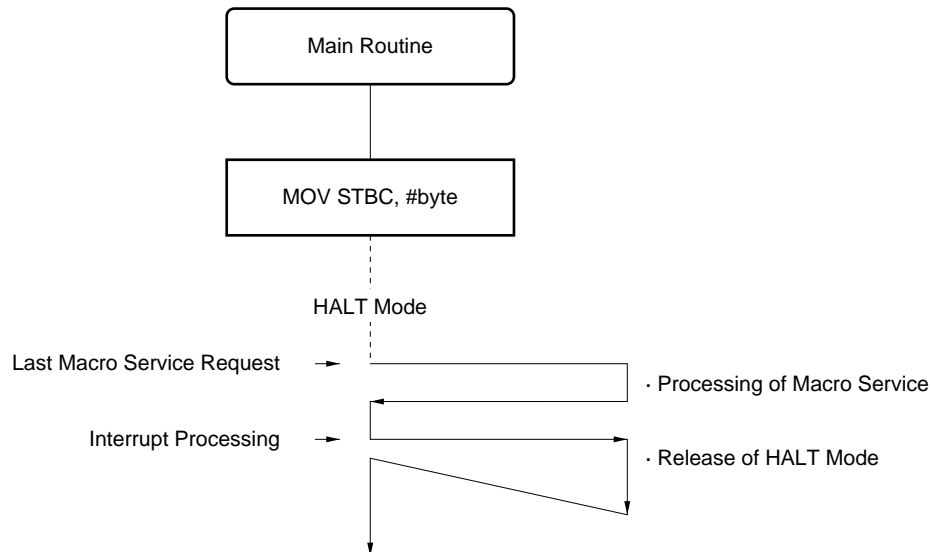
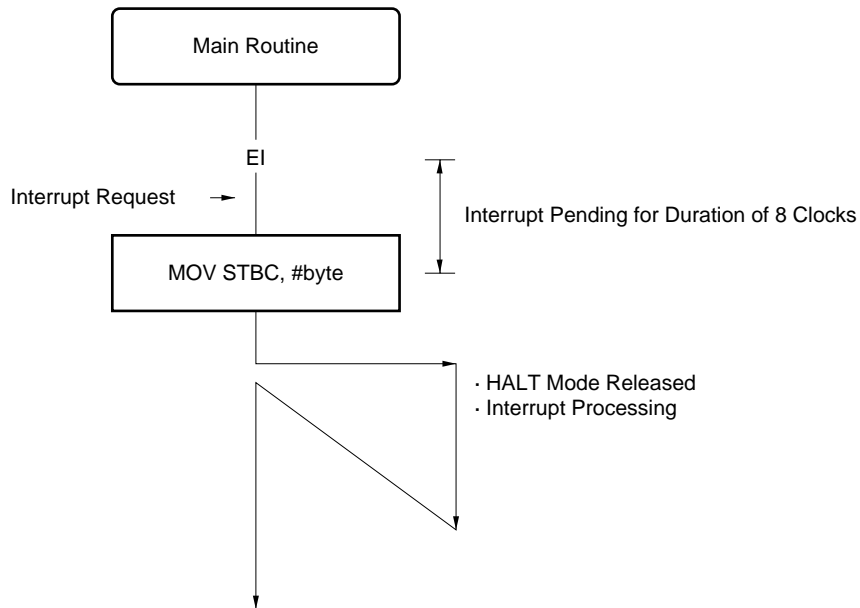
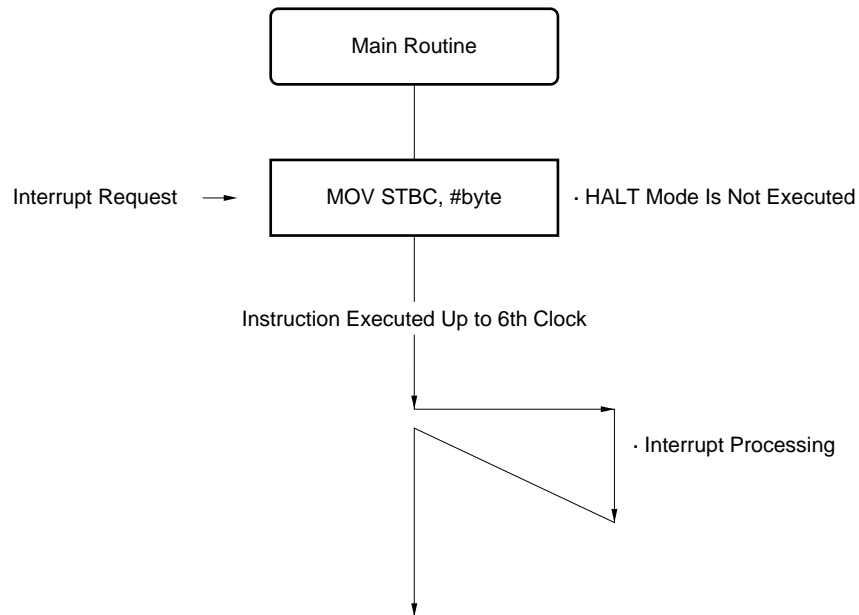


Figure 22-5. Operation After HALT Mode Has Been Released (4/4)

- (6) If interrupt occurs while instruction that temporarily keeps interrupt request pending, and if HALT mode is set while that interrupt is kept pending



- (7) If HALT instruction and interrupt conflict



**(1) Release by non-maskable interrupt**

When a non-maskable interrupt is generated, the  $\mu$ PD784026 is released from HALT mode irrespective of whether the interrupt acknowledgment enabled state (EI) or disabled state (DI) is in effect.

When the  $\mu$ PD784026 is released from HALT mode, if the non-maskable interrupt that released HALT mode can be acknowledged, acknowledgment of that non-maskable interrupt is performed and a branch is made to the service program. If the interrupt cannot be acknowledged, the instruction following the instruction that set the HALT mode (the MOV STBC, #byte instruction) is executed, and the non-maskable interrupt that released the HALT mode is acknowledged when acknowledgment becomes possible. See **20.6 NON-MASKABLE INTERRUPT ACKNOWLEDGMENT OPERATION** for details of non-maskable interrupt acknowledgment.

★ **Caution** The HALT mode cannot be released by the watchdog timer.

**(2) Release by maskable interrupt request**

HALT mode release by a maskable interrupt request can only be performed by an interrupt for which the interrupt mask flag is 0.

When HALT mode is released, if an interrupt can be acknowledged when the interrupt request enable flag (IE) is set (1), a branch is made to the interrupt service program. If the interrupt cannot be acknowledged and if the IE flag is cleared (0), execution is resumed from the instruction following the instruction that set the HALT mode. See **20.7 MASKABLE INTERRUPT ACKNOWLEDGMENT OPERATION** for details of interrupt acknowledgment.

With macro service, HALT mode is released temporarily, service is performed once, then HALT mode is restored. When macro service has been performed the specified number of times, HALT mode is released if the VCIE bit in the macro service mode register of the macro service control word is cleared (0). The operation after release in this case is the same as for release by a maskable interrupt described earlier. If the VCIE bit is set (1), the HALT mode is entered again and is released by the next interrupt request.

Table 22-3 HALT Mode Release by Maskable Interrupt Request

Release Source	MK* <sup>1</sup>	IE* <sup>2</sup>	State on Release	Operation after Release
Maskable interrupt request (excluding macro service request)	0	1	<ul style="list-style-type: none"> <li>Interrupt service program not being executed</li> <li>Low-priority maskable interrupt service program being executed</li> <li>PRSL bit*<sup>4</sup> cleared (0) during execution of priority level 3 interrupt service program</li> </ul>	Interrupt request acknowledgment
			<ul style="list-style-type: none"> <li>Same-priority maskable interrupt service program being executed (If PRSL bit*<sup>4</sup> is cleared (0), excluding execution of priority level 3 interrupt service program)</li> <li>High-priority interrupt service program being executed</li> </ul>	Execution of instruction after MOV STBC, #byte instruction (interrupt request that released HALT mode is held pending* <sup>3</sup> )
	0	0	—	
	1	×	—	HALT mode retained
Macro service request	0	×	—	Macro service processing execution End condition not established → HALT mode again End condition established → If VCIE* <sup>5</sup> = 1: HALT mode again If VCIE* <sup>5</sup> = 0: Same as release by maskable interrupt request
	1	×	—	HALT mode retained

\* 1. Interrupt mask bit in individual interrupt request source

2. Interrupt enable flag in program status word (PSW)

3. Pending interrupt requests are acknowledged when acknowledgment becomes possible.

4. Bit in interrupt mode control register (IMC)

5. Bit in macro service mode register of macro service control word in individual macro service request source

### (3) Release by RESET input

The program is executed after branching to the reset vector address, as in a normal reset operation. However, internal RAM contents retain their value directly before HALT mode was set.

## 22.4 STOP MODE

### 22.4.1 STOP Mode Setting and Operating States

The STOP mode is selected by setting (1) the STP bit of the standby control register (STBC).

The only writes that can be performed on the STBC register are 8-bit data writes by means of a dedicated instruction. STOP mode setting is therefore performed by means of the “MOV STBC, #byte” instruction,

- ★ Write an NOP instruction three times after the instruction that has set the STOP mode (or after the STOP mode has been released). Otherwise, several instructions may be executed after the STOP mode has been released and before the interrupt is acknowledged. This may change the sequential relation between interrupt processing and instruction execution, and cause a problem.

**Caution** If the STOP mode is set when the condition to release the HALT mode is satisfied (refer to 22.3.2 HALT Mode Release), the STOP mode is not set, but the next instruction is executed or execution branches to a vectored interrupt service program. To accurately set the STOP mode, clear the interrupt request before setting the STOP mode.

Table 22-4 Operating States in STOP Mode

Clock oscillator		Oscillation stopped
Internal system clock		Stopped
CPU		Operation stopped
I/O lines		Retain state prior to STOP mode setting
Peripheral functions		All operation stopped*
Internal RAM		Retained
Bus lines	AD0 to AD7	High-impedance
	A8 to A19	High-impedance
$\overline{\text{RD}}$ , $\overline{\text{WR}}$ output		High-impedance
ASTB output		High-impedance
$\overline{\text{REFRQ}}$ output		Retained
HLDRQ input		High-impedance
HLDAK output		Low level

- \* A/D converter operation is stopped, but if the CS bit of the A/D converter mode register (ADM) is set (1), the current consumption does not decrease. D/A converter operation is not stopped.



- ★ **Cautions**
1. When the STOP mode is used in a system that uses an external clock, the EXTC bit of the OSTS must be set (1). If STOP mode setting is performed in a system to which an external clock is input when the EXTC bit of the OSTS is cleared (0), the the current consumption will increase. When setting the EXTC bit of OSTS to 1, be sure to input a clock in phase reverse to that of the clock input to the X1 pin, to the X2 pin (refer to 4.3.1 Clock Oscillator).
  2. The CS bit of the A/D converter mode (ADM) register should be cleared (0).
  3. D/A converter operation is not stopped simply by setting the STOP mode. In order to reduce the current consumption, the DACEn (n = 0, 1) bits of the D/A converter mode register (DAM) must both be cleared (0). When DACEn is cleared (0), the ANOn (n = 0, 1) pin output level becomes high-impedance.

### 22.4.2 STOP Mode Release

STOP mode is released by NMI input, INTP4 input, INTP5 input, and  $\overline{\text{RESET}}$  input.

A summary of the release source and the operations after release is shown in Table 22-5. The operations after releasing the STOP mode is shown in Figure 22-6.

**Table 22-5 STOP Mode Release and Operations after Release**

Release Source	MK* <sup>1</sup>	ISM* <sup>2</sup>	IE* <sup>3</sup>	State after Release	Operation after Release
$\overline{\text{RESET}}$ input	×	×	×	—	Normal reset operation
NMI pin input	×	×	×	<ul style="list-style-type: none"> <li>Non-maskable interrupt service program not being executed</li> <li>Low-priority non-maskable interrupt service program being executed</li> </ul>	Interrupt request acknowledgment
				<ul style="list-style-type: none"> <li>NMI pin input service program being executed</li> <li>High-priority non-maskable interrupt service program being executed</li> </ul>	Execution of instruction after MOV STBC, #byte instruction (interrupt request that released STOP mode is held pending* <sup>4</sup> )
INTP4/INTP5 pin input	0	0	1	<ul style="list-style-type: none"> <li>Interrupt service program not being executed</li> <li>Low-priority maskable interrupt service program being executed</li> <li>PRSL bit*<sup>5</sup> cleared (0) during execution of priority level 3 interrupt service program</li> </ul>	Interrupt request acknowledgment
				<ul style="list-style-type: none"> <li>Same-priority maskable interrupt service program being executed (If PRSL bit*<sup>5</sup> is cleared (0), excluding execution of priority level 3 interrupt service program)</li> <li>High-priority interrupt service program being executed</li> </ul>	Execution of instruction after MOV STBC, #byte instruction (interrupt request that released STOP mode is held pending* <sup>4</sup> )
	0	0	0	—	STOP mode retained
	1	0	×	—	
	×	1	×	—	

\* 1. Interrupt mask bit in individual interrupt request source

2. Macro service enable flag in individual interrupt request source

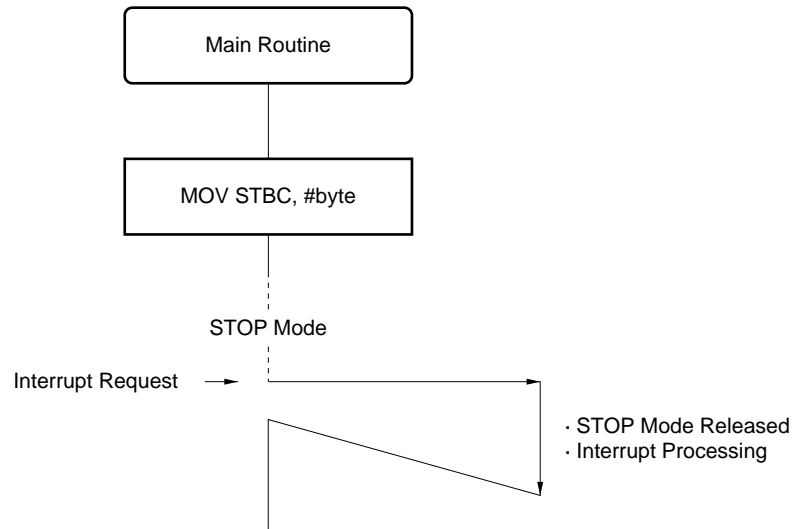
3. Interrupt enable flag in program status word (PSW)

4. Pending interrupt requests are acknowledged when acknowledgment becomes possible.

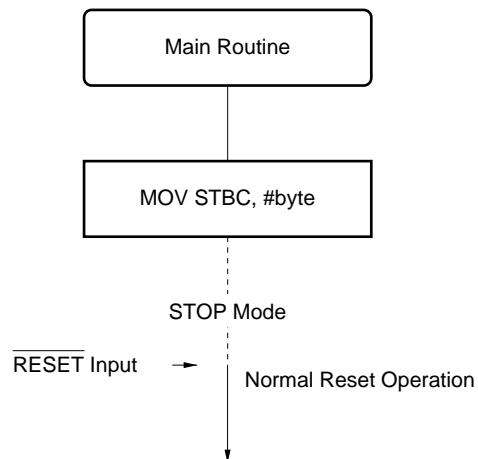
5. Bit in interrupt mode control register (IMC)

Figure 22-6. Operation After STOP Mode Has Been Released (1/2)

(1) If interrupt occurs after STOP mode has been set

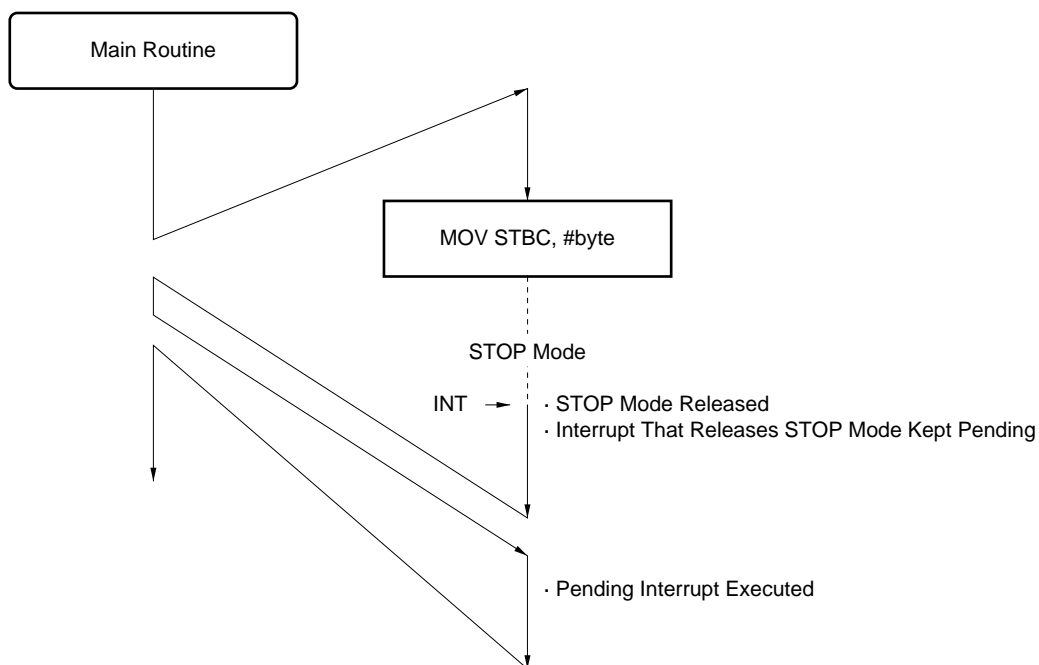


(2) Reset after STOP mode has been set

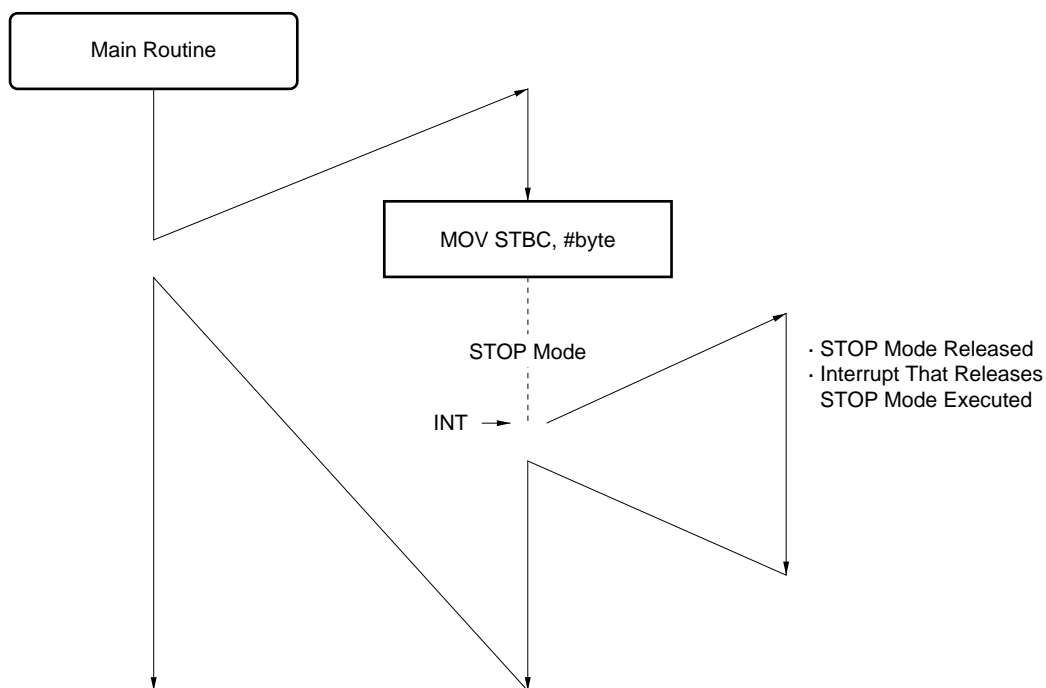


**Figure 22-6. Operation After STOP Mode Has Been Released (2/2)**

- (3) If STOP mode is set while interrupt routine of interrupt with priority higher than or same as the interrupt that releases the STOP mode is executed



- (4) If STOP mode is set while interrupt routine of interrupt with priority lower than interrupt that releases STOP mode is executed



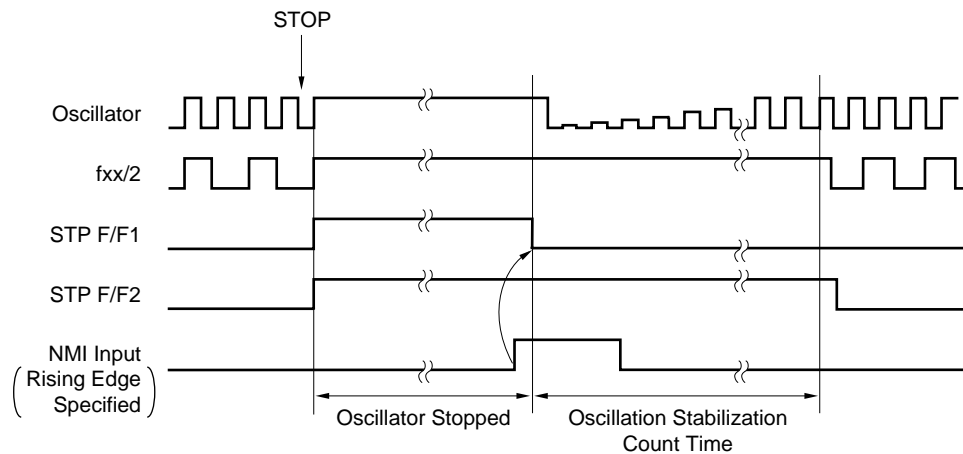
**(1) STOP mode release by NMI input**

The oscillator resumes oscillation when the valid edge specified by external interrupt mode register 0 (INTM0) is input to the NMI input. STOP mode is released after the oscillation stabilization time specified by the oscillation stabilization time specification register (OSTS) elapses.

When the  $\mu$ PD784026 is released from STOP mode, if a non-maskable interrupt by NMI pin input can be acknowledged, a branch is made to the NMI interrupt service program. If the interrupt cannot be acknowledged (if the STOP mode is set in an NMI interrupt service program, etc.), execution is resumed from the instruction following the instruction that set the STOP mode, and a branch is made to the NMI interrupt service program when acknowledgment becomes possible (by execution of an RETI instruction, etc.).

See **20.6 NON-MASKABLE INTERRUPT ACKNOWLEDGMENT OPERATION** for details of NMI interrupt acknowledgment.

**Figure 22-7 STOP Mode Release by NMI Input**

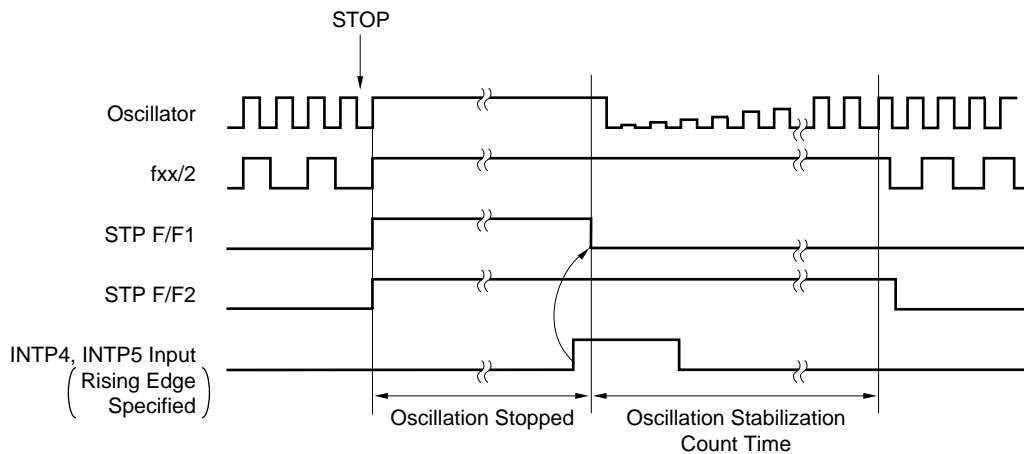


**(2) STOP mode release by INTP4 or INTP5 input**

When masking of interrupts by INTP4 and INTP5 input is released and macro service is disabled, the oscillator resumes oscillation when the valid edge specified by external interrupt mode register 1 (INTM1) is input to the INTP4 or INTP5 input. Following this, STOP mode is released after the oscillation stabilization time specified by the oscillation stabilization time specification register (OSTS) elapses.

When the  $\mu$ PD784026 is released from STOP mode, if an interrupt can be acknowledged when the interrupt enable flag (IE) is set (1), a branch is made to the interrupt service program. If the interrupt cannot be acknowledged and if the IE flag is cleared (0), execution is resumed from the instruction following the instruction that set the STOP mode. See **20.7 MASKABLE INTERRUPT ACKNOWLEDGMENT OPERATION** for details of interrupt acknowledgment.

**Figure 22-8 STOP Mode Release by INTP4/INTP5 Input**

**(3) STOP mode release by  $\overline{\text{RESET}}$  input**

When  $\overline{\text{RESET}}$  input falls from high to low and the reset state is established, the oscillator resumes oscillation. The oscillation stabilization time should be secured while  $\overline{\text{RESET}}$  is active. Thereafter, normal operation is started when  $\overline{\text{RESET}}$  rises.

Unlike an ordinary reset operation, data memory retains its contents prior to STOP mode setting.

## 22.5 IDLE MODE

### 22.5.1 IDLE Mode Setting and Operating States

The IDLE mode is selected by setting (1) both the STP bit and the HLT bit of the standby control (STBC) register.

The only writes that can be performed on the STBC are 8-bit data writes by means of a dedicated instruction. IDLE mode setting is therefore performed by means of the “MOV STBC, #byte” instruction.

- ★ Write an NOP instruction three times after the instruction that has set the IDLE mode (or after the IDLE mode has been released). Otherwise, several instructions may be executed after the IDLE mode has been released and before the interrupt is acknowledged. This may change the sequential relation between interrupt processing and instruction execution, and cause a problem.

**Caution** If the IDLE mode is set when the condition to release the HALT mode is satisfied (refer to 22.3.2 HALT Mode Release), the IDLE mode is not set, but the next instruction is executed or execution branches to a vectored interrupt service program. To accurately set the IDLE mode, clear the interrupt request before setting the IDLE mode.

Table 22-6 Operating States in IDLE Mode

Clock oscillator		Oscillation continues
Internal system clock		Stopped
CPU		Operation stopped
I/O lines		Retain state prior to IDLE mode setting
Peripheral functions		All operation stopped*
Internal RAM		Retained
Bus lines	AD0 to AD7	High-impedance
	A8 to A19	High-impedance
$\overline{\text{RD}}$ , $\overline{\text{WR}}$ output		High-impedance
ASTB output		High-impedance
$\overline{\text{REFRQ}}$ output		Retained
HLDRQ input		High-impedance
HLDAK output		Low level

- \* A/D converter operation is stopped, but if the CS bit of the A/D converter mode register (ADM) is set, the current consumption does not decrease. D/A converter operation is not stopped.

- Cautions**
1. The CS bit of the A/D converter mode (ADM) register should be reset.
  2. D/A converter operation is not stopped simply by setting the IDLE mode. In order to reduce the current consumption, the DACEn (n = 0, 1) bits of the D/A converter mode register (DAM) must both be cleared (0). When DACEn is cleared (0), the ANOn (n = 0, 1) pin output level becomes high-impedance.

### 22.5.2 IDLE Mode Release

IDLE mode is released by NMI input, INTP4 input, INTP5 input, or  $\overline{\text{RESET}}$  input.

A summary of the release sources and the operations after release is shown in Table 22-7. The operation after releasing the IDLE mode is shown in Figure 22-9.

**Table 22-7 IDLE Mode Release and Operations after Release**

Release Source	MK* <sup>1</sup>	ISM* <sup>2</sup>	IE* <sup>3</sup>	State after Release	Operation after Release
$\overline{\text{RESET}}$ input	×	×	×	—	Normal reset operation
NMI pin input	×	×	×	<ul style="list-style-type: none"> <li>Non-maskable interrupt service program not being executed</li> <li>Low-priority non-maskable interrupt service program being executed</li> </ul>	Interrupt request acknowledgment
				<ul style="list-style-type: none"> <li>NMI pin input service program being executed</li> <li>High-priority non-maskable interrupt service program being executed</li> </ul>	Execution of instruction after MOV STBC, #byte instruction (interrupt request that released IDLE mode is held pending* <sup>4</sup> )
INTP4/INTP5 pin input	0	0	1	<ul style="list-style-type: none"> <li>Interrupt service program not being executed</li> <li>Low-priority maskable interrupt service program being executed</li> <li>PRSL bit*<sup>5</sup> cleared (0) during execution of priority level 3 interrupt service program</li> </ul>	Interrupt request acknowledgment
				<ul style="list-style-type: none"> <li>Same-priority maskable interrupt service program being executed (If PRSL bit*<sup>5</sup> is cleared (0), excluding execution of priority level 3 interrupt service program)</li> <li>High-priority interrupt service program being executed</li> </ul>	Execution of instruction after MOV STBC, #byte instruction (interrupt request that released IDLE mode is held pending* <sup>4</sup> )
				—	
				—	IDLE mode retained
				—	

\* 1. Interrupt mask bit in individual interrupt request source

2. Macro service enable flag in individual interrupt request source

3. Interrupt enable flag in program status word (PSW)

4. Pending interrupt requests are acknowledged when acknowledgment becomes possible.

5. Bit in interrupt mode control register (IMC)



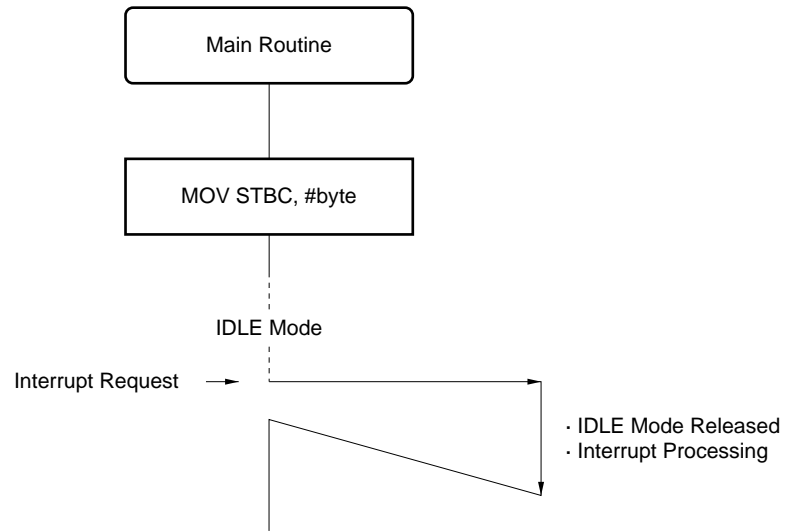
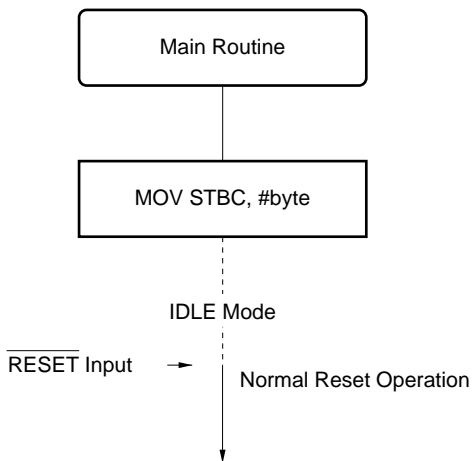
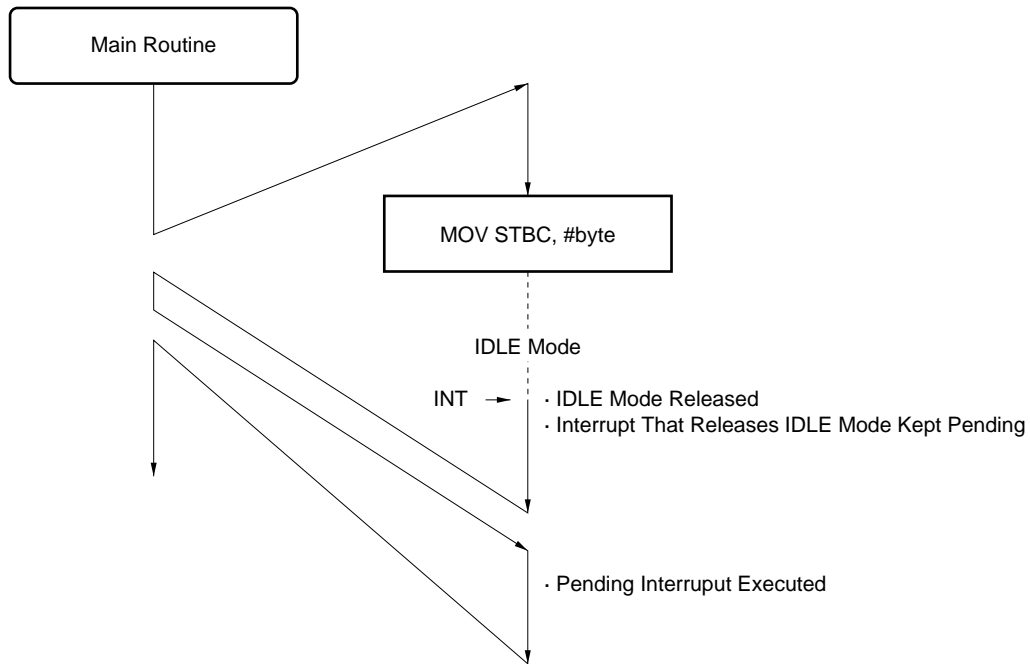
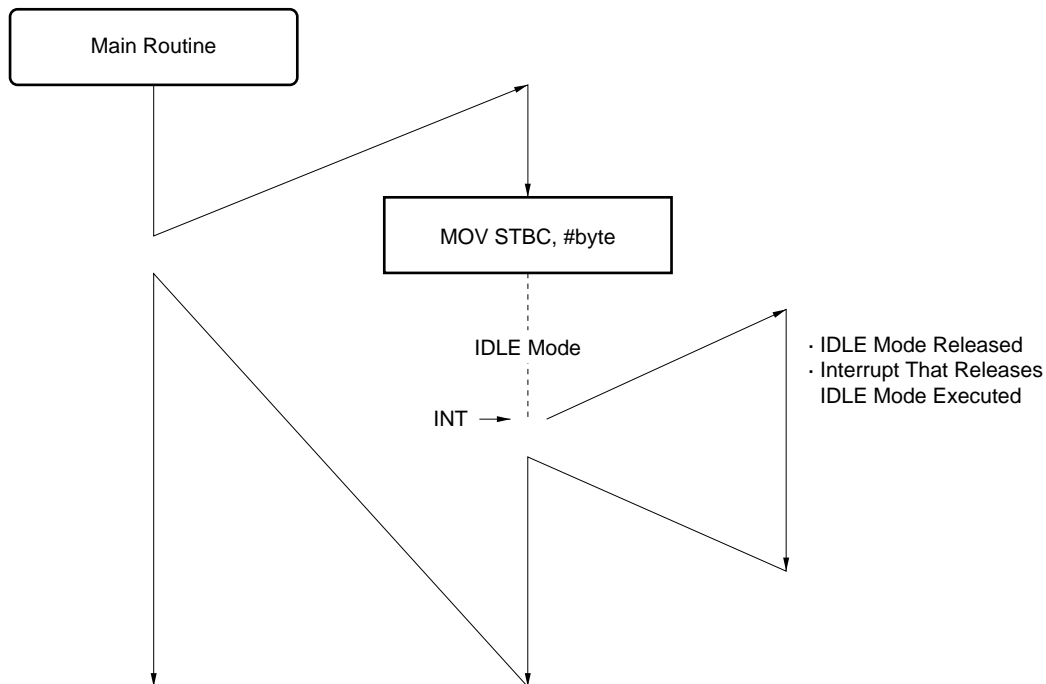
**Figure 22-9. Operation After IDLE Mode Has Been Released (1/2)****(1) If interrupt occurs after IDLE mode has been set****(2) Reset after IDLE mode has been set**

Figure 22-9. Operation After IDLE Mode Has Been Released (2/2)

- (3) If IDLE mode is set while interrupt routine of interrupt with priority higher than or same as the interrupt that releases the IDLE mode is executed



- (4) If IDLE mode is set while interrupt routine of interrupt with priority lower than interrupt that releases IDLE mode is executed



**(1) IDLE mode release by NMI input**

IDLE mode is released when the valid edge specified by external interrupt mode register 0 (INTM0) is input to the NMI input.

When the  $\mu$ PD784026 is released from IDLE mode, if a non-maskable interrupt by NMI pin input can be acknowledged, a branch is made to the NMI interrupt service program. If the interrupt cannot be acknowledged (if the IDLE mode is set in an NMI interrupt service program, etc.), execution is resumed from the instruction following the instruction that set the IDLE mode, and a branch is made to the NMI interrupt service program when acknowledgment becomes possible (by execution of an RETI instruction, etc.).

See **20.6 NON-MASKABLE INTERRUPT ACKNOWLEDGMENT OPERATION** for details of NMI interrupt acknowledgment.

**(2) IDLE mode release by INTP4 or INTP5 input**

When masking of interrupts by INTP4 and INTP5 input is released and macro service is disabled, IDLE mode is released when the valid edge specified by external interrupt mode register 1 (INTM1) is input to the INTP4 or INTP5 input.

When the  $\mu$ PD784026 is released from IDLE mode, if an interrupt can be acknowledged when the interrupt enable flag (IE) is set (1), a branch is made to the interrupt service program. If the interrupt cannot be acknowledged and if the IE flag is cleared (0), execution is resumed from the instruction following the instruction that set the IDLE mode.

See **20.7 MASKABLE INTERRUPT ACKNOWLEDGMENT OPERATION** for details of interrupt acknowledgment.

## 22.6 CHECK ITEMS WHEN STOP MODE/IDLE MODE IS USED

Check items required to reduce the current consumption when STOP mode/IDLE mode is used are shown below.

### (1) Is the output level of each output pin appropriate?

The appropriate output level for each pin varies according to the next-stage circuit. You should select the output level that minimizes the current consumption.

- If high level is output when the input impedance of the next-stage circuit is low, a current will flow from the power supply to the port, resulting in an increased current consumption. This applies when the next-stage circuit is a CMOS IC, etc. When the power supply is off, the input impedance of a CMOS IC is low. In order to suppress the current consumption, or to prevent an adverse effect on the reliability of the CMOS IC, low level should be output. If a high level is output, latchup may result when power is turned on again.
- Depending on the next-stage circuit, inputting low level may increase the current consumption. In this case, high-level or high-impedance output should be used to reduce the current consumption.
- If the next-stage circuit is a CMOS IC, the current consumption of the CMOS IC may increase if the output is made high-impedance when power is supplied to it (the CMOS IC may also be overheated and damaged). In this case you should output an appropriate level, or pull the output high or low with a resistor.

The method of setting the output level depends on the port mode.

- When a port is in control mode, the output level is determined by the status of the internal hardware, and therefore the internal hardware status must be taken into consideration when setting the output level.
- In port mode, the output level can be set by writing to the port output latch and port mode register by software.

When a port is in control mode, its output level can be set easily by changing to port mode.

**(2) Is the input pin level appropriate?**

The voltage level input to each pin should be in the range between  $V_{SS}$  potential and  $V_{DD}$  potential. If a voltage outside this range is applied, the current consumption will increase and the reliability of the  $\mu$ PD784026 may be adversely affected.

Also ensure that an intermediate potential is not applied.

**(3) Are pull-up resistors necessary?**

An unnecessary pull-up resistor will increase the current consumption and cause a latchup of other devices. A mode should be specified in which pull-up resistors are used only for parts that require them.

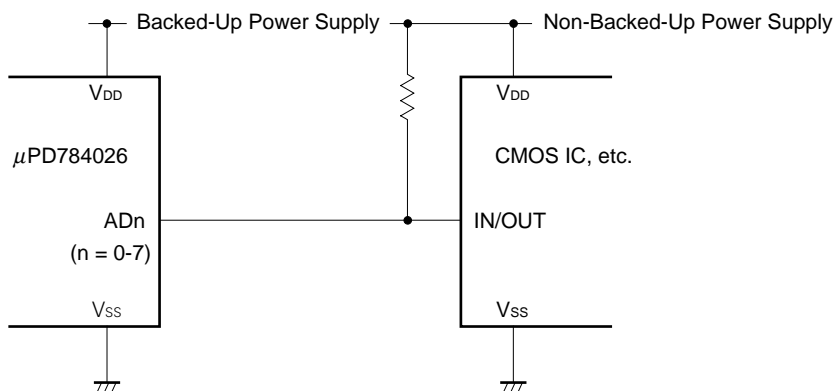
If there is a mixture of parts that do and do not require pull-up resistors, for parts that do, you should connect a pull-up resistor externally and specify a mode in which the internal pull-up resistor is not used.

**(4) Is processing of the address bus, address/data bus, etc., appropriate?**

In STOP mode and IDLE mode, the address bus, address/data bus,  $\overline{RD}$  and  $\overline{WR}$  pins become high-impedance. Normally, these pins are pulled high with a pull-up resistor. If this pull-up resistor is connected to the backed-up power supply, then if the input impedance of circuitry connected to the non-backed-up power supply is low, a current will flow through the pull-up resistor, and the current consumption will increase. Therefore, the pull-up resistor should be connected to the non-backed-up power supply side as shown in Figure 22-10.

Also, in STOP mode and IDLE mode the  $\overline{ASTB}$  pin also becomes high impedance, and the  $\overline{REFRQ}/\overline{HLDAK}$  pin adopts a fixed level. Countermeasures should be taken with reference to the points noted in (1).

**Figure 22-10 Example of Address/Data Bus Processing**



The voltage level input to the  $\overline{WAIT}/\overline{HLDRQ}$  pin should be in the range between  $V_{SS}$  potential and  $V_{DD}$  potential. If a voltage outside this range is applied, the current consumption will increase and the reliability of the  $\mu$ PD784026 may be adversely affected.

**(5) A/D converter**

The current flowing to the  $AV_{DD}$ ,  $AV_{REF1}$  pins can be reduced by clearing (0) the CS bit (bit 7) of the A/D converter mode register (ADM). The current can be further reduced, if required, by cutting the current supply to the  $AV_{REF1}$  pin with external circuitry.

Make sure that the  $AV_{DD}$  pin is not at the same potential as the  $V_{DD}$  pin. Unless power is supplied to the  $AV_{DD}$  pin in the STOP mode, not only does the current consumption increase, but the reliability is also affected.

**(6) D/A converter**

In the STOP mode and IDLE mode the D/A converter still consumes a certain current. Clearing (0) the both DACEn ( $n = 0, 1$ ) bits of the D/A converter mode register (DAM) sets the ANOn ( $n = 0/1$ ) output to high impedance, enabling the current consumption to be reduced. (power consumption is not reduced if only one of the DACEn bits is cleared to 0).

The current consumption at resistor string can be eliminated by setting the voltage input to the  $AV_{REF2}$  pin to the same potential as  $AV_{REF3}$ . The ANOn output when the DACEn bit of the DAM is set (1) will be at the same potential as  $AV_{REF3}$ , and therefore the  $AV_{REF3}$  pin voltage should be set so as to minimize the current consumption of the next-stage circuit. The current consumption of the  $\mu$ PD784026 can be minimized by making the input voltage of the  $AV_{REF2}$  pin the same as the input voltage of the  $AV_{REF3}$  pin, and by clearing both the DACEn bits of DAM to 0. However, the output of the ANOn pin goes into a high-impedance state.

Also, a voltage should not be applied to the ANOn pins from off-chip, as this may result in an increase in the current consumption, and the  $\mu$ PD784026 may suffer damage or reduced reliability.

## 22.7 CAUTIONS

- (1) If HALT/STOP/IDLE mode (standby mode hereafter) setting is performed when a condition that releases HALT mode (refer to **22.3.2 HALT Mode Release**) is satisfied, standby mode is not entered, and execution of the next instruction, or a branch to a vectored interrupt service program, is performed. To ensure that a definite standby mode setting is made, interrupt requests should be cleared, etc. before entering standby mode.
- (2) When crystal/ceramic oscillation is used, the EXTC bit must be cleared (0) before use. If the EXTC bit is set (1), oscillation will stop.
- ★ (3) When the STOP mode is used in a system that uses an external clock, the EXTC bit of the OSTS must be set (1). If STOP mode setting is performed in a system to which an external clock is input when the EXTC bit of the OSTS is cleared (0), the current consumption will increase.  
When setting the EXTC bit of OSTS to 1, be sure to input a clock in phase reverse to that of the clock input to the X1 pin, to the X2 pin (refer to **4.3.1 Clock Generation Circuit**).
- (4) In STOP mode and IDLE mode, the CS bit of the A/D converter mode ADM register should be cleared (0).
- (5) D/A converter operation is not stopped simply by setting the STOP mode or IDLE mode. In order to reduce the current consumption, the DACEn ( $n = 0, 1$ ) bits of the D/A converter mode register (DAM) must both be cleared (0). When DACEn is cleared (0), the ANOn ( $n = 0, 1$ ) pin output level becomes high-impedance.
- (6) Execute a NOP instruction three times after the standby instruction (after the standby mode has been released). Otherwise, the standby instruction is not executed if execution of the standby instruction conflicts with an interrupt request, and the interrupt is acknowledged after several instructions that follow the standby instruction have been executed. The instruction that is executed before the interrupt is acknowledged is the instruction that is started to be executed within 6 clocks after the standby instruction has been executed.

**Example**   MOV STBC, #byte  
              NOP  
              NOP  
              NOP  
              :  
              :

[MEMO]



## CHAPTER 23 RESET FUNCTION

### 23.1 RESET FUNCTION

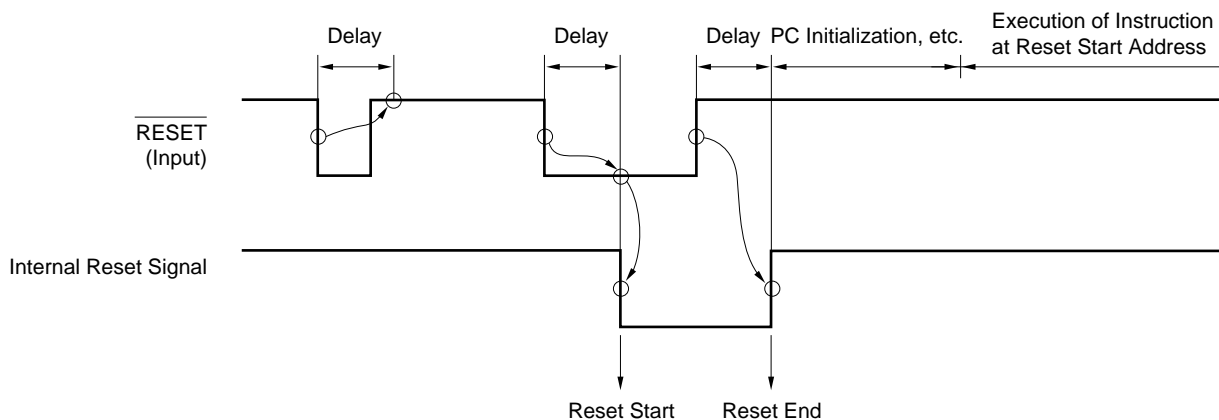
When low level is input to the  $\overline{\text{RESET}}$  input pin, a system reset is affected, the various hardware units are set to the states shown in Table 23-2, and all pins except the power supply pins and the X1 and X2 pins are placed in the high-impedance state. Table 23-1 shows the pin statuses on reset and after reset release.

When the  $\overline{\text{RESET}}$  input changes from low to high level, the reset state is released, the contents of address 00000H of the reset vector table are set in bits 0 to 7 of the program counter (PC), the contents of address 00001H in bits 8 to 15, and 0000B in bits 16 to 19, a branch is made, and program execution is started at the branch destination address. A reset can therefore be performed from any address in the base area.

The contents of the various registers should be initialized as required in the program in the base area.

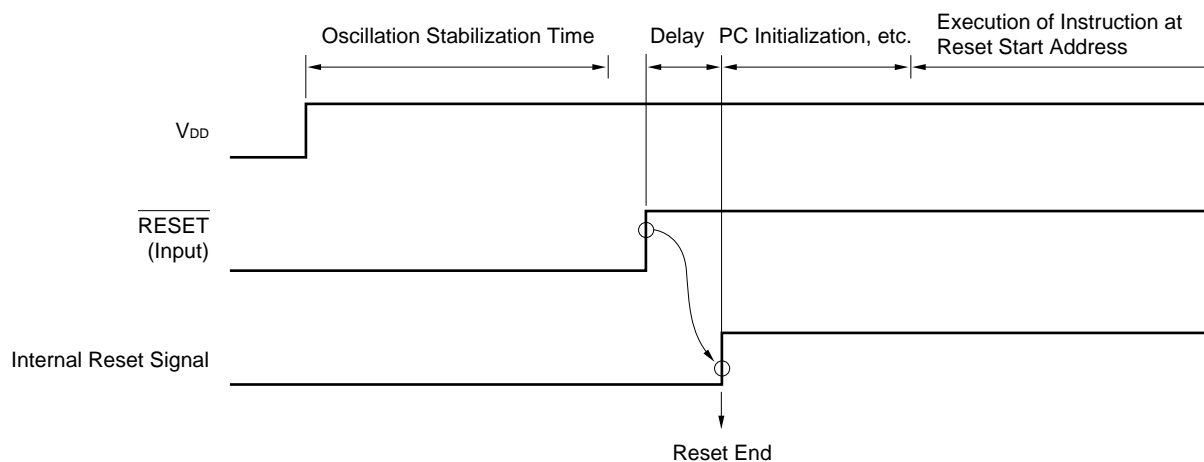
To prevent misoperation due to noise, the  $\overline{\text{RESET}}$  input pin incorporates an analog delay noise elimination circuit (see Figure 23-1).

**Figure 23-1 Reset Signal Acknowledgment**



In a reset operation upon powering on, the  $\overline{\text{RESET}}$  signal must be kept active until the oscillation stabilization time has elapsed (approx. 40 ms, depending on the resonator used).

**Figure 23-2 Power-On Reset Operation**



**Remark**  $f_{\text{CLK}}$ : Internal system clock frequency

**Table 23-1 Pin Statuses During Reset Input and After Reset Release**

Pin Name	Input/Output	On Reset	Directly After Reset Release
P00-P07	Input/output	Hi-Z	Hi-Z (input port mode)
P10/PWM0-P17	Input/output	Hi-Z	Hi-Z (input port mode)
P20/NMI-P27/SI	Input	Hi-Z	Hi-Z (input port)
P30/RxD-P37/TO3	Input/output	Hi-Z	Hi-Z (input port mode)
P40/AD0-P47/AD7	Input/output	Hi-Z	Hi-Z (input port mode)* <sup>1</sup>
P50/A8-P57/A15	Input/output	Hi-Z	Hi-Z (input port mode)* <sup>1</sup>
P60/A16-P63/A19* <sup>2</sup>	Input/output	Hi-Z	Hi-Z (input port mode)* <sup>1</sup>
P64/ $\overline{\text{RD}}$ , P65/ $\overline{\text{WR}}$	Input/output	Hi-Z	Hi-Z (input port mode)* <sup>1</sup>
P66/ $\overline{\text{WAIT}}$ , P67/ $\overline{\text{REFRQ}}$	Input/output	Hi-Z	Hi-Z (input port mode)
P70/ANI0-P77/ANI7	Input/output	Hi-Z	Hi-Z (input port mode)
ASTB/CLKOUT	Output	Hi-Z	0
ANO0, ANO1	Output	Hi-Z	Outputs $\text{AV}_{\text{REF3}}$ pin input voltage

- \* 1. In the  $\mu\text{PD784021}$  these pins function as the address/data bus, and output signals to fetch the reset vector address from address 0000H (see **Figure 23-3 (a)**).
2. In the  $\mu\text{PD784021}$ , these pins operate as output port pins only, and output "0" after reset release.

**Table 23-2 Hardware States After Reset (1/2)**

Hardware			State After Reset	
Program counter (PC)			Set with contents of reset vector table (0000H/0001H).	
Stack pointer (SP)			Undefined* <sup>1</sup>	
Program status word (PSW)			02H	
Internal RAM	Data memory		Undefined* <sup>1</sup>	
	General-purpose registers			
Ports	Ports 0, 1, 2, 3, 4, 5, 6* <sup>2</sup> , 7		Undefined (high impedance)	
Port mode registers	PM0, PM1, PM3, PM4, PM5, PM6* <sup>3</sup> , PM7		FFH	
Port mode control registers (PMC1, PMC3)			00H	
Pull-up resistor option register (PUO)			00H	
Real-time output port control register (RTPC)			00H	
Timer/counter	Timer registers (TM0, TM1W, TM2W, TM3W)		0000H	
	Compare registers (CR00, CR01, CR10LW, CR20W, CR30W)		Undefined	
	Capture registers (CR02, CR12W, CR22W)			
	Capture/compare registers (CR11W, CR21W)			
	Timer control registers (TMC0, TMC1)		00H	
	Timer output control register (TOC)			
	Capture/compare control registers		CRC0	10H
			CRC1, CRC2	00H
	Prescaler mode registers (PRM0, PRM1)		00H	
	One-shot pulse output control register (OSPC)		00H	
PWM	PWM control register (PWMC)		05H	
	PWM prescaler register (PWPR)		00H	
	PWM modulo registers (PWM0, PWM1)		Undefined	
A/D converter	A/D converter mode register (ADM)		00H	
	A/D conversion result register (ADCR)		Undefined	
D/A converter	D/A converter mode register (DAM)		03H	
	D/A conversion value setting registers (DACS0, DACS1)		00H	

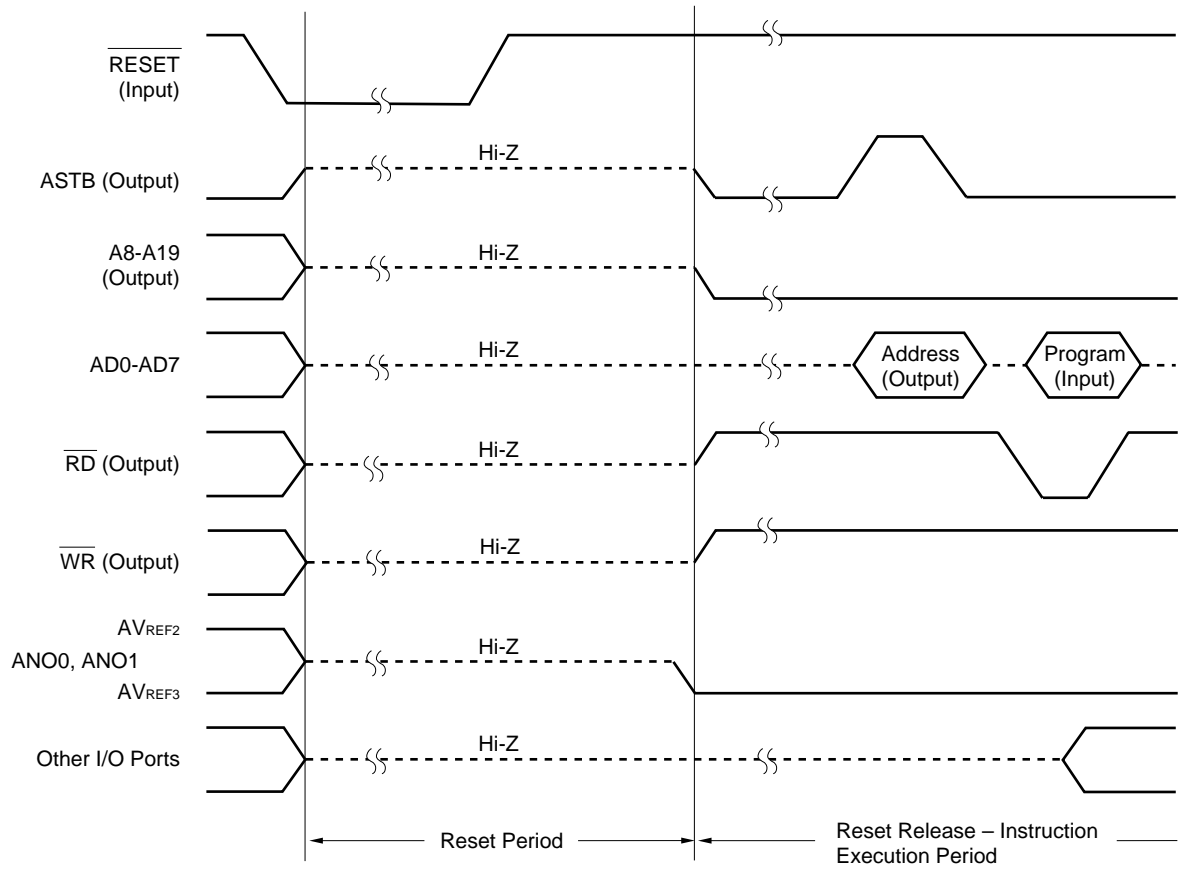
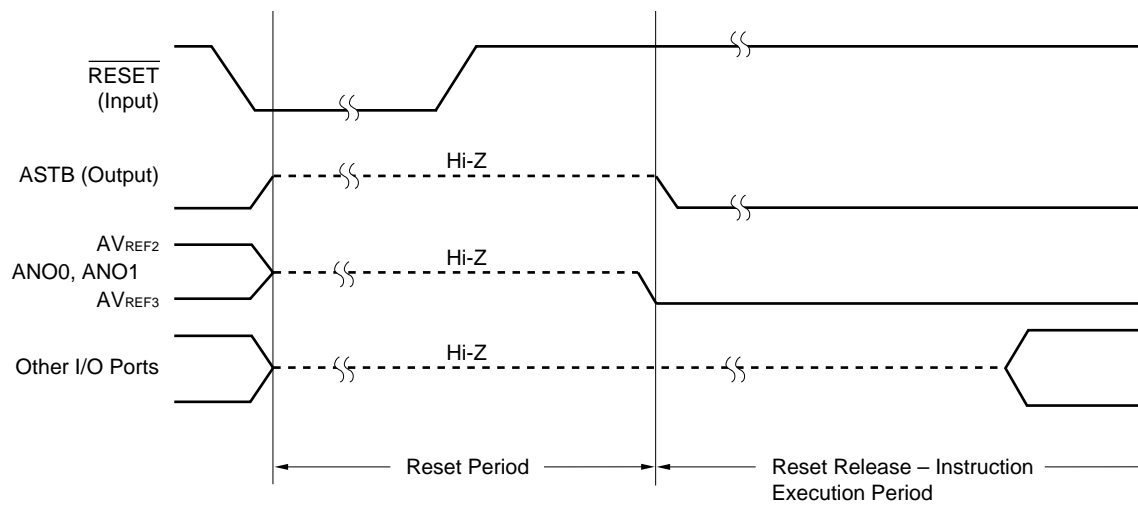
\* 1. When HALT mode, STOP mode or IDLE mode is released by  $\overline{\text{RESET}}$  input, the value before that mode was set is retained.

2.  $\mu\text{PD784021}$ :  $\times 0\text{H}$

3.  $\mu\text{PD784021}$ :  $\text{F}\times\text{H}$

**Table 23-2 Hardware States After Reset (2/2)**

Hardware			State After Reset
Serial interface	Clocked serial interface mode registers (CSIM, CSIM1, CSIM2)		00H
	Shift registers (SIO, SIO1, SIO2)		Undefined
	Asynchronous serial interface mode registers (ASIM, ASIM2)		00H
	Asynchronous serial interface status registers (ASIS, ASIS2)		00H
	Serial bus interface control register (SBIC)		00H
	Serial receive buffers (RXB, RXB2)		Undefined
	Serial transmit shift registers (TXS, TXS2)		Undefined
	Baud rate generator control registers (BRGC, BRGC2)		00H
Clock output function (CLOM)			00H
Memory extension mode register (MM)			20H
Programmable wait control registers		PWC1	AAH
		PWC2	AAAAH
Refresh function	Refresh mode register (RFM)		00H
	Refresh area specification register (RFA)		00H
Hold mode register (HLDM)			00H
Interrupts	Interrupt control registers (PIC0, PIC1, PIC2, PIC3, PIC4, PIC5, CIC00, CIC01, CIC10, CIC11, CIC20, CIC21, CIC30, ADIC, SERIC, SRIC, STIC, SERIC2, SRIC2, STIC2, CSIIC, CSIIC1, CSIIC2)		43H
	Interrupt mask registers	MK0	FFFFH
		MK1L	FFH
	In-service priority register (ISPR)		00H
	Interrupt mode control register (IMC)		00H
External interrupt mode registers (INTM0, INTM1)			00H
Sampling clock selection register (SCS0)			00H
Standby control register (STBC)			30H
Oscillation stabilization time specification register (OSTS)			00H
Internal memory size switching register (IMS)			FFH

**Figure 23-3 Reset Input Timing****(a)  $\mu$ PD784021****(b)  $\mu$ PD784026**

## 23.2 CAUTIONS

Reset input when powering on must remain at the low level until oscillation stabilizes after the supply voltage has reached the prescribed voltage.

## CHAPTER 24 $\mu$ PD78P4026 PROGRAMMING

The  $\mu$ PD78P4026 incorporates a 64 K-byte PROM as program memory. When programming the  $\mu$ PD78P4026, the PROM programming mode is set by means of the  $V_{PP}$  pin and the  $\overline{\text{RESET}}$  pin. For the connection of unused pins, see **1.3.2 PROM Programming Mode** in **1.3 PIN CONFIGURATION (Top View)**.

### 24.1 OPERATING MODES

When +5 V or +12.5 V is applied to the  $V_{PP}$  pin and a low-level signal is applied to the  $\overline{\text{RESET}}$  pin, the  $\mu$ PD78P4026 is placed in the PROM programming mode. This is one of the operating modes shown in Table 24-1 below according to the setting of the  $\overline{\text{CE}}$ ,  $\overline{\text{OE}}$ , and  $\overline{\text{PGM}}$  pins.

The PROM contents can be read by setting the read mode.

**Table 24-1 PROM Programming Operating Modes**

<div>Pins</div> <div>Operating Mode</div>	<div><math>\overline{\text{RESET}}</math></div>	<div><math>V_{\text{PP}}</math></div>	<div><math>V_{\text{DD}}</math></div>	<div><math>\overline{\text{CE}}</math></div>	<div><math>\overline{\text{OE}}</math></div>	<div><math>\overline{\text{PGM}}</math></div>	<div>D0-D7</div>
Page data latch	L	+12.5 V	+6.5 V	H	L	H	Data input
Page write				H	H	L	High-impedance
Byte write				L	H	L	Data input
Program verify				L	L	H	Data output
Program inhibit				×	H	H	High-impedance
				×	L	L	
Read	L	+5 V	+5 V	L	L	H	Data output
Output disable				L	H	×	High-impedance
Standby				H	×	×	High-impedance

**Remark** ×: L or H

**(1) Read mode**

Read mode is set by setting  $\overline{CE}$  to L and  $\overline{OE}$  to L.

**(2) Output disable mode**

If  $\overline{OE}$  is set to H, data output becomes high impedance and the output disable mode is set.

Therefore, if multiple  $\mu$ PD78P4026s are connected to the data bus, data can be read from any one device by controlling the  $\overline{OE}$  pin.

**(3) Standby mode**

Setting  $\overline{CE}$  to H sets the standby mode.

In this mode, data output becomes high-impedance irrespective of the status of  $\overline{OE}$ .

**(4) Page data latch mode**

Setting  $\overline{CE}$  to H,  $\overline{PGM}$  to H, and  $\overline{OE}$  to L at the start of the page write mode sets the page data latch mode.

In this mode, 1-page 4-byte data is latched in the internal address/data latch circuit.

**(5) Page write mode**

After 1-page 4-byte address and data are latched in the page data latch mode, a page write is executed by applying a 0.1 ms program pulse (active-low) to the  $\overline{PGM}$  pin while  $\overline{CE} = H$  and  $\overline{OE} = H$ . After this, program verification can be performed by setting  $\overline{CE}$  to L and  $\overline{OE}$  to L.

If programming is not performed by one program pulse, repeated write and verify operations are executed X times ( $X \leq 10$ ).

**(6) Byte write mode**

A byte write is executed by applying a 0.1 ms program pulse (active-low) to the  $\overline{PGM}$  pin while  $\overline{CE} = L$  and  $\overline{OE} = H$ . After this, program verification can be performed by setting  $\overline{OE}$  to L.

If programming is not performed by one program pulse, repeated write and verify operations are executed X times ( $X \leq 10$ ).

**(7) Program verify mode**

Setting  $\overline{CE}$  to L,  $\overline{PGM}$  to H, and  $\overline{OE}$  to L sets the program verify mode.

After writing is performed, this mode should be used to check whether the data has been written correctly.

**(8) Program inhibit mode**

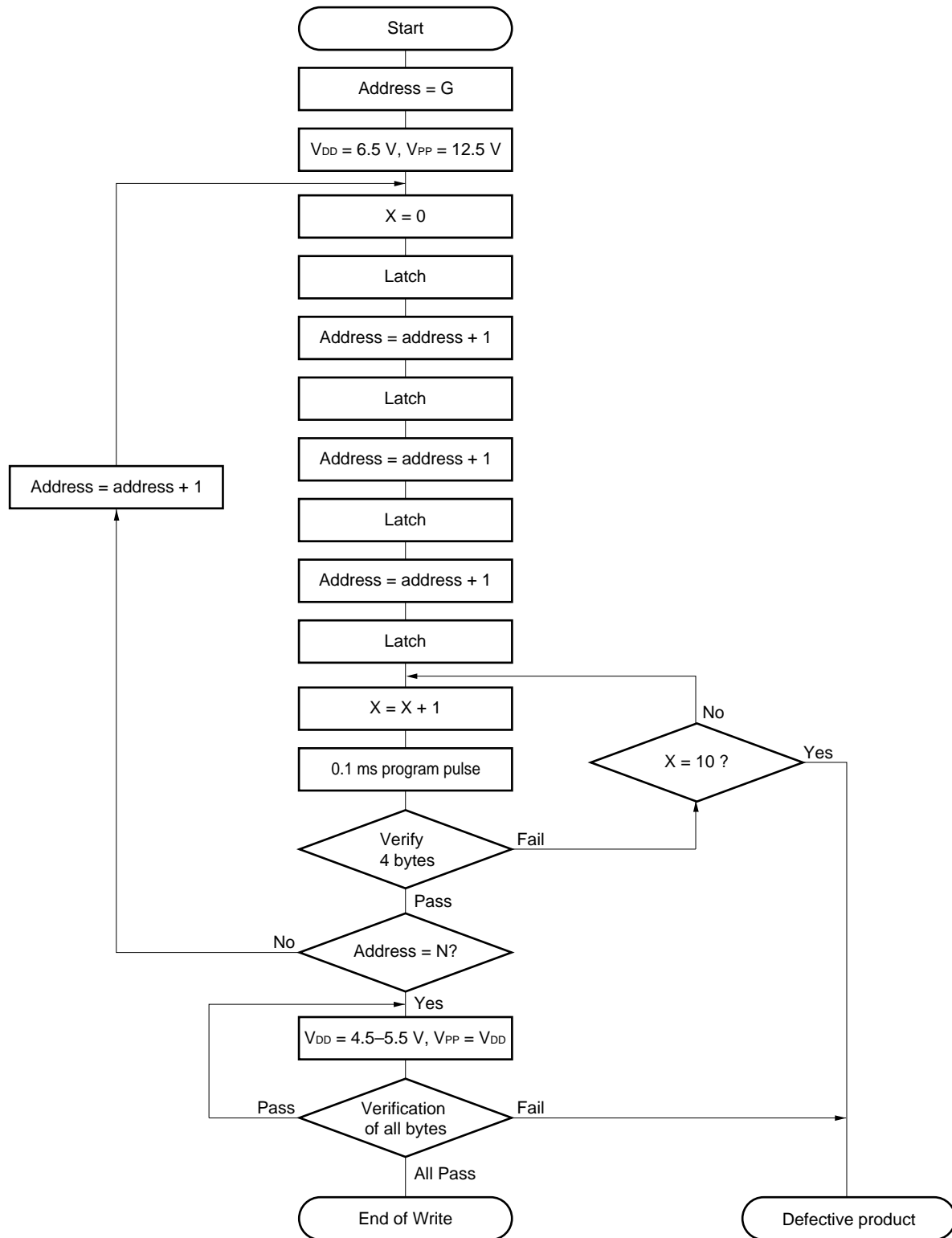
The program inhibit mode is used when the  $\overline{OE}$  pins,  $V_{PP}$  pins and pins D0 to D7 of multiple  $\mu$ PD78P4026s are connected in parallel, and you wish to write to one of these devices.

The page write mode or byte write mode described above is used to perform a write. At this time, a write is not performed on devices on which the  $\overline{PGM}$  pin is driven high.

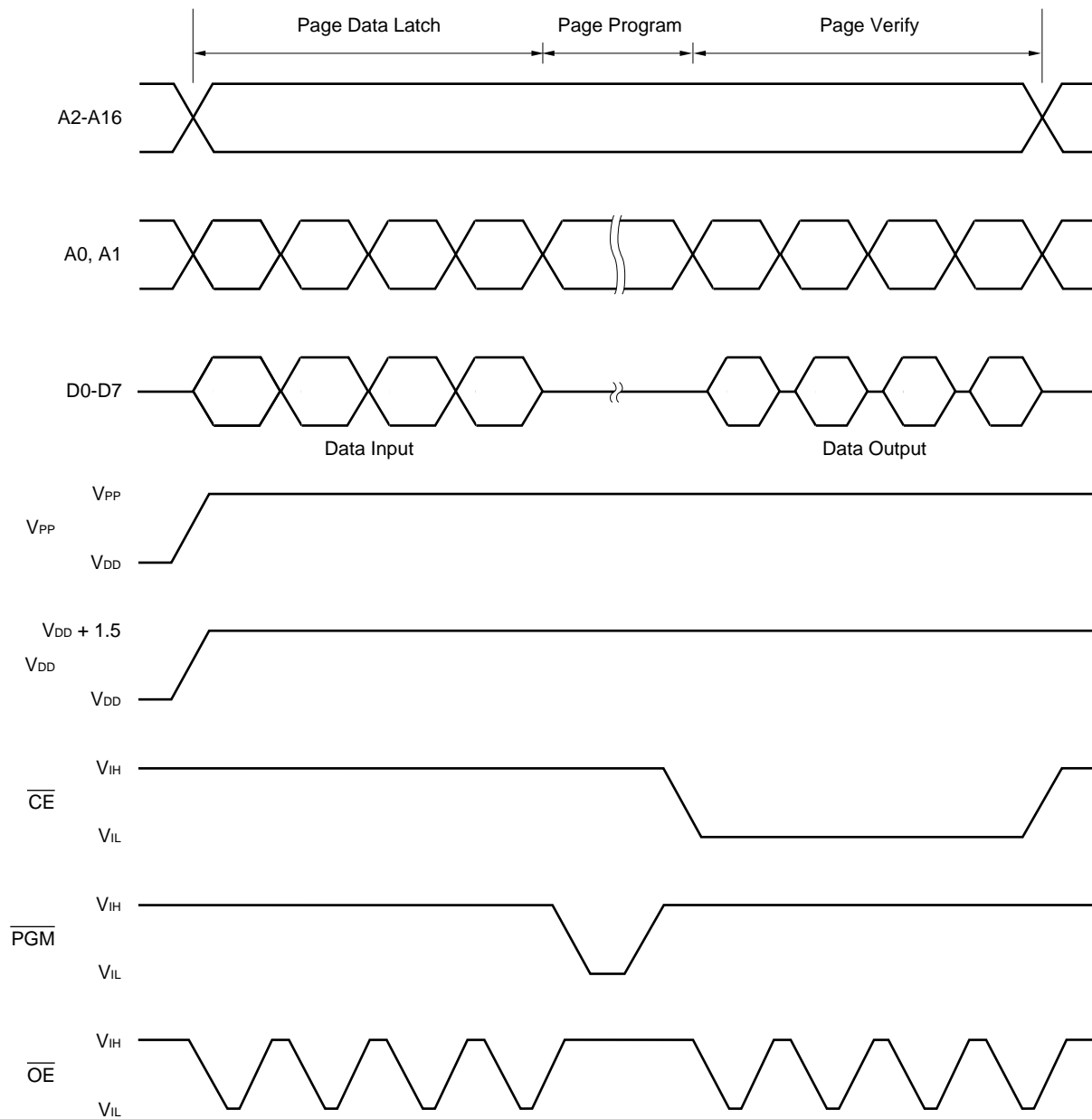


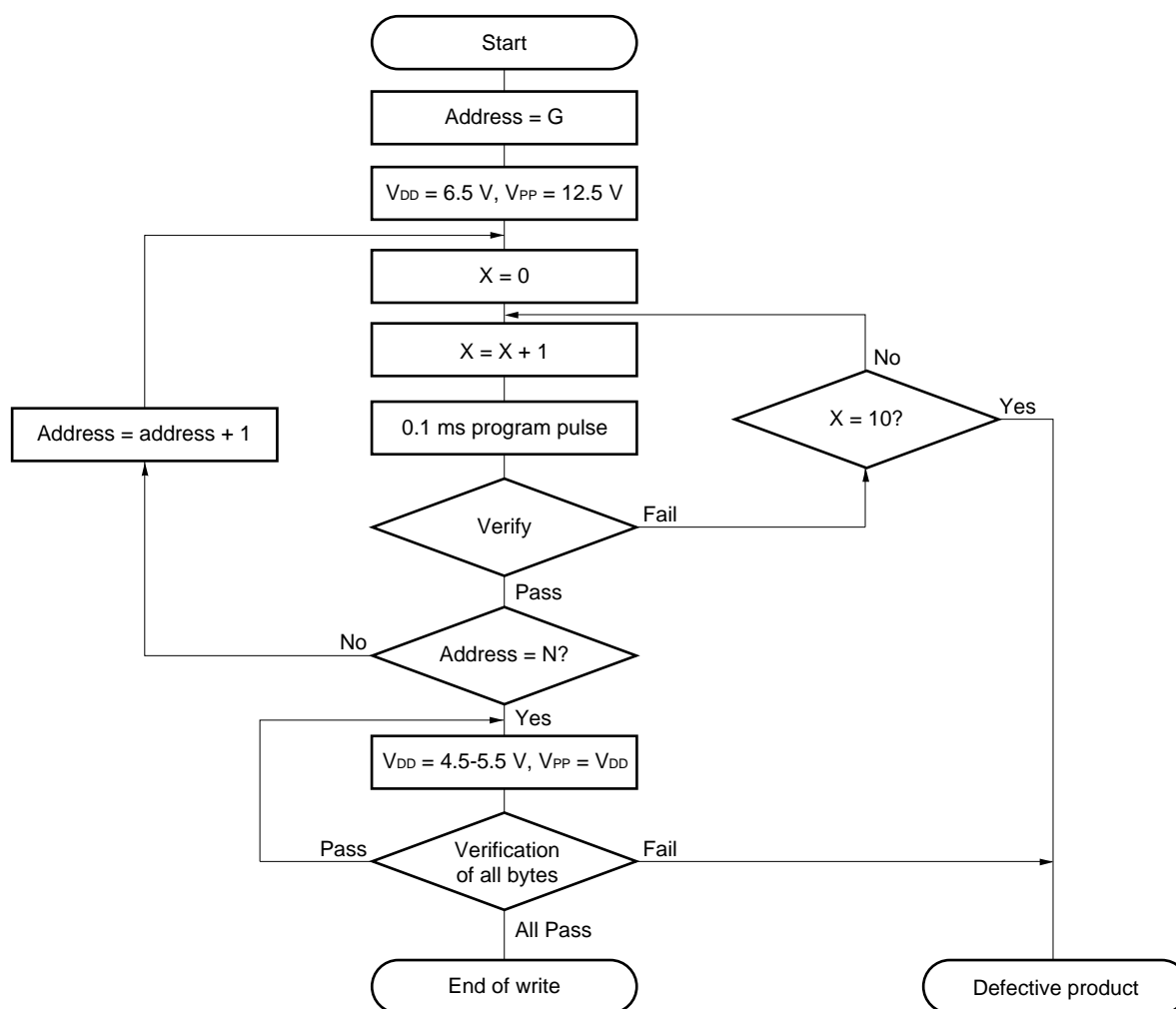
## 24.2 PROM WRITE PROCEDURE

Figure 24-1 Page Program Mode Flowchart

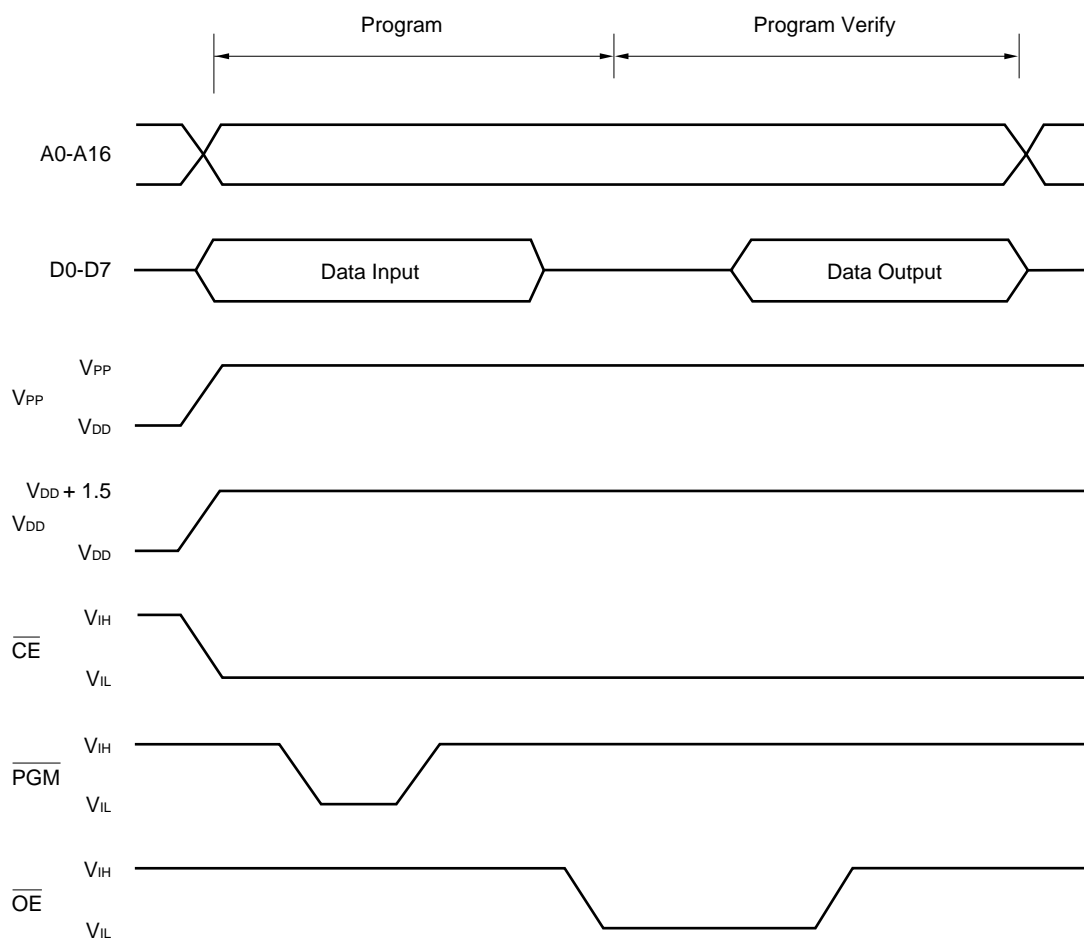


**Remark** G = Start address  
N = Last address of program

**Figure 24-2 Page Program Mode Timing**

**Figure 24-3 Byte Program Mode Flowchart**

**Remark** G = Start address  
N = Last address of program

**Figure 24-4 Byte Program Mode Timing**

- Cautions**
1. Ensure that V<sub>DD</sub> is applied before V<sub>PP</sub>, and cut after V<sub>PP</sub>.
  2. Ensure that V<sub>PP</sub> does not become +13.5 V or over including overshoot.
  3. Removing the device while +12.5 V is being applied to V<sub>PP</sub> may have an adverse affect on reliability.

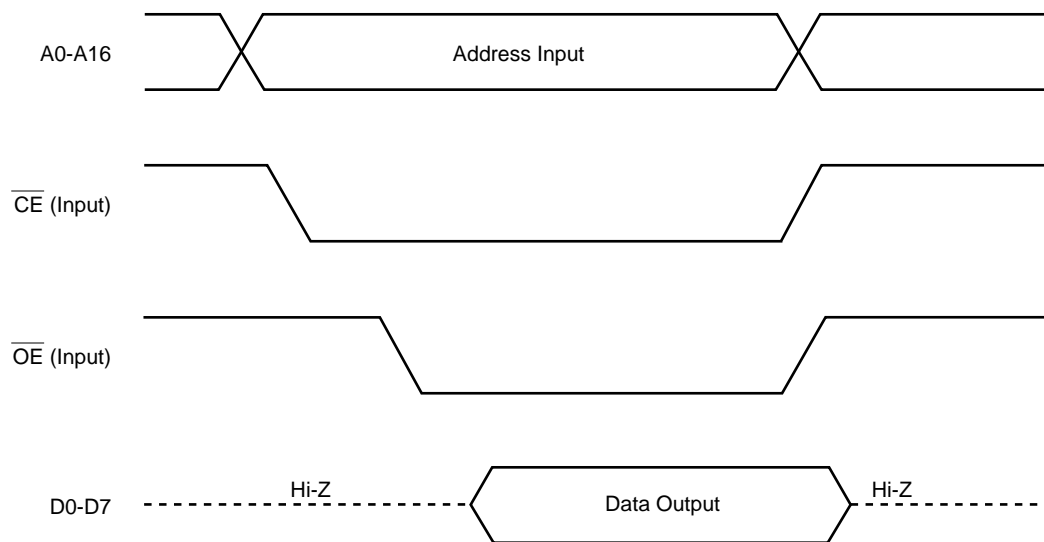
### 24.3 PROM READING PROCEDURE

PROM contents can be read onto the external data bus (D0 to D7) using the following procedure.

- (1) Fix the  $\overline{\text{RESET}}$  pin low, and supply +5 V to the  $V_{PP}$  pin. Unused pins are handled as shown in **1.3.2 PROM Programming Mode** in **1.3 PIN CONFIGURATION (Top View)**.
- (2) Supply +5 V to the  $V_{DD}$  and  $V_{PP}$  pins.
- (3) Input address of data to be read to pins A0 to A16.
- (4) Read mode.
- (5) Output data to pins D0 to D7.

The timing for steps (2) to (5) above is shown in Figure 24-5.

**Figure 24-5 PROM Read Timing**



#### 24.4 SCREENING OF ONE-TIME PROM PRODUCT

Because of its construction, the one-time PROM product ( $\mu$ PD78P4026GC-3B9) cannot be fully tested by NEC before shipment. After the necessary data has been written, it is recommended that screening be carried out by performing PROM verification after high-temperature storage under the following conditions.

Storage Temperature	Storage Time
125 °C	24 hours

## 24.5 ERASING PROCEDURE ( $\mu$ PD78P4026KK-T ONLY)

With the  $\mu$ PD78P4026KK-T, the data contents written to the program memory can be erased (to FFH) and new data can be written to the program memory.

To erase the data contents, the erasure window of the program memory must be exposed to a light with a wavelength of approximately 400 nm or shorter through erasure window. Usually, an ultraviolet ray of 254 nm is used. The time required to completely erase the data contents is as follows:

- Ultraviolet ray intensity x erase time: 28.8 W·s/cm<sup>2</sup> MIN.
- Erase time: about 40 minutes (with an ultraviolet lamp of 12,000  $\mu$ W/cm<sup>2</sup> used. The longer time may be required if the performance of the ultraviolet lamp degrades or if the erasure window is dirty.)

To erase the data, place the ultraviolet lamp within 2.5 cm from the erasure window. If a filter is attached to the ultraviolet lamp, remove the filter.

## 24.6 STICKER ON ERASURE WINDOW ( $\mu$ PD78P4026KK-T ONLY)

To prevent the contents of the EPROM from being erased by accident by light other than that from the erasure lamp, or the internal circuits other than EPROM from malfunctioning due to light, attach a protective sticker to the erasure window when the contents of the EPROM are not being erased.

## 24.7 QUALITY

The  $\mu$ PD78P4026KK-T is not intended to be used in your mass-produced systems. Only use the  $\mu$ PD78P4026KK-T for experiments and evaluation of the functions of a trial system.

## 24.8 CAUTIONS

- (1) Ensure that  $V_{DD}$  is applied before  $V_{PP}$ , and cut after  $V_{PP}$ .
- (2) Ensure that  $V_{PP}$  does not become +13.5 V or over including overshoot.
- (3) Removing the device while +12.5 V is being applied to  $V_{PP}$  may have an adverse affect on reliability.

[MEMO]



## CHAPTER 25 INSTRUCTION OPERATIONS

### 25.1 LEGEND

#### (1) Explanation of Operand identifiers (1/2)

Identifier	Explanation
r, r' <sup>*1</sup>	X(R0), A(R1), C(R2), B(R3), R4, R5, R6, R7, R8, R9, R10, R11, E(R12), D(R13), L(R14), H(R15)
r1 <sup>*1</sup>	X(R0), A(R1), C(R2), B(R3), R4, R5, R6, R7
r2	R8, R9, R10, R11, E(R12), D(R13), L(R14), H(R15)
r3	V, U, T, W
rp, rp' <sup>*2</sup>	AX(RP0), BC(RP1), RP2, RP3, VP(RP4), UP(RP5), DE(RP6), HL(RP7)
rp1 <sup>*2</sup>	AX(RP0), BC(RP1), RP2, RP3
rp2	VP(RP4), UP(RP5), DE(RP6), HL(RP7)
rg, rg'	VVP(RG4), UUP(RG5), TDE(RG6), WHL(RG7)
sfr	Special function register symbol (see <b>Special Function Register Application Table</b> )
sfrp	Special function register symbol (register for which 16-bit operation is possible: see <b>Special Function Register Application Table</b> )
post <sup>*2</sup>	AX(RP0), BC(RP1), RP2, RP3, VP(RP4), UP(RP5)/PSW, DE(RP6), HL(RP7) Multiple definitions are permissible. However, UP is only used with PUSH/POP instructions, and PSW with PUSHU/POPU instructions.
mem	[TDE], [WHL], [TDE+], [WHL+], [TDE-], [WHL-], [VVP], [UUP]: Register indirect addressing [TDE+byte], [WHL+byte], [SP+byte], [UUP+byte], [VVP+byte]: Based addressing imm24 [A], imm24 [B], imm24 [DE], imm24 [HL]: Indexed addressing [TDE+A], [TDE+B], [TDE+C], [WHL+A], [WHL+B], [WHL+C], [VVP+DE], [VVP+HL]: Based indexed addressing
mem1	All mem except [WHL+] and [WHL-]
mem2	[TDE], [WHL]
mem3	[AX], [BC], [RP2], [RP3], [VVP], [UUP], [TDE], [WHL]

- \* 1. Setting the RSS bit to 1 enables R4 to R7 to be used as X, A, C and B, but this function should only be used when using a 78K/III series program.
- 2. Setting the RSS bit to 1 enables RP2 and RP3 to be used as AX and BC, but this function should only be used when using a 78K/III series program.

**(1) Explanation of Operand identifiers (2/2)**

Identifier	Explanation
* saddr, saddr' saddr1 saddr2 saddrp saddrp1 saddrp2 saddrg saddrg1 saddrg2	FD20H to FF1FH immediate data or label FE00H to FEFFH immediate data or label FD20H to FDFFH, FF00H to FF1FH immediate data or label FD20H to FF1EH immediate data or label (16-bit operation) FE00H to FEFFH immediate data or label (16-bit operation) FD20H to FDFFH, FF00H to FF1EH immediate data or label (16-bit operation) FD20H to FEFDH immediate data or label (24-bit operation) FE00H to FEFDH immediate data or label (24-bit operation) FD20H to FDFFH immediate data or label (24-bit operation)
addr24 addr20 addr16 addr11 addr8 addr5	0H to FFFFFFFH immediate data or label 0H to FFFFFFFH immediate data or label 0H to FFFFFH immediate data or label 800H to FFFH immediate data or label 0FE00H to 0FEFFH* immediate data or label 40H to 7EH immediate data or label
imm24 word byte bit n locaddr	24-bit immediate data or label 16-bit immediate data or label 8-bit immediate data or label 3-bit immediate data or label 3-bit immediate data 00H or 0FH

- \* The addresses shown here apply when 00H is specified by the LOCATION instruction.  
When 0FH is specified by the LOCATION instruction, F000H should be added to the address values shown.

**(2) Operand column symbols**

Symbol	Explanation
+	Auto-increment
–	Auto-decrement
#	Immediate data
!	16-bit absolute address
!!	24-bit/20-bit absolute address
\$	8-bit relative address
\$\$	16-bit relative address
/	Bit inversion
[ ]	Indirect addressing
[%]	24-bit indirect addressing

**(3) Flag column symbols**

Symbol	Explanation
(Blank)	No change
0	Cleared to 0
1	Set to 1
×	Set or cleared depending on result
P	P/V flag operates as parity flag
V	P/V flag operates as overflow flag
R	Previously saved value is restored

**(4) Operation column symbols**

Symbol	Explanation
jdisp8	Signed two's complement data (8 bits) indicating relative address distance between start address of next instruction and branch address
jdisp16	Signed two's complement data (16 bits) indicating relative address distance between start address of next instruction and branch address
PC <sub>HW</sub>	PC bits 16 to 19
PC <sub>LW</sub>	PC bits 0 to 15

**(5) Number of bytes of instruction that includes mem in operands**

mem Mode	Register Indirect Addressing		Based Addressing	Indexed Addressing	Based Indexed Addressing
Number of bytes	1	2*	3	5	2

\* One-byte instruction only when [TDE], [WHL], [TDE+], [TDE-], [WHL+] or [WHL-] is written as mem in an MOV instruction.

**(6) Number of bytes of instruction that includes saddr, saddrp, r or rp in operands**

For some instructions that include saddr, saddrp, r or rp in their operands, two “Bytes” entries are given, separated by a slash (“/”). The entry that applies is shown in the table below.

Identifier	Left-Hand “Bytes” Figure	Right-Hand “Bytes” Figure
saddr	saddr2	saddr1
saddrp	saddrp2	saddrp1
r	r1	r2
rp	rp1	rp2

**(7) Code of instructions that include mem in operands and string instructions**

Operands TDE, WHL, VVP and UUP (24-bit registers) can also be written as DE, HL, VP and UP respectively. However, they are still treated as TDE, WHL, VVP and UUP (24-bit registers) when written as DE, HL, VP and UP.

## 25.2 LIST OF OPERATIONS

## (1) 8-bit data transfer instruction: MOV

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MOV	r, #byte	2/3	$r \leftarrow \text{byte}$					
	saddr, #byte	3/4	$(\text{saddr}) \leftarrow \text{byte}$					
	sfr, #byte	3	$\text{sfr} \leftarrow \text{byte}$					
	!addr16, #byte	5	$(\text{saddr16}) \leftarrow \text{byte}$					
	!!addr24, #byte	6	$(\text{addr24}) \leftarrow \text{byte}$					
	r, r'	2/3	$r \leftarrow r'$					
	A, r	1/2	$A \leftarrow r$					
	A, saddr2	2	$A \leftarrow (\text{saddr2})$					
	r, saddr	3	$r \leftarrow (\text{saddr})$					
	saddr2, A	2	$(\text{saddr2}) \leftarrow A$					
	saddr, r	3	$(\text{saddr}) \leftarrow r$					
	A, sfr	2	$A \leftarrow \text{sfr}$					
	r, sfr	3	$r \leftarrow \text{sfr}$					
	sfr, A	2	$\text{sfr} \leftarrow A$					
	sfr, r	3	$\text{sfr} \leftarrow r$					
	saddr, saddr'	4	$(\text{saddr}) \leftarrow (\text{saddr}')$					
	r, !addr16	4	$r \leftarrow (\text{addr16})$					
	!addr16, r	4	$(\text{addr16}) \leftarrow r$					
	r, !!addr24	5	$r \leftarrow (\text{addr24})$					
	!!addr24, r	5	$(\text{addr24}) \leftarrow r$					
	A, [saddrp]	2/3	$A \leftarrow ((\text{saddrp}))$					
	A, [%saddrg]	3/4	$A \leftarrow ((\text{saddrg}))$					
	A, mem	1-5	$A \leftarrow (\text{mem})$					
	[saddrp], A	2/3	$((\text{saddrp})) \leftarrow A$					
	[%saddrg], A	3/4	$((\text{saddrg})) \leftarrow A$					
	mem, A	1-5	$(\text{mem}) \leftarrow A$					
	PSWL, #byte	3	$\text{PSWL} \leftarrow \text{byte}$	x	x	x	x	x
	PSWH, #byte	3	$\text{PSWH} \leftarrow \text{byte}$					
	PSWL, A	2	$\text{PSWL} \leftarrow A$	x	x	x	x	x
	PSWH, A	2	$\text{PSWH} \leftarrow A$					
	A, PSWL	2	$A \leftarrow \text{PSWL}$					
	A, PSWH	2	$A \leftarrow \text{PSWH}$					
	r3, #byte	3	$r3 \leftarrow \text{byte}$					
	A, r3	2	$A \leftarrow r3$					
	r3, A	2	$r3 \leftarrow A$					

**(2) 16-bit data transfer instruction: MOVW**

Mnemonic	Operands	Bytes	Operation	Flags					
				S	Z	AC	P/V	CY	
MOVW	rp, #word	3	rp ← word						
	saddrp, #word	4/5	(saddrp) ← word						
	sfrp, #word	4	sfrp ← word						
	!addr16, #word	6	(addr16) ← word						
	!!addr24, #word	7	(addr24) ← word						
	rp, rp'	2	rp ← rp'						
	AX, saddrp2	2	AX ← (saddrp2)						
	rp, saddrp	3	rp ← (saddrp)						
	saddrp2, AX	2	(saddrp2) ← AX						
	saddrp, rp	3	(saddrp) ← rp						
	AX, sfrp	2	AX ← sfrp						
	rp, sfrp	3	rp ← sfrp						
	sfrp, AX	2	sfrp ← AX						
	sfrp, rp	3	sfrp ← rp						
	saddrp, saddrp'	4	(saddrp) ← (saddrp')						
	rp, !addr16	4	rp ← (addr16)						
	!addr16, rp	4	(addr16) ← rp						
	rp, !!addr24	5	rp ← (addr24)						
	!!addr24, rp	5	(addr24) ← rp						
	AX, [saddrp]	3/4	AX ← ((saddrp))						
	AX, [%saddrg]	3/4	AX ← ((saddrg))						
	AX, mem	2-5	AX ← (mem)						
	[saddrp], AX	3/4	((saddrp)) ← AX						
	[%saddrg], AX	3/4	((saddrg)) ← AX						
	mem, AX	2-5	(mem) ← AX						

**(3) 24-bit data transfer instruction: MOVG**

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MOVG	rg, #imm24	5	$rg \leftarrow \text{imm24}$					
	rg, rg'	2	$rg \leftarrow rg'$					
	rg, !addr24	5	$rg \leftarrow (\text{addr24})$					
	!addr24, rg	5	$(\text{addr24}) \leftarrow rg$					
	rg, saddrg	3	$rg \leftarrow (\text{saddrg})$					
	saddrg, rg	3	$(\text{saddrg}) \leftarrow rg$					
	WHL, [%saddrg]	3/4	$WHL \leftarrow ((\text{saddrg}))$					
	[%saddrg], WHL	3/4	$((\text{saddrg})) \leftarrow WHL$					
	WHL, mem1	2-5	$WHL \leftarrow (\text{mem1})$					
	mem1, WHL	2-5	$(\text{mem1}) \leftarrow WHL$					

**(4) 8-bit data exchange instruction: XCH**

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
XCH	r, r'	2/3	$r \leftrightarrow r'$					
	A, r	1/2	$A \leftrightarrow r$					
	A, saddr2	2	$A \leftrightarrow (\text{saddr2})$					
	r, saddr	3	$r \leftrightarrow (\text{saddr})$					
	r, sfr	3	$r \leftrightarrow \text{sfr}$					
	saddr, saddr'	4	$(\text{saddr}) \leftrightarrow (\text{saddr}')$					
	r, !addr16	4	$r \leftrightarrow (\text{addr16})$					
	r, !addr24	5	$r \leftrightarrow (\text{addr24})$					
	A, [saddrp]	2/3	$A \leftrightarrow ((\text{saddrp}))$					
	A, [%saddrg]	3/4	$A \leftrightarrow ((\text{saddrg}))$					
	A, mem	2-5	$A \leftrightarrow (\text{mem})$					

## (5) 16-bit data exchange instruction: XCHW

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
XCHW	rp, rp'	2	rp $\leftrightarrow$ rp'					
	AX, saddrp2	2	AX $\leftrightarrow$ (saddrp2)					
	rp, saddrp	3	rp $\leftrightarrow$ (saddrp)					
	rp, sfrp	3	rp $\leftrightarrow$ sfrp					
	AX, [saddrp]	3/4	AX $\leftrightarrow$ ((saddrp))					
	AX, [%saddrg]	3/4	AX $\leftrightarrow$ ((saddrg))					
	AX, !addr16	4	AX $\leftrightarrow$ (addr16)					
	AX, !!addr24	5	AX $\leftrightarrow$ (addr24)					
	saddrp, saddrp'	4	(saddrp) $\leftrightarrow$ (saddrp')					
	AX, mem	2-5	AX $\leftrightarrow$ (mem)					

## (6) 8-bit operation instructions: ADD, ADDC, SUB, SUBC, CMP, AND, OR, XOR

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ADD	A, #byte	2	A, CY $\leftarrow$ A + byte	×	×	×	V	×
	r, #byte	3	r, CY $\leftarrow$ r + byte	×	×	×	V	×
	saddr, #byte	3/4	(saddr), CY $\leftarrow$ (saddr) + byte	×	×	×	V	×
	sfr, #byte	4	sfr, CY $\leftarrow$ sfr + byte	×	×	×	V	×
	r, r'	2/3	r, CY $\leftarrow$ r + r'	×	×	×	V	×
	A, saddr2	2	A, CY $\leftarrow$ A + (saddr2)	×	×	×	V	×
	r, saddr	3	r, CY $\leftarrow$ r + (saddr)	×	×	×	V	×
	saddr, r	3	(saddr), CY $\leftarrow$ (saddr) + r	×	×	×	V	×
	r, sfr	3	r, CY $\leftarrow$ r + sfr	×	×	×	V	×
	sfr, r	3	sfr, CY $\leftarrow$ sfr + r	×	×	×	V	×
	saddr, saddr'	4	(saddr), CY $\leftarrow$ (saddr) + (saddr')	×	×	×	V	×
	A, [saddrp]	3/4	A, CY $\leftarrow$ A + ((saddrp))	×	×	×	V	×
	A, [%saddrg]	3/4	A, CY $\leftarrow$ A + ((saddrg))	×	×	×	V	×
	[saddrp], A	3/4	((saddrp)), CY $\leftarrow$ ((saddrp)) + A	×	×	×	V	×
	[%saddrg], A	3/4	((saddrg)), CY $\leftarrow$ ((saddrg)) + A	×	×	×	V	×
	A, !addr16	4	A, CY $\leftarrow$ A + (addr16)	×	×	×	V	×
	A, !!addr24	5	A, CY $\leftarrow$ A + (addr24)	×	×	×	V	×
	!addr16, A	4	(addr16), CY $\leftarrow$ (addr16) + A	×	×	×	V	×
	!!addr24, A	5	(addr24), CY $\leftarrow$ (addr24) + A	×	×	×	V	×
	A, mem	2-5	A, CY $\leftarrow$ A + (mem)	×	×	×	V	×
	mem, A	2-5	(mem), CY $\leftarrow$ (mem) + A	×	×	×	V	×



Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ADDC	A, #byte	2	$A, CY \leftarrow A + \text{byte} + CY$	×	×	×	V	×
	r, #byte	3	$r, CY \leftarrow r + \text{byte} + CY$	×	×	×	V	×
	saddr, #byte	3/4	$(saddr), CY \leftarrow (saddr) + \text{byte} + CY$	×	×	×	V	×
	sfr, #byte	4	$sfr, CY \leftarrow sfr + \text{byte} + CY$	×	×	×	V	×
	r, r'	2/3	$r, CY \leftarrow r + r' + CY$	×	×	×	V	×
	A, saddr2	2	$A, CY \leftarrow A + (saddr2) + CY$	×	×	×	V	×
	r, saddr	3	$r, CY \leftarrow r + (saddr) + CY$	×	×	×	V	×
	saddr, r	3	$(saddr), CY \leftarrow (saddr) + r + CY$	×	×	×	V	×
	r, sfr	3	$r, CY \leftarrow r + sfr + CY$	×	×	×	V	×
	sfr, r	3	$sfr, CY \leftarrow sfr + r + CY$	×	×	×	V	×
	saddr, saddr'	4	$(saddr), CY \leftarrow (saddr) + (saddr') + CY$	×	×	×	V	×
	A, [saddrp]	3/4	$A, CY \leftarrow A + ((saddrp)) + CY$	×	×	×	V	×
	A, [%saddrg]	3/4	$A, CY \leftarrow A + ((saddrg)) + CY$	×	×	×	V	×
	[saddrp], A	3/4	$((saddrp)), CY \leftarrow ((saddrp)) + A + CY$	×	×	×	V	×
	[%saddrg], A	3/4	$((saddrg)), CY \leftarrow ((saddrg)) + A + CY$	×	×	×	V	×
	A, !addr16	4	$A, CY \leftarrow A + (\text{addr16}) + CY$	×	×	×	V	×
	A, !!addr24	5	$A, CY \leftarrow A + (\text{addr24}) + CY$	×	×	×	V	×
	!addr16, A	4	$(\text{addr16}), CY \leftarrow (\text{addr16}) + A + CY$	×	×	×	V	×
	!!addr24, A	5	$(\text{addr24}), CY \leftarrow (\text{addr24}) + A + CY$	×	×	×	V	×
	A, mem	2-5	$A, CY \leftarrow A + (\text{mem}) + CY$	×	×	×	V	×
	mem, A	2-5	$(\text{mem}), CY \leftarrow (\text{mem}) + A + CY$	×	×	×	V	×

Mnemonic	Operands	Bytes	Operation	Flags					
				S	Z	AC	P/V	CY	
SUB	A, #byte	2	A, CY $\leftarrow$ A – byte	×	×	×	V	×	
	r, #byte	3	r, CY $\leftarrow$ r – byte	×	×	×	V	×	
	saddr, #byte	3/4	(saddr), CY $\leftarrow$ (saddr) – byte	×	×	×	V	×	
	sfr, #byte	4	sfr, CY $\leftarrow$ sfr – byte	×	×	×	V	×	
	r, r'	2/3	r, CY $\leftarrow$ r – r'	×	×	×	V	×	
	A, saddr2	2	A, CY $\leftarrow$ A – (saddr2)	×	×	×	V	×	
	r, saddr	3	r, CY $\leftarrow$ r – (saddr)	×	×	×	V	×	
	saddr, r	3	(saddr), CY $\leftarrow$ (saddr) – r	×	×	×	V	×	
	r, sfr	3	r, CY $\leftarrow$ r – sfr	×	×	×	V	×	
	sfr, r	3	sfr, CY $\leftarrow$ sfr – r	×	×	×	V	×	
	saddr, saddr'	4	(saddr), CY $\leftarrow$ (saddr) – (saddr')	×	×	×	V	×	
	A, [saddrp]	3/4	A, CY $\leftarrow$ A – ((saddrp))	×	×	×	V	×	
	A, [%saddrg]	3/4	A, CY $\leftarrow$ A – ((saddrg))	×	×	×	V	×	
	[saddrp], A	3/4	((saddrp)), CY $\leftarrow$ ((saddrp)) – A	×	×	×	V	×	
	[%saddrg], A	3/4	((saddrg)), CY $\leftarrow$ ((saddrg)) – A	×	×	×	V	×	
	A, !addr16	4	A, CY $\leftarrow$ A – (addr16)	×	×	×	V	×	
	A, !!addr24	5	A, CY $\leftarrow$ A – (addr24)	×	×	×	V	×	
	!addr16, A	4	(addr16), CY $\leftarrow$ (addr16) – A	×	×	×	V	×	
	!!addr24, A	5	(addr24), CY $\leftarrow$ (addr24) – A	×	×	×	V	×	
	A, mem	2-5	A, CY $\leftarrow$ A – (mem)	×	×	×	V	×	
	mem, A	2-5	(mem), CY $\leftarrow$ (mem) – A	×	×	×	V	×	

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
SUBC	A, #byte	2	A, CY $\leftarrow$ A – byte – CY	×	×	×	V	×
	r, #byte	3	r, CY $\leftarrow$ r – byte – CY	×	×	×	V	×
	saddr, #byte	3/4	(saddr), CY $\leftarrow$ (saddr) – byte – CY	×	×	×	V	×
	sfr, #byte	4	sfr, CY $\leftarrow$ sfr – byte – CY	×	×	×	V	×
	r, r'	2/3	r, CY $\leftarrow$ r – r' – CY	×	×	×	V	×
	A, saddr2	2	A, CY $\leftarrow$ A – (saddr2) – CY	×	×	×	V	×
	r, saddr	3	r, CY $\leftarrow$ r – (saddr) – CY	×	×	×	V	×
	saddr, r	3	(saddr), CY $\leftarrow$ (saddr) – r – CY	×	×	×	V	×
	r, sfr	3	r, CY $\leftarrow$ r – sfr – CY	×	×	×	V	×
	sfr, r	3	sfr, CY $\leftarrow$ sfr – r – CY	×	×	×	V	×
	saddr, saddr'	4	(saddr), CY $\leftarrow$ (saddr) – (saddr') – CY	×	×	×	V	×
	A, [saddrp]	3/4	A, CY $\leftarrow$ A – ((saddrp)) – CY	×	×	×	V	×
	A, [%saddrg]	3/4	A, CY $\leftarrow$ A – ((saddrg)) – CY	×	×	×	V	×
	[saddrp], A	3/4	((saddrp)), CY $\leftarrow$ ((saddrp)) – A – CY	×	×	×	V	×
	[%saddrg], A	3/4	((saddrg)), CY $\leftarrow$ ((saddrg)) – A – CY	×	×	×	V	×
	A, !addr16	4	A, CY $\leftarrow$ A – (addr16) – CY	×	×	×	V	×
	A, !!addr24	5	A, CY $\leftarrow$ A – (addr24) – CY	×	×	×	V	×
	!addr16, A	4	(addr16), CY $\leftarrow$ (addr16) – A – CY	×	×	×	V	×
	!!addr24, A	5	(addr24), CY $\leftarrow$ (addr24) – A – CY	×	×	×	V	×
	A, mem	2-5	A, CY $\leftarrow$ A – (mem) – CY	×	×	×	V	×
	mem, A	2-5	(mem), CY $\leftarrow$ (mem) – A – CY	×	×	×	V	×

Mnemonic	Operands	Bytes	Operation	Flags					
				S	Z	AC	P/V	CY	
CMP	A, #byte	2	A – byte	×	×	×	V	×	
	r, #byte	3	r – byte	×	×	×	V	×	
	saddr, #byte	3/4	(saddr) – byte	×	×	×	V	×	
	sfr, #byte	4	sfr – byte	×	×	×	V	×	
	r, r'	2/3	r – r'	×	×	×	V	×	
	A, saddr2	2	A – (saddr2)	×	×	×	V	×	
	r, saddr	3	r – (saddr)	×	×	×	V	×	
	saddr, r	3	(saddr) – r	×	×	×	V	×	
	r, sfr	3	r – sfr	×	×	×	V	×	
	sfr, r	3	sfr – r	×	×	×	V	×	
	saddr, saddr'	4	(saddr) – (saddr')	×	×	×	V	×	
	A, [saddrp]	3/4	A – ((saddrp))	×	×	×	V	×	
	A, [%saddrg]	3/4	A – ((saddrg))	×	×	×	V	×	
	[saddrp], A	3/4	((saddrp)) – A	×	×	×	V	×	
	[%saddrg], A	3/4	((saddrg)) – A	×	×	×	V	×	
	A, !addr16	4	A – (addr16)	×	×	×	V	×	
	A, !!addr24	5	A – (addr24)	×	×	×	V	×	
	!addr16, A	4	(addr16) – A	×	×	×	V	×	
	!!addr24, A	5	(addr24) – A	×	×	×	V	×	
	A, mem	2-5	A – (mem)	×	×	×	V	×	
	mem, A	2-5	(mem) – A	×	×	×	V	×	

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
AND	A, #byte	2	$A \leftarrow A \wedge \text{byte}$	x	x		P	
	r, #byte	3	$r \leftarrow r \wedge \text{byte}$	x	x		P	
	saddr, #byte	3/4	$(\text{saddr}) \leftarrow (\text{saddr}) \wedge \text{byte}$	x	x		P	
	sfr, #byte	4	$\text{sfr} \leftarrow \text{sfr} \wedge \text{byte}$	x	x		P	
	r, r'	2/3	$r \leftarrow r \wedge r'$	x	x		P	
	A, saddr2	2	$A \leftarrow A \wedge (\text{saddr2})$	x	x		P	
	r, saddr	3	$r \leftarrow r \wedge (\text{saddr})$	x	x		P	
	saddr, r	3	$(\text{saddr}) \leftarrow (\text{saddr}) \wedge r$	x	x		P	
	r, sfr	3	$r \leftarrow r \wedge \text{sfr}$	x	x		P	
	sfr, r	3	$\text{sfr} \leftarrow \text{sfr} \wedge r$	x	x		P	
	saddr, saddr'	4	$(\text{saddr}) \leftarrow (\text{saddr}) \wedge (\text{saddr}')$	x	x		P	
	A, [saddrp]	3/4	$A \leftarrow A \wedge ((\text{saddrp}))$	x	x		P	
	A, [%saddrg]	3/4	$A \leftarrow A \wedge ((\text{saddrg}))$	x	x		P	
	[saddrp], A	3/4	$((\text{saddrp})) \leftarrow ((\text{saddrp})) \wedge A$	x	x		P	
	[%saddrg], A	3/4	$((\text{saddrg})) \leftarrow ((\text{saddrg})) \wedge A$	x	x		P	
	A, !addr16	4	$A \leftarrow A \wedge (\text{addr16})$	x	x		P	
	A, !!addr24	5	$A \leftarrow A \wedge (\text{addr24})$	x	x		P	
	!addr16, A	4	$(\text{addr16}) \leftarrow (\text{addr16}) \wedge A$	x	x		P	
	!!addr24, A	5	$(\text{addr24}) \leftarrow (\text{addr24}) \wedge A$	x	x		P	
	A, mem	2-5	$A \leftarrow A \wedge (\text{mem})$	x	x		P	
	mem, A	2-5	$(\text{mem}) \leftarrow (\text{mem}) \wedge A$	x	x		P	

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
OR	A, #byte	2	$A \leftarrow A \vee \text{byte}$	x	x		P	
	r, #byte	3	$r \leftarrow r \vee \text{byte}$	x	x		P	
	saddr, #byte	3/4	$(\text{saddr}) \leftarrow (\text{saddr}) \vee \text{byte}$	x	x		P	
	sfr, #byte	4	$\text{sfr} \leftarrow \text{sfr} \vee \text{byte}$	x	x		P	
	r, r'	2/3	$r \leftarrow r \vee r'$	x	x		P	
	A, saddr2	2	$A \leftarrow A \vee (\text{saddr2})$	x	x		P	
	r, saddr	3	$r \leftarrow r \vee (\text{saddr})$	x	x		P	
	saddr, r	3	$(\text{saddr}) \leftarrow (\text{saddr}) \vee r$	x	x		P	
	r, sfr	3	$r \leftarrow r \vee \text{sfr}$	x	x		P	
	sfr, r	3	$\text{sfr} \leftarrow \text{sfr} \vee r$	x	x		P	
	saddr, saddr'	4	$(\text{saddr}) \leftarrow (\text{saddr}) \vee (\text{saddr}')$	x	x		P	
	A, [saddrp]	3/4	$A \leftarrow A \vee ((\text{saddrp}))$	x	x		P	
	A, [%saddrg]	3/4	$A \leftarrow A \vee ((\text{saddrg}))$	x	x		P	
	[saddrp], A	3/4	$((\text{saddrp})) \leftarrow ((\text{saddrp})) \vee A$	x	x		P	
	[%saddrg], A	3/4	$((\text{saddrg})) \leftarrow ((\text{saddrg})) \vee A$	x	x		P	
	A, !addr16	4	$A \leftarrow A \vee (\text{addr16})$	x	x		P	
	A, !!addr24	5	$A \leftarrow A \vee (\text{addr24})$	x	x		P	
	!addr16, A	4	$(\text{addr16}) \leftarrow (\text{addr16}) \vee A$	x	x		P	
	!!addr24, A	5	$(\text{addr24}) \leftarrow (\text{addr24}) \vee A$	x	x		P	
	A, mem	2-5	$A \leftarrow A \vee (\text{mem})$	x	x		P	
	mem, A	2-5	$(\text{mem}) \leftarrow (\text{mem}) \vee A$	x	x		P	

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
XOR	A, #byte	2	$A \leftarrow A \nabla \text{byte}$	×	×		P	
	r, #byte	3	$r \leftarrow r \nabla \text{byte}$	×	×		P	
	saddr, #byte	3/4	$(\text{saddr}) \leftarrow (\text{saddr}) \nabla \text{byte}$	×	×		P	
	sfr, #byte	4	$\text{sfr} \leftarrow \text{sfr} \nabla \text{byte}$	×	×		P	
	r, r'	2/3	$r \leftarrow r \nabla r'$	×	×		P	
	A, saddr2	2	$A \leftarrow A \nabla (\text{saddr2})$	×	×		P	
	r, saddr	3	$r \leftarrow r \nabla (\text{saddr})$	×	×		P	
	saddr, r	3	$(\text{saddr}) \leftarrow (\text{saddr}) \nabla r$	×	×		P	
	r, sfr	3	$r \leftarrow r \nabla \text{sfr}$	×	×		P	
	sfr, r	3	$\text{sfr} \leftarrow \text{sfr} \nabla r$	×	×		P	
	saddr, saddr'	4	$(\text{saddr}) \leftarrow (\text{saddr}) \nabla (\text{saddr}')$	×	×		P	
	A, [saddrp]	3/4	$A \leftarrow A \nabla ((\text{saddrp}))$	×	×		P	
	A, [%saddrg]	3/4	$A \leftarrow A \nabla ((\text{saddrg}))$	×	×		P	
	[saddrp], A	3/4	$((\text{saddrp})) \leftarrow ((\text{saddrp})) \nabla A$	×	×		P	
	[%saddrg], A	3/4	$((\text{saddrg})) \leftarrow ((\text{saddrg})) \nabla A$	×	×		P	
	A, !addr16	4	$A \leftarrow A \nabla (\text{addr16})$	×	×		P	
	A, !!addr24	5	$A \leftarrow A \nabla (\text{addr24})$	×	×		P	
	!addr16, A	4	$(\text{addr16}) \leftarrow (\text{addr16}) \nabla A$	×	×		P	
	!!addr24, A	5	$(\text{addr24}) \leftarrow (\text{addr24}) \nabla A$	×	×		P	
	A, mem	2-5	$A \leftarrow A \nabla (\text{mem})$	×	×		P	
	mem, A	2-5	$(\text{mem}) \leftarrow (\text{mem}) \nabla A$	×	×		P	

**(7) 16-bit operation instructions: ADDW, SUBW, CMPW**

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ADDW	AX, #word	3	AX, CY $\leftarrow$ AX + word	x	x	x	V	x
	rp, #word	4	rp, CY $\leftarrow$ rp + word	x	x	x	V	x
	rp, rp'	2	rp, CY $\leftarrow$ rp + rp'	x	x	x	V	x
	AX, saddrp2	2	AX, CY $\leftarrow$ AX + (saddrp2)	x	x	x	V	x
	rp, saddrp	3	rp, CY $\leftarrow$ rp + (saddrp)	x	x	x	V	x
	saddrp, rp	3	(saddrp), CY $\leftarrow$ (saddrp) + rp	x	x	x	V	x
	rp, sfrp	3	rp, CY $\leftarrow$ rp + sfrp	x	x	x	V	x
	sfrp, rp	3	sfrp, CY $\leftarrow$ sfrp + rp	x	x	x	V	x
	saddrp, #word	4/5	(saddrp), CY $\leftarrow$ (saddrp) + word	x	x	x	V	x
	sfrp, #word	5	sfrp, CY $\leftarrow$ sfrp + word	x	x	x	V	x
	saddrp, saddrp'	4	(saddrp), CY $\leftarrow$ (saddrp) + (saddrp')	x	x	x	V	x
SUBW	AX, #word	3	AX, CY $\leftarrow$ AX – word	x	x	x	V	x
	rp, #word	4	rp, CY $\leftarrow$ rp – word	x	x	x	V	x
	rp, rp'	2	rp, CY $\leftarrow$ rp – rp'	x	x	x	V	x
	AX, saddrp2	2	AX, CY $\leftarrow$ AX – (saddrp2)	x	x	x	V	x
	rp, saddrp	3	rp, CY $\leftarrow$ rp – (saddrp)	x	x	x	V	x
	saddrp, rp	3	(saddrp), CY $\leftarrow$ (saddrp) – rp	x	x	x	V	x
	rp, sfrp	3	rp, CY $\leftarrow$ rp – sfrp	x	x	x	V	x
	sfrp, rp	3	sfrp, CY $\leftarrow$ sfrp – rp	x	x	x	V	x
	saddrp, #word	4/5	(saddrp), CY $\leftarrow$ (saddrp) – word	x	x	x	V	x
	sfrp, #word	5	sfrp, CY $\leftarrow$ sfrp – word	x	x	x	V	x
	saddrp, saddrp'	4	(saddrp), CY $\leftarrow$ (saddrp) – (saddrp')	x	x	x	V	x
CMPW	AX, #word	3	AX – word	x	x	x	V	x
	rp, #word	4	rp – word	x	x	x	V	x
	rp, rp'	2	rp – rp'	x	x	x	V	x
	AX, saddrp2	2	AX – (saddrp2)	x	x	x	V	x
	rp, saddrp	3	rp – (saddrp)	x	x	x	V	x
	saddrp, rp	3	(saddrp) – rp	x	x	x	V	x
	rp, sfrp	3	rp – sfrp	x	x	x	V	x
	sfrp, rp	3	sfrp – rp	x	x	x	V	x
	saddrp, #word	4/5	(saddrp) – word	x	x	x	V	x
	sfrp, #word	5	sfrp – word	x	x	x	V	x
	saddrp, saddrp'	4	(saddrp) – (saddrp')	x	x	x	V	x



**(8) 24-bit operation instructions: ADDG, SUBG**

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ADDG	rg, rg'	2	$rg, CY \leftarrow rg + rg'$	x	x	x	V	x
	rg, # imm24	5	$rg, CY \leftarrow rg + \# \text{ imm24}$	x	x	x	V	x
	WHL, saddrg	3	$WHL, CY \leftarrow WHL + (\text{saddrg})$	x	x	x	V	x
SUBG	rg, rg'	2	$rg, CY \leftarrow rg - rg'$	x	x	x	V	x
	rg, # imm24	5	$rg, CY \leftarrow rg - \text{imm24}$	x	x	x	V	x
	WHL, saddrg	3	$WHL, CY \leftarrow WHL - (\text{saddrg})$	x	x	x	V	x

**(9) Multiplication instructions: MULU, MULUW, MULW, DIVUW, DIVUX**

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MULU	r	2/3	$AX \leftarrow A \times r$					
MULUW	rp	2	$AX \text{ (high-order)}, rp \text{ (low-order)} \leftarrow AX \times rp$					
MULW	rp	2	$AX \text{ (high-order)}, rp \text{ (low-order)} \leftarrow AX \times rp$					
DIVUW	r	2/3	$AX \text{ (quotient)}, r \text{ (remainder)} \leftarrow AX \div r^{*1}$					
DIVUX	rp	2	$AXDE \text{ (quotient)}, rp \text{ (remainder)} \leftarrow AXDE \div rp^{*2}$					

\* 1. When  $r = 0$ ,  $r \leftarrow X$ ,  $AX \leftarrow \text{FFFFH}$

2. When  $rp = 0$ ,  $pr \leftarrow \text{DE}$ ,  $AXDE \leftarrow \text{FFFFFFFFH}$

**(10) Special operation instructions: MACW, MACSW, SACW**

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MACW	byte	3	$AXDE \leftarrow (B) \times (C) + AXDE$ , $B \leftarrow B + 2$ , $C \leftarrow C + 2$ , $\text{byte} \leftarrow \text{byte} - 1$ End if( $\text{byte} = 0$ or $P/V = 1$ )	x	x	x	V	x
MACSW	byte	3	$AXDE \leftarrow (B) \times (C) + AXDE$ , $B \leftarrow B + 2$ , $C \leftarrow C + 2$ , $\text{byte} \leftarrow \text{byte} - 1$ if $\text{byte} = 0$ then End if $P/V = 1$ then if overflow $AXDE \leftarrow \text{7FFFFFFFH}$ , End if underflow $AXDE \leftarrow \text{80000000H}$ , End	x	x	x	V	x
SACW	[TDE + ], [WHL + ]	4	$AX \leftarrow  (TDE) - (WHL)  + AX$ , $TDE \leftarrow TDE + 2$ , $WHL \leftarrow WHL + 2$ $C \leftarrow C - 1$ End if( $C = 0$ or $CY = 1$ )	x	x	x	V	x

**(11) Increment/decrement instructions: INC, DEC, INCW, DECW, INCG, DECG**

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
INC	r	1/2	$r \leftarrow r + 1$	×	×	×	V	
	saddr	2/3	$(saddr) \leftarrow (saddr) + 1$	×	×	×	V	
DEC	r	1/2	$r \leftarrow r - 1$	×	×	×	V	
	saddr	2/3	$(saddr) \leftarrow (saddr) - 1$	×	×	×	V	
INCW	rp	2/1	$rp \leftarrow rp + 1$					
	saddrp	3/4	$(saddrp) \leftarrow (saddrp) + 1$					
DECW	rp	2/1	$rp \leftarrow rp - 1$					
	saddrp	3/4	$(saddrp) \leftarrow (saddrp) - 1$					
INCG	rg	2	$rg \leftarrow rg + 1$					
DECG	rg	2	$rg \leftarrow rg - 1$					

**(12) Adjustment instructions: ADJBA, ADJBS, CVTBW**

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ADJBA		2	Decimal Adjust Accumulator after Addition	×	×	×	P	×
ADJBS		2	Decimal Adjust Accumulator after Subtract	×	×	×	P	×
CVTBW		1	$X \leftarrow A, A \leftarrow 00H$ if $A_7 = 0$ $X \leftarrow A, A \leftarrow FFH$ if $A_7 = 1$					

## (13) Shift/rotate instructions: ROR, ROL, RORC, ROLC, SHR, SHL, SHRW, SHLW, ROR4, ROL4

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ROR	r, n	2/3	$(CY, r7 \leftarrow r0, rm - 1 \leftarrow rm) \times n \text{ times } n = 0 - 7$				P	×
ROL	r, n	2/3	$(CY, r0 \leftarrow r7, rm + 1 \leftarrow rm) \times n \text{ times } n = 0 - 7$				P	×
RORC	r, n	2/3	$(CY \leftarrow r0, r7 \leftarrow CY, rm - 1 \leftarrow rm) \times n \text{ times } n = 0 - 7$				P	×
ROLC	r, n	2/3	$(CY \leftarrow r7, r0 \leftarrow CY, rm + 1 \leftarrow rm) \times n \text{ times } n = 0 - 7$				P	×
SHR	r, n	2/3	$(CY \leftarrow r0, r7 \leftarrow 0, rm - 1 \leftarrow rm) \times n \text{ times } n = 0 - 7$	×	×	0	P	×
SHL	r, n	2/3	$(CY \leftarrow r7, r0 \leftarrow 0, rm + 1 \leftarrow rm) \times n \text{ times } n = 0 - 7$	×	×	0	P	×
SHRW	rp, n	2	$(CY \leftarrow rp0, rp15 \leftarrow 0, rpm - 1 \leftarrow rpm) \times n \text{ times } n = 0 - 7$	×	×	0	P	×
SHLW	rp, n	2	$(CY \leftarrow rp15, rp0 \leftarrow 0, rpm + 1 \leftarrow rpm) \times n \text{ times } n = 0 - 7$	×	×	0	P	×
ROR4	mem3	2	$A3 - 0 \leftarrow (mem3)3 - 0, (mem3)7 - 4 \leftarrow A3 - 0, (mem3)3 - 0 \leftarrow (mem3)7 - 4$					
ROL4	mem3	2	$A3 - 0 \leftarrow (mem3)7 - 4, (mem3)3 - 0 \leftarrow A3 - 0, (mem3)7 - 4 \leftarrow (mem3)3 - 0$					

## (14) Bit manipulation instructions: MOV1, AND1, OR1, XOR1, NOT1, SET1, CLR1

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MOV1	CY, saddr. bit	3/4	$CY \leftarrow (saddr. \text{ bit})$					×
	CY, sfr. bit	3	$CY \leftarrow \text{sfr. bit}$					×
	CY, X. bit	2	$CY \leftarrow X. \text{ bit}$					×
	CY, A. bit	2	$CY \leftarrow A. \text{ bit}$					×
	CY, PSWL. bit	2	$CY \leftarrow \text{PSWL. bit}$					×
	CY, PSWH. bit	2	$CY \leftarrow \text{PSWH. bit}$					×
	CY, !addr16. bit	5	$CY \leftarrow !\text{addr16. bit}$					×
	CY, !!addr24. bit	2	$CY \leftarrow !!\text{addr24. bit}$					×
	CY, mem2. bit	2	$CY \leftarrow \text{mem2. bit}$					×
	saddr. bit, CY	3/4	$(saddr. \text{ bit}) \leftarrow CY$					
	sfr. bit, CY	3	$\text{sfr. bit} \leftarrow CY$					
	X. bit, CY	2	$X. \text{ bit} \leftarrow CY$					
	A. bit, CY	2	$A. \text{ bit} \leftarrow CY$					
	PSWL. bit, CY	2	$\text{PSWL. bit} \leftarrow CY$	×	×	×	×	×
	PSWH. bit, CY	2	$\text{PSWH. bit} \leftarrow CY$					
	!addr16. bit, CY	5	$!\text{addr16. bit} \leftarrow CY$					
	!!addr24. bit, CY	6	$!!\text{addr24. bit} \leftarrow CY$					
	mem2. bit, CY	2	$\text{mem2. bit} \leftarrow CY$					

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
AND1	CY, saddr. bit	3/4	$CY \leftarrow CY \wedge (\text{saddr. bit})$					×
	CY, /saddr. bit	3/4	$CY \leftarrow CY \wedge \overline{(\text{saddr. bit})}$					×
	CY, sfr. bit	3	$CY \leftarrow CY \wedge \text{sfr. bit}$					×
	CY, /sfr. bit	3	$CY \leftarrow CY \wedge \overline{\text{sfr. bit}}$					×
	CY, X. bit	2	$CY \leftarrow CY \wedge X. \text{ bit}$					×
	CY, /X. bit	2	$CY \leftarrow CY \wedge \overline{X. \text{ bit}}$					×
	CY, A. bit	2	$CY \leftarrow CY \wedge A. \text{ bit}$					×
	CY, /A. bit	2	$CY \leftarrow CY \wedge \overline{A. \text{ bit}}$					×
	CY, PSWL. bit	2	$CY \leftarrow CY \wedge \text{PSWL. bit}$					×
	CY, /PSWL. bit	2	$CY \leftarrow CY \wedge \overline{\text{PSWL. bit}}$					×
	CY, PSWH. bit	2	$CY \leftarrow CY \wedge \text{PSWH. bit}$					×
	CY, /PSWH. bit	2	$CY \leftarrow CY \wedge \overline{\text{PSWH. bit}}$					×
	CY, !addr16. bit	5	$CY \leftarrow CY \wedge \text{!addr16. bit}$					×
	CY, /!addr16. bit	5	$CY \leftarrow CY \wedge \overline{\text{!addr16. bit}}$					×
	CY, !!addr24. bit	2	$CY \leftarrow CY \wedge \text{!!addr24. bit}$					×
	CY, /!!addr24. bit	6	$CY \leftarrow CY \wedge \overline{\text{!!addr24. bit}}$					×
	CY, mem2. bit	2	$CY \leftarrow CY \wedge \text{mem2. bit}$					×
	CY, /mem2. bit	2	$CY \leftarrow CY \wedge \overline{\text{mem2. bit}}$					×
OR1	CY, saddr. bit	3/4	$CY \leftarrow CY \vee (\text{saddr. bit})$					×
	CY, /saddr. bit	3/4	$CY \leftarrow CY \vee \overline{(\text{saddr. bit})}$					×
	CY, sfr. bit	3	$CY \leftarrow CY \vee \text{sfr. bit}$					×
	CY, /sfr. bit	3	$CY \leftarrow CY \vee \overline{\text{sfr. bit}}$					×
	CY, X. bit	2	$CY \leftarrow CY \vee X. \text{ bit}$					×
	CY, /X. bit	2	$CY \leftarrow CY \vee \overline{X. \text{ bit}}$					×
	CY, A. bit	2	$CY \leftarrow CY \vee A. \text{ bit}$					×
	CY, /A. bit	2	$CY \leftarrow CY \vee \overline{A. \text{ bit}}$					×
	CY, PSWL. bit	2	$CY \leftarrow CY \vee \text{PSWL. bit}$					×
	CY, /PSWL. bit	2	$CY \leftarrow CY \vee \overline{\text{PSWL. bit}}$					×
	CY, PSWH. bit	2	$CY \leftarrow CY \vee \text{PSWH. bit}$					×
	CY, /PSWH. bit	2	$CY \leftarrow CY \vee \overline{\text{PSWH. bit}}$					×
	CY, !addr16. bit	5	$CY \leftarrow CY \vee \text{!addr16. bit}$					×
	CY, /!addr16. bit	5	$CY \leftarrow CY \vee \overline{\text{!addr16. bit}}$					×
	CY, !!addr24. bit	2	$CY \leftarrow CY \vee \text{!!addr24. bit}$					×
	CY, /!!addr24. bit	6	$CY \leftarrow CY \vee \overline{\text{!!addr24. bit}}$					×
	CY, mem2. bit	2	$CY \leftarrow CY \vee \text{mem2. bit}$					×
	CY, /mem2. bit	2	$CY \leftarrow CY \vee \overline{\text{mem2. bit}}$					×

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
XOR1	CY, saddr. bit	3/4	$CY \leftarrow CY \nabla (\text{saddr. bit})$					×
	CY, sfr. bit	3	$CY \leftarrow CY \nabla \text{sfr. bit}$					×
	CY, X. bit	2	$CY \leftarrow CY \nabla X. \text{ bit}$					×
	CY, A. bit	2	$CY \leftarrow CY \nabla A. \text{ bit}$					×
	CY, PSWL. bit	2	$CY \leftarrow CY \nabla \text{PSWL. bit}$					×
	CY, PSWH. bit	2	$CY \leftarrow CY \nabla \text{PSWH. bit}$					×
	CY, !addr16. bit	5	$CY \leftarrow CY \nabla \text{!addr16. bit}$					×
	CY, !!addr24. bit	2	$CY \leftarrow CY \nabla \text{!!addr24. bit}$					×
	CY, mem2. bit	2	$CY \leftarrow CY \nabla \text{mem2. bit}$					×
NOT1	saddr. bit	3/4	$(\text{saddr. bit}) \leftarrow \overline{(\text{saddr. bit})}$					
	sfr. bit	3	$\text{sfr. bit} \leftarrow \overline{\text{sfr. bit}}$					
	X. bit	2	$X. \text{ bit} \leftarrow \overline{X. \text{ bit}}$					
	A. bit	2	$A. \text{ bit} \leftarrow \overline{A. \text{ bit}}$					
	PSWL. bit	2	$\text{PSWL. bit} \leftarrow \overline{\text{PSWL. bit}}$	×	×	×	×	×
	PSWH. bit	2	$\text{PSWH. bit} \leftarrow \overline{\text{PSWH. bit}}$					
	!addr16. bit	5	$\text{!addr16. bit} \leftarrow \overline{\text{!addr16. bit}}$					
	!!addr24. bit	2	$\text{!!addr24. bit} \leftarrow \overline{\text{!!addr24. bit}}$					
	mem2. bit	2	$\text{mem2. bit} \leftarrow \overline{\text{mem2. bit}}$					
	CY	1	$CY \leftarrow \overline{CY}$					×
SET1	saddr. bit	2/3	$(\text{saddr. bit}) \leftarrow 1$					
	sfr. bit	3	$\text{sfr. bit} \leftarrow 1$					
	X. bit	2	$X. \text{ bit} \leftarrow 1$					
	A. bit	2	$A. \text{ bit} \leftarrow 1$					
	PSWL. bit	2	$\text{PSWL. bit} \leftarrow 1$	×	×	×	×	×
	PSWH. bit	2	$\text{PSWH. bit} \leftarrow 1$					
	!addr16. bit	5	$\text{!addr16. bit} \leftarrow 1$					
	!!addr24. bit	2	$\text{!!addr24. bit} \leftarrow 1$					
	mem2. bit	2	$\text{mem2. bit} \leftarrow 1$					
	CY	1	$CY \leftarrow 1$					1
CLR1	saddr. bit	2/3	$(\text{saddr. bit}) \leftarrow 0$					
	sfr. bit	3	$\text{sfr. bit} \leftarrow 0$					
	X. bit	2	$X. \text{ bit} \leftarrow 0$					
	A. bit	2	$A. \text{ bit} \leftarrow 0$					
	PSWL. bit	2	$\text{PSWL. bit} \leftarrow 0$	×	×	×	×	×
	PSWH. bit	2	$\text{PSWH. bit} \leftarrow 0$					
	!addr16. bit	5	$\text{!addr16. bit} \leftarrow 0$					
	!!addr24. bit	2	$\text{!!addr24. bit} \leftarrow 0$					
	mem2. bit	2	$\text{mem2. bit} \leftarrow 0$					
	CY	1	$CY \leftarrow 0$					0

**(15) Stack manipulation instructions: PUSH, PUSHU, POP, POPU, MOVG, ADDWG, SUBWG, INCG, DECG**

Mnemonic	Operands	Bytes	Operation	Flags					
				S	Z	AC	P/V	CY	
PUSH	PSW	1	$(SP - 2) \leftarrow PSW, SP \leftarrow SP - 2$						
	sfrp	3	$(SP - 2) \leftarrow sfrp, SP \leftarrow SP - 2$						
	sfr	3	$(SP - 1) \leftarrow sfr, SP \leftarrow SP - 1$						
	post	2	$\{(SP - 2) \leftarrow post, SP \leftarrow SP - 2\} \times m \text{ times}^*$						
	rg	2	$(SP - 3) \leftarrow rg, SP \leftarrow SP - 3$						
PUSHU	post	2	$\{(UUP - 2) \leftarrow post, UUP \leftarrow UUP - 2\} \times m \text{ times}^*$						
POP	PSW	1	$PSW \leftarrow (SP), SP \leftarrow SP + 2$	R	R	R	R	R	
	sfrp	3	$sfrp \leftarrow (SP), SP \leftarrow SP + 2$						
	sfr	3	$sfr \leftarrow (SP), SP \leftarrow SP + 1$						
	post	2	$\{post \leftarrow (SP), SP \leftarrow SP + 2\} \times m \text{ times}^*$						
	rg	2	$rg \leftarrow (SP), SP \leftarrow SP + 3$						
POPU	post	2	$\{post \leftarrow (UUP), UUP \leftarrow UUP + 2\} \times m \text{ times}^*$						
MOVG	SP, # imm24	5	$SP \leftarrow imm24$						
	SP, WHL	2	$SP \leftarrow WHL$						
	WHL, SP	2	$WHL \leftarrow SP$						
ADDWG	SP, #word	4	$SP \leftarrow SP + word$						
SUBWG	SP, #word	4	$SP \leftarrow SP - word$						
INCG	SP	2	$SP \leftarrow SP + 1$						
DECG	SP	2	$SP \leftarrow SP - 1$						

\* m = number of registers specified by "post"

**(16) Call/return instructions: CALL, CALLF, CALLT, BRK, BRKCS, RET, RETI, RETB, RETCS, RETCSB**

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
CALL	!addr16	3	$(SP - 3) \leftarrow (PC + 3)$ , $SP \leftarrow SP - 3$ , $PC_{HW} \leftarrow 0$ , $PC_{LW} \leftarrow \text{addr16}$					
	!!addr20	4	$(SP - 3) \leftarrow (PC + 4)$ , $SP \leftarrow SP - 3$ , $PC \leftarrow \text{addr20}$					
	rp	2	$(SP - 3) \leftarrow (PC + 2)$ , $SP \leftarrow SP - 3$ , $PC_{HW} \leftarrow 0$ , $PC_{LW} \leftarrow rp$					
	rg	2	$(SP - 3) \leftarrow (PC + 2)$ , $SP \leftarrow SP - 3$ , $PC \leftarrow rg$					
	[rp]	2	$(SP - 3) \leftarrow (PC + 2)$ , $SP \leftarrow SP - 3$ , $PC_{HW} \leftarrow 0$ , $PC_{LW} \leftarrow (rp)$					
	[rg]	2	$(SP - 3) \leftarrow (PC + 2)$ , $SP \leftarrow SP - 3$ , $PC \leftarrow (rg)$					
	!addr20	3	$(SP - 3) \leftarrow (PC + 3)$ , $SP \leftarrow SP - 3$ , $PC \leftarrow PC + 3 + \text{jdisp16}$					
CALLF	!addr11	2	$(SP - 3) \leftarrow (PC + 2)$ , $SP \leftarrow SP - 3$ , $PC_{19-12} \leftarrow 0$ , $PC_{11} \leftarrow 1$ , $PC_{10-0} \leftarrow \text{addr11}$					
CALLT	[addr5]	1	$(SP - 3) \leftarrow (PC + 1)$ , $SP \leftarrow SP - 3$ , $PC_{HW} \leftarrow 0$ , $PC_{LW} \leftarrow (\text{addr5})$					
BRK		1	$(SP - 2) \leftarrow PSW$ , $(SP - 1)_{0-3} \leftarrow (PC + 1)_{HW}$ , $(SP - 4) \leftarrow (PC + 1)_{LW}$ , $SP \leftarrow SP - 4$ $PC_{HW} \leftarrow 0$ , $PC_{LW} \leftarrow (003EH)$					
BRKCS	RBn	2	$PC_{LW} \leftarrow RP2$ , $RP3 \leftarrow PSW$ , $RBS2 - 0 \leftarrow n$ , $RSS \leftarrow 0$ , $IE \leftarrow 0$ , $RP3_{8-11} \leftarrow PC_{HW}$ , $PC_{HW} \leftarrow 0$					
RET		1	$PC \leftarrow (SP)$ , $SP \leftarrow SP + 3$					
RETI		1	$PC_{LW} \leftarrow (SP)$ , $PC_{HW} \leftarrow (SP + 3)_{0-3}$ , $PSW \leftarrow (SP + 2)$ , $SP \leftarrow SP + 4$ Clears to 0 flag with highest priority of flags of ISPR that are set (1)	R	R	R	R	R
RETB		1	$PC_{LW} \leftarrow (SP)$ , $PC_{HW} \leftarrow (SP + 3)_{0-3}$ , $PSW \leftarrow (SP + 2)$ , $SP \leftarrow SP + 4$	R	R	R	R	R
RETCS	!addr16	3	$PSW \leftarrow RP3$ , $PC_{LW} \leftarrow RP2$ , $RP2 \leftarrow \text{addr16}$ , $PC_{HW} \leftarrow RP3_{8-11}$ Clears to 0 flag with highest priority of flags of ISPR that are set (1)	R	R	R	R	R
RETCSB	!addr16	4	$PSW \leftarrow RP3$ , $PC_{LW} \leftarrow RP2$ , $RP2 \leftarrow \text{addr16}$ , $PC_{HW} \leftarrow RP3_{8-11}$	R	R	R	R	R

**(17) Unconditional branch instruction: BR**

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
BR	!addr16	3	$PC_{HW} \leftarrow 0, PC_{LW} \leftarrow \text{addr16}$					
	!!addr20	4	$PC \leftarrow \text{addr20}$					
	rp	2	$PC_{HW} \leftarrow 0, PC_{LW} \leftarrow rp$					
	rg	2	$PC \leftarrow rg$					
	[rp]	2	$PC_{HW} \leftarrow 0, PC_{LW} \leftarrow (rp)$					
	[rg]	2	$PC \leftarrow (rg)$					
	\$addr20	2	$PC \leftarrow PC + 2 + \text{jdisp8}$					
	\$!addr20	3	$PC \leftarrow PC + 3 + \text{jdisp16}$					



(18) Conditional branch instructions: BNZ, BNE, BZ, BE, BNC, BNL, BC, BL, BN, BZ, BPE, BP, BN, BLT, BGE, BLE, BGT, BNH, BH, BF, BT, BTCLR, BFSET, DBNZ

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
BNZ	\$addr20	2	$PC \leftarrow PC + 2 + \text{jdisp8}$ if $Z = 0$					
BNE								
BZ	\$addr20	2	$PC \leftarrow PC + 2 + \text{jdisp8}$ if $Z = 1$					
BE								
BNC	\$addr20	2	$PC \leftarrow PC + 2 + \text{jdisp8}$ if $CY = 0$					
BNL								
BC	\$addr20	2	$PC \leftarrow PC + 2 + \text{jdisp8}$ if $CY = 1$					
BL								
BNV	\$addr20	2	$PC \leftarrow PC + 2 + \text{jdisp8}$ if $P/V = 0$					
BPO								
BV	\$addr20	2	$PC \leftarrow PC + 2 + \text{jdisp8}$ if $P/V = 1$					
BPE								
BP	\$addr20	2	$PC \leftarrow PC + 2 + \text{jdisp8}$ if $S = 0$					
BN	\$addr20	2	$PC \leftarrow PC + 2 + \text{jdisp8}$ if $S = 1$					
BLT	\$addr20	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if $P/V \nabla S = 1$					
BGE	\$addr20	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if $P/V \nabla S = 0$					
BLE	\$addr20	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if $(P/V \nabla S) \vee Z = 1$					
BGT	\$addr20	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if $(P/V \nabla S) \vee Z = 0$					
BNH	\$addr20	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if $Z \vee CY = 1$					
BH	\$addr20	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if $Z \vee CY = 0$					
BF	saddr. bit, \$addr20	4/5	$PC \leftarrow PC + 4^* + \text{jdisp8}$ if (saddr. bit) = 0					
	sfr. bit, \$addr20	4	$PC \leftarrow PC + 4 + \text{jdisp8}$ if sfr. bit = 0					
	X. bit, \$addr20	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if X. bit = 0					
	A. bit, \$addr20	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if A. bit = 0					
	PSWL. bit, \$addr20	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if PSWL. bit = 0					
	PSWH. bit, \$addr20	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if PSWH. bit = 0					
	!addr16. bit, \$addr20	6	$PC \leftarrow PC + 3 + \text{jdisp8}$ if !addr16. bit = 0					
	!!addr24. bit, \$addr20	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if !!addr24. bit = 0					
	mem2. bit, \$addr20	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if mem2. bit = 0					

\* When the number of bytes is 4. When 5, the operation is:  $PC \leftarrow PC + 5 + \text{jdisp8}$ .

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
BT	saddr. bit, \$addr20	3/4	$PC \leftarrow PC + 3^{*1} + \text{jdisp8}$ if (saddr. bit) = 1					
	sfr. bit, \$addr20	4	$PC \leftarrow PC + 4 + \text{jdisp8}$ if sfr. bit = 1					
	X. bit, \$addr20	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if X. bit = 1					
	A. bit, \$addr20	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if A. bit = 1					
	PSWL. bit, \$addr20	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if PSWL. bit = 1					
	PSWH. bit, \$addr20	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if PSWH. bit = 1					
	!addr16. bit, \$addr20	6	$PC \leftarrow PC + 3 + \text{jdisp8}$ if !addr16. bit = 1					
	!!addr24. bit, \$addr20	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if !!addr24. bit = 1					
	mem2. bit, \$addr20	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if mem2. bit = 1					
BTCLR	saddr. bit, \$addr20	4/5	{ $PC \leftarrow PC + 4^{*2} + \text{jdisp8}$ , (saddr. bit) $\leftarrow 0$ } if (saddr. bit) = 1					
	sfr. bit, \$addr20	4	{ $PC \leftarrow PC + 4 + \text{jdisp8}$ , sfr. bit $\leftarrow 0$ } if sfr. bit = 1					
	X. bit, \$addr20	3	{ $PC \leftarrow PC + 3 + \text{jdisp8}$ , X. bit $\leftarrow 0$ } if X. bit = 1					
	A. bit, \$addr20	3	{ $PC \leftarrow PC + 3 + \text{jdisp8}$ , A. bit $\leftarrow 0$ } if A. bit = 1					
	PSWL. bit, \$addr20	3	{ $PC \leftarrow PC + 3 + \text{jdisp8}$ , PSWL. bit $\leftarrow 0$ } if PSWL. bit = 1	×	×	×	×	×
	PSWH. bit, \$addr20	3	{ $PC \leftarrow PC + 3 + \text{jdisp8}$ , PSWH. bit $\leftarrow 0$ } if PSWH. bit = 1					
	!addr16. bit, \$addr20	6	{ $PC \leftarrow PC + 3 + \text{jdisp8}$ , !addr16. bit $\leftarrow 0$ } if !addr16. bit = 1					
	!!addr24. bit, \$addr20	3	{ $PC \leftarrow PC + 3 + \text{jdisp8}$ , !!addr24. bit $\leftarrow 0$ } if !!addr24. bit = 1					
	mem2. bit, \$addr20	3	{ $PC \leftarrow PC + 3 + \text{jdisp8}$ , mem2. bit $\leftarrow 0$ } if mem2. bit = 1					

\* 1. When the number of bytes is 3. When 4, the operation is:  $PC \leftarrow PC + 4 + \text{jdisp8}$ .

2. When the number of bytes is 4. When 5, the operation is:  $PC \leftarrow PC + 5 + \text{jdisp8}$ .

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
BFSET	saddr. bit, \$addr20	4/5	{PC ← PC + 4*2 + jdisp8, (saddr. bit) ← 1} if (saddr. bit) = 0					
	sfr. bit, \$addr20	4	{PC ← PC + 4 + jdisp8, sfr. bit ← 1} if sfr. bit = 0					
	X. bit, \$addr20	3	{PC ← PC + 3 + jdisp8, X. bit ← 1} if X. bit = 0					
	A. bit, \$addr20	3	{PC ← PC + 3 + jdisp8, A. bit ← 1} if A. bit = 0					
	PSWL. bit, \$addr20	3	{PC ← PC + 3 + jdisp8, PSWL. bit ← 1} if PSWL. bit = 0	×	×	×	×	×
	PSWH. bit, \$addr20	3	{PC ← PC + 3 + jdisp8, PSWH. bit ← 1} if PSWH. bit = 0					
	!addr16. bit, \$addr20	6	{PC ← PC + 3 + jdisp8, !addr16. bit ← 1} if !addr16. bit = 0					
	!!addr24. bit, \$addr20	3	{PC ← PC + 3 + jdisp8, !!addr24. bit ← 1} if !!addr24. bit = 0					
	mem2. bit, \$addr20	3	{PC ← PC + 3 + jdisp8, mem2. bit ← 1} if mem2. bit = 0					
DBNZ	B, \$addr20	2	B ← B - 1, PC ← PC + 2 + jdisp8 if B ≠ 0					
	C, \$addr20	2	C ← C - 1, PC ← PC + 2 + jdisp8 if C ≠ 0					
	\$addr, \$addr20	3/4	(saddr) ← (saddr) - 1, PC ← PC + 3*1 = jdisp8 if (saddr) ≠ 0					

\* 1. When the number of bytes is 3. When 4, the operation is: PC ← PC + 4 + jdisp8.

2. When the number of bytes is 4. When 5, the operation is: PC ← PC + 5 + jdisp8.

**(19) CPU control instructions: MOV, LOCATION, SEL, SWRS, NOP, EI, DI**

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MOV	STBC, #byte	4	STBC ← byte					
	WDM, #byte	4	WDM ← byte					
LOCATION	locaddr	4	SFR, internal data area location address high-order word specification					
SEL	RBn	2	RSS ← 0, RBS2 - 0 ← n					
	RBn, ALT	2	RSS ← 1, RBS2 - 0 ← n					
SWRS		2	RSS ← $\overline{\text{RSS}}$					
NOP		1	No Operaton					
EI		1	IE ← 1 (Enable interrupt)					
DI		1	IE ← 0 (Disable interrupt)					

**(20) Special instructions: CHKL, CHKLA**

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
CHKL	sfr	3	(Pin level) $\nabla$ (output latch)	×	×		P	
CHKLA	sfr	3	$A \leftarrow$ (pin level) $\nabla$ (output latch)	×	×		P	

**(21) String instructions: MOVTBLW, MOVW, XCHM, MOVBK, XCHBK, CMPME, CMPMNE, CMPMC, CMPMNC, CMPBKE, CMPBKNE, CMPBKC, CMPBKNC**

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MOVTBLW	!addr8, byte	4	$(\text{addr8} + 2) \leftarrow (\text{addr8}), \text{byte} \leftarrow \text{byte} - 1, \text{addr8} \leftarrow \text{addr8} - 2$ End if byte = 0					
MOVW	[TDE + ], A	2	$(\text{TDE}) \leftarrow A, \text{TDE} \leftarrow \text{TDE} + 1, C \leftarrow C - 1$ End if C = 0					
	[TDE - ], A	2	$(\text{TDE}) \leftarrow A, \text{TDE} \leftarrow \text{TDE} - 1, C \leftarrow C - 1$ End if C = 0					
XCHM	[TDE + ], A	2	$(\text{TDE}) \leftrightarrow A, \text{TDE} \leftarrow \text{TDE} + 1, C \leftarrow C - 1$ End if C = 0					
	[TDE - ], A	2	$(\text{TDE}) \leftrightarrow A, \text{TDE} \leftarrow \text{TDE} - 1, C \leftarrow C - 1$ End if C = 0					
MOVBK	[TDE + ], [WHL +]	2	$(\text{TDE}) \leftarrow (\text{WHL}), \text{TDE} \leftarrow \text{TDE} + 1, \text{WHL} \leftarrow \text{WHL} + 1, C \leftarrow C - 1$ End if C = 0					
	[TDE - ], [WHL -]	2	$(\text{TDE}) \leftarrow (\text{WHL}), \text{TDE} \leftarrow \text{TDE} - 1, \text{WHL} \leftarrow \text{WHL} - 1, C \leftarrow C - 1$ End if C = 0					
XCHBK	[TDE + ], [WHL +]	2	$(\text{TDE}) \leftrightarrow (\text{WHL}), \text{TDE} \leftarrow \text{TDE} + 1, \text{WHL} \leftarrow \text{WHL} + 1, C \leftarrow C - 1$ End if C = 0					
	[TDE - ], [WHL -]	2	$(\text{TDE}) \leftrightarrow (\text{WHL}), \text{TDE} \leftarrow \text{TDE} - 1, \text{WHL} \leftarrow \text{WHL} - 1, C \leftarrow C - 1$ End if C = 0					
CMPME	[TDE + ], A	2	$(\text{TDE}) - A, \text{TDE} \leftarrow \text{TDE} + 1, C \leftarrow C - 1$ End if C = 0 or Z = 0	×	×	×	V	×
	[TDE - ], A	2	$(\text{TDE}) - A, \text{TDE} \leftarrow \text{TDE} - 1, C \leftarrow C - 1$ End if C = 0 or Z = 0	×	×	×	V	×
CMPMNE	[TDE + ], A	2	$(\text{TDE}) - A, \text{TDE} \leftarrow \text{TDE} + 1, C \leftarrow C - 1$ End if C = 0 or Z = 1	×	×	×	V	×
	[TDE - ], A	2	$(\text{TDE}) - A, \text{TDE} \leftarrow \text{TDE} - 1, C \leftarrow C - 1$ End if C = 0 or Z = 1	×	×	×	V	×
CMPMC	[TDE + ], A	2	$(\text{TDE}) - A, \text{TDE} \leftarrow \text{TDE} + 1, C \leftarrow C - 1$ End if C = 0 or CY = 0	×	×	×	V	×
	[TDE - ], A	2	$(\text{TDE}) - A, \text{TDE} \leftarrow \text{TDE} - 1, C \leftarrow C - 1$ End if C = 0 or CY = 0	×	×	×	V	×
CMPMNC	[TDE + ], A	2	$(\text{TDE}) - A, \text{TDE} \leftarrow \text{TDE} + 1, C \leftarrow C - 1$ End if C = 0 or CY = 1	×	×	×	V	×
	[TDE - ], A	2	$(\text{TDE}) - A, \text{TDE} \leftarrow \text{TDE} - 1, C \leftarrow C - 1$ End if C = 0 or CY = 1	×	×	×	V	×
CMPBKE	[TDE + ], [WHL +]	2	$(\text{TDE}) \leftarrow (\text{WHL}), \text{TDE} \leftarrow \text{TDE} + 1, \text{WHL} \leftarrow \text{WHL} + 1, C \leftarrow C - 1$ End if C = 0 or Z = 0	×	×	×	V	×
	[TDE - ], [WHL -]	2	$(\text{TDE}) \leftarrow (\text{WHL}), \text{TDE} \leftarrow \text{TDE} - 1, \text{WHL} \leftarrow \text{WHL} - 1, C \leftarrow C - 1$ End if C = 0 or Z = 0	×	×	×	V	×

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
CMPBKNE	[TDE + ], [WHL +]	2	(TDE) – (WHL), TDE $\leftarrow$ TDE + 1, WHL $\leftarrow$ WHL + 1, C $\leftarrow$ C – 1 End if C = 0 or Z = 1	×	×	×	V	×
	[TDE – ], [WHL –]	2	(TDE) – (WHL), TDE $\leftarrow$ TDE – 1, WHL $\leftarrow$ WHL – 1, C $\leftarrow$ C – 1 End if C = 0 or Z = 1	×	×	×	V	×
CMPBKC	[TDE + ], [WHL +]	2	(TDE) – (WHL), TDE $\leftarrow$ TDE + 1, WHL $\leftarrow$ WHL + 1, C $\leftarrow$ C – 1 End if C = 0 or CY = 0	×	×	×	V	×
	[TDE – ], [WHL –]	2	(TDE) – (WHL), TDE $\leftarrow$ TDE – 1, WHL $\leftarrow$ WHL – 1, C $\leftarrow$ C – 1 End if C = 0 or CY = 0	×	×	×	V	×
CMPBKNC	[TDE + ], [WHL +]	2	(TDE) – (WHL), TDE $\leftarrow$ TDE + 1, WHL $\leftarrow$ WHL + 1, C $\leftarrow$ C – 1 End if C = 0 or CY = 1	×	×	×	V	×
	[TDE – ], [WHL –]	2	(TDE) – (WHL), TDE $\leftarrow$ TDE – 1, WHL $\leftarrow$ WHL – 1, C $\leftarrow$ C – 1 End if C = 0 or CY = 1	×	×	×	V	×

**25.3 INSTRUCTIONS LISTED BY TYPE OF ADDRESSING****(1) 8-bit instructions (combinations expressed by writing A for r are shown in parentheses)**

MOV, XCH, ADD, ADDC, SUB, SUBC, AND OR XOR, CMP, MULU, DIVUW, INC, DEC, ROR, ROL, RORC, ROLC, SHR, SHL, ROR4, ROL4, DBNZ, PUSH, POP, MOVM, XCHM, CMPME, CMPMNE, CMPMNC, CMPMC, MOVBK, XCHBK, CMPBKE, CMPBKNE, CMPBKNC, CMPBKC, CHKL, CHKLA

**Table 25-1 List of Instructions by 8-Bit Addressing**

2nd Operand 1st Operand	# byte	A	r r'	saddr saddr'	sfr	!addr16 !!addr24	mem [saddrp] [%saddrg]	r3 PSWL PSWH	[WHL +] [WHL -]	n	None*2
A	(MOV) ADD*1	(MOV) (XCH) (ADD)*1	MOV XCH (ADD)*1	(MOV)*6 (XCH)*6 (ADD)*1, 6	MOV (XCH) (ADD)*1	(MOV) (XCH) ADD*1	MOV XCH ADD*1	MOV	(MOV) (XCH) (ADD)*1		
r	MOV ADD*1	(MOV) (XCH) (ADD)*1	MOV XCH ADD*1	MOV XCH ADD*1	MOV XCH ADD*1	MOV XCH				ROR*3	MULU DIVUW INC DEC
saddr	MOV ADD*1	MOV*6 (ADD)*1	MOV ADD*1	MOV XCH ADD*1							INC DEC DBNZ
sfr	MOV ADD*1	MOV (ADD)*1	MOV ADD*1								PUSH POP CHKL CHKLA
!addr16 !!addr24	MOV	(MOV) (ADD)*1	MOV								
mem [saddrp] [%saddrg]		MOV ADD*1									
mem3											ROR4 ROL4
r3 PSWL PSWH	MOV	MOV									
B, C											DBNZ
STBC, WDM	MOV										
[TDE +] [TDE -]		(MOV) (ADD)*1 MOVM*4							MOVBK*5		

\* 1. ADDC, SUB, SUBC, AND, OR, XOR and CMP are the same as ADD.

2. There is no 2nd operand, or the 2nd operand is not an operand address.

3. ROL, RORC, ROLC, SHR and SHL are the same as ROR.

4. XCHM, CMPME, CMPMNE, CMPMNC and CMPMC are the same as MOVM.

5. XCHBK, CMPBKE, CMPBKNE, CMPBKNC and CMPBKC are the same as MOVBK.

6. If saddr is saddr2 in this combination, there is a short code length instruction.

**(2) 16-bit instructions (combinations expressed by writing AX for rp are shown in parentheses)**

MOVM, XCHW, ADDW, SUBW, CMPW, MULUW, MULW, DIVUX, INCW, DECW, SHRW, SHLW, PUSH, POP, ADDWG, SUBWG, PUSHU, POPU, MOVTBLW, MACW, MACSW, SACW

**Table 25-2 List of Instructions by 16-Bit Addressing**

2nd Operand 1st Operand	# word	AX	r r'	saddrp saddrp'	sfrp	!addr16 !!addr24	mem [saddrp] [%saddrg]	[WHL +]	byte	n	None*2
AX	(MOVW) ADDW*1	(MOVW) (XCHW) (ADD)*1	(MOVW) (XCHW) (ADDW)*1	(MOVW)*3 (XCHW)*3 (ADDW)*1,3	MOVW (XCHW) (ADDW)*1	(MOVW) XCHW	MOVW XCHW	(MOVW) (XCHW)			
rp	MOVW ADDW*1	(MOVW) (XCHW) (ADDW)*1	MOVW XCHW ADDW*1	MOVW XCHW ADDW*1	MOVW XCHW ADDW*1	MOVW				SHRW SHLW	MULW*4 INCW DECW
saddrp	MOVW ADDW*1	(MOVW)*3 (ADDW)*1	MOVW ADDW*1	MOVW XCHW ADDW*1							INCW DECW
sfrp	MOVW ADDW*1	MOVW (ADDW)*1	MOVW ADDW*1								PUSH POP
!addr16 !!addr24	MOVW	(MOVW)	MOVW						MOVTBLW		
mem [saddrp] [%saddrg]		MOVW									
PSW											PUSH POP
SP	ADDWG SUBWG										
post											PUSH POP PUSHU POPU
[TDE +]		(MOVW)						SACW			
byte											MACW MACSW

\* 1. SUBW and CMPW are the same as ADDW.

2. There is no 2nd operand, or the 2nd operand is not an operand address.

3. If saddrp is saddrp2 in this combination, there is a short code length instruction.

4. MULUW and DIVUX are the same as MULW.

**(3) 24-bit instructions (combinations expressed by writing WHL for rg are shown in parentheses)**

MOVG, ADDG, SUBG, INCG, DECG, PUSH, POP

**Table 25-3 List of Instructions by 24-Bit Addressing**

2nd Operand 1st Operand	# imm24	WHL	rg rg'	saddr	!!addr24	mem1	[%saddr]	SP	None*
WHL	(MOVG) (ADDG) (SUBG)	(MOVG) (ADDG) (SUBG)	(MOVG) (ADDG) (SUBG)	(MOVG) ADDG SUBG	(MOVG)	MOVG	MOVG	MOVG	
rg	MOVG ADDG SUBG	(MOVG) (ADDG) (SUBG)	MOVG ADDG SUBG	MOVG	MOVG				INCG DECG PUSH POP
saddr		(MOVG)	MOVG						
!!addr24		(MOVG)	MOVG						
mem1		MOVG							
[%saddr]		MOVG							
SP	MOVG	MOVG							INCG DECG

\* There is no 2nd operand, or the 2nd operand is not an operand address.

**(4) Bit manipulation instructions**

MOV1, AND1, OR1, XOR1, SET1, CLR1, NOT1, BT, BF, BTCLR, BFSET

**Table 25-4 List of Instructions by Bit Manipulation Instruction Addressing**

2nd Operand 1st Operand	CY	saddr. bit sfr. bit A.bit X. bit PSWL. bit PSWH. bit mem2. bit !addr16. bit !!addr24. bit	/saddr.bit /sfr. bit /A. bit /X. bit /PSWL. bit /PSWH. bit /mem2. bit /!addr16. bit /!!addr24. bit	None*
CY		MOV1 AND1 OR1 XOR1	AND1 OR1	NOT SET1 CLR1
saddr. bit sfr. bit A. bit X. bit PSWL. bit PSWH. bit mem2. bit !addr16. bit !!addr24. bit	MOV1			NOT1 SET1 CLR1 BF BT BTCLR BFSET

\* There is no 2nd operand, or the 2nd operand is not an operand address.



**(5) Call/return instructions / branch instructions**

CALL, CALLF, CALLT, BRK, RET, RETI, RETB, RETCS, RETCSB, BRKCS, BR, BNZ, BNE, BZ, BE, BNC, BNL, BC, BL, BN, BV, BPO, BV, BPE, BP, BN, BLT, BGE, BLE, BGT, BNH, BH, BF, BT, BTCLR, BFSET, DBNZ

**Table 25-5 List of Instructions by Call/Return Instruction / Branch Instruction Addressing**

Instruction Address Operand	\$addr20	!addr20	!addr16	!!addr20	rp	rg	[rp]	[rg]	!addr11	[addr5]	RBn	None
Basic instructions	BC* BR	CALL BR	CALL BR RETCS RETCSB	CALL BR	CALL BR	CALL BR	CALL BR	CALL BR	CALLF	CALLT	BRKCS	BRK RET RETI RETB
Compound instructions	BF BT BTCLR BFSET DBNZ											

\* BNZ, BNE, BZ, BE, BNC, BNL, BL, BN, BV, BPO, BV, BPE, BP, BN, BLT, BGE, BLE, BGT, BNH, and BH are the same as BC.

**(6) Other instructions**

ADJBA, ADJBS, CVTBW, LOCATION, SEL, NOT, EI, DI, SWRS

[MEMO]

## APPENDIX A DIFFERENCES BETWEEN $\mu$ PD784026 SUBSERIES AND $\mu$ PD78234 SUBSERIES

**Table A-1 Differences Between  $\mu$ PD784026 Subseries and  $\mu$ PD78234 Subseries (1/4)**

Item		Series Name	$\mu$ PD784026 Subseries	$\mu$ PD78234 Subseries
Number of basic instructions (mnemonics)			113	65
General-purpose registers			8 bits $\times$ 16 $\times$ 8 banks or 16 bits $\times$ 8 $\times$ 8 banks (memory mapping)	8 bits $\times$ 8 $\times$ 4 banks (memory mapping)
Minimum instruction execution time			160 ns (25 MHz operation)	333 ns (12 MHz operation)
Supply voltage			$V_{DD} = 2.7$ to $5.5$ V	$V_{DD} = +5$ V $\pm$ 10 %
Internal memory capacity	ROM		48 K/64 K bytes	16 K/32 K bytes
	RAM		2048 bytes	640/1024 bytes
Memory space			Total of 1 M bytes	<ul style="list-style-type: none"> <li>• Program : 64 K bytes</li> <li>• Data : 1 M bytes</li> </ul>
Clock generator	Clock frequency divider		1/2, 1/4, 1/8 or 1/16 frequency division selectable	1/2 frequency division only
I/O pins	Port 0 (P00 to P07)		Input/output	Input
	Port 4 (P40 to P47)		Input/output specified bit-wise	Input/output specified in 8-bit units
	Port 6 (P60 to P67)		Input/output (in $\mu$ PD784021, P60 to P63 are output-only)	P60 to P63 : Output-only P64 to P67 : Input/output
	Port 7 (P70 to P77)		Input/output	Input
Real-time output port	Output triggers		<ul style="list-style-type: none"> <li>• INTC10</li> <li>• INTC11</li> <li>• INTC10 or INTP0</li> <li>• INTP0</li> </ul>	<ul style="list-style-type: none"> <li>• INTC10</li> <li>• INTC11</li> <li>• INTC10 or INTP0</li> </ul>
Timer/counter	Count clock		$f_{xx}/8$ is fastest	$f_{CLK}/8$ is fastest
	External event counter		Possible with 3 timer/counters: 16-bit timer/counters 0 to 2	Possible with 8-bit timer/counter 2 only
	Number of counter bits		16-bit timer/counter $\times$ 1 8/16-bit switchable timer/counter $\times$ 3 8/16-bit switchable timer $\times$ 1	16-bit timer/counter $\times$ 1 8-bit timer/counter $\times$ 2 8-bit timer $\times$ 1
	Timer/counter 0 prescaler		Yes	No

Table A-1 Differences Between  $\mu$ PD784026 Subseries and  $\mu$ PD78234 Subseries (2/4)

Series Name		Item	μPD784026 Subseries	μPD78234 Subseries
Timer/counter	Timer/counter 1 capture registers	Dual-function capture/compare register and dedicated capture register Common capture trigger	Dual-function capture/compare register only	
	Timer/counter 2 capture registers	Dedicated capture register and dual-function capture/compare register Independent capture triggers input Dedicated capture register retains capture value after a read	Dedicated capture register only Dedicated capture register value is undefined after a read	
	Timer 3 interrupt	Independent interrupt	Shared with INTP4	
Watchdog timer		Provided	Not provided	
PWM output	Prescaler	Selectable from 1/1 to 1/4	Not provided	
A/D converter	INTAD	Independent	Shared with INTP5	
	INTAD in software start select mode	Generated	Not generated	
	Post-ADCR-read A/D conversion start mode	Provided	Not provided	
	Conversion speed	9.6 μs (fCLK = 12.5 MHz)	20 μs (fCLK = 6 MHz)	
D/A converter	Analog output enabling/disabling	Specifiable by DACEn bit	Not specifiable	
Asynchronous serial interface	Number of channels	2	1	
	3-wire serial I/O mode switching	Possible	Not possible	
	Reception completion interrupt disabling function in case of receive error	Provided	Not provided	
	Baud rate generator baud rate calculation expression	$\frac{f_{xx}}{2^{n+3} \cdot m}$ n = 0-11 m = 16-30	$\frac{f_{xx}}{2} \cdot \frac{1}{n} \cdot \frac{1}{2^m} \cdot \frac{1}{16}$ n = 1-15 m = 1-7	
	UART mode transmission enable bits (TXE0/TXE1)	Provided	Not provided	
	Start bit sampling clock	Baud rate × m clock (m = 16 to 30)	Baud rate × 16 clock	

Table A-1 Differences Between  $\mu$ PD784026 Subseries and  $\mu$ PD78234 Subseries (3/4)

Item		Series Name	$\mu$ PD784026 Subseries	$\mu$ PD78234 Subseries
Asynchronous serial interface	Baud rate generation by timer/counter 3		Not possible	Possible
Clocked serial interface 3-wire serial I/O	Dual-function mode		Asynchronous serial interface mode	SBI mode
	Transfer order		MSB-first/LSB-first switchable	MSB-first only
	Internal serial clock		$\frac{f_{xx}}{2} \cdot \frac{1}{2^{n+3}} \cdot \frac{1}{m}$ n = 0-11 m = 1, 16-30	TM3 output, $f_{CLK}/8$ , $f_{CLK}/16$ switchable
Clocked serial interface	Transfer order in 3-wire serial I/O mode		MSB-first/LSB-first switchable	MSB-first only
	SO0 pin level control in 3-wire serial I/O mode		Possible	Not possible
Clock output function			Provided	Not provided
Edge detection function	INTP4/INTP5 noise elimination method		Analog delay	Clock sampling
	INTP0 noise elimination sampling clock		Selectable	Fixed
Interrupt function	Interrupt sources		BRK & BRKCS instructions, operand error interrupt, and 23 hardware interrupts	BRK instruction and 19 hardware interrupts
	Context switching function		Provided	Not provided
	Multiprocessing priority		4 levels	2 levels
	Control register for each interrupt request source		Register provided for each interrupt request source	Registers provided on function-by-function basis
	Vector table & macro service control word address		Varies. Compare individual device user's manuals, etc., for details	
	Macro service function	SFR address when type A or type C is used	Programmable	Fixed for each interrupt request source
		Type C MPD & MPT decrementing	Possible	Not possible
		Type A, type B & type C timing data 16-bit data transfer	Possible	Not possible
		Operation when macro service counter reaches 0 selectable	Generation or non-generation of vectored interrupt request	Vectored interrupt request always generated
		Type B & type C macro service counter	Fixed 16-bit	8/16-bit switchable
		Counter mode	Provided	Not provided

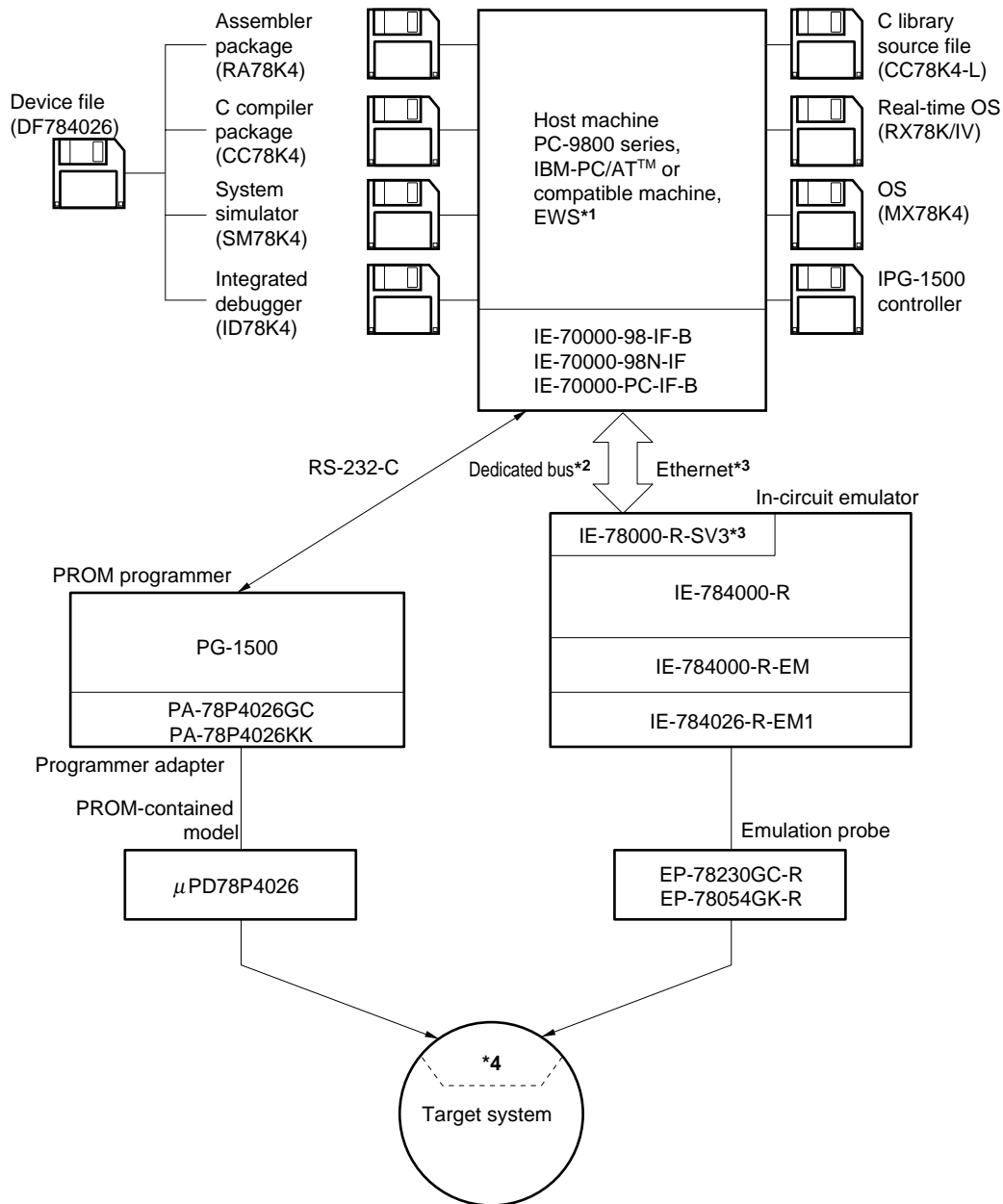
Table A-1 Differences Between  $\mu$ PD784026 Subseries and  $\mu$ PD78234 Subseries (4/4)

Series Name		$\mu$ PD784026 Subseries	$\mu$ PD78234 Subseries
Local bus interface function	External expansion function	Switchable from 256 bytes to 1M bytes in 7 stages	Switchable in 2 stages: 64 K bytes & 1 M bytes
	Programmable wait memory space division	8 divisions specifiable	2 divisions specifiable
	Address wait function	Provided	No
	Refresh operations & memory access operations	Performed simultaneously in specified space Otherwise performed mutually exclusively	Performed mutually exclusively
	Bus hold function	Provided	Not provided
Standby function	IDLE mode	Provided	Not provided
	STOP mode release signal	RESET & NMI input, and unmasked INTP4 & INTP5	RESET & NMI input
	HALT mode release by maskable interrupt	Released by unmasked interrupt	Released only by acknowledgeable interrupt
	Oscillation stabilization time in case of STOP mode release by NMI	Oscillation stabilization timer 16 to 19-bit count time 16 to 19 selectable	NMI active level time + oscillation stabilization timer 16-bit count time or oscillation stabilization timer 15-bit count time
Packages		<ul style="list-style-type: none"> <li>• 80-pin plastic QFP (14 × 14 mm)</li> <li>• 80-pin plastic TQFP (fine-pitch) (12 × 12 mm): <math>\mu</math>PD784021 only</li> <li>• 80-pin ceramic WQFN (14 × 14 mm) : <math>\mu</math>PD78P4026 only</li> </ul>	<ul style="list-style-type: none"> <li>• 80-pin plastic QFP (14 × 14 mm)</li> <li>• 94-pin plastic QFP (20 × 20 mm)</li> <li>• 84-pin plastic QFJ (1150 × 1150 mil)</li> <li>• 94-pin ceramic WQFN (14 × 14 mm) : <math>\mu</math>PD78P238 only</li> </ul>

## APPENDIX B DEVELOPMENT TOOLS

This appendix describes the tools required for the development of a system using a  $\mu$ PD784026 subseries product. A check sheet of the necessary tools for development are provided in Appendix B.6.

Figure B-1 Development Tool Configuration



\* 1. SM78K4, ID78K4, MX78K4, and PG-1500 controller are not supported.

2. When using PC-9800 series or IBM PC/AT as the host machine

3. When EWS is used as the host machine

4. Conversion socket (EV-9200GC-80) or conversion adapter (TGK-080SDW)

**Remark** This figure shows 3.5-inch FDs as the media in which software is supplied.



## B.1 LANGUAGE PROCESSING SOFTWARE (1/2)

RA78K4 assembler package	<p>This is a relocatable assembler that can be used in common with all 78K/IV series products by changing device file. It includes a macro function for greater development efficiency. A structured assembler is also provided which enables the program control structure to be clearly written, resulting in improved program productivity and maintainability.</p> <p>It is used in combination with the optionally available device file (DF784026).</p>	
	Structured Assembler Preprocessor (Program Name: ST78K4)	Converts a source program written in structured assembler assembly language to a form that can be input to the relocatable assembler.
	Relocatable Assembler (Program Name: RA78K4)	Program that converts a source program written in assembly language to machine language and generates a relocatable object module file.
	Linker (Program Name: LK78K4)	Links object module files generated by the relocatable assembler to a library file, determines program absolute addresses, and generates a load module file.
	Object Converter (Program Name: OC78K4)	Converts the load module file generated by the linker to a form that can be downloaded to an in-circuit emulator or PROM programmer.
	Librarian (Program Name: LB78K4)	Links object module files output by the relocatable assembler, and creates a single library file. Also performs library file updating.
	List Converter (Program Name: LCN78K4)	Creates an assembly list incorporating absolute values using the assembly list file, object module file and load module file output by the relocatable assembler.
Part number: $\mu$ SxxxxRA78K4		

## B.1 LANGUAGE PROCESSING SOFTWARE (2/2)

CC78K4 C compiler package	This C compiler can be used with all the products in the 78K/IV series by changing the device file. Its language specifications conform to ANSI, and the program can be written to ROM. This compiler has functions such as special function register manipulation function, bit manipulation function, variables using short direct addressing, and interrupt control function. By using these functions, programs can be written efficiently, and a higher object efficiency can be achieved. In addition, a sample program of the start-up routine and an object library of standard functions are also provided. Use this compiler in combination with the assembler package (RA78K4) and device file (DF784026) optionally available.
	Part number: $\mu S \times \times \times \times$ CC78K4
CC78K4-L C library source file	This is a source program of the library supplied with the C compiler package (CC78K4) and is necessary for modifying the library (to satisfy the user's specifications).
	Part number: $\mu S \times \times \times \times$ CC78K4-L
DF784026* device file	This file contains information peculiar to the device. It is used in combination with RA78K4, CC78K4, SM78K4, or ID78K4 optionally available.
	Part number: $\mu S \times \times \times \times$ DF784026

\* The DF784026 can be used with all of the RA78K4, CC78K4, SM78K4, and ID78K4.

**Remark**  $\times \times \times \times$  in the part number differs as follows, depending on the host machine and OS to be used.

$\mu S \times \times \times \times$ RA78K4

$\mu S \times \times \times \times$ CC78K4

$\mu S \times \times \times \times$ CC78K4-L

$\mu S \times \times \times \times$ DF784026

$\times \times \times \times$	Host Machine	OS	Supply Media
5A13	PC-9800 series	MS-DOS (Ver.3.30 to Ver.6.2*)	3.5" 2HD
5A10			5" 2HD
7B13	IBM PC/AT and compatible machines	Refer to <b>B.4.</b>	3.5" 2HC
7B10			5" 2HC
3P16	HP9000 series 700	HP-UX (rel.9.01)	Digital audio tape (DAT)
3K15	SPARCstation™	SunOS™ (rel.4.1.1)	Cartridge tape (QIC-24)
3R15	NEWS™	NEWS-OS™	Cartridge tape (QIC-24)

\* Ver. 5.0 or above of MS-DOS has a task swap function, but this function cannot be used with the above software.

## B.2 DEBUGGING TOOLS

## B.2.1 Hardware

IE-784000-R	The IE-784000-R is an in-circuit emulator that can be used with the entire 78K/IV series. It is used in combination with the optionally available IE-784000-R-EM and IE-784026-R-EM1. Debugging is performed with a host machine connected. The optionally available integrated debugger and device file are required, and in combination with these it is possible to perform debugging at the C language or structured assembly language source program level. More efficient debugging or program testing is possible by means of the C0 coverage function, etc. Connection to the host machine is performed by means of Ethernet or a dedicated bus, and a optionally available interface adapter is required.
IE-784000-R-EM	Emulation board for use with the entire 78K/IV series.
IE-784026-R-EM1	Board which emulates $\mu$ PD784026 subseries peripheral functions. Used in combination with the IE-784000-R.
IE-70000-98-IF-B	Interface adapter for use when a PC-9800 series model is used as the host machine.
IE-70000-98N-IF	Interface adapter and cable for use when a PC-9800 series notebook computer is used as the host machine.
IE-70000-PC-IF-B	Interface adapter for use when an IBM PC/AT model is used as the host machine.
IE-78000-R-SV3	This is an interface board when an EWS is used as the host machine and is connected to the board in the IE-784000-R. 10Base-5 is supported as Ethernet. For the other methods, a commercially available conversion adapter is necessary.
IE-784000-R-BK	This board is necessary for upgrading the in-circuit emulator for the 75X/XL series or 78K series to the IE-784000-R.
EP-78230GC-R	This is an emulation probe for the $\mu$ PD784026 subseries of 80-pin plastic QFP (GC-3B9 type). One 80-pin conversion socket, EV-9200GC-80, that facilitates development of the target system is provided.
EV-9200GC-80	This is a conversion socket that connects the PC board of the target system designed to mount an 80-pin plastic QFP (GC-3B9 type) and the EP-78230GC-R. Instead of the EP-78230GC-R, the $\mu$ PD78P4026KK-T (ceramic WQFN) can be also connected.
EP-78054GK-R	This is an emulation probe for the $\mu$ PD784026 subseries of 80-pin plastic TQFP (GK-BE9 type). One 80-pin conversion adapter, T GK-080SDW, that facilitates development of the target system is provided.
T GK-080SDW	This is a conversion adapter that connects the board of the target system designed to mount an 80-pin plastic TQFP (fine pitch) (GK-BE9 type) and the EP-78054GK-R.
EV-9900	This is a jig used to remove the $\mu$ PD78P4026KK-T from the EV-9200GC-80.

**Remarks 1.** Five EV-9200GC-80s are available as a set.

The EV-9500GK-80 is available separately.

- 2.** T GK-080SDW is a product of Tokyo Eletech Corporation (Tokyo (03) 5295-1161) and is available separately. To purchase this product, contact an NEC distributor.

## B.2.2 Software (1/2)

SM78K4 system simulator	<p>This simulator can debug the target system at C source level or assembler level while simulating the operation of the target system on the host machine.</p> <p>The SM78K4 runs on Windows.</p> <p>By using the SM78K4, the logic and performance of the application can be verified independently of the hardware without the IE-784000-R, enhancing development efficiency and improving the quality of the software.</p> <p>This system simulator is used in combination with a device file (DF784026) optionally available.</p>
	Part number: $\mu$ SxxxxSM78K4
ID78K4 integrated debugger	<p>This is a program that controls the IE-784000-R.</p> <p>With this program, efficient debugging can be performed by debugging the source program described in C language, structured assembly language, or assembly language, by dividing the screen of the host machine and displaying various information at the same time.</p> <p>This program is used in combination with a device file (DF784026) optionally available.</p>
	Part number: $\mu$ SxxxxID78K4
DF784026* device file	<p>This file contains information peculiar to the device.</p> <p>It is used in combination with SM78K4, ID78K4, RA78K4, or CC78K4 optionally available.</p>
	Part number: $\mu$ SxxxxDF784026

\* The DF784026 can be used with all of the RA78K4, CC78K4, SM78K4, and ID78K4.

**Remark** xxxx in the part number differs as follows, depending on the host machine and OS to be used.

$\mu$ SxxxxSM78K4

$\mu$ SxxxxID78K4

xxxx	Host Machine	OS	Supply Media
AA13	PC-9800 series	MS-DOS (Ver.3.30 to Ver.6.2*) + Windows (Ver.3.1)	3.5" 2HD
AB13	IBM PC/AT and compatible machines (Japanese Windows)	Refer to <b>B.4.</b>	3.5" 2HC
BB13	IBM PC/AT and compatible machines (English Windows)		

\* Ver. 5.0 or above of MS-DOS has a task swap function, but this function cannot be used with the above software.

**B.2.2 Software (2/2)** $\mu$ SxxxxDF784026

xxxx	Host Machine	OS	Supply Media
5A13	PC-9800 series	MS-DOS	3.5" 2HD
5A10		(Ver.3.30 to Ver.6.2*)	5" 2HD
7B13	IBM PC/AT and compatible machines	Refer to <b>B.4.</b>	3.5" 2HC
7B10			5" 2HC
3P16	HP9000 series 700	HP-UX(rel.9.01)	Digital audio tape (DAT)
3K15	SPARCstation	SunOS(rel.4.1.1)	Cartridge tape (QIC-24)

\* Ver. 5.0 or above of MS-DOS has a task swap function, but this function cannot be used with the above software.

**B.3 PROM WRITING TOOLS****(1) Hardware**

PG-1500	This PROM programmer can program single-chip microcontrollers containing PROM in a stand-alone mode or under the control of the host machine, when a board supplied as an accessory and an optionally available PROM programmer adapter are connected. It can also program representative PROMs, from 256K-bit to 4 M-bit models.
PA-78P4026GC PA-78P4026KK	This is a PROM programmer adapter for the $\mu$ PD78P4026 and is connected to the PG-1500. PA-78P4026GC-3B9 : for 80-pin plastic QFP (GC-3B9 type) PA-78P4026KK : for 80-pin ceramic WQFN (KK-T type)

**(2) Software**

PG-1500 controller	The PG-1500 controller connects the PG-1500 and the host machine with a serial and parallel interfaces to control the PG-1500 on the host machine.			
	Host Machine	OS	Supply Media	Order Code
	PC-9800 series	MS-DOS (Ver.3.30 to Ver.6.2*)	3.5" 2HD	$\mu$ S5A13PG1500
			5" 2HD	$\mu$ S5A10PG1500
	IBM PC/AT and compatible machines	Refer to <b>B.4.</b>	3.5" 2HD	$\mu$ S7B13PG1500
			5" 2HD	$\mu$ S7B10PG1500

\* Ver. 5.0 or above of MS-DOS has a task swap function, but this function cannot be used with the above software.

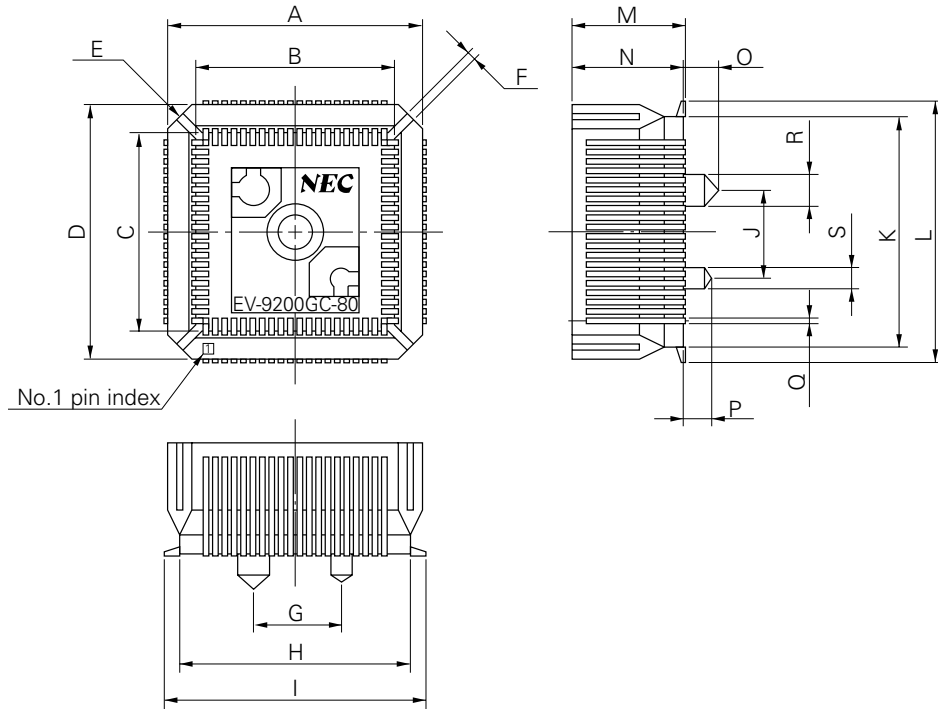
## B.4 OS OF IBM PC

The following OSs for the IBM PC are supported.

OS	Version
PC DOS	Ver.5.02 to Ver.6.3 J6.1/V <sup>*2</sup> to J6.3/V <sup>*2</sup>
Windows*1	Ver.3.1 to Ver.3.11
MS-DOS	Ver.5.0 to Ver.6.22 5.0/V <sup>*2</sup> to 6.2/V <sup>*2</sup>
IBM DOS™	J5.02/V <sup>*2</sup>

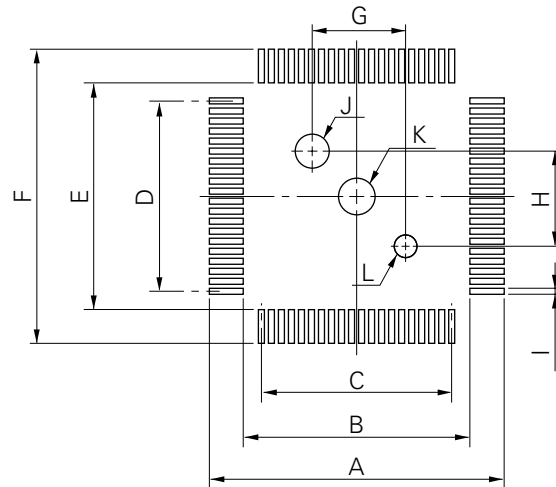
- \* 1. PC DOS and Windows are used in combination when the integrated debugger is used.
- 2. Only the English mode is supported.

**Caution** Ver. 5.0 or above of MS-DOS has a task swap function, but this function cannot be used with the above software.

**B.5 CONVERSION SOCKET (EV-9200GC-80) AND CONVERSION ADAPTER (TGK-080SDW)****(1) Dimensions of conversion socket (EV-9200GC-80) and recommended board mounting pattern****Figure B-2 Dimensions of EV-9200GC-80 (reference)**

EV-9200GC-80-G1E

ITEM	MILLIMETERS	INCHES
A	18.0	0.709
B	14.4	0.567
C	14.4	0.567
D	18.0	0.709
E	4-C 2.0	4-C 0.079
F	0.8	0.031
G	6.0	0.236
H	16.0	0.63
I	18.7	0.736
J	6.0	0.236
K	16.0	0.63
L	18.7	0.736
M	8.2	0.323
N	8.0	0.315
O	2.5	0.098
P	2.0	0.079
Q	0.35	0.014
R	ø2.3	ø0.091
S	ø1.5	ø0.059

**Figure B-3 Recommended Board Mounting Pattern of EV-9200GC-80 (reference)**

EV-9200GC-80-P1E

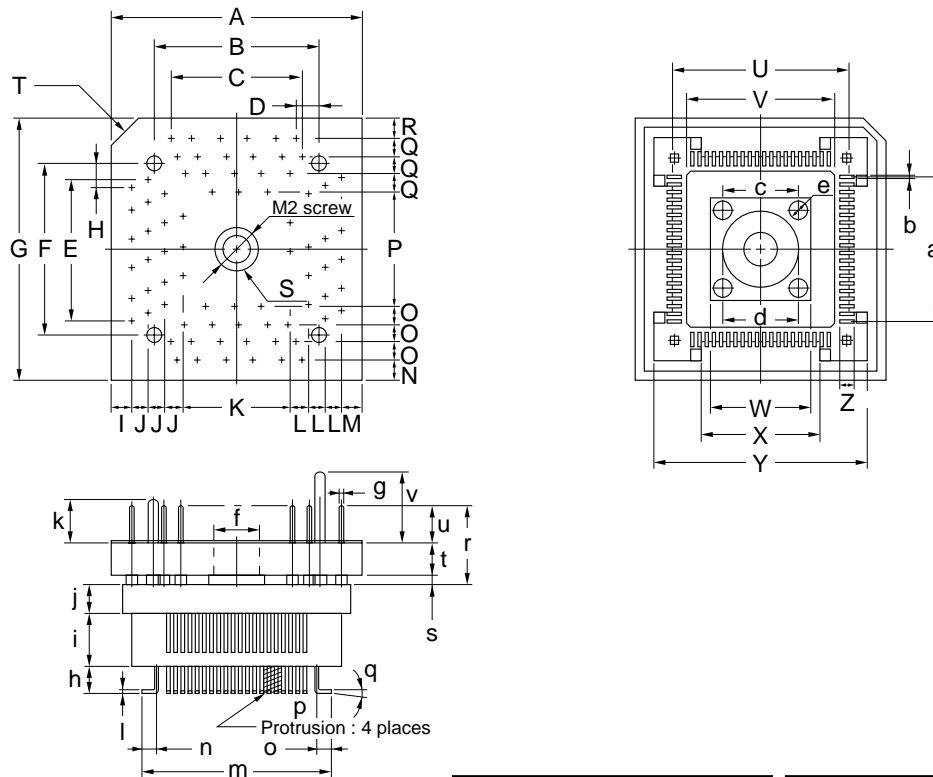
ITEM	MILLIMETERS	INCHES
A	19.7	0.776
B	15.0	0.591
C	$0.65 \pm 0.02 \times 19 = 12.35 \pm 0.05$	$0.026^{+0.001}_{-0.002} \times 0.748 = 0.486^{+0.003}_{-0.002}$
D	$0.65 \pm 0.02 \times 19 = 12.35 \pm 0.05$	$0.026^{+0.001}_{-0.002} \times 0.748 = 0.486^{+0.003}_{-0.002}$
E	15.0	0.591
F	19.7	0.776
G	$6.0 \pm 0.05$	$0.236^{+0.003}_{-0.002}$
H	$6.0 \pm 0.05$	$0.236^{+0.003}_{-0.002}$
I	$0.35 \pm 0.02$	$0.014^{+0.001}_{-0.001}$
J	$\phi 2.36 \pm 0.03$	$\phi 0.093^{+0.001}_{-0.002}$
K	$\phi 2.3$	$\phi 0.091$
L	$\phi 1.57 \pm 0.03$	$\phi 0.062^{+0.001}_{-0.002}$

**Caution** Dimensions of mount pad for EV-9200 and that for target device (QFP) may be different in some parts. For the recommended mount pad dimensions for QFP, refer to "SEMICONDUCTOR DEVICE MOUNTING TECHNOLOGY MANUAL" (C10535E).



## (2) Dimensions of conversion adapter (TGK-080SDW)

Figure B-4 Dimensions of TGK-080SDW (reference)



ITEM	MILLIMETERS	INCHES	ITEM	MILLIMETERS	INCHES
A	18.0	0.709	a	0.5x19=9.5±0.10	0.020x0.748=0.374±0.004
B	11.77	0.463	b	0.25	0.010
C	0.5x19=9.5	0.020x0.748=0.374	c	φ5.3	φ0.209
D	0.5	0.020	d	φ5.3	φ0.209
E	0.5x19=9.5	0.020x0.748=0.374	e	φ1.3	φ0.051
F	11.77	0.463	f	φ3.55	φ0.140
G	18.0	0.709	g	φ0.3	φ0.012
H	0.5	0.020	h	1.85±0.2	0.073±0.008
I	1.58	0.062	i	3.5	0.138
J	1.2	0.047	j	2.0	0.079
K	7.64	0.301	k	3.0	0.118
L	1.2	0.047	l	0.25	0.010
M	1.58	0.062	m	14.0	0.551
N	1.58	0.062	n	1.4±0.2	0.055±0.008
O	1.2	0.047	o	1.4±0.2	0.055±0.008
P	7.64	0.301	p	h=1.8 φ1.3	h=0.071 φ0.051
Q	1.2	0.047	q	0~5°	0.000~0.197°
R	1.58	0.062	r	5.9	0.232
S	φ3.55	φ0.140	s	0.8	0.031
T	C 2.0	C 0.079	t	2.4	0.094
U	12.31	0.485	u	2.7	0.106
V	10.17	0.400	v	3.9	0.154
W	6.8	0.268	TGK-080SDW-G1E		
X	8.24	0.324			
Y	14.8	0.583			
Z	1.4±0.2	0.055±0.008			

**note:** Product by TOKYO ELETECH CORPORATION.

**B.6 CHECK SHEET FOR  $\mu$ PD784026 SUBSERIES DEVELOPMENT TOOLS**

The following development tools are necessary for using the  $\mu$ PD784026 subseries products. Check if the necessary tools are at hand (the dotted line in the table below indicates either of the tools above or below the line should be selected).

- **Host machine: PC-9800 series**

Part Number	Check	Remark
IE-784000-R		
IE-784000-R-EM		
IE-784026-R-EM1		
IE-70000-98-IF-B (other than notebook type personal computer)		
IE-70000-98N-IF (for notebook type personal computer)		
EP-78230GC-R		
EP-78054GK-R* <sup>1</sup>		
EV-9200GC-80		
TGK-080SDW* <sup>1</sup>		
EV-9900 (necessary for using PROM version)* <sup>2</sup>		
PA-78P4026GC (necessary for using PROM version)		
PA-78P4026KK (necessary for using PROM version)		
$\mu$ SAA131D78K4 (3.5")		
$\mu$ S5A13DF784026 (3.5")		
$\mu$ S5A10DF784026 (5")		
$\mu$ S5A13RA78K4 (3.5")		
$\mu$ S5A10RA78K4 (5")		
$\mu$ S5A13CC78K4 (3.5")* <sup>3</sup>		
$\mu$ S5A10CC78K4 (5")* <sup>3</sup>		
$\mu$ S5A13CC78K4-L (3.5")* <sup>4</sup>		
$\mu$ S5A10CC78K4-L (5")* <sup>4</sup>		

\* 1. Selectable only when developing the  $\mu$ PD784021GK-BE9.

2. Jig used to remove WQFN from the EV-9200GC-80 (Use two of these jigs. Instead of this jig, tweezers can also be used.)

3. Necessary for using the C compiler.

4. Necessary for remodeling the library of the C compiler.

- Host machine: IBM PC/AT

Part Number	Check	Remark
IE-784000-R		
IE-784000-R-EM		
IE-784026-R-EM1		
IE-70000-PC-IF-B		
EP-78230GC-R		
EP-78054GK-R* <sup>1</sup>		
EV-9200GC-R		
TGK-080SDW* <sup>1</sup>		
EV-9900 (necessary for using PROM version)* <sup>2</sup>		
PA-78P4026GC (necessary for using PROM version)		
PA-78P4026KK (necessary for using PROM version)		
μSBB13ID78K4 (3.5") (English version)		
μSAB13ID78K4 (3.5") (Japanese version)		
μS5A13DF784026 (3.5")		
μS5A10DF784026 (5")		
μS5A13RA78K4 (3.5")		
μS5A10RA78K4 (5")		
μS5A13CC78K4 (3.5")* <sup>3</sup>		
μS5A10CC78K4 (5")* <sup>3</sup>		
μS5A13CC78K4-L (3.5")* <sup>4</sup>		
μS5A10CC78K4-L (5")* <sup>4</sup>		

\* 1. Selectable only when developing the μPD784021GK-BE9.

2. Jig used to remove WQFN from the EV-9200GC-80 (Use two of these jigs. Instead of these jigs, tweezers can also be used.)

3. Necessary for using the C compiler.

4. Necessary for remodeling the library of the C compiler.

[MEMO]

## APPENDIX C SOFTWARE FOR EMBEDDING

### C.1 REAL-TIME OS (1/2)

RX78K/IV real-time OS	<p>The RX78K/IV is intended to realize multi-task environments in areas where real-time features are required.</p> <p>This OS can improve the performance of the entire system by allocating the idle time of the CPU to the other processing.</p> <p>The RX78K/IV supplies 31 system calls conforming to the <math>\mu</math>ITRON specifications. It also supplies a tool (configurator) that is used to create the nucleus of the RX78K/IV and several information tables. This OS is used in combination with the assembler package (RA78K4) and device file (DF784026) optionally available.</p> <p>Part number: <math>\mu</math>SxxxxRX78K4-<math>\Delta\Delta\Delta\Delta</math></p>
--------------------------	---

**Caution** When purchasing the RX78K/IV, fill out the necessary forms and conclude a contract with NEC.

**Remark** xxxx and  $\Delta\Delta\Delta\Delta$  in the part number differs as follows, depending on the host machine and OS to be used.

$\mu$ SxxxxRX78K4- $\Delta\Delta\Delta\Delta$

$\Delta\Delta\Delta\Delta$	Product Outline	Upper Limit of Quantity for Mass Production
001	Object code for evaluation	Do not use this for mass production.
100K	Object code for mass production	0.1 million units
001M		1 million units
010M		10 million units
S01	Source program	Source program of object code for mass production

xxxx	Host Machine	OS	Supply Media
5A13	PC-9800 series	MS-DOS (Ver.3.30 to Ver.6.2*)	3.5" 2HD
5A10			5" 2HD
7B13	IBM PC/AT and compatible machines	Refer to <b>B.4.</b>	3.5" 2HC
7B10			5" 2HC
3P16	HP9000 series 700	HP-UX (rel.9.01)	Digital audio tape (DAT)
3K15	SPARCstation	SunOS (rel.4.1.1)	Cartridge tape (QIC-24)

\* Ver. 5.0 or above of MS-DOS has a task swap function, but this function cannot be used with the above software.

**C.1 REAL-TIME OS (2/2)**

MX78K4 - OS	This is an OS of the $\mu$ ITRON specification subset. The nucleus of the MX78K4 is supplied. This OS manages tasks, events, and time. Task management is to control the execution sequence of tasks and select the task to be executed next.
	Part number: $\mu$ SxxxxMX78K4- $\Delta\Delta\Delta$

**Remark** xxxx and  $\Delta\Delta\Delta$  in the part number differs as follows, depending on the host machine and OS to be used.

$\mu$ SxxxxMX78K4- $\Delta\Delta\Delta$

$\Delta\Delta\Delta$	Product Outline	Note
001	Object for evaluation	Use this for trial production.
xx	Object for mass production	Use this for mass production.
S01	Source program	Can be purchased only when object code for mass production is purchased.

xxxx	Host Machine	OS	Supply Media
5A13	PC-9800 series	MS-DOS (Ver.3.30 to Ver.6.2*)	3.5" 2HD
5A10			5" 2HD
7B13	IBM PC/AT and compatible machines	Refer to <b>B.4.</b>	3.5" 2HC
7B10			5" 2HC

\* Ver. 5.0 or above of MS-DOS has a task swap function, but this function cannot be used with the above software.

## APPENDIX D REGISTER INDEX

### [A]

ADCR	: A/D Conversion Result Register .....	376
ADIC	: Interrupt Control Register .....	504
ADM	: A/D Converter Mode Register .....	378
ASIM	: Asynchronous Serial Interface Mode Register .....	407
ASIM2	: Asynchronous Serial Interface Mode Register 2 .....	407
ASIS	: Asynchronous Serial Interface Status Register .....	409
ASIS2	: Asynchronous Serial Interface Status Register 2 .....	409

### [B]

BRGC	: Baud Rate Generator Control Register .....	427
BRGC2	: Baud Rate Generator Control Register 2 .....	427

### [C]

CIC00	: Interrupt Control Register .....	503
CIC01	: Interrupt Control Register .....	503
CIC10	: Interrupt Control Register .....	503
CIC11	: Interrupt Control Register .....	503
CIC20	: Interrupt Control Register .....	504
CIC21	: Interrupt Control Register .....	504
CIC30	: Interrupt Control Register .....	504
CLOM	: Clock Output Mode Register .....	481
CR00	: Compare Register .....	176
CR01	: Compare Register .....	176
CR02	: Capture Register .....	176
CR10/CR10W	: Compare Register .....	237
CR11/CR11W	: Capture/Compare Register .....	237
CR20/CR20W	: Compare Register .....	275
CR21/CR21W	: Capture/Compare Register .....	275
CR30/CR30W	: Compare Register .....	345
CRC0	: Capture/Compare Control Register 0 .....	180
CRC1	: Capture/Compare Control Register 1 .....	241
CRC2	: Capture/Compare Control Register 2 .....	279
CSIIC	: Interrupt Control Register .....	505
CSIIC1	: Interrupt Control Register .....	505
CSIIC2	: Interrupt Control Register .....	505
CSIM	: Clocked Serial Interface Mode Register .....	438, 454
CSIM1	: Clocked Serial Interface Mode Register 1 .....	418
CSIM2	: Clocked Serial Interface Mode Register 2 .....	418

### [D]

DACS0	: D/A Conversion Value Setting Register 0 .....	396
DACS1	: D/A Conversion Value Setting Register 1 .....	396
DAM	: D/A Converter Mode Register .....	397

[H]		
HLDM	: Hold Mode Register .....	604
[I]		
IMC	: Interrupt Mode Control Register .....	509
IMS	: Internal Memory Size Switching Register .....	49
INTM0	: External Interrupt Mode Register 0 .....	486
INTM1	: External Interrupt Mode Register 1 .....	487
ISPR	: In-Service Priority Register .....	508
[M]		
MK0H	: Interrupt Mask Register H .....	506
HK0L	: Interrupt Mask Register L .....	506
MK1L	: Interrupt Mask Register 1L .....	506
MM	: Memory Extension Mode Register .....	574, 585
[O]		
OSPC	: One-Shot Pulse Output Control Register .....	182
OSTS	: Oscillation Stabilization Time Specification Register .....	77, 614
[P]		
P0	: Port 0 .....	87
P0H	: Port 0 Buffer Register H .....	160
P0L	: Port 0 Buffer Register L .....	160
P1	: Port 1 .....	95
P2	: Port 2 .....	108
P3	: Port 3 .....	114
P4	: Port 4 .....	125
P5	: Port 5 .....	132
P6	: Port 6 .....	140
P7	: Port 7 .....	150
PIC0	: Interrupt Control Register .....	503
PIC1	: Interrupt Control Register .....	503
PIC2	: Interrupt Control Register .....	503
PIC3	: Interrupt Control Register .....	503
PIC4	: Interrupt Control Register .....	504
PIC5	: Interrupt Control Register .....	504
PM0	: Port 0 Mode Register .....	88
PM1	: Port 1 Mode Register .....	99
PM3	: Port 3 Mode Register .....	118
PM4	: Port 4 Mode Register .....	126
PM5	: Port 5 Mode Register .....	133
PM6	: Port 6 Mode Register .....	146
PM7	: Port 7 Mode Register .....	151
PMC1	: Port 1 Mode Control Register .....	100
PMC3	: Port 3 Mode Control Register .....	119
PRM0	: Prescaler Mode Register 0 .....	179, 347
PRM1	: Prescaler Mode Register 1 .....	240, 278



PSWL	: Program Status Word .....	511
PUO	: Pull-Up Resistor Option Register .....	91, 104, 110, 123, 129, 136, 148
PWC1	: Programmable Wait Control Register 1 .....	587
PWC2	: Programmable Wait Control Register 2 .....	587
PWM0	: PWM Modulo Register 0 .....	368
PWM1	: PWM Modulo Register 1 .....	368
PWMC	: PWM Control Register .....	367
PWPR	: PWM Prescaler Register .....	368
[R]		
RFA	: Refresh Area Specification Register .....	600
RFM	: Refresh Mode Register .....	599
RTPC	: Real-Time Output Port Control Register .....	159
RXB	: Serial Receive Buffer .....	405
RXB2	: Serial Receive Buffer 2 .....	405
[S]		
SBIC	: Serial Bus Interface Control Register .....	439, 440
SCS0	: Sampling Clock Selection Register .....	488
SERIC	: Interrupt Control Register .....	504
SERIC2	: Interrupt Control Register .....	505
SIO	: Serial Shift Register .....	437, 457
SIO1	: Serial Shift Register 1 .....	417
SIO2	: Serial Shift Register 2 .....	417
SRIC	: Interrupt Control Register .....	504
SRIC2	: Interrupt Control Register .....	505
STBC	: Standby Control Register .....	76, 612
STIC	: Interrupt Control Register .....	505
STIC2	: Interrupt Control Register .....	505
[T]		
TM0	: Timer Register 0 .....	176
TM1/TM1W	: Timer Register 1 .....	237
TM2/TM2W	: Timer Register 2 .....	275
TM3/TM3W	: Timer Register 3 .....	345
TMC0	: Timer Control Register 0 .....	178, 346
TMC1	: Timer Control Register 1 .....	239, 277
TOC	: Timer Output Control Register .....	181, 280
TXS	: Serial Transmit Shift Register .....	405
TXS2	: Serial Transmit Shift Register 2 .....	405
[W]		
WDM	: Watchdog Timer Mode Register .....	361, 510

[MEMO]

## APPENDIX E REVISION HISTORY

The following lists the revision history. The “Applied to” column shows the chapters in each edition.

(1/4)

Edition	Contents Revised from the Previous Edition	Applied to
2nd edition	<ul style="list-style-type: none"> <li>• Addition of description on <math>\mu</math>PD784020</li> <li>• Addition of <math>\mu</math>PD784021GK-BE9 package</li> </ul>	Throughout
	Introduction of $\mu$ PD784038, 784038Y, 784046, 784208, and 784208Y subseries in 78K/IV Series Product Development Diagram	78K/IV Series Product Development Diagram
	Change of W (write only) of following addresses to R/W (read/write) in <b>Table 3-5 List of Special Function Registers (SFRs)</b> in <b>CHAPTER 3 CPU ARCHITECTURE</b> 0FF30H, 0FF32H, 0FF33H, 0FF5CH, 0FF5EH, 0FF90H, 0FF91H, 0FFCEH, 0FFFCH	CHAPTER 3 CPU ARCHITECTURE
	<b>CHAPTER 4 CLOCK GENERATOR</b> <ul style="list-style-type: none"> <li>• Dividing <b>4.3.1 Clock Oscillator</b> into <b>(1) When using crystal/ceramic oscillation</b> and <b>(2) When using external clock</b></li> <li>• Addition of <b>Table 4-1 Time Required to Change Division Ratio</b> in <b>4.3.2 Divider</b></li> </ul>	CHAPTER 4 CLOCK GENERATOR
	Dividing <b>CHAPTER 7 TIMER/COUNTER UNIT</b> into following five chapters for more detailed explanation and clear operation <ul style="list-style-type: none"> <li>• <b>CHAPTER 7 OUTLINE OF TIMER/COUNTER</b></li> <li>• <b>CHAPTER 8 TIMER/COUNTER 0</b> <ul style="list-style-type: none"> <li>* Addition of description on selector</li> </ul> </li> <li>• <b>CHAPTER 9 TIMER/COUNTER 1</b> <ul style="list-style-type: none"> <li>* Change of description in note (*) of <b>Table 9-2 Timer/Counter 1 Pulse Measurement Range</b></li> <li>* Addition of description on selector</li> <li>* Addition of <b>Table 9-4 Maximum Input Frequency and Minimum Input Pulse Width That Can be Counted as Events</b></li> </ul> </li> <li>• <b>CHAPTER 10 TIMER/COUNTER 2</b> <ul style="list-style-type: none"> <li>* Addition of description on selector</li> <li>* Change of <b>Figure 10-18 Capture Operation in 8-Bit Operating Mode</b></li> <li>* Change of <b>Figure 10-19 Capture Operation in 16-Bit Operating Mode</b></li> </ul> </li> <li>• <b>CHAPTER 11 TIMER 3</b> <ul style="list-style-type: none"> <li>* Change of <b>Figure 11-1 Timer 3 Block Diagram</b></li> <li>* Addition of description on selector</li> </ul> </li> </ul>	CHAPTERS 7, 8, 9, 10, 11
	<b>CHAPTER 12 WATCHDOG TIMER FUNCTION</b> <ul style="list-style-type: none"> <li>• Change of <b>Figure 12-1 Watchdog Timer Block Diagram</b></li> <li>• Change of contents of WD12 and WD11 in <b>Figure 12-2 Watchdog Timer Mode Register (WDM) Format</b></li> <li>• Change of <b>12.3.1 Count Operation</b></li> </ul>	CHAPTER 12 WATCHDOG TIMER FUNCTION
	Addition of <b>14.4 EXTERNAL CIRCUIT OF A/D CONVERTER</b> to <b>CHAPTER 14 A/D CONVERTER</b>	CHAPTER 14 A/D CONVERTER
	Addition of <b>Figure 15-3 Example of Connecting Capacitors to Reference Voltage Input Pins of D/A Converter</b> in <b>15.3.1 Basic Operation</b> of <b>CHAPTER 15 D/A CONVERTER</b>	CHAPTER 15 D/A CONVERTER

(2/4)

Edition	Contents Revised from the Previous Edition	Applied to
2nd edition	<p>The following changes and additions in <b>CHAPTER 20 INTERRUPT FUNCTIONS</b></p> <ul style="list-style-type: none"> <li>• Change of (c) in <b>Figure 20-9 Non-Maskable Interrupt Request Acknowledgment Operations</b></li> <li>• Change of ISPR value in <b>Table 20-5 Multiple Interrupt Servicing</b></li> <li>• Addition of (4) <b>Counter mode</b> to <b>20.8.2 Types of Macro Service</b></li> <li>• Change of <b>Figure 20-21 Macro Service Mode Register Format</b></li> <li>• Addition of <b>20.8.9 Counter Mode</b></li> <li>• Addition of <b>20.11.1 Interrupt Acknowledgment Processing Time</b></li> <li>• Addition of <b>20.11.2 Processing Time of Macro Service</b></li> </ul>	CHAPTER 20 INTERRUPT FUNCTIONS
	<p><b>25.2 LIST OF OPERATIONS</b> in <b>CHAPTER 25 INSTRUCTION OPERATIONS</b></p> <ul style="list-style-type: none"> <li>• Addition of following bit manipulation instructions <ul style="list-style-type: none"> <li>MOV1 CY, !addr16. bit</li> <li>!addr16. bit, CY</li> <li>!addr24. bit, CY</li> <li>AND1, OR1</li> <li>CY, !addr16. bit</li> <li>CY, /!addr16. bit</li> <li>CY, /!addr24. bit</li> <li>XOR1 CY, !addr16. bit</li> <li>NOT1, SET1, CLR1</li> <li>!addr16. bit</li> </ul> </li> <li>• Change of contents of operation of call return instruction BRK, RETI, RETB</li> <li>• Addition of following conditional branch instructions BF, BT, BFSET, BTCLR</li> <li>!addr16. bit, \$addr20</li> </ul>	CHAPTER 25 INSTRUCTION OPERATIONS
	<p>Change of <b>APPENDIX B DEVELOPMENT TOOLS</b></p> <ul style="list-style-type: none"> <li>• Addition of system simulator and integrated debugger to <b>B.2.2 Software</b></li> <li>• Addition of <b>B.5 CONVERSION SOCKET (EV-9200GC-80) AND CONVERSION ADAPTER (EV-9500GK-80)</b></li> <li>• Addition of <b>B.6 CHECK SHEET FOR <math>\mu</math>PD784026 SUBSERIES DEVELOPMENT TOOLS</b></li> </ul>	APPENDIX B DEVELOPMENT TOOLS
	<p>Addition of MX78K4 to <b>APPENDIX C SOFTWARE FOR EMBEDDING</b></p>	APPENDIX C SOFTWARE FOR EMBEDDING

(3/4)

Edition	Contents Revised from the Previous Edition	Applied to
3rd edition	Change of <b>78K/IV SERIES PRODUCT DEVELOPMENT DIAGRAM</b>	CHAPTER 1 GENERAL
	<ul style="list-style-type: none"> <li>• Correction of <b>Figure 3-1 <math>\mu</math>PD784020 Memory Map</b></li> <li>• Addition of caution on prohibiting external wait setting for internal ROM area</li> <li>• Addition of note on reading timer register</li> </ul>	CHAPTER 3 CPU ARCHITECTURE
	<ul style="list-style-type: none"> <li>• Addition of caution on reading timer register</li> <li>• <b>8.5 EXTERNAL EVENT COUNTER FUNCTION</b> Correction of timing of TM0 in <b>Figure 8-10 Timer/Counter 0 External Event Count Timing</b></li> <li>• <b>8.7.2 Toggle Output</b> Change from “the output level at the time it was stopped is retained” to “the inactive level (<math>\overline{ALVn}</math>: <math>n = 0, 1</math>) is output” when timer/counter 0 is stopped by timer control register 0</li> <li>• <b>8.7.3 PWM Output (3) Stopping PWM output</b> Change from “the output level at the time it was stopped is retained” to “the active level is output” when timer/counter 0 is stopped by timer control register 0</li> </ul>	CHAPTER 8 TIMER/COUNTER 0
	<ul style="list-style-type: none"> <li>• Addition of caution on reading timer register</li> <li>• Correction of timing of TM1 in <b>Figure 9-10 Timing of External Event Count by Timer/Counter 1</b></li> </ul>	CHAPTER 9 TIMER/ COUNTER 1
	<ul style="list-style-type: none"> <li>• <b>10.2 CONFIGURATION</b> Addition of caution on reading timer register to <b>(1) Timer register 2 (TM2/TM2W)</b> Deletion of “Also, the count value can be cleared by a match (CR21/CR21W)” from description in <b>(2) Compare register (CR20/CR20W)</b> Addition of “Also, TM2/TM2W can be cleared after capture operation” to <b>(b) When specified as capture register in (3) Capture/compare register (CR21/CR21W)</b></li> <li>• <b>10.5 EXTERNAL EVENT COUNTER FUNCTION</b> Correction of timing of TM2 in <b>Figure 10-11 Timer/Counter 2 External Event Count Timing</b></li> <li>• <b>10.8.2 Toggle Output</b> Change from “the output level at the time it was stopped is retained” to “the inactive level (<math>\overline{ALVn}</math>: <math>n = 0, 1</math>) is output” when timer/counter 2 is stopped by timer control register 1</li> <li>• <b>10.8.3 PWM Output (3) Stopping PWM output</b> Change from “the output level at the time it was stopped is retained” to “the active level is output” when timer/counter 2 is stopped by timer control register 1</li> </ul>	CHAPTER 10 TIMER/COUNTER 2
	Addition of caution on reading timer register	CHAPTER 11 TIMER 3
	Change of “If the STOP mode or IDLE mode is entered as the result of an inadvertent program loop” to “If the STOP mode, <u>HALT mode</u> , or IDLE mode is entered as the result of an inadvertent program loop” in <5> of (2) in <b>12.4.1 General Cautions on Use of Watchdog Timer</b>	CHAPTER 12 WATCHDOG TIMER FUNCTION
	Low-speed conversion ( $f_{CLK} = 12.5 \text{ MHz}$ ) Change from $240/f_{CLK}$ ( $19 \mu\text{s}$ ) to $180/f_{CLK}$ ( $14.4 \mu\text{s}$ )	CHAPTER 14 A/D CONVERTER
	<ul style="list-style-type: none"> <li>• Addition of note on disabling reception completion interrupt in case of receive error and wait time calculation method</li> <li>• Addition of caution on selecting MSB/LSB first</li> </ul>	CHAPTER 16 ASYNCHRONOUS SERIAL INTER- FACE/3-WIRE SERIAL I/O
	Unification from CLO to CLKOUT pin	CHAPTER 18 CLOCK OUTPUT FUNCTION
	Addition of instructions to <b>20.9 WHEN INTERRUPT REQUESTS AND MACRO SERVICE ARE TEMPORARILY HELD PENDING</b>	CHAPTER 20 INTERRUPT FUNCTIONS

(4/4)

Edition	Contents Revised from the Previous Edition	Applied to
3rd edition	<ul style="list-style-type: none"> <li>• <b>21.2.1 Wait Function Control Registers and 21.2.3 Access Waits</b> Addition of cautions on disabling external wait setting to internal ROM area Change of formats in <b>Figure 21-9 Programmable Wait Control Registers (PWC1/PWC2) Format</b></li> <li>• <b>21.2.2 Address Waits and 21.3 PSEUDO-STATIC RAM REFRESH FUNCTION</b> Deletion of cautions on “using pseudo-static RAM refresh function is prohibited while the address wait function is used, and use of the address wait function is prohibited when the pseudo-static RAM refresh function is used”</li> <li>• <b>21.4.1 Hold Mode Register (HLDLM)</b> Change of <b>Figure 21-20 Hold Mode Register (HLDLM) Format</b></li> <li>• <b>21.4.2 Operation</b> Correction and addition of description of operation when hold mode is set while instruction not using local bus interface is executed</li> </ul>	CHAPTER 21 LOCAL BUS INTERFACE FUNCTION
	<ul style="list-style-type: none"> <li>• Change and addition from “The watchdog timer must not be used to release the standby mode (STOP or IDLE mode).” to “The watchdog timer must not be used to release the standby mode (STOP/<u>HALT</u>/IDLE mode).”</li> <li>• Addition of note on operation after release of standby mode to <b>22.2.1 Standby Control Register (STBC)</b></li> <li>• <b>22.3.1 HALT Mode Setting and Operation States</b> Addition of note on operation after release of HALT mode Addition of <b>Figure 22-5 Operation After of HALT Mode Has Been Released</b></li> <li>• <b>22.4.1 STOP Mode Setting and Operation States</b> Addition of note on operation after release of STOP mode Addition of <b>Figure 22-6 Operation After STOP Mode Has Been Released</b></li> <li>• <b>22.5.1 IDLE Mode Setting and Operation States</b> Addition of note on operation after release of IDLE mode Addition of <b>Figure 22-9 Operation After IDLE Mode Has Been Released</b></li> </ul>	CHAPTER 22 STANDBY FUNCTION
	Change from EV-9500GK-80 to TKG080SDW (Tokyo Eletech Corp.)	APPENDIX B DEVELOPMENT TOOLS

## 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 Corporation  
Semiconductor Solution Engineering Division  
Technical Information Support Dept.  
Fax: 044-548-7900

**South America**

NEC do Brasil S.A.  
Fax: +55-11-889-1689

**Taiwan**

NEC Electronics Taiwan Ltd.  
Fax: 02-719-5951

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"/>