

**NEC**

**User's Manual**

# **$\mu$ PD784976A Subseries**

**16-Bit Single-Chip Microcontroller**

**Hardware**

---

**$\mu$ PD784975A**

**$\mu$ PD78F4976A**

Document No. U15017EJ2V0UD00 (2nd edition)  
Date Published March 2002 N CP(K)

© NEC Corporation 2000, 2002  
Printed in Japan

[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.

**IEBus and FIP are trademarks of NEC Corporation.**

**Windows and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.**

**PC/AT is a trademark of IBM Corporation.**

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

**SPARCstation is a trademark of SPARC International, Inc.**

**Solaris and SunOS are trademarks of Sun Microsystems, Inc.**

**OSF/Motif is a trademark of Open Software Foundation, Inc.**

**TRON is an abbreviation of The Real-time 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.

License not needed:  $\mu$ PD78F4976AGF-3BA

The customer must judge the need for license:  $\mu$ PD784975AGF-xxx-3BA

• **The information in this document is current as of February, 2002. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC's data sheets or data books, etc., for the most up-to-date specifications of NEC semiconductor products. Not all products and/or types are available in every country. Please check with an NEC sales representative for availability and additional information.**

- No part of this document may be copied or reproduced in any form or by any means without prior written consent of NEC. NEC assumes no responsibility for any errors that may appear in this document.
- NEC 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 NEC semiconductor products listed in this document or any other liability arising from the use of such products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC or others.
- Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of customer's equipment shall be done under the full responsibility of customer. NEC assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.
- While NEC endeavours to enhance the quality, reliability and safety of NEC semiconductor products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC semiconductor products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment, and anti-failure features.
- NEC semiconductor products are classified into the following three quality grades:  
"Standard", "Special" and "Specific". The "Specific" quality grade applies only to semiconductor products developed based on a customer-designated "quality assurance program" for a specific application. The recommended applications of a semiconductor product depend on its quality grade, as indicated below. Customers must check the quality grade of each semiconductor product 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": Aircraft, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems and medical equipment for life support, etc.

The quality grade of NEC semiconductor products is "Standard" unless otherwise expressly specified in NEC's data sheets or data books, etc. If customers wish to use NEC semiconductor products in applications not intended by NEC, they must contact an NEC sales representative in advance to determine NEC's willingness to support a given application.

(Note)

(1) "NEC" as used in this statement means NEC Corporation and also includes its majority-owned subsidiaries.

(2) "NEC semiconductor products" means any semiconductor product developed or manufactured by or for NEC (as defined above).

M8E 00.4

# Regional Information

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

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

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

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

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

## **NEC do Brasil S.A.**

Electron Devices Division  
Guarulhos-SP, Brasil  
Tel: 11-6462-6810  
Fax: 11-6462-6829

## **NEC Electronics (Europe) GmbH**

Duesseldorf, Germany  
Tel: 0211-65 03 01  
Fax: 0211-65 03 327

- Branch The Netherlands  
Eindhoven, The Netherlands  
Tel: 040-244 58 45  
Fax: 040-244 45 80

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

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

Vélizy-Villacoublay, France  
Tel: 01-3067-58-00  
Fax: 01-3067-58-99

## **NEC Electronics (France) S.A. Representación en España**

Madrid, Spain  
Tel: 091-504-27-87  
Fax: 091-504-28-60

## **NEC Electronics Italiana S.R.L.**

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

## **NEC Electronics (UK) Ltd.**

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

## **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 Shanghai, Ltd.**

Shanghai, P.R. China  
Tel: 021-6841-1138  
Fax: 021-6841-1137

## **NEC Electronics Taiwan Ltd.**

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

## **NEC Electronics Singapore Pte. Ltd.**

Novena Square, Singapore  
Tel: 253-8311  
Fax: 250-3583

J02.3

## Major Revisions in This Edition

Page	Description
p.24 p.26	<b>CHAPTER 1 GENERAL</b> <ul style="list-style-type: none"> <li>• Modification of <b>78K/IV Series Lineup</b></li> <li>• Modification of <b>Caution</b> and <b>Remark</b> in <b>1.4 Pin Configuration (Top View)</b></li> </ul>
p.39 p.41	<b>CHAPTER 2 PIN FUNCTIONS</b> <ul style="list-style-type: none"> <li>• Modification of description in <b>2.2.13 AV<sub>DD</sub></b></li> <li>• Modification of <b>Table 2-1 Types of Pin I/O Circuits and Recommended Connection of Unused Pins</b></li> </ul>
p.73	<b>CHAPTER 3 CPU ARCHITECTURE</b> <ul style="list-style-type: none"> <li>• Addition of interrupt mask register 1L to <b>Table 3-6 Special Function Register (SFR) List</b></li> </ul>
p.95 p.99	<b>CHAPTER 4 PORT FUNCTIONS</b> <ul style="list-style-type: none"> <li>• Correction of <b>Table 4-3 Port Mode Register and Output Latch Setting When Alternate Function Is Used</b></li> <li>• Modification of description in <b>4.5 Selecting Mask Option</b></li> </ul>
p.176 p.185 pp.186, 187	<b>CHAPTER 11 A/D CONVERTER</b> <ul style="list-style-type: none"> <li>• Modification of description in <b>(8) AV<sub>DD</sub> pin</b> in <b>11.2 Configuration of A/D Converter</b></li> <li>• Modification of <b>Figure 11-9 Timing of A/D Conversion End Interrupt Request Generation</b></li> <li>• <b>11.5 Notes on A/D Converter</b> Addition of <b>(10) Timing that makes A/D conversion result undefined</b> and <b>(11) Cautions on board design</b> and modification of description in <b>(12) Reading A/D conversion result register (ADCR)</b></li> </ul>
p.192 p.193 p.196	<b>CHAPTER 12 SERIAL INTERFACE</b> <ul style="list-style-type: none"> <li>• Modification of <b>Remark</b> in <b>Figure 12-4 Format of Serial Operation Mode Register 2</b></li> <li>• Addition of <b>Table 12-2 Serial Interface Operation Mode Settings</b></li> <li>• Modification of <b>Remark</b> in <b>12.4.2 3-wire serial I/O mode (b) Format of serial operation mode register 2</b></li> </ul>
p.207	<b>CHAPTER 13 ASYNCHRONOUS SERIAL INTERFACE</b> <ul style="list-style-type: none"> <li>• Addition of <b>Table 13-3 Serial Interface Operation Mode Settings</b></li> </ul>
p.241 p.246 p.247	<b>CHAPTER 16 INTERRUPT FUNCTION</b> <ul style="list-style-type: none"> <li>• Modification of <b>Table 16-3 Control Registers</b></li> <li>• Modification of description in <b>16.3.2 Interrupt mask registers (MK0, MK1L)</b></li> <li>• Modification of <b>Figure 16-2 Format of Interrupt Mask Registers (MK0, MK1L)</b></li> </ul>
p.316 p.323	<b>CHAPTER 17 STANDBY FUNCTION</b> <ul style="list-style-type: none"> <li>• Modification of <b>Figure 17-6 STOP Mode Release by INTP0 to INTP2 Input</b></li> <li>• Modification of description in <b>(4) A/D converter</b> in <b>17.6 Check Items When STOP Mode/IDLE Mode Is Used</b></li> </ul>
p.325	<b>CHAPTER 18 RESET FUNCTION</b> <ul style="list-style-type: none"> <li>• Modification of <b>Figure 18-1 Oscillation of Main System Clock in Reset Period</b></li> </ul>
p.363	Addition of <b>CHAPTER 21 ELECTRICAL SPECIFICATIONS</b>
p.378	Addition of <b>CHAPTER 22 PACKAGE DRAWINGS</b>
p.379	Addition of <b>CHAPTER 23 RECOMMENDED SOLDERING CONDITIONS</b>
p.381 p.382 p.383 p.384 pp.385, 386 p.387	<b>APPENDIX A DEVELOPMENT TOOLS</b> <ul style="list-style-type: none"> <li>• Modification of <b>Figure A-1 Development Tool Configuration</b></li> <li>• Addition of SP78K4 to <b>A.1 Language Processing Software</b> Modification of <b>Remark</b></li> <li>• Modification of <b>A.3.1 Hardware</b></li> <li>• Modification of <b>Remark</b> in <b>A.3.2 Software</b></li> <li>• Addition of <b>A.4 Notes on Target System Design</b></li> </ul>

The mark ★ shows major revised points.

## INTRODUCTION

**Target Readers** This manual is intended for users who wish to understand the functions of the  $\mu$ PD784976A Subseries and to design and develop application systems and programs using these microcontrollers.

**Purpose** This manual is intended for users to understand the functions described in the Organization below.

**Organization** The  $\mu$ PD784976A Subseries User's Manual is divided into two parts: this manual and Instructions (common to the 78K/IV Series)

$\mu$ PD784976A Subseries User's Manual (This manual)
---

78K/IV Series User's Manual Instructions
--

- |   |   |
|---|---|
| <ul style="list-style-type: none"><li>• Pin functions</li><li>• Internal block functions</li><li>• Interrupt functions</li><li>• Other on-chip peripheral functions</li><li>• Electrical specifications</li></ul> | <ul style="list-style-type: none"><li>• CPU functions</li><li>• Instruction set</li><li>• Explanation of each instruction</li></ul> |
|---|---|

### How to Read This Manual

It is assumed that the reader of this manual has general knowledge in the fields of electrical engineering, logic circuits, and microcontrollers.

- To understand the functions in general:
  - Read this manual in the order of the contents.
- How to interpret the register format:
  - For the bit whose number is in angle brackets, its bit name is defined as a reserved word in the RA78K4, and in CC78K4, already defined in the header file named sfrbit.h.
- When you know a register name and want to confirm its details:
  - Read **APPENDIX C REGISTER INDEX**.
- To know the  $\mu$ PD784976A Subseries instruction function in details:
  - Refer to **78K/IV Series User's Manual: Instruction (U10905E)** separately available.
- When you want to know the application examples of each function of the  $\mu$ PD784976A Subseries
  - Refer to **78K/IV Series Software Basics (U10095E)** separately available.
- To learn about the electrical specifications:
  - Refer to **CHAPTER 21 ELECTRICAL SPECIFICATIONS**.

<b>Conventions</b>	Data significance:	Higher digits on the left and lower digits on the right
	Active low representation:	$\overline{\text{xxx}}$ (overscore over pin or signal name)
	<b>Note:</b>	Footnote for item marked with <b>Note</b> in the text
	<b>Caution:</b>	Information requiring particular attention
	<b>Remark:</b>	Supplementary information
	Numeral representation:	Binary ..... XXXX or XXXXB Decimal ..... XXXX Hexadecimal ..... XXXXH

## Related Documents

The related documents indicated in this publication may include preliminary versions. However, preliminary versions are not marked as such.

### Documents Related to Devices

Document Name	Document No.
μPD784976A Subseries Hardware User's Manual	This manual
78K/IV Series Instructions User's Manual	U10905E
78K/IV Series Software Basics Application Note	U10095E

### Documents Related to Development Software Tools (User's Manuals)

Document Name		Document No.
RA78K4 Assembler Package	Operation	U15254E
	Language	U15255E
	Structured Assembler Preprocessor	U11743E
CC78K4 C Compiler	Operation	U15557E
	Language	U15556E
SM78K4 System Simulator Ver. 1.40 or Later Windows Based	Reference	U10093E
SM78K Series System Simulator Ver. 1.40 or Later	External Part User Open Interface Specifications	U10092E
ID78K Series Integrated Debugger Ver. 2.30 or Later Windows Based	Operation	U15185E
RX78K4 Real-time OS	Fundamental	U10603E
	Installation	U10604E

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



**Documents Related to Development Hardware Tools (User's Manuals)**

Document Name	Document No.
IE-78K4-NS In-Circuit Emulator	U13356E
IE-784976-NS-EM1 Emulation Board	U13745E

**Documents Related to Flash Memory Writing**

Document Name	Document No.
PG-FP3 Flash Memory Programmer User's Manual	U13502E

**Other Related Documents**

Document Name	Document No.
SEMICONDUCTOR SELECTION GUIDE - Products & Packages -	X13769E
Semiconductor Device Mounting Technology Manual	C10535E
Quality Grades on NEC Semiconductor Devices	C11531E
NEC Semiconductor Device Reliability/Quality Control System	C10983E
Guide to Prevent Damage for Semiconductor Devices by Electrostatic Discharge (ESD)	C11892E

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

## CONTENTS

<b>CHAPTER 1 GENERAL</b> .....	<b>23</b>
<b>1.1 Features</b> .....	<b>25</b>
<b>1.2 Application Fields</b> .....	<b>25</b>
<b>1.3 Ordering Information</b> .....	<b>25</b>
<b>1.4 Pin Configuration (Top View)</b> .....	<b>26</b>
<b>1.5 Block Diagram</b> .....	<b>28</b>
<b>1.6 Functional Outline</b> .....	<b>29</b>
<b>1.7 Mask Option</b> .....	<b>31</b>
<b>CHAPTER 2 PIN FUNCTIONS</b> .....	<b>32</b>
<b>2.1 Pin Function List</b> .....	<b>32</b>
<b>2.2 Pin Functions</b> .....	<b>36</b>
2.2.1 P00 to P03 (Port 0) .....	36
2.2.2 P10 to P17 (Port 1) .....	36
2.2.3 P20, P25 to P27 (Port 2) .....	36
2.2.4 P40 to P47 (Port 4) .....	37
2.2.5 P50 to P57 (Port 5) .....	37
2.2.6 P60 to P67 (Port 6) .....	37
2.2.7 P70 to P77 (Port 7) .....	38
2.2.8 P80 to P87 (Port 8) .....	38
2.2.9 P90 to P97 (Port 9) .....	38
2.2.10 P100 to P107 (Port 10) .....	39
2.2.11 FIP0 to FIP15 .....	39
2.2.12 V <sub>LOAD</sub> .....	39
2.2.13 AV <sub>DD</sub> .....	39
2.2.14 AV <sub>SS</sub> .....	39
2.2.15 $\overline{\text{RESET}}$ .....	39
2.2.16 X1 and X2 .....	39
2.2.17 V <sub>DD0</sub> to V <sub>DD2</sub> .....	39
2.2.18 V <sub>SS0</sub> and V <sub>SS1</sub> .....	39
2.2.19 V <sub>PP</sub> ( $\mu$ PD78F4976A only) .....	39
2.2.20 IC (Mask ROM product only) .....	40
<b>2.3 Pin I/O Circuits and Connections of Unused Pins</b> .....	<b>41</b>
<b>CHAPTER 3 CPU ARCHITECTURE</b> .....	<b>44</b>
<b>3.1 Memory Space</b> .....	<b>44</b>
<b>3.2 Internal ROM Area</b> .....	<b>48</b>
<b>3.3 Base Area</b> .....	<b>49</b>
3.3.1 Vector table area .....	50
3.3.2 CALLT instruction table area .....	50
3.3.3 CALLF instruction entry area .....	51
<b>3.4 Internal Data Area</b> .....	<b>52</b>
3.4.1 Internal RAM area .....	52
3.4.2 Special function register (SFR) area .....	54
<b>3.5 <math>\mu</math>PD78F4976A Memory Mapping</b> .....	<b>55</b>

<b>3.6</b>	<b>Control Registers .....</b>	<b>56</b>
3.6.1	Program counter (PC) .....	56
3.6.2	Program status word (PSW) .....	56
3.6.3	Using the RSS bit .....	59
3.6.4	Stack pointer (SP) .....	62
<b>3.7</b>	<b>General-Purpose Registers .....</b>	<b>66</b>
3.7.1	Configuration .....	66
3.7.2	Functions .....	68
<b>3.8</b>	<b>Special Function Registers (SFRs).....</b>	<b>71</b>
<b>3.9</b>	<b>Cautions.....</b>	<b>75</b>
<b>CHAPTER 4 PORT FUNCTIONS .....</b>		<b>76</b>
<b>4.1</b>	<b>Port Functions .....</b>	<b>76</b>
<b>4.2</b>	<b>Port Configuration.....</b>	<b>78</b>
4.2.1	Port 0.....	78
4.2.2	Port 1.....	79
4.2.3	Port 2.....	80
4.2.4	Port 4.....	83
4.2.5	Port 5.....	84
4.2.6	Port 6.....	88
4.2.7	Port 7.....	91
4.2.8	Port 8.....	92
4.2.9	Port 9.....	93
4.2.10	Port 10.....	94
<b>4.3</b>	<b>Port Function Control Registers .....</b>	<b>95</b>
<b>4.4</b>	<b>Port Function Operations.....</b>	<b>98</b>
4.4.1	Writing to I/O port .....	98
4.4.2	Reading from I/O port .....	98
4.4.3	Operations on I/O port.....	98
<b>4.5</b>	<b>Selecting Mask Option.....</b>	<b>99</b>
<b>CHAPTER 5 CLOCK GENERATOR .....</b>		<b>100</b>
<b>5.1</b>	<b>Functions of Clock Generator .....</b>	<b>100</b>
<b>5.2</b>	<b>Configuration of Clock Generator .....</b>	<b>100</b>
<b>5.3</b>	<b>Control Register .....</b>	<b>102</b>
<b>5.4</b>	<b>Main System Clock Oscillator .....</b>	<b>106</b>
5.4.1	Divider .....	108
<b>5.5</b>	<b>Operations of Clock Generator .....</b>	<b>109</b>
<b>5.6</b>	<b>Changing CPU Clock Setting.....</b>	<b>110</b>
<b>CHAPTER 6 TIMER COUNTER OVERVIEW .....</b>		<b>111</b>
<b>CHAPTER 7 16-BIT TIMER/EVENT COUNTER .....</b>		<b>113</b>
<b>7.1</b>	<b>Function.....</b>	<b>113</b>
<b>7.2</b>	<b>Configuration .....</b>	<b>114</b>
<b>7.3</b>	<b>Control Register .....</b>	<b>118</b>
<b>7.4</b>	<b>Operation .....</b>	<b>123</b>
7.4.1	Operation as interval timer (16 bits) .....	123

7.4.2	Pulse width measurement .....	125
7.4.3	Operation as external event counter .....	132
<b>7.5</b>	<b>Generation of Remote Controller Receive Interrupt .....</b>	<b>135</b>
7.5.1	Operating procedure .....	135
7.5.2	Cautions on generation of remote controller interrupt .....	138
<b>7.6</b>	<b>Noise Eliminator of Remote Controller Receive Interrupt Generator .....</b>	<b>141</b>
<b>7.7</b>	<b>Cautions .....</b>	<b>143</b>
<b>CHAPTER 8</b>	<b>8-BIT PWM TIMERS .....</b>	<b>147</b>
<b>8.1</b>	<b>Functions of 8-Bit PWM Timers .....</b>	<b>147</b>
<b>8.2</b>	<b>Configuration of 8-Bit PWM Timers .....</b>	<b>148</b>
<b>8.3</b>	<b>8-Bit PWM Timer Control Registers .....</b>	<b>151</b>
<b>8.4</b>	<b>Operations of 8-Bit PWM Timers .....</b>	<b>154</b>
8.4.1	Operation as interval timer (8-bit operation) .....	154
8.4.2	Operation as external event counter .....	157
8.4.3	Square-wave (8-bit resolution) output operation .....	158
8.4.4	8-bit PWM output operation .....	159
<b>8.5</b>	<b>Cautions on 8-Bit PWM Timers .....</b>	<b>164</b>
<b>CHAPTER 9</b>	<b>WATCHDOG TIMER .....</b>	<b>165</b>
<b>9.1</b>	<b>Configuration .....</b>	<b>165</b>
<b>9.2</b>	<b>Control Register .....</b>	<b>166</b>
<b>9.3</b>	<b>Operations .....</b>	<b>168</b>
<b>9.4</b>	<b>Cautions .....</b>	<b>168</b>
9.4.1	General cautions when using the watchdog timer .....	168
9.4.2	Cautions about the $\mu$ PD784976A Subseries watchdog timer .....	168
<b>CHAPTER 10</b>	<b>WATCH TIMER .....</b>	<b>169</b>
<b>10.1</b>	<b>Functions .....</b>	<b>169</b>
<b>10.2</b>	<b>Configuration .....</b>	<b>170</b>
<b>10.3</b>	<b>Watch Timer Control Registers .....</b>	<b>171</b>
<b>CHAPTER 11</b>	<b>A/D CONVERTER .....</b>	<b>174</b>
<b>11.1</b>	<b>Function of A/D Converter .....</b>	<b>174</b>
<b>11.2</b>	<b>Configuration of A/D Converter .....</b>	<b>174</b>
<b>11.3</b>	<b>A/D Converter Control Registers .....</b>	<b>177</b>
<b>11.4</b>	<b>Operation of A/D Converter .....</b>	<b>179</b>
11.4.1	Basic operation of A/D converter .....	179
11.4.2	Input voltage and conversion result .....	181
11.4.3	Operation mode of A/D converter .....	182
<b>11.5</b>	<b>Notes on A/D Converter .....</b>	<b>183</b>
<b>CHAPTER 12</b>	<b>SERIAL INTERFACE .....</b>	<b>188</b>
<b>12.1</b>	<b>Function of Serial Interface .....</b>	<b>188</b>
<b>12.2</b>	<b>Configuration of Serial Interface .....</b>	<b>188</b>
<b>12.3</b>	<b>Serial Interface Control Registers .....</b>	<b>191</b>
<b>12.4</b>	<b>Operation of Serial Interface .....</b>	<b>194</b>
12.4.1	Operation stop mode .....	194

12.4.2	3-wire serial I/O mode .....	195
<b>12.5</b>	<b>Functions of Serial Interface 2 (SIO2) .....</b>	<b>198</b>
<b>12.6</b>	<b>Cautions on Using Serial Interface 2 (SIO2) .....</b>	<b>199</b>
<b>CHAPTER 13</b>	<b>ASYNCHRONOUS SERIAL INTERFACE .....</b>	<b>200</b>
<b>13.1</b>	<b>Functions of Asynchronous Serial Interface .....</b>	<b>200</b>
<b>13.2</b>	<b>Configuration of Asynchronous Serial Interface .....</b>	<b>202</b>
<b>13.3</b>	<b>Asynchronous Serial Interface Control Registers .....</b>	<b>203</b>
<b>13.4</b>	<b>Operation of Asynchronous Serial Interface .....</b>	<b>208</b>
13.4.1	Operation stop mode .....	208
13.4.2	Asynchronous serial interface (UART) mode .....	209
<b>CHAPTER 14</b>	<b>VFD CONTROLLER/DRIVER .....</b>	<b>220</b>
<b>14.1</b>	<b>Function of VFD Controller/Driver .....</b>	<b>220</b>
<b>14.2</b>	<b>Configuration of VFD Controller/Driver .....</b>	<b>221</b>
<b>14.3</b>	<b>VFD Controller/Driver Control Registers .....</b>	<b>222</b>
14.3.1	Control registers .....	222
14.3.2	One display period and blanking width .....	225
<b>14.4</b>	<b>Display Data Memory .....</b>	<b>226</b>
<b>14.5</b>	<b>Key Scan Flag and Key Scan Data .....</b>	<b>227</b>
14.5.1	Key scan flag .....	227
14.5.2	Key scan data .....	227
<b>14.6</b>	<b>Leakage Emission of Fluorescent Indicator Panel .....</b>	<b>228</b>
<b>14.7</b>	<b>Calculation of Total Power Dissipation .....</b>	<b>231</b>
<b>CHAPTER 15</b>	<b>EDGE DETECTION FUNCTION .....</b>	<b>234</b>
<b>15.1</b>	<b>Control Registers .....</b>	<b>234</b>
<b>15.2</b>	<b>Edge Detection of P64, P65, and P67 Pins .....</b>	<b>235</b>
<b>CHAPTER 16</b>	<b>INTERRUPT FUNCTION .....</b>	<b>236</b>
<b>16.1</b>	<b>Interrupt Request Sources .....</b>	<b>237</b>
16.1.1	Software interrupts .....	239
16.1.2	Operand error interrupts .....	239
16.1.3	Non-maskable interrupts .....	239
16.1.4	Maskable interrupts .....	239
<b>16.2</b>	<b>Interrupt Service Modes .....</b>	<b>240</b>
16.2.1	Vectored interrupt service .....	240
16.2.2	Macro service .....	240
16.2.3	Context switching .....	240
<b>16.3</b>	<b>Interrupt Servicing Control Registers .....</b>	<b>241</b>
16.3.1	Interrupt control registers .....	243
16.3.2	Interrupt mask registers (MK0, MK1L) .....	246
16.3.3	In-service priority register (ISPR) .....	248
16.3.4	Interrupt mode control register (IMC) .....	249
16.3.5	Watchdog timer mode register (WDM) .....	250
16.3.6	Interrupt select control register (SNMI) .....	251
16.3.7	Program status word (PSW) .....	252
<b>16.4</b>	<b>Software Interrupt Acknowledgment Operations .....</b>	<b>253</b>

16.4.1	BRK instruction software interrupt acknowledgment operation .....	253
16.4.2	BRKCS instruction software interrupt (software context switching) acknowledgment operation .....	253
<b>16.5</b>	<b>Operand Error Interrupt Acknowledgment Operation .....</b>	<b>254</b>
<b>16.6</b>	<b>Non-Maskable Interrupt Acknowledgment Operation .....</b>	<b>254</b>
<b>16.7</b>	<b>Maskable Interrupt Acknowledgment Operation .....</b>	<b>257</b>
16.7.1	Vectored interrupt .....	259
16.7.2	Context switching .....	259
16.7.3	Maskable interrupt priority levels .....	261
<b>16.8</b>	<b>Macro Service Function .....</b>	<b>267</b>
16.8.1	Outline of macro service function .....	267
16.8.2	Types of macro service .....	267
16.8.3	Basic macro service operation .....	270
16.8.4	Operation at end of macro service .....	271
16.8.5	Macro service control registers .....	274
16.8.6	Macro service type A .....	276
16.8.7	Macro service type B .....	279
16.8.8	Macro service type C .....	283
16.8.9	Counter mode .....	288
<b>16.9</b>	<b>When Interrupt Requests and Macro Service Are Temporarily Held Pending .....</b>	<b>290</b>
<b>16.10</b>	<b>Instructions Whose Execution Is Temporarily Suspended by an Interrupt or Macro Service .....</b>	<b>290</b>
<b>16.11</b>	<b>Interrupt and Macro Service Operation Timing .....</b>	<b>291</b>
16.11.1	Interrupt acknowledge processing time .....	292
16.11.2	Processing time of macro service .....	293
<b>16.12</b>	<b>Restoring Interrupt Function to Initial State .....</b>	<b>294</b>
<b>16.13</b>	<b>Cautions .....</b>	<b>295</b>
<b>CHAPTER 17</b>	<b>STANDBY FUNCTION .....</b>	<b>297</b>
<b>17.1</b>	<b>Configuration and Function .....</b>	<b>297</b>
<b>17.2</b>	<b>Control Registers .....</b>	<b>298</b>
17.2.1	Standby control register (STBC) .....	298
17.2.2	Oscillation stabilization time specification register (OSTS) .....	300
<b>17.3</b>	<b>HALT Mode .....</b>	<b>302</b>
17.3.1	HALT mode setting and operating states .....	302
17.3.2	HALT mode release .....	302
<b>17.4</b>	<b>STOP Mode .....</b>	<b>310</b>
17.4.1	STOP mode setting and operating states .....	310
17.4.2	STOP mode release .....	312
<b>17.5</b>	<b>IDLE Mode .....</b>	<b>317</b>
17.5.1	IDLE mode setting and operating states .....	317
17.5.2	IDLE mode release .....	318
<b>17.6</b>	<b>Check Items When STOP Mode/IDLE Mode Is Used .....</b>	<b>323</b>
<b>17.7</b>	<b>Cautions .....</b>	<b>324</b>
<b>CHAPTER 18</b>	<b>RESET FUNCTION .....</b>	<b>325</b>

	<b>CHAPTER 19 <math>\mu</math>PD78F4976A PROGRAMMING.....</b>	<b>327</b>
	<b>19.1 Selecting Communication Protocol .....</b>	<b>327</b>
	<b>19.2 Flash Memory Programming Functions.....</b>	<b>328</b>
	<b>19.3 Connecting Flashpro III .....</b>	<b>329</b>
	<b>CHAPTER 20 INSTRUCTION OPERATION .....</b>	<b>330</b>
	<b>20.1 Examples .....</b>	<b>330</b>
	<b>20.2 List of Operations .....</b>	<b>334</b>
	<b>20.3 Lists of Addressing Instructions .....</b>	<b>359</b>
★	<b>CHAPTER 21 ELECTRICAL SPECIFICATIONS.....</b>	<b>363</b>
★	<b>CHAPTER 22 PACKAGE DRAWINGS .....</b>	<b>378</b>
★	<b>CHAPTER 23 RECOMMENDED SOLDERING CONDITIONS .....</b>	<b>379</b>
	<b>APPENDIX A DEVELOPMENT TOOLS .....</b>	<b>380</b>
	<b>A.1 Language Processing Software .....</b>	<b>382</b>
	<b>A.2 Flash Memory Writing Tools .....</b>	<b>383</b>
	<b>A.3 Debugging Tools .....</b>	<b>384</b>
	A.3.1 Hardware.....	384
	A.3.2 Software .....	385
★	<b>A.4 Notes on Target System Design.....</b>	<b>387</b>
	<b>A.5 Conversion Socket (EV-9200GF-100).....</b>	<b>389</b>
	<b>APPENDIX B SOFTWARE FOR EMBEDDED USE .....</b>	<b>391</b>
	<b>APPENDIX C REGISTER INDEX .....</b>	<b>392</b>
	<b>C.1 Register Name Index (Alphabetic Order) .....</b>	<b>392</b>
	<b>C.2 Register Symbol Index .....</b>	<b>395</b>
★	<b>APPENDIX D REVISION HISTORY .....</b>	<b>398</b>

## LIST OF FIGURES (1/5)

Figure No.	Title	Page
2-1	Types of Pin I/O Circuits .....	42
3-1	Format of Memory Expansion Mode Register (MM) .....	44
3-2	Memory Map of $\mu$ PD784975A .....	46
3-3	Memory Map of $\mu$ PD78F4976A .....	47
3-4	Memory Map of Internal RAM .....	53
3-5	Format of Internal Memory Size Switching Register (IMS) .....	55
3-6	Format of Program Counter (PC) .....	56
3-7	Format of Program Status Word (PSW) .....	57
3-8	Format of Stack Pointer (SP) .....	62
3-9	Data Saved to the Stack .....	63
3-10	Data Restored from the Stack .....	64
3-11	Format of General-Purpose Register .....	66
3-12	General-Purpose Register Addresses .....	67
4-1	Port Types .....	76
4-2	Block Diagram of P00 to P03 .....	78
4-3	Block Diagram of P10 to P17 .....	79
4-4	Block Diagram of P20 and P25 .....	80
4-5	Block Diagram of P26 .....	81
4-6	Block Diagram of P27 .....	82
4-7	Block Diagram of P40 to P47 .....	83
4-8	Block Diagram of P50 to P54 .....	84
4-9	Block Diagram of P55 .....	85
4-10	Block Diagram of P56 .....	86
4-11	Block Diagram of P57 .....	87
4-12	Block Diagram of P60, P64, P65, and P67 .....	88
4-13	Block Diagram of P61 .....	89
4-14	Block Diagram of P62, P63, and P66 .....	90
4-15	Block Diagram of P70 to P77 .....	91
4-16	Block Diagram of P80 to P87 .....	92
4-17	Block Diagram of P90 to P97 .....	93
4-18	Block Diagram of P100 to P107 .....	94
4-19	Format of Port Mode Register .....	96
4-20	Format of Pull-Up Resistor Option Register 2 (PU2) .....	96
4-21	Format of Pull-Up Resistor Option Register (PUO) .....	97
5-1	Block Diagram of Clock Generator .....	101
5-2	Format of Standby Control Register (STBC) .....	103
5-3	Format of Oscillation Mode Select Register (CC) .....	104
5-4	Format of Oscillation Stabilization Specification Register (OSTS) .....	105
5-5	External Circuit of Main System Clock Oscillator .....	106
5-6	Examples of Oscillator Connected Incorrectly .....	107
5-7	Changing CPU Clock .....	110



## LIST OF FIGURES (2/5)

Figure No.	Title	Page
6-1	Block Diagram of Timer Counter .....	111
7-1	Block Diagram of 16-Bit Timer/Event Counter 0 .....	115
7-2	Format of 16-Bit Timer Mode Control Register 0 (TMC0) .....	118
7-3	Format of Capture/Compare Control Register 0 (CRC0) .....	119
7-4	Format of Prescaler Mode Register 0 (PRM0) .....	120
7-5	Format of Remote Controller Receive Mode Register (REMM) .....	121
7-6	Control Register Settings When 16-Bit Timer/Event Counter Operates as Interval Timer .....	123
7-7	Configuration of Interval Timer .....	124
7-8	Timing of Interval Timer Operation .....	124
7-9	Control Register Settings for Pulse Width Measurement with Free-Running Counter and One Capture Register .....	125
7-10	Configuration for Pulse Width Measurement with Free-Running Counter .....	126
7-11	Timing of Pulse Width Measurement with Free-Running Counter and One Capture Register (with Both Edges Specified) .....	126
7-12	Control Register Settings for Measurement of Two Pulse Widths with Free-Running Counter .....	127
7-13	CR01 Capture Operation with Rising Edge Specified .....	128
7-14	Timing of Pulse Width Measurement with Free-Running Counter (with Both Edges Specified) .....	128
7-15	Control Register Settings for Pulse Width Measurement with Free-Running Counter and Two Capture Registers .....	129
7-16	Timing of Pulse Width Measurement with Free-Running Counter and Two Capture Registers (with Rising Edge Specified) .....	130
7-17	Control Register Settings for Pulse Width Measurement by Restarting .....	131
7-18	Timing of Pulse Width Measurement by Restarting (with Rising Edge Specified) .....	132
7-19	Control Register Settings in External Event Counter Mode .....	133
7-20	Configuration of External Event Counter .....	133
7-21	Timing of External Event Counter Operation (with Rising Edge Specified) .....	134
7-22	Operation Timing When Remote Controller Receive Interrupt Is Generated .....	136
7-23	Block Diagram of INTREM .....	141
7-24	Sampling Timing Chart .....	142
7-25	Noise Eliminator Output Signal .....	142
7-26	Start Timing of 16-Bit Timer Counter 0 (TM0) .....	143
7-27	Timing After Changing Compare Register During Timer Count Operation .....	143
7-28	Data Hold Timing of Capture Register .....	144
7-29	Operation Timing of OVFO Flag .....	145
8-1	Block Diagram of 8-Bit PWM Timer 50 .....	148
8-2	Block Diagram of 8-Bit PWM Timer 51 .....	149
8-3	Format of Timer Clock Select Register 5n (TCL5n) .....	151
8-4	Format of 8-Bit Timer Control Register 5n (TMC5n) .....	153
8-5	Timing of Interval Timer Operation .....	155
8-6	Timing of External Event Counter Operation (with Rising Edge Specified) .....	157
8-7	PWM Output Operation Timing .....	160

## LIST OF FIGURES (3/5)

Figure No.	Title	Page
8-8	Operation Timing When CR5n Is Changed .....	161
8-9	16-Bit Resolution Cascade Mode .....	163
8-10	Start Timing of 8-Bit Timer Counter 5n (TM5n) .....	164
8-11	Timing After Changing Compare Register Value During Timer Count Operation .....	164
9-1	Block Diagram of Watchdog Timer .....	165
9-2	Format of Watchdog Timer Mode Register (WDM) .....	167
10-1	Block Diagram of Watch Timer .....	169
10-2	Watch Timer Mode Control Register (WTM) .....	171
10-3	Watch Timer Clock Select Register (WTCL) .....	173
11-1	Block Diagram of A/D Converter .....	175
11-2	Format of A/D Converter Mode Register .....	177
11-3	Format of A/D Converter Input Select Register .....	178
11-4	Basic Operation of A/D Converter .....	180
11-5	Relation Between Analog Input Voltage and A/D Conversion Result .....	181
11-6	A/D Conversion by Software Start .....	182
11-7	Example of Reducing Current Consumption in Standby Mode .....	183
11-8	Processing of Analog Input Pin .....	184
11-9	Timing of A/D Conversion End Interrupt Request Generation .....	185
11-10	Processing of AV <sub>DD</sub> Pin .....	185
11-11	Result of Conversion Immediately After A/D Conversion Is Started .....	186
11-12	Conversion Result Read Timing (When Conversion Result Is Undefined) .....	186
11-13	Conversion Result Read Timing (When Conversion Result Is Normal) .....	187
12-1	Block Diagram of Serial Interface 0, 1 .....	189
12-2	Block Diagram of Serial Interface 2 .....	189
12-3	Format of Serial Operation Mode Register n .....	191
12-4	Format of Serial Operation Mode Register 2 .....	192
12-5	Timing in 3-Wire Serial I/O Mode .....	197
13-1	Block Diagram of Asynchronous Serial Interface (UART) .....	201
13-2	Format of Asynchronous Serial Interface Mode Register 0 (ASIM0) .....	204
13-3	Format of Asynchronous Serial Interface Status Register 0 (ASIS0) .....	205
13-4	Format of Baud Rate Generator Control Register 0 (BRGC0) .....	206
13-5	Baud Rate Capacity Error Considering Sampling Errors (When k = 0) .....	214
13-6	Format of Asynchronous Serial Interface Transmit/Receive Data .....	215
13-7	Asynchronous Serial Interface Transmit Completion Interrupt Timing .....	217
13-8	Asynchronous Serial Interface Receive Completion Interrupt Timing .....	218
13-9	Receive Error Timing .....	219

## LIST OF FIGURES (4/5)

Figure No.	Title	Page
14-1	Block Diagram of VFD Controller/Driver .....	221
14-2	Format of Display Mode Register 0 .....	222
14-3	Format of Display Mode Register 1 (DSPM1) .....	223
14-4	Format of Display Mode Register 2 (DSPM2) .....	224
14-5	Blanking Width of VFD Output Signal .....	225
14-6	Relation Between Address Location of Display Data Memory and VFD Output (with 48 VFD Output Pins and 16 Patterns) .....	226
14-7	Leakage Emission Because of Short Blanking Time .....	228
14-8	Leakage Emission Caused by $C_{SG}$ .....	229
14-9	Leakage Emission Caused by $C_{SG}$ .....	230
14-10	Total Power Dissipation $P_T$ ( $T_A = -40^\circ\text{C}$ to $+85^\circ\text{C}$ ) .....	231
14-11	Relationship Between Display Data Memory and VFD Output with 10 Segments $\times$ 11 Digits Displayed .....	233
15-1	Format of External Interrupt Rising Edge Enable Register (EGP0) and External Interrupt Falling Edge Enable Register (EGN0) .....	234
15-2	Edge Detection of P64, P65, and P67 Pins .....	235
16-1	Interrupt Control Register ( $\times\times\text{ICn}$ ) .....	244
16-2	Format of Interrupt Mask Registers (MK0, MK1L) .....	247
16-3	Format of In-Service Priority Register (ISPR) .....	248
16-4	Format of Interrupt Mode Control Register (IMC) .....	249
16-5	Format of Watchdog Timer Mode Register (WDM) .....	250
16-6	Format of Interrupt Select Control Register (SNMI) .....	251
16-7	Format of Program Status Word (PSWL) .....	252
16-8	Context Switching Operation by Execution of BRKCS Instruction .....	253
16-9	Return from BRKCS Instruction Software Interrupt (RETCSB Instruction Operation) .....	254
16-10	Non-Maskable Interrupt Request Acknowledgment Operations .....	255
16-11	Interrupt Request Acknowledgment Processing Algorithm .....	258
16-12	Context Switching Operation by Generation of Interrupt Request .....	259
16-13	Return from Interrupt that Uses Context Switching by Means of RETCS Instruction .....	260
16-14	Examples of Servicing When Another Interrupt Request Is Generated During Interrupt Service .....	262
16-15	Examples of Servicing of Simultaneously Generated Interrupts .....	265
16-16	Differences in Level 3 Interrupt Acknowledgment According to IMC Register Setting .....	266
16-17	Differences Between Vectored Interrupt and Macro Service Processing .....	267
16-18	Macro Service Processing Sequence .....	270
16-19	Operation at End of Macro Service When VCIE = 0 .....	272
16-20	Operation at End of Macro Service When VCIE = 1 .....	273
16-21	Format of Macro Service Control Word .....	274
16-22	Format of Macro Service Mode Register .....	275
16-23	Macro Service Data Transfer Processing Flow (Type A) .....	277
16-24	Type A Macro Service Channel .....	278

## LIST OF FIGURES (5/5)

Figure No.	Title	Page
16-25	Macro Service Data Transfer Processing Flow (Type B) .....	280
16-26	Type B Macro Service Channel .....	281
16-27	Parallel Data Input Synchronized with External Interrupts .....	282
16-28	Timing of Parallel Data Input .....	283
16-29	Macro Service Data Transfer Processing Flow (Type C) .....	284
16-30	Type C Macro Service Channel .....	286
16-31	Macro Service Data Transfer Processing Flow (Counter Mode) .....	288
16-32	Counter Mode .....	289
16-33	Counting Number of Edges .....	289
16-34	Interrupt Request Generation and Acknowledgment (Unit: Clock = 1/f <sub>CLK</sub> ) .....	291
17-1	Standby Mode Transition Diagram .....	297
17-2	Format of Standby Control Register (STBC) .....	299
17-3	Format of Oscillation Stabilization Time Specification Register (OSTS) .....	301
17-4	Operation After HALT Mode Release .....	304
17-5	Operation After STOP Mode Release .....	313
17-6	STOP Mode Release by INTP0 to INTP2 Input .....	316
17-7	Operation After IDLE Mode Release .....	319
18-1	Oscillation of Main System Clock in Reset Period .....	325
18-2	Accepting Reset Signal .....	326
19-1	Format of Communication Protocol Selection .....	328
19-2	Connecting Flashpro III in 3-Wire Serial I/O Mode (When Using 3-Wire Serial I/O0) .....	329
21-1	Power Supply Voltage and Clock Cycle Time .....	364
A-1	Development Tool Configuration .....	381
A-2	Distance Between In-Circuit Emulator and Conversion Socket .....	387
A-3	Conditions for Target System Connection (1) .....	388
A-4	Conditions for Target System Connection (2) .....	388
A-5	Package Drawing of EV-9200GF-100 (Reference) (Units: mm) .....	389
A-6	Recommended Board Installation Pattern of EV-9200GF-100 (Reference) (Units: mm) .....	390

## LIST OF TABLES (1/2)

Table No.	Title	Page
1-1	Mask Options of Mask ROM Versions .....	31
2-1	Types of Pin I/O Circuits and Recommended Connection of Unused Pins .....	41
3-1	Vector Table Address .....	50
3-2	Internal RAM Area List .....	52
3-3	Settings of the Internal Memory Size Switching Register (IMS) .....	55
3-4	Register Bank Selection .....	59
3-5	Correspondence Between Function Names and Absolute Names .....	70
3-6	Special Function Register (SFR) List .....	72
4-1	Port Function .....	77
4-2	Port Configuration .....	78
4-3	Port Mode Register and Output Latch Setting When Alternate Function Is Used .....	95
4-4	Comparison Between Mask Options of Mask ROM Version and $\mu$ PD78F4976A .....	99
5-1	Configuration of Clock Generator .....	100
6-1	Timer Counter Operation .....	111
7-1	Configuration of 16-Bit Timer/Event Counter 0 .....	114
7-2	Valid Edge of TI00 Pin and Valid Edge of Capture Trigger of CR00 .....	116
7-3	Valid Edge of TIO51 Pin and Valid Edge of Capture Trigger of CR00 .....	117
7-4	Valid Edge of Pin TI00 and Valid Edge of Capture Trigger of CR01 .....	117
7-5	Selection of TI00 Pin Valid Edge and Signal Identifier .....	135
7-6	Setting Range of CR00 (Min) and CR01 (Max), and Generation of INTREM .....	136
8-1	Configuration of 8-Bit PWM Timers .....	148
10-1	Configuration of Watch Timer .....	170
10-2	Setting of Watch Timer Interrupt Request .....	173
11-1	Configuration of A/D Converter .....	174
12-1	Configuration of Serial Interface .....	188
12-2	Serial Interface Operation Mode Settings .....	193
13-1	Switching Asynchronous Serial Interface Mode and 3-Wire Serial I/O Mode .....	200
13-2	Configuration of Asynchronous Serial Interface .....	202
13-3	Serial Interface Operation Mode Settings .....	207
13-4	Relation Between 5-Bit Counter Source Clock and m Value .....	213
13-5	Relation Between BR0 Selection Clock and Baud Rate .....	214
13-6	Receive Error Causes .....	219

## LIST OF TABLES (2/2)

Table No.	Title	Page
14-1	VFD Output Pins and Multiplexed Port Pins .....	220
14-2	Configuration of VFD Controller/Driver .....	221
16-1	Interrupt Request Service Modes .....	236
16-2	Interrupt Request Sources .....	237
16-3	Control Registers .....	241
16-4	Flag List of Interrupt Control Registers for Interrupt Request Sources .....	242
16-5	Multiple Interrupt Processing .....	261
16-6	Interrupts for Which Macro Service Can be Used .....	268
16-7	Interrupt Acknowledge Processing Time .....	292
16-8	Macro Service Processing Time .....	293
17-1	Operating States in HALT Mode .....	302
17-2	HALT Mode Release and Operations After Release .....	303
17-3	HALT Mode Release by Maskable Interrupt Request .....	309
17-4	Operating States in STOP Mode .....	310
17-5	STOP Mode Release and Operations After Release .....	312
17-6	Operating States in IDLE Mode .....	317
17-7	IDLE Mode Release and Operations After Release .....	318
18-1	State During/After Reset for All Hardware Resets .....	326
19-1	Communication Protocols .....	327
19-2	Major Functions in Flash Memory Programming .....	328
20-1	8-Bit Addressing Instructions .....	359
20-2	16-bit Addressing Instructions .....	360
20-3	24-bit Addressing Instructions .....	361
20-4	Bit Manipulation Instruction Addressing Instructions .....	361
20-5	Call Return Instructions and Branch Instruction Addressing Instructions .....	362
23-1	Surface Mounting Type Soldering Conditions .....	379

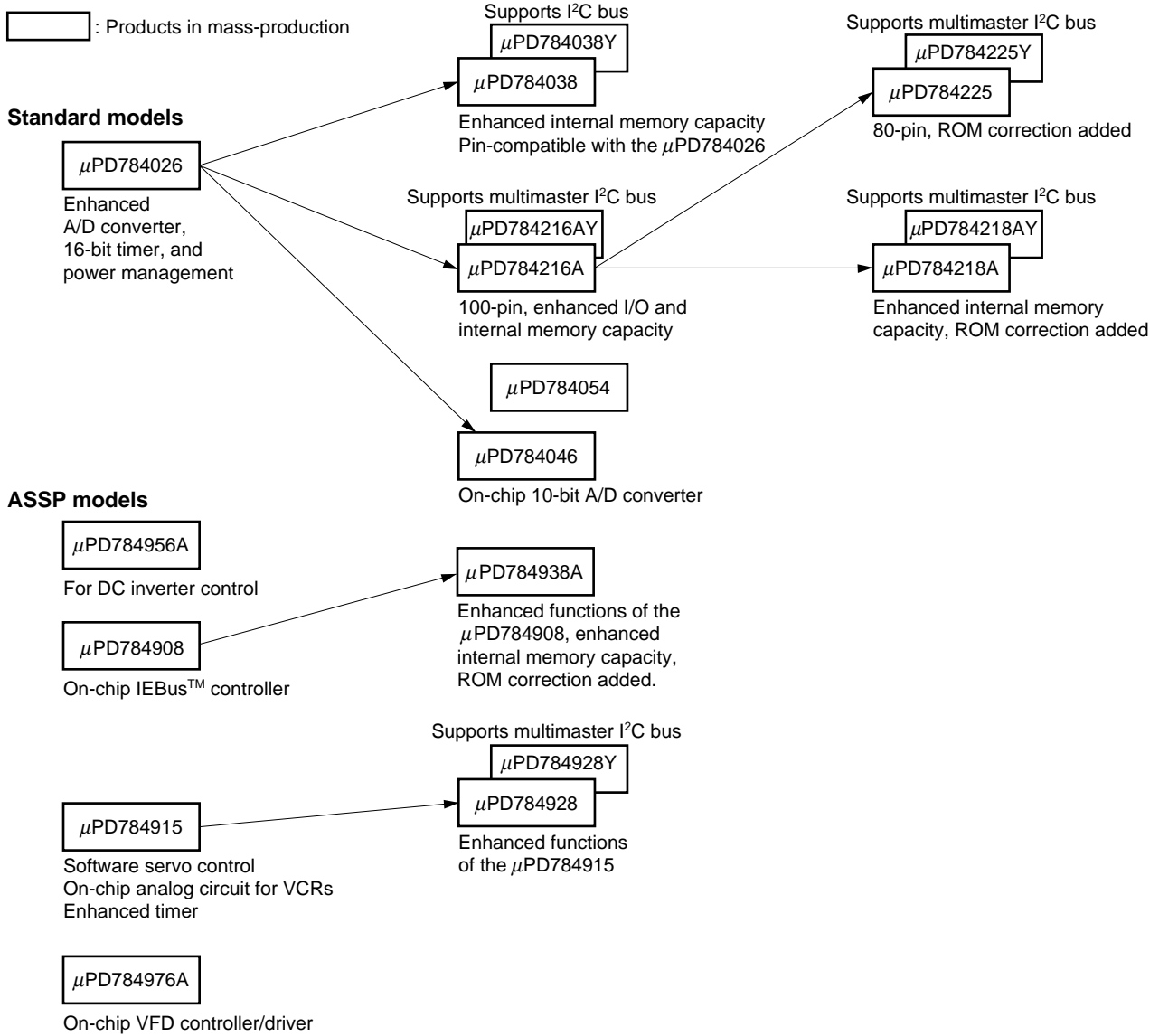
## CHAPTER 1 GENERAL

The  $\mu$ PD784976A Subseries is part of the 78K/IV Series designed for ASSP and housed in a 100-pin QFP. Each of the 78K/IV Series of 16-bit single-chip microcontrollers features a powerful CPU that can access 1 MB of memory.

The  $\mu$ PD784975A incorporates 96 KB of mask ROM and 3,584 bytes of RAM. It also incorporates a VFD controller/driver, advanced timer/event counters, and two independent serial interfaces.

The  $\mu$ PD78F4976A incorporates 128 KB of flash memory and 5,120 bytes of RAM.

★ 78K/IV Series Lineup



**Remark** VFD (Vacuum Fluorescent Display) is referred to as FIP™ (Fluorescent Indicator Panel) in some documents, but the functions of the two are the same.



## 1.1 Features

- High-capacity ROM and RAM

Item Part Number	Program Memory		Data Memory		
	Mask ROM	Flash Memory	Peripheral RAM	High-Speed RAM	VFD Display RAM
μPD784975A	96 KB	—	3,072 bytes	512 bytes	96 bytes
μPD78F4976A	—	128 KB <sup>Note</sup>	4,608 bytes		

**Note** 96 KB can be selected by the memory size switching register (IMS).

- Minimum instruction execution time
  - 160 ns (@f<sub>xx</sub> = 12.5 MHz operation)
- I/O port: 72 pins
- VFD controller/driver: Total display output pins: 48 (universal grid compatible)
  - Display current 10 mA: 16 pins
  - Display current 3 mA: 32 pins
- 8-bit resolution A/D converter: 12 channels
  - Supply voltage (AV<sub>DD</sub> = 4.5 to 5.5 V)
- Serial interface: 3 channels
  - 3-wire serial I/O mode: 2 channels
  - UART/IOE (3-wire serial I/O): 1 channel
- Timer: 5 channels
  - 16-bit timer/event counter: 1 channel
  - 8-bit PWM timer: 2 channels
  - Watch timer: 1 channel
  - Watchdog timer: 1 channel
- Vectored interrupt source: 22
- Supply voltage: V<sub>DD</sub> = 4.5 to 5.5 V

## 1.2 Application Fields

Combined mini-component audio systems, separate mini-component audio systems, tuners, cassette decks, CD players, and audio amplifiers

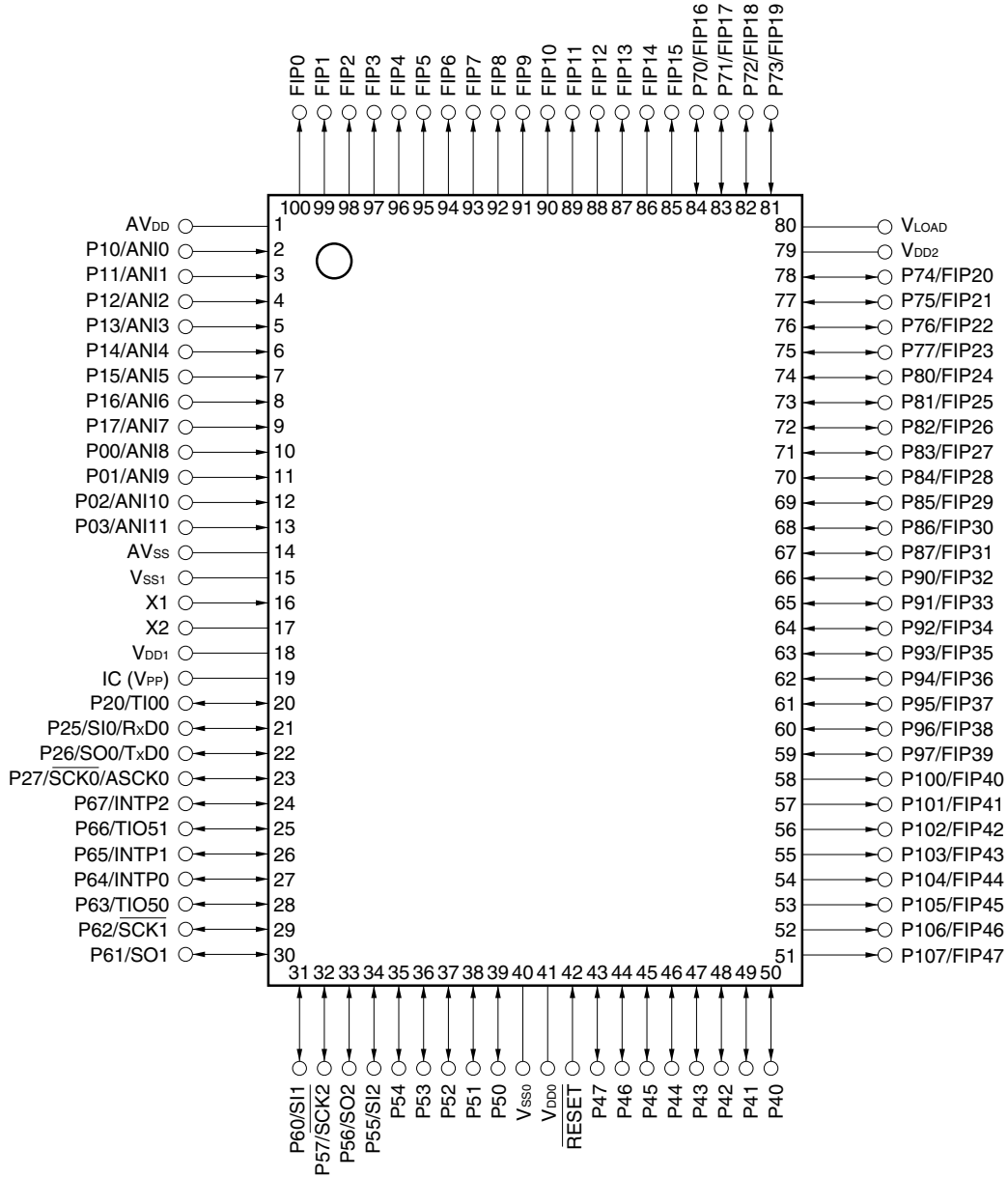
## 1.3 Ordering Information

Part Number	Package	Internal ROM
μPD784975AGF-xxx-3BA	100-pin plastic QFP (14 × 20)	Mask ROM
μPD78F4976AGF-3BA	100-pin plastic QFP (14 × 20)	Flash memory

**Remark** xxx indicates ROM code suffix.

1.4 Pin Configuration (Top View)

- 100-pin plastic QFP (14 × 20):  $\mu$ PD784975AGF-~~xxx~~-3BA, 78F4976AGF-3BA



**Cautions** 1. Directly connect the IC (Internally Connected) pin to V<sub>SS1</sub> in the normal operation mode.

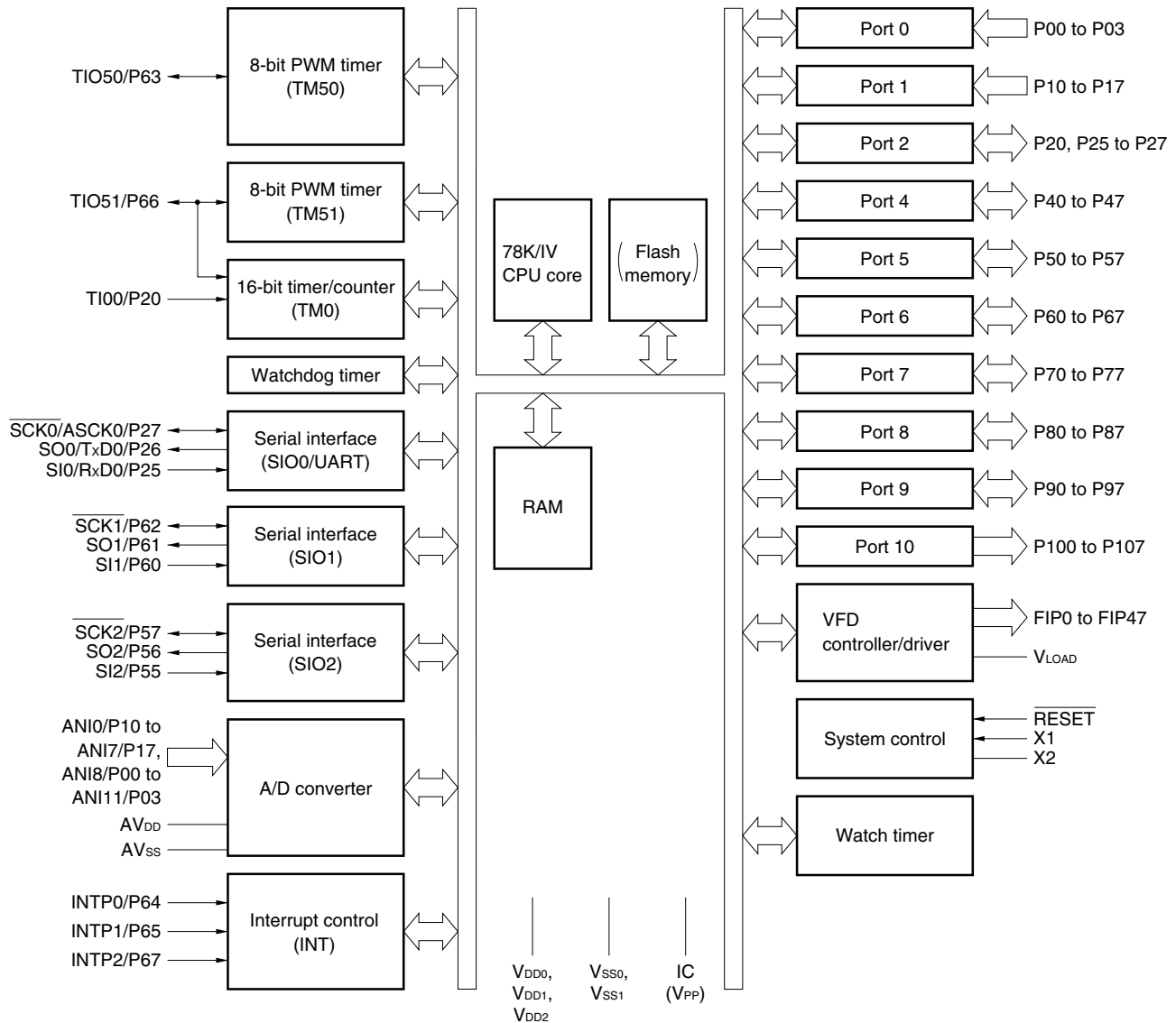
- ★ 2. When the A/D converter is used (ADCS = 1), use the AV<sub>DD</sub> pin with the same potential as V<sub>DD1</sub>. When the A/D converter is not used (ADCS = 0), the AV<sub>DD</sub> pin can be used with the same potential as V<sub>SS1</sub>.
- 3. Connect the AV<sub>SS</sub> pin to V<sub>SS1</sub>.

- ★ **Remarks** 1. When the  $\mu$ PD784976A Subseries is used in applications where the noise generated inside the microcontroller needs to be reduced, the implementation of noise reduction measures, such as supplying voltage to V<sub>DD0</sub> and V<sub>DD1</sub> individually and connecting V<sub>SS0</sub> and V<sub>SS1</sub> to different ground lines, is recommended. Always keep V<sub>DD0</sub> at the same potential as the V<sub>DD1</sub>. In addition, always keep V<sub>SS0</sub> at the same potential as V<sub>SS1</sub>.
- 2. The value in parentheses is valid for the  $\mu$ PD78F4976A.

ANI0 to ANI11:	Analog input	P90 to P97:	Port 9
ASCK0:	Asynchronous serial clock	P100 to P107:	Port 10
AV <sub>DD</sub> :	Analog power supply	$\overline{\text{RESET}}$ :	Reset
AV <sub>SS</sub> :	Analog ground	RxD0:	Receive data
FIP0 to FIP47:	Fluorescent indicator panel	$\overline{\text{SCK0}}$ , $\overline{\text{SCK1}}$ , $\overline{\text{SCK2}}$ :	Serial clock
IC:	Internally connected	SI0 to SI2:	Serial input
INTP0 to INTP2:	External interrupt input	SO0 to SO2:	Serial output
P00 to P03:	Port 0	TI00:	Timer input
P10 to P17:	Port 1	TIO50, TIO51:	Timer input/output
P20, P25 to P27:	Port 2	TxD0:	Transmit data
P40 to P47:	Port 4	V <sub>DD0</sub> to V <sub>DD2</sub> :	Power supply
P50 to P57:	Port 5	V <sub>LOAD</sub> :	Negative power supply
P60 to P67:	Port 6	V <sub>PP</sub> <sup>Note</sup> :	Programming power supply
P70 to P77:	Port 7	V <sub>SS0</sub> , V <sub>SS1</sub> :	Ground
P80 to P87:	Port 8	X1, X2:	Crystal

**Note** The V<sub>PP</sub> pin is available in the  $\mu$ PD78F4976A only.

### 1.5 Block Diagram



- Remarks**
1. The internal ROM capacity varies depending on the product.
  2. The flash memory pin and V<sub>PP</sub> pin are available in the  $\mu$ PD78F4976A only.

## 1.6 Functional Outline

Part Number		$\mu$ PD784975A	$\mu$ PD78F4976A
Internal memory	ROM	Mask ROM	Flash memory
		96 KB	128 KB <sup>Note</sup>
	Peripheral RAM	3,072 bytes	4,608 bytes
	High-speed RAM	512 bytes	
	VFD display RAM	96 bytes	
General-purpose register		8 bits $\times$ 16 registers $\times$ 8 banks, or 16 bits $\times$ 8 registers $\times$ 8 banks	
Minimum instruction execution time		160 ns (@f <sub>xx</sub> = 12.5 MHz operation)	
Instruction set		<ul style="list-style-type: none"> <li>• 16-bit operation</li> <li>• Multiplication/division (8 bits <math>\times</math> 8 bits, 16 bits <math>\div</math> 8 bits)</li> <li>• Bit manipulation (set, reset, test, Boolean operation)</li> <li>• BCD adjustment, etc.</li> </ul>	
I/O port (including VFD-multiplexed pins)		<ul style="list-style-type: none"> <li>• Total: 72 pins</li> <li>• CMOS input: 12 pin</li> <li>• CMOS I/O: 20 pins</li> <li>• N-ch open-drain I/O: 8 pins</li> <li>• P-ch open-drain I/O: 24 pins</li> <li>• P-ch open-drain output: 8 pins</li> </ul>	
VFD controller/driver		<ul style="list-style-type: none"> <li>• Total display output: 48 pins</li> <li>• Display current 10 mA: 16 pins</li> <li>• Display current: 3 mA: 32 pins</li> </ul>	
A/D converter		<ul style="list-style-type: none"> <li>• 8-bit resolution <math>\times</math> 12 channels</li> <li>• Supply voltage: AV<sub>DD</sub> = 4.5 to 5.5 V</li> </ul>	
Serial interface		<ul style="list-style-type: none"> <li>• 3-wire serial I/O mode: 2 channels</li> <li>• UART/IOE (3-wire serial I/O): 1 channel</li> </ul>	
Timer		<ul style="list-style-type: none"> <li>• 16-bit timer/event counter: 1 channel</li> <li>• 8-bit PWM timer: 2 channels</li> <li>• Watch timer: 1 channel</li> <li>• Watchdog timer: 1 channel</li> </ul>	
Timer output		2 pins (8-bit PWM output)	
Vectored interrupt source	Maskable	Internal: 14, external: 3, internal/external: 2	
	Non-maskable	Internal: 1	
	Software	BRK, BRKCS instructions, operand error	
Supply voltage		V <sub>DD</sub> = 4.5 to 5.5 V	
Package		100-pin plastic QFP (14 $\times$ 20)	

**Note** 96 KB can be selected by the memory size switching register (IMS)

The following table summarizes the timers (for details, refer to **CHAPTERS 7 16-BIT TIMER/EVENT COUNTER** and **CHAPTER 8 8-BIT PWM TIMER**).

Item		Name	16-Bit Timer/Event Counter	8-Bit PWM Timer (TM50)	8-Bit PWM Timer (TM51)
Count width	8 bits		—	○	○
	16 bits		○	○	
Operation mode	Interval timer		1 ch	1 ch	1 ch
	External event counter		○	○	○
Function	Timer output		—	○	○
	PWM output		—	○	○
	Square wave output		—	○	○
	Pulse width measurement		Two inputs	—	—
	Number of interrupt requests		2	1	1

The following table summarizes the serial interface (for details, refer to **CHAPTER 12 SERIAL INTERFACE**).

Function	SI0	SI1
3-wire serial I/O mode	○ (fixed to MSB)	○ (fixed to MSB)

## 1.7 Mask Option

The mask ROM version ( $\mu$ PD794975A) has mask options. By specifying the mask options when placing an order for these versions, the pull-up and pull-down resistors shown in Table 1-1 can be used. If these mask options are used when pull-up and pull-down resistors are necessary, the number of components can be decreased and the mounting area can be reduced.

Table 1-1 shows the mask options available for the  $\mu$ PD784976A Subseries.

**Table 1-1. Mask Options of Mask ROM Versions**

Pin Name	Mask Option
P50 to P57	Pull-up resistors can be specified in 1-bit units.
P70/FIP16 to P77/FIP23, P80/FIP24 to P87/FIP31, P90/FIP32 to P97/FIP39, P100/FIP40 to P107/FIP47	Pull-down resistors can be specified in 1-bit units.

**Caution** The emulation function using a mask option resistor is not available in the in-circuit emulator (IE). Utilize the function by externally mounting a resistor on the board.

## CHAPTER 2 PIN FUNCTIONS

### 2.1 Pin Function List

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

Pin Name	I/O	Function	After Reset	Alternate Function
P00 to P03	Input	Port 0. 4-bit input port.	—	ANI8 to ANI11
P10 to P17	Input	Port 1. 8-bit input port.	—	ANI0 to ANI7
P20	I/O	Port 2. 4-bit I/O port. Input/output can be specified in 1-bit units. On-chip pull-up resistor can be specified by a software setting in 1-bit or 8-bit units.	Input	TI00
P25				SI0/RxD0
P26				SO0/TxD0
P27				$\overline{\text{SCK0}}$ /ASCK0
P40 to P47	I/O	Port 4. 8-bit I/O port. Input/output can be specified in 1-bit units. On-chip pull-up resistor can be specified by a software setting per port. Can directly drive LED.	Input	—
P50 to P54	I/O	Port 5. N-ch open-drain 8-bit medium-voltage I/O port. Input/output can be specified in 1-bit units. On-chip pull-up resistor can be specified by mask option in 1-bit units (mask ROM versions only). The $\mu$ PD78F4976A does not have pull-up resistors, however. Can directly drive LED.	Input	—
P55				SI2
P56				SO2
P57				$\overline{\text{SCK2}}$
P60	I/O	Port 6. 8-bit I/O port. Input/output can be specified in 1-bit units. On-chip pull-up resistor can be specified by a software setting per port.	Input	SI1
P61				SO1
P62				$\overline{\text{SCK1}}$
P63				TIO50
P64				INTP0
P65				INTP1
P66				TIO51
P67				INTP2
P70 to P77	I/O	Port 7. P-ch open-drain 8-bit high-voltage I/O port. Input/output can be specified in 1-bit units. On-chip pull-down resistor can be specified by mask option in 1-bit units (mask ROM versions only). The $\mu$ PD78F4976A does not have pull-down resistors, however.	Input	FIP16 to FIP23



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

Pin Name	I/O	Function	After Reset	Alternate Function
P80 to P87	I/O	Port 8. P-ch open-drain 8-bit high-voltage I/O port. Input/output can be specified in 1-bit units. On-chip pull-down resistor can be specified by mask option in 1-bit units (mask ROM versions only). The $\mu$ PD78F4976A does not have pull-down resistors, however.	Input	FIP24 to FIP31
P90 to P97	I/O	Port 9. P-ch open-drain 8-bit high-voltage I/O port. Input/output can be specified in 1-bit units. On-chip pull-down resistor can be specified by mask option in 1-bit units (mask ROM versions only). The $\mu$ PD78F4976A does not have pull-down resistors, however.	Input	FIP32 to FIP39
P100 to P107	Output	Port 10. P-ch open-drain 8-bit high-voltage output port. On-chip pull-down resistor can be specified by mask option in 1-bit units (mask ROM versions only). The $\mu$ PD78F4976A does not have pull-down resistors, however.	—	FIP40 to FIP47

(2) Non-port pins (1/2)

Pin Name	I/O	Function	After Reset	Alternate Function	
INTP0	Input	External interrupt request input for which a valid edge (rising, falling, or both rising and falling) can be specified.	Input	P64	
INTP1				P65	
INTP2				P67	
SI0	Input	Serial data output (3-wire serial I/O 0)	Input	P25	
SO0	Output	Serial data input (3-wire serial I/O 0)		P26	
$\overline{\text{SCK0}}$	I/O	Serial clock input/output (3-wire serial I/O 0)		P27	
SI1	Input	Serial data input (3-wire serial I/O 1)		P60	
SO1	Output	Serial data output (3-wire serial I/O 1)		P61	
$\overline{\text{SCK1}}$	I/O	Serial clock input/output (3-wire serial I/O 1)		P62	
SI2	Input	Serial data input (3-wire serial I/O2)		P55	
SO2	Output	Serial data output (3-wire serial I/O2)		P56	
$\overline{\text{SCK2}}$	I/O	Serial clock input/output (3-wire serial I/O2)		P57	
RxD0	Input	Serial data input (UART)		P25/SI0	
TxD0	Output	Serial data output (UART)		P26/SO0	
ASCK0	Input	Serial clock input (UART)		P27/ $\overline{\text{SCK0}}$	
TI00	Input	External count clock input to 16-bit timer/event counter 0 (TM0) or capture trigger signal input to the 16-bit capture/compare register (CR00/CR01), or remote controller signal input		P20	
TIO50	I/O	External count clock input to or timer output from the 8-bit PWM timer (TM50)		P63	
TIO51		External count clock input to or timer output from the 8-bit PWM timer (TM51) or capture trigger signal input to the 16-bit capture/compare register (CR00)		P66	
ANI0 to ANI7	Input	Analog voltage input for A/D converter		—	P10 to P17
ANI8 to ANI11			P00 to P03		
★ AV <sub>DD</sub>	—	Analog supply voltage for A/D converter.	—	—	
AV <sub>SS</sub>		Ground potential for A/D converter. To be set to the same potential as V <sub>SS1</sub>			
FIP0 to FIP15	Output	High-voltage high-current output of VFD controller/driver.	Output	—	
FIP16 to FIP23			Input		P70 to P77
FIP24 to FIP31					P80 to P87
FIP32 to FIP39					P90 to P97
FIP40 to FIP47			—		P100 to P107
V <sub>LOAD</sub>	—	Pull-down resistor connection of VFD controller/driver.	—	—	
$\overline{\text{RESET}}$	Input	System reset input.	—	—	
X1		Crystal connection for main system clock oscillation.			
X2	—				
V <sub>DD0</sub>	Positive power supply to ports.				
V <sub>DD1</sub>	Positive power supply (except ports, analog block, and VFD controller/driver)				
V <sub>DD2</sub>	Positive power supply to VFD controller/driver.				
V <sub>SS0</sub>	Ground potential for ports.				
V <sub>SS1</sub>	Ground potential (except ports and analog block).				

**(2) Non-port pins (2/2)**

Pin Name	I/O	Function	After Reset	Alternate Function
$V_{PP}^{\text{Note}}$	—	High voltage is applied to this pin when program is written/verified. In the normal operation mode, directly connect this pin to $V_{SS1}$ .	—	—
IC		Internally connected. Directly connect this pin to $V_{SS1}$ .		

**Note**  $V_{PP}$  is provided to the  $\mu\text{PD78F4976A}$  only.

## 2.2 Pin Functions

### 2.2.1 P00 to P03 (Port 0)

P00 to P03 constitute a 4-bit input port. These pins function alternately as A/D converter analog inputs. The following operation modes can be specified in 1-bit units.

**(1) Port mode**

In this mode, P00 to P03 function as a 4-bit input port.

**(2) Control mode**

In this mode, P00 to P03 function as A/D converter analog input pins (ANI8 to ANI11).

### 2.2.2 P10 to P17 (Port 1)

P10 to P17 constitute an 8-bit input port. These pins function alternately as A/D converter analog inputs. The following operation modes can be specified in 1-bit units.

**(1) Port mode**

In this mode, P10 to P17 functions as an 8-bit input port.

**(2) Control mode**

In this mode, P10 to P17 function as analog input pins (ANI0 to ANI7) of the A/D converter.

### 2.2.3 P20, P25 to P27 (Port 2)

P20 and P25 to P27 constitute a 4-bit I/O port. These pins function alternately as data input/output for serial interface 0, asynchronous serial interface, clock, and data input for the timer signal.

The following operation modes can be specified in 1-bit units.

**(1) Port mode**

In this mode, P20 and P25 to P27 function as a 4-bit I/O port. These pins can be set in the input or output mode in 1-bit units by using the port 2 mode register (PM2). The on-chip pull-up resistor can be specified in 1-bit units using the pull-up resistor option register 2 (PU2).

**(2) Control mode**

In this mode, P20 and P25 to P27 function as data input/output for serial interface 0, the asynchronous serial interface, the clock, and data input for the timer signal.

**(a) SI0, SO0**

These are serial data I/O pins of the serial interface 0.

**(b)  $\overline{\text{SCK0}}$**

This is a serial clock I/O pin of the serial interface 0.

**(c) TI00**

This is a timer input pin of 16-bit timer counter 0 (TM0).

**(d) RxD0, TxD0**

These are serial data I/O pins of the asynchronous serial interface.

**(e) ASCK0**

These are baud rate clock I/O pins of the asynchronous serial interface.

**2.2.4 P40 to P47 (Port 4)**

P40 to 47 constitute an 8-bit I/O port. These pins can be set in the input or output mode in 1-bit units using the port 4 mode register (PM4). The on-chip pull-up resistor can be specified in 8-bit units using bit 4 (PUO4) of the pull-up resistor option register (PUO) only when it is used as an input port.

This port can directly drive an LED.

**2.2.5 P50 to P57 (Port 5)**

P50 to P57 constitute an 8-bit I/O port. These pins function alternately as a data input/output for serial interface 2 and clock input/output.

The following operation modes can be specified in 1-bit units.

**(1) Port mode**

In this mode, port 6 functions as an 8-bit I/O port. The port 5 mode register (PM5) can specify whether the port functions as an input or output port, in 1-bit units.

These pins are N-ch open-drain pins. Pull-up resistors can be specified for these pins in the mask ROM versions in 1-bit units using a mask option. The  $\mu$ PD78F4976A does not have pull-up resistors.

**(2) Control mode**

In this mode, port 5 is provided with functions such as data input/output for serial interface 2 and clock input/output.

**(a) SI2 and SO2**

These pins function as serial data I/O pins for serial interface 2.

**(b)  $\overline{\text{SCK2}}$** 

This pin functions as a serial clock I/O pin for serial interface 2.

**2.2.6 P60 to P67 (Port 6)**

P60 to P67 constitute an 8-bit I/O port. These pins function alternately as data input/output for serial interface 1, clock input/output, timer input/output, and external interrupt request input.

The following operation modes can be specified in 1-bit units.

**(1) Port mode**

In this mode, port 6 functions as an 8-bit I/O port. The port 6 mode register (PM6) can specify whether the port functions as an input or output port in 1-bit units.

Use of on-chip pull-up resistors can be specified in 8-bit units using bit 6 of the pull-up resistor option register (PUO) only when port 6 is used as an input port.

**(2) Control mode**

In this mode, port 6 is provided with functions such as data input/output for serial interface 1, clock input/output, timer input/output, timer capture trigger signal input, and external interrupt request input.

**(a) SI1 and SO1**

These pins function as serial data I/O pins for serial interface 1.

**(b)  $\overline{\text{SCK1}}$** 

This pin functions as a serial clock I/O pin for serial interface 1.

**(c) TIO50 and TIO51**

TIO50: This pin functions as an external count clock input pin and timer output pin for the 8-bit PWM timer.

TIO51: This pin functions as an external count clock input pin and timer output pin for the 8-bit PWM timer as well as a capture trigger signal input pin for the 16-bit capture/compare register 00 (CR00).

**(d) INTP0, INTP1, and INTP2**

These pins function as external interrupt request input pins, for which a valid edge (rising, falling, or rising and falling) can be specified.

**2.2.7 P70 to P77 (Port 7)**

P70 to P77 constitute an 8-bit I/O port. These pins are also used as VFD controller/driver output pins. The following operation modes can be specified in 1-bit units.

**(1) Port mode**

In this mode, P70 to P77 function as an 8-bit I/O port.

These pins are P-ch open-drain pins. Pull-down resistors can be specified for these pins in the mask ROM versions in 1-bit units using a mask option. The  $\mu\text{PD78F4976A}$  does not have pull-down resistors.

**(2) Control mode**

In this mode, P70 to P77 function as the output pins of the VFD controller/driver (FIP16 to FIP23).

**2.2.8 P80 to P87 (Port 8)**

P80 to P87 constitute an 8-bit I/O port. These pins are also used as VFD controller/driver output pins. The following operation modes can be specified in 1-bit units.

**(1) Port mode**

In this mode, P80 to P87 function as an 8-bit I/O port.

These pins are P-ch open-drain pins. Pull-down resistors can be specified for these pins in the mask ROM versions in 1-bit units using a mask option. The  $\mu\text{PD78F4976A}$  does not have pull-down resistors.

**(2) Control mode**

In this mode, P80 to P87 function as the output pins of the VFD controller/driver (FIP24 to FIP31).

**2.2.9 P90 to P97 (Port 9)**

P90 to P97 constitute an 8-bit I/O port. These pins are also used as VFD controller/driver output pins. The following operation modes can be specified in 1-bit units.

**(1) Port mode**

In this mode, P90 to P97 function as an 8-bit I/O port.

These pins are P-ch open-drain pins. Pull-down resistors can be specified for these pins in the mask ROM versions in 1-bit units using a mask option. The  $\mu\text{PD78F4976A}$  does not have pull-down resistors.

**(2) Control mode**

In this mode, P90 to P97 function as the output pins of the VFD controller/driver (FIP32 to FIP39).

**2.2.10 P100 to P107 (Port 10)**

P100 to P107 constitute an 8-bit output port. These pins are also used as VFD controller/driver output pins. The following operation modes can be specified in 1-bit units.

**(1) Port mode**

In this mode, P100 to P107 function as an 8-bit output port.

These pins are P-ch open-drain pins. Pull-down resistors can be specified for these pins in the mask ROM versions in 1-bit units using a mask option. The  $\mu$ PD78F4976A does not have pull-down resistors.

**(2) Control mode**

In this mode, P100 to P107 function as the output pins of the VFD controller/driver (FIP40 to FIP47).

**2.2.11 FIP0 to FIP15**

These are the output pins of the VFD controller/driver.

**2.2.12 V<sub>LOAD</sub>**

This pin connects a pull-down resistor to the VFD controller/driver.

**2.2.13 AV<sub>DD</sub>**

This pin supplies an analog voltage to the A/D converter.

Always keep this pin at the same potential as the V<sub>DD1</sub> pin even when the A/D converter is not used.

**2.2.14 AV<sub>SS</sub>**

This is the ground pin of the A/D converter.

- ★ When the A/D converter is used (ADCS = 1), use the AV<sub>DD</sub> pin with the same potential as V<sub>DD1</sub>.
- ★ When the A/D converter is not used (ADCS = 0), the AV<sub>DD</sub> pin can be used with the same potential as V<sub>SS1</sub>.

**2.2.15  $\overline{\text{RESET}}$** 

This pin inputs an active-low system reset signal.

**2.2.16 X1 and X2**

These pins connect a crystal resonator for main system clock oscillation.

To supply an external clock, input it to X1, and input a signal reverse to that input to X1, to X2.

**2.2.17 V<sub>DD0</sub> to V<sub>DD2</sub>**

V<sub>DD0</sub> supplies a positive voltage to the ports.

V<sub>DD1</sub> supplies a positive voltage to the internal function blocks other than the ports, analog block, and VFD controller/driver.

V<sub>DD2</sub> supplies a positive voltage to the VFD controller/driver.

**2.2.18 V<sub>SS0</sub> and V<sub>SS1</sub>**

V<sub>SS0</sub> is the ground pin for the ports.

V<sub>SS1</sub> is the ground pin for the internal function blocks other than the ports and analog block.

**2.2.19 V<sub>PP</sub> ( $\mu$ PD78F4976A only)**

A high voltage is applied to this pin when the flash memory programming mode is used and when a program is written or verified.

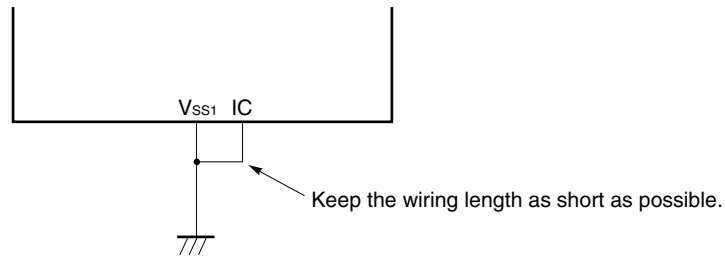
Directly connect this pin to V<sub>SS1</sub> in the normal operation mode.

**2.2.20 IC (Mask ROM product only)**

The IC (Internally Connected) pin sets a test mode in which the  $\mu$ PD784975A is tested before shipment. Usually, connect the IC pin directly to  $V_{SS1}$  with as short a wiring length as possible.

If there is a potential difference between the IC and  $V_{SS1}$  pins because the wiring length between the IC and  $V_{SS1}$  pin is too long, or external noise is superimposed on the IC pin, your program may not run correctly.

- Directly connect the IC pin to the  $V_{SS1}$ .**





### 2.3 Pin I/O Circuits and Connections of Unused Pins

The I/O circuit type of each pin and recommended connections of unused pins are shown in Table 2-1. For the configuration of each I/O circuit type, refer to Figure 2-1.

**Table 2-1. Types of Pin I/O Circuits and Recommended Connection of Unused Pins**

Pin Name	I/O Circuit Type	I/O	Recommended Connection of Unused Pins
P00/ANI8 to P03/ANI11	9	Input	Connect to $V_{DD0}$ or $V_{SS0}$
P10/ANI0 to P17/ANI7			
P20/TI00	8-C	I/O	Input: Independently connect to $V_{SS0}$ via a resistor Output: Leave open
P25/SI0/RxD0			
P26/SO0/TxD0	5-H		
P27/ $\overline{\text{SCK0}}$ /ASCK0	8-C		
P40 to P47	5-H		
P60/SI1	8-C		
P61/SO1	5-H		
P62/ $\overline{\text{SCK1}}$	8-C		
P63/TIO50			
P64/INTP0			
P65/INTP1			
P66/TIO51			
P67/INTP2			
Mask ROM version			
P50 to P54, P55/SI2, P56/SO2, P57/ $\overline{\text{SCK2}}$	13-J	I/O	Input: Independently connect to $V_{DD0}$ via a resistor Output: Leave open
P70/FIP16 to P77/FIP23	15-F	I/O	Input: Independently connect to $V_{SS0}$ via a resistor Output: Leave open
P80/FIP24 to P87/FIP31			
P90/FIP32 to P97/FIP39			
P100/FIP40 to P107/FIP47	14-F	Output	Leave open
IC	—	—	Directly connect to $V_{SS1}$
$\mu$ PD78F4976A			
P50 to P54, P55/SI2, P56/SO2, P57/ $\overline{\text{SCK2}}$	13-K	I/O	Input: Independently connect to $V_{DD0}$ via a resistor Output: Leave open
P70/FIP16 to P77/FIP23	15-E	I/O	Input: Independently connect to $V_{SS0}$ via a resistor Output: Leave open
P80/FIP24 to P87/FIP31			
P90/FIP32 to P97/FIP39			
P100/FIP40 to P107/FIP47	14-E	Output	Leave open
$V_{PP}$	—	—	Directly connect to $V_{SS1}$
FIP0 to FIP15	14-C	Output	Leave open
$\overline{\text{RESET}}$	2	Input	—
$A_{VDD}$	—	—	Connect to $V_{DD1}$ or $V_{SS1}$
$A_{VSS}$			Connect to $V_{SS1}$
$V_{LOAD}$			

★

Figure 2-1. Types of Pin I/O Circuits (1/2)

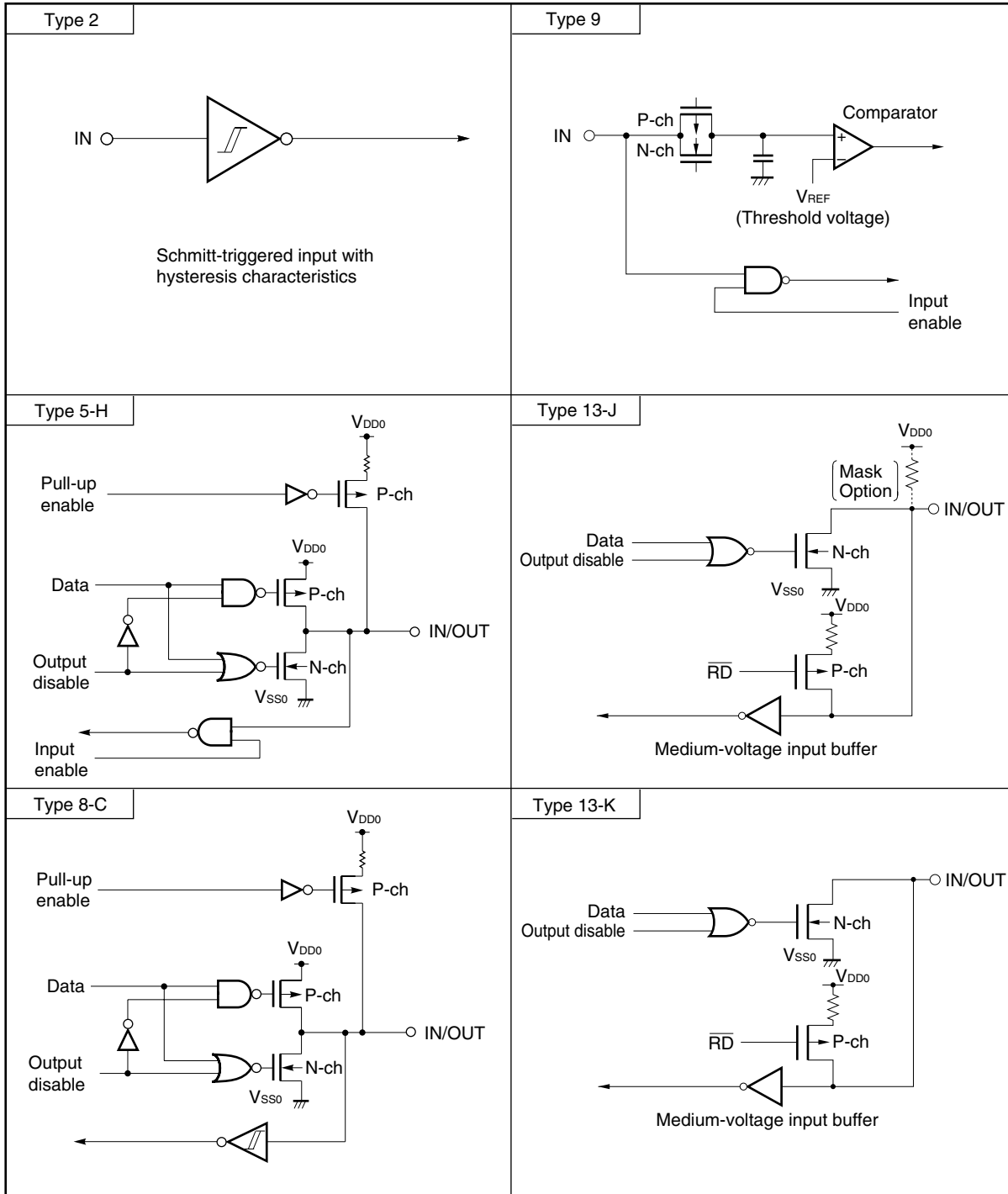
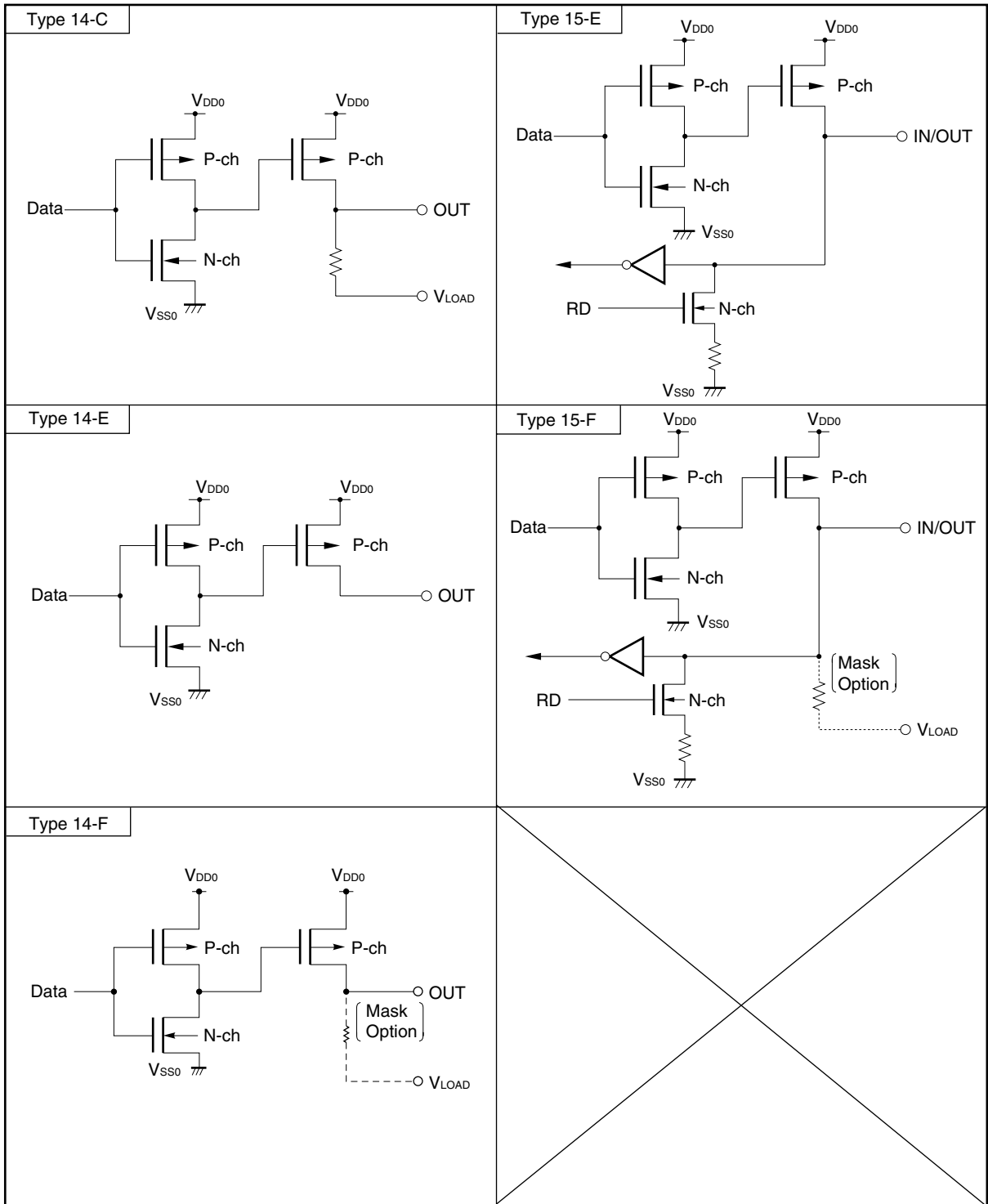


Figure 2-1. Types of Pin I/O Circuits (2/2)



## CHAPTER 3 CPU ARCHITECTURE

### 3.1 Memory Space

The  $\mu$ PD784975A can access a 1 MB space. The mapping of the internal data space differs with the LOCATION instruction (special function register and internal RAM). The LOCATION instruction must always be executed after clearing a reset and cannot be used more than once.

The program after clearing a reset must be as follows.

```

RSTVCT    CSEG AT 0
          DW    RSTSTRT

          to

INITSEG   CSEG BASE
RSTSTRT:  LOCATION 0H; or LOCATION 0FH
          MOVG  SP, #STKBGN
MOV       MM, #80H
    
```

After clearing a reset, set the memory expansion mode register (MM) to 80H.

MM is used for selecting the speed at which instructions are fetched from internal ROM.

8-bit memory manipulation instructions are used to read and write MM.

Figure 3-1 shows the format of MM.

$\overline{\text{RESET}}$  input sets MM to 20H.

**Figure 3-1. Format of Memory Expansion Mode Register (MM)**

Address: 0FFC4H After reset: 20H R/W

Symbol	7	6	5	4	3	2	1	0
MM	IFCH	0	AW	0	0	0	0	0

IFCH	AW	Selection of speed instruction fetching from internal ROM
0	1	Low-speed mode (speed at which one byte is fetched every six cycles)
1	0	Normal-speed mode (speed at which two bytes are fetched every two cycles)
Other than above		Setting prohibited

**Caution** After clearing a reset, be sure to set this register to 80H.

(1) When the LOCATION 0H instruction is executed

- **Internal memory**

The internal data area and internal ROM area are as follows.

Part Number	Internal Data Area	Internal ROM Area
μPD784975A	0F100H to 0FFFFH 0EA00H to 0EA5FH	00000H to 0E9FFH 10000H to 17FFFH
μPD78F4976A	0EB00H to 0FFFFH 0EA00H to 0EA5FH	00000H to 0E9FFH 10000H to 1FFFFH

**Remark** The following area, that overlaps the internal data area, cannot be used while the LOCATION 0H instruction is executing.

Part Name	Unused Area
μPD784975A	0EA00H to 0FFFFH (5,632 bytes)
μPD78F4976A	

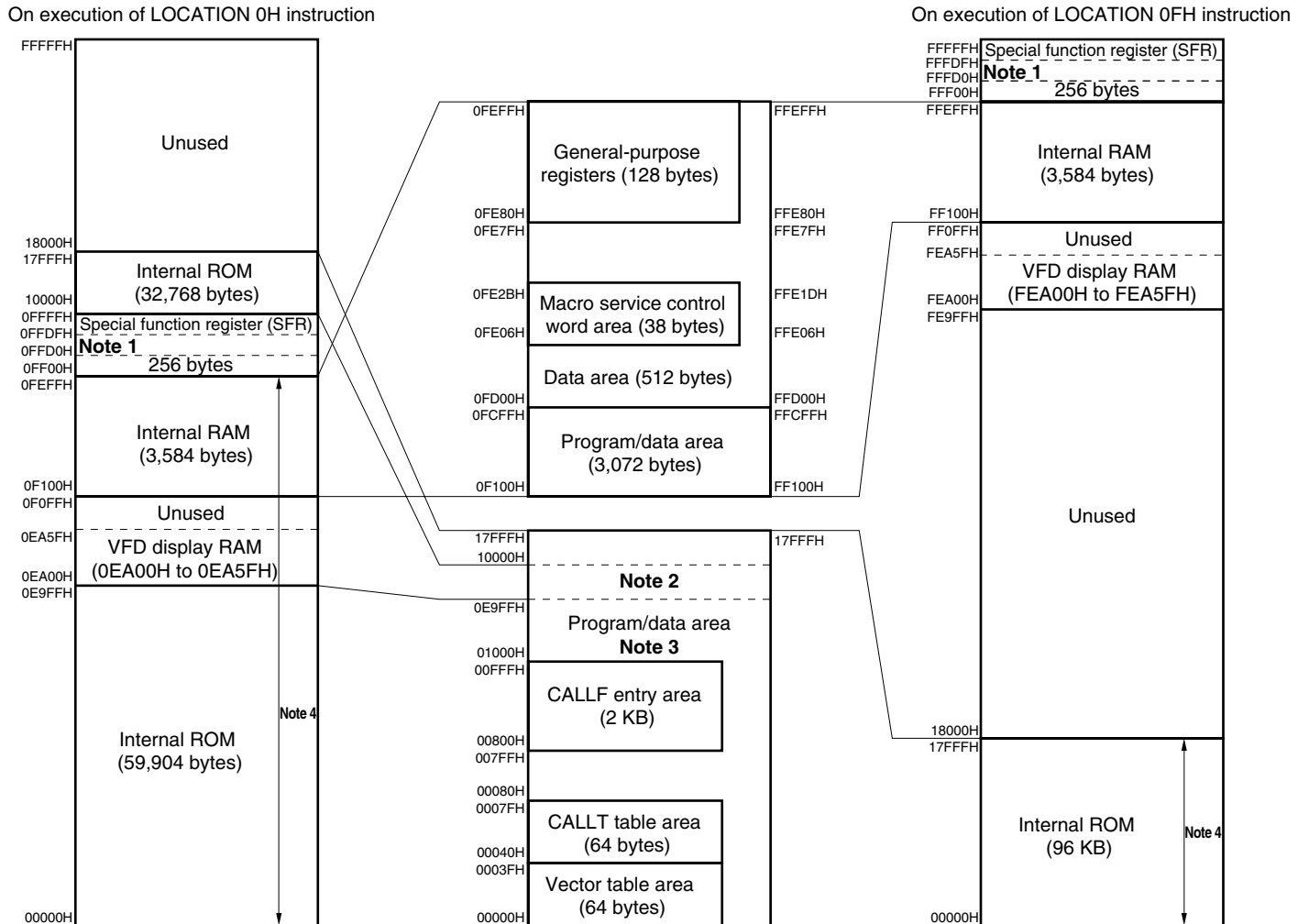
(2) When the LOCATION 0FH instruction is executed

- **Internal memory**

The internal data area and internal ROM area are as follows.

Part Number	Internal Data Area	Internal ROM Area
μPD784975A	FF100H to FFFFFH FEA00H to FEA5FH	00000H to 17FFFH
μPD78F4976A	FEB00H to FFFFFH FEA00H to FEA5FH	00000H to 1FFFFH

Figure 3-2. Memory Map of  $\mu$ PD784975A



- Notes**
1. Unused area
  2. The 5,632 bytes in this area can be used as internal ROM only when the LOCATION 0FH instruction is executing.
  3. LOCATION 0H instruction execution: 92,672 bytes; LOCATION 0FH instruction execution: 98,304 bytes
  4. This is the base area. It is used as an entry area upon the occurrence of a reset (except for internal RAM) or interrupt.



### 3.2 Internal ROM Area

The following products in the  $\mu$ PD784976A Subseries have internal ROMs that can store the programs and table data.

If the internal ROM area or internal data area overlap when the LOCATION 0H instruction is executing, the internal data area becomes the access target. The internal ROM area in the overlapping part cannot be accessed.

Part Number	Internal ROM	Address Area	
		LOCATION 0H Instruction	LOCATION 0FH Instruction
$\mu$ PD784975A	96 K $\times$ 8 bits (Mask ROM)	00000H to 0E9FFH 10000H to 17FFFH	00000H to 17FFFH
$\mu$ PD78F4976A	128 K $\times$ 8 bits (Flash memory)	00000H to 0E9FFH 10000H to 1FFFFH	00000H to 1FFFFH

The internal ROM can be accessed at high speed. Usually, a fetch is at the same speed as an external ROM fetch. By setting (to 1) the IFCH bit of the memory expansion mode register (MM), the high-speed fetch function is used. An internal ROM fetch is a high-speed fetch (fetch in two system clocks in 2-byte units).

If the instruction execution cycle similar to the external ROM fetch is selected, waits are inserted by the wait function. However, when a high-speed fetch is used, waits cannot be inserted for the internal ROM. However, do not set external waits for the internal ROM area. If an external wait is set for the internal ROM area, the CPU enters the deadlock state. The deadlock state is only released by a reset input.

$\overline{\text{RESET}}$  input causes an instruction execution cycle similar to the external ROM fetch cycle.



### 3.3 Base Area

The area from 0 to FFFFH becomes the base area. The base area is the target in the following uses.

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

This base area is allocated in the vector table area, CALLT instruction table area, and CALLF instruction entry area.

When the LOCATION 0H instruction is executing, the internal data area is placed in the base area. Be aware that the program is not fetched from the internal high-speed RAM area and special function register (SFR) area in the internal data area. Also, use the data in the internal RAM area after initialization.

### 3.3.1 Vector table area

The 64-byte area from 00000H to 0003FH is reserved as the vector table area. The program start addresses for branching by interrupt requests and  $\overline{\text{RESET}}$  input are stored in the vector table area. If context switching is used by each interrupt, the register bank number of the switch destination is stored.

The portion that is not used as the vector table can be used as program memory or data memory.

The values written in the vector table are 16-bit values. Therefore, branching can only be to the base area.

**Table 3-1. Vector Table Address**

Interrupt Source	Vector Table Address
BRK instruction	003EH
Operand error	003CH
INTWDT (non-maskable)	0004H
INTWDT (maskable)	0006H
INTP0	0008H
INTP1	000AH
INTP2	000CH
INTTM00	000EH
INTTM01	0010H
INTKS	0012H
INTCSI1	0016H
INTTM50	0018H
INTTM51	001AH
INTAD	001CH
INTREM	00IEH
INTCSI2	0020H
INTSER0	0022H
INTSR0	0024H
INTST0	0026H
INTWTI	0028H
INTWT	002AH

### 3.3.2 CALLT instruction table area

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

In a CALLT instruction, this table is referenced and the base area address written in the table is branched to as the subroutine. Since a CALLT instruction is one byte, many subroutine call descriptions in the program can be CALLT instructions, so the object size of the program can be reduced. Since a maximum of 32 subroutine entry addresses can be described in the table, they should be registered in order from the most frequently described.

When not used as the CALLT instruction table, the area can be used as normal program memory or data memory.

### 3.3.3 CALLF instruction entry area

The area from 00800H to 00FFFH can be for direct subroutine calls in the 2-byte call instruction (CALLF).

Since a CALLF instruction is a 2-byte call instruction, compared to when using the CALL instruction (3 bytes or 4 bytes) of a direct subroutine call, the object size can be reduced.

When you want to achieve high speed, describing direct subroutines in this area is effective.

If you want to decrease the object size, an unconditional branch (BR) is described in this area, and the actual subroutine is placed outside of this area. When a subroutine is called from five or more locations, reducing the object size is attempted. In this case, since only a 4-byte location for the BR instruction is used in the CALLF entry area, the object size of many subroutines can be reduced.

### 3.4 Internal Data Area

The internal data space consists of the internal RAM area and the special function register area (refer to **Figures 3-1** and **3-2**).

The final address in the internal data area can be set to 0FFFFH (when executing the LOCATION 0H instruction) or FFFFFH (when executing the LOCATION 0FH instruction) by the LOCATION instruction. The address selection of the internal data area by this LOCATION 0H must be executed once immediately after a reset is cleared. After one selection, updating is not possible. The program following a reset clear must be as shown in the example. If the internal data area and another area are allocated to the same address, the internal data area becomes the access target, and the other area cannot be accessed.

```

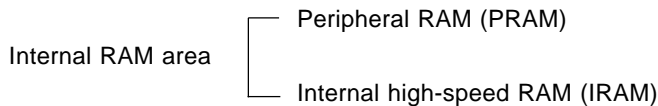
Example  RSTVCT    CSEG AT 0
           DW        RSTSTRT
           to

           INITSEG  CSEG  BASE
           RSTSTRT: LOCATION 0H ; or LOCATION 0FH
           MOVG SP, #STKBGN
           MOV MM, #80H
    
```

**Caution** When the LOCATION 0H instruction is executing, the program after clearing the reset must not overlap the internal data area. In addition, make sure the entry address of the servicing routine for a non-maskable interrupt does not overlap the internal data area. The entry area for a maskable interrupt must be initialized before referencing the internal data area.

#### 3.4.1 Internal RAM area

The  $\mu$ PD784975A has an on-chip general-purpose static RAM. This space has the following configuration.



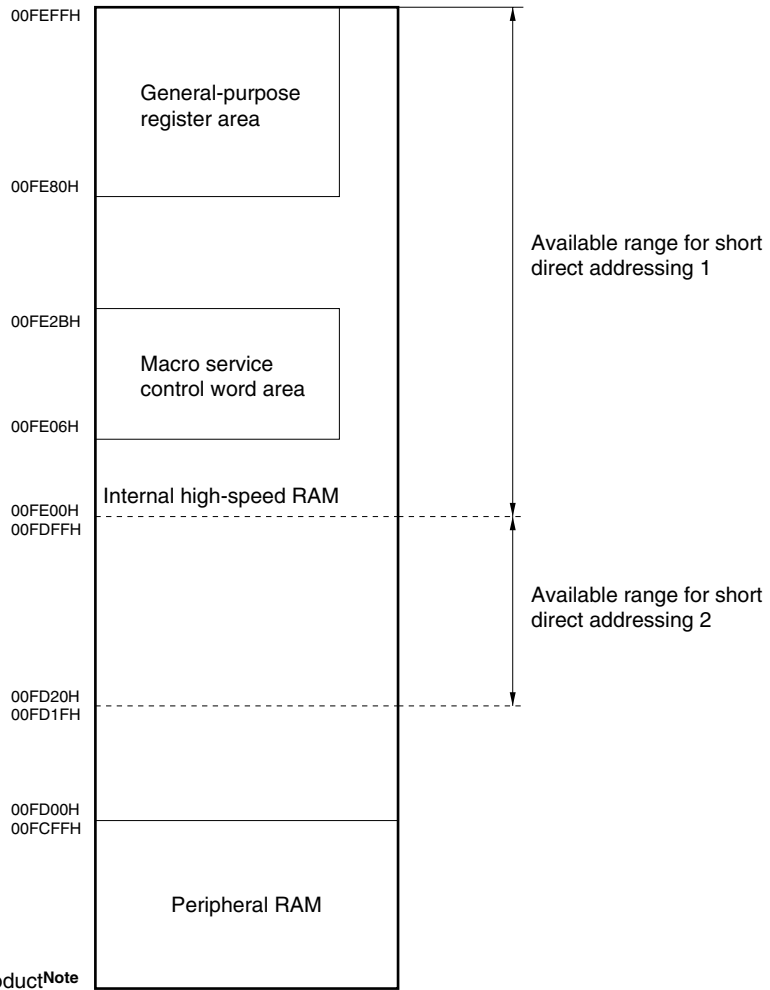
**Table 3-2. Internal RAM Area List**

Internal RAM Part Number	Internal RAM Area		
		Peripheral RAM: PRAM	Internal High-speed RAM: IRAM
$\mu$ PD784975A	3,584 bytes (0F100H to 0FEFFH)	3,072 bytes (0F100H to 0FCFFH)	512 bytes (0FD00H to 0FEFFH)
$\mu$ PD78F4976A	5,120 bytes (0EB00H to 0FEFFH)	4,608 bytes (0EB00H to 0FCFFH)	

**Remark** The addresses in the table are the values when the LOCATION 0H instruction is executing. When the LOCATION 0FH instruction is executing, 0F0000H is added to the above values.

Figure 3-4 shows the internal RAM memory map.

**Figure 3-4. Memory Map of Internal RAM**



**Note**  $\mu$ PD784975A: 00F100H  
 $\mu$ PD78F4976A: 00EB00H

**Remark** The addresses in the figure are the values when the LOCATION 0H instruction is executing. When the LOCATION 0FH instruction is executing, 0F0000H is added to the above values.

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

The internal high-speed RAM can be accessed at high speed. FD20H to FEFFH can use the short direct addressing mode for high-speed access. The two short direct addressing modes are short direct addressing 1 and short direct addressing 2 that are based on the address of the target. Both addressing modes have the same function. In a portion of the instructions, short direct addressing 2 has a shorter word length than short direct addressing 1. For details, refer to **78K/IV Series User's Manual Instruction (U10905E)**.

A program cannot be fetched from IRAM. If a program is fetched from an address that is mapped by IRAM, the CPU runs wild.

The following areas are reserved in IRAM.

- General-purpose register area: FE80H to FEFFH
- Macro service control word area: FE06H to FE2BH
- Macro service channel area: FE00H to FEFFH (The address is set by a macro service control word.)

When reserved functions are not used in these areas, they can be used as normal data memory.

**Remark** The addresses in this text are the addresses when the LOCATION 0H instruction is executing. When the LOCATION 0FH instruction is executing, 0F0000H is added to the values in this text.

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

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

A program fetch from the peripheral RAM is high speed and can occur in two clocks in 2-byte units.

**3.4.2 Special function register (SFR) area**

The special function register (SFR) of the on-chip peripheral hardware is mapped to the area from 0FF00H to 0FFFFH (refer to **Figures 3-2** and **3-3**).

**Caution** In this area, do not access an address that is not mapped in SFR. If mistakenly accessed, the CPU enters the deadlock state. The deadlock state is released only by reset input.

**Remark** The addresses in this text are the addresses only when the LOCATION 0H instruction is executing. If the LOCATION 0FH instruction is executing, 0F0000H is added to the values in the text.

### 3.5 $\mu$ PD78F4976A Memory Mapping

The  $\mu$ PD78F4976A has a 128 KB flash memory and 5,120-byte internal RAM.

The  $\mu$ PD78F4976A has a function (memory size switching function) so that a part of the internal memory is not used by the software.

The memory size is switched by the internal memory size switching register (IMS).

Based on the IMS setting, the memory mapping can be the same memory mapping as the mask ROM products with different internal memories (ROM, RAM).

IMS can only be written by an 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$  input sets IMS to FFH.

**Figure 3-5. Format of Internal Memory Size Switching Register (IMS)**

Address: 0FFFCH After reset: FFH W

Symbol	7	6	5	4	3	2	1	0
IMS	1	1	ROM1	ROM0	1	1	RAM1	RAM0

ROM1	ROM0	Internal ROM capacity selection
0	0	Setting prohibited
0	1	Setting prohibited
1	0	96 KB
1	1	128 KB

RAM1	RAM0	Internal RAM capacity selection <sup>Note</sup>
0	0	Setting prohibited
0	1	3,584 bytes
1	0	Setting prohibited
1	1	5,120 bytes

**Note** The internal RAM capacity is the sum of the peripheral RAM capacity and high-speed RAM capacity.

**Cautions 1.** The mask ROM version ( $\mu$ PD784975A) does not have an IMS.

Even if the IMS write instruction is executed in the mask ROM version, it does not have any effect on operations.

**2.** In the case that the  $\mu$ PD78F4976A is selected as the emulation CPU in the in-circuit emulator, the memory size would always be “FFH” even if a write instruction other than FFH is executed to IMS.

Table 3-3 shows the IMS settings that have the same memory map as the mask ROM version.

**Table 3-3. Settings of the Internal Memory Size Switching Register (IMS)**

Target Mask ROM Version	IMS Setting
$\mu$ PD784975A	EDH

### 3.6 Control Registers

The control registers are the program counter (PC), program status word (PSW), and stack pointer (SP).

#### 3.6.1 Program counter (PC)

This is a 20-bit binary counter that saves address information about the program to be executed next (refer to **Figure 3-6**).

Usually, this counter is automatically incremented based on the number of bytes in the instruction to be fetched. When the instruction that is branched is executed, the immediate data or register contents are set.

$\overline{\text{RESET}}$  input sets the lower 16 bits of the PC to the 16-bit data at addresses 0 and 1, and 0000 in the higher 4 bits of the PC.

**Figure 3-6. Format of Program Counter (PC)**



#### 3.6.2 Program status word (PSW)

The program status word (PSW) is a 16-bit register that consists of various flags that are set and reset based on the result of the instruction execution.

A read or write access is performed in units of higher 8 bits (PSWH) and lower 8 bits (PSWL). In addition, bit manipulation instructions can manipulate each flag.

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

$\overline{\text{RESET}}$  input resets all of the bits to 0.

Always write 0 in the bits indicated by "0" in Figure 3-7. The contents of bits indicated by "-" are undefined when read.



**Figure 3-7. Format of Program Status Word (PSW)**

Symbol	7	6	5	4	3	2	1	0
PSWH	UF	RBS2	RBS1	RBS0	—	—	—	—
PSWL	S	Z	RSS	AC	IE	P/V	0	CY

Each flag is described below.

**(1) Carry flag (CY)**

This is the flag that stores the carry or borrow of an operation result.

When a shift rotate instruction is executed, the shifted out value is stored. When a bit manipulation instruction is executed, this flag functions as the bit accumulator.

The CY flag state can be tested by a conditional branch instruction.

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

The P/V flag has the following two actions in accordance with the execution of the operation instruction.

The state of the P/V flag can be tested by a conditional branch instruction.

- **Parity flag action**

The results of executing the logical instructions, shift rotate instructions, and CHKL and CHKLA instructions are set to 1 when an even number of bits is set to 1. If the number of bits is odd, the result is reset to 0. However, for 16-bit shift instructions, the parity flag from only the lower 8 bits of the operation result is valid.

- **Overflow flag action**

The result of executing an arithmetic operation instruction is set to 1 only when the numerical range expressed in two's complement is exceeded. Otherwise, the result is reset to 0. Specifically, the result is the exclusive Or of the carry from the MSB and the carry to the MSB and becomes the flag contents. For example, in 8-bit arithmetic operations, the two's complement range is 80H (−128) to 7FH (+127). If the operation result is outside this range, the flag is set to 1. If inside the range, it is reset to 0.

**Example** The action of the overflow flag when an 8-bit addition instruction is executed is described next.

When 78H (+120) and 69H (+105) are added, the operation result becomes E1H (+225). Since the upper limit of two's complement is exceeded, the P/V flag is set to 1. In a two's complement expression, E1H becomes -31.

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

Next, since the operation result of the addition of the following two negative numbers falls within the two's complement range, the P/V flag is reset to 0.

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

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

This flag controls the CPU interrupt request acknowledgement.

If IE is 0, interrupts are disabled, and only non-maskable interrupts and unmasked macro services can be accepted. Otherwise, everything is disabled.

If IE is 1, the interrupt enable state is entered. Enabling the acknowledgement of interrupt requests is controlled by the interrupt mask flags that correspond to each interrupt request and the priority of each interrupt.

This flag is set to 1 by executing the EI instruction and is reset to 0 by executing the DI instruction or acknowledging an interrupt.

**(4) Auxiliary carry flag (AC)**

If the operation result has a carry from bit 3 or a borrow to bit 3, this flag is set to 1. Otherwise, the flag is reset to 0.

This flag is used when the ADJBA and ADJBS instructions are executing.

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

This flag sets the general-purpose registers that function as X, A, C, and B and the general-purpose register pairs (16 bits) that function as AX and BC.

This flag is used to maintain compatibility with the 78K/III Series. Always set this flag to 0 except when using a 78K/III Series program.

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

This flag indicates that the operation result is 0.

If the operation result is 0, this flag is set to 1. Otherwise, it is reset to 0. The state of the Z flag can be tested by conditional branch instructions.

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

This flag indicates that the MSB in the operation result is 1.

The flag is set to 1 when the MSB of the operation result is 1. If 0, the flag is reset to 0. The S flag state can be tested by the conditional branch instructions.

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

This is the 3-bit flag that selects one of the eight register banks (register banks 0 to 7). (Refer to **Table 3-4.**)

Three bit information that indicates the register bank selected by executing the SEL RBn instruction is stored.

**Table 3-4. Register Bank Selection**

RBS2	RBS1	RBS0	Set 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 is set and reset by a user program and can be used in program control.

**3.6.3 Using the RSS bit**

Basically, always use with the RSS bit fixed at 0.

The following descriptions discuss using a 78K/III Series program and a program that sets the RSS bit to 1. Reading is not necessary if the RSS bit is fixed at 0.

The RSS bit enables the functions in A (R1), X (R0), B (R3), C (R2), AX (RP0), and BC (RP1) to also be used in registers R4 to R7 (RP2, RP3). When this bit is effectively used, efficient programs in terms of program size and program execution can be written.

Sometimes, however, unexpected problems arise if used carelessly. Consequently, always set the RSS bit to 0. Use with the RSS bit set to 1 only when 78K/III series programs will be used.

By setting the RSS bit to 0 in all programs, writing and debugging programs become more efficient.

Even if a program where the RSS bit is set to 1 is used, when possible, it is recommended to use the program after modifying the program so that the RSS bit is not set to 1.

**(1) Using the RSS bit**

- Registers used in instructions where the A, X, B, C, and AX registers are directly described in the operand column of the operation list (refer to **20.2**)
- Registers that are implicitly specified in instructions that use the A, AX, B, and C registers by implied addressing
- Registers that are used in addressing in instructions that use the A, B, and C registers in indexed addressing and based indexed addressing

The registers used in these cases are switched in the following ways by 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

The registers used in other cases always become the same registers regardless of the contents of the RSS bit. For registers A, X, B, C, AX, and BC in NEC assembler RA78K4, instruction code is generated for any register described by name or for registers set by an RSS pseudo instruction in the assembler.

When the RSS bit is set or reset, always specify an RSS pseudo instruction immediately before (or immediately after) that instruction (refer to the following examples).

**<Program examples>**

- When RSS = 0

```
RSS 0          ; RSS pseudo instruction
CLR1 PSWL. 5
MOV B, A      ; This description corresponds to "MOV R3, R1".
```

- When RSS = 1

```
RSS 1          ; RSS pseudo instruction
SET1 PSWL. 5
MOV B, A      ; This description corresponds to "MOV R7, R5".
```

**(2) Generation of instruction code in the RA78K4**

- In the RA78K4, when an instruction with the same function as an instruction that directly specifies A or AX in the operand column in the operation list of the instruction is used, the instruction code that directly describes A or AX in the operand column is given priority and generated.

**Example** The MOV A, r instruction where r is B has the same function as the MOV r, r' instruction where r is A and r' is B. In addition, they have the same (MOV A, B) description in the assembler source program. In this case, RA78K4 generates code that corresponds to the MOV A, r instruction.

- If A, X, B, C, AX, or BC is described in an instruction that specifies r, r', rp, or rp' in the operand column, the A, X, B, C, AX, or BC instruction code generates the instruction code that specifies the following registers based on the operand of the RSS pseudo instruction in RA78K4.

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 and RP0 to RP4 are specified in r, r', rp, and rp' in the operand column, an instruction code that conforms to the specification is output. (Instruction code that directly describes A or AX in the operand column is not output.)
- The A, B, and C registers that are used in indexed addressing and based indexed addressing cannot be described as R1, R3, R2, or R5, R7, R6.

**(3) Cautions on use**

Switching the RSS bit obtains the same effect as holding two register sets. However, be careful and write the program so that implicit descriptions in the program and dynamically changing the RSS bit during program execution always agree.

Also, since a program with RSS = 1 cannot be used in a program that uses context switching, the portability of the program becomes poor. Furthermore, since different registers having the same name are used, the readability of the program worsens, and debugging becomes difficult. Therefore, when RSS = 1 must be used, write the program while taking these problems into consideration.

A register that does not have the RSS bit set can be accessed by specifying the absolute name.

### 3.6.4 Stack pointer (SP)

The 24-bit register saves the starting address of the stack (LIFO: 00000H to FFFFFFFH) (refer to **Figure 3-8**). The stack is used for addressing during subroutine processing or interrupt servicing. Always set the higher 4 bits to zero.

The contents of the SP are decremented before writing to the stack area and incremented after reading from the stack (refer to **Figures 3-9** and **3-10**).

SP is accessed by special instructions.

Since the SP contents become undefined when  $\overline{\text{RESET}}$  is input, always initialize the SP from the initialization program immediately after clearing the reset (before accepting a subroutine call or interrupt).

#### Example Initializing SP

```
MOVG SP, #0FEE0H ; SP ← 0FEE0H (when used from FEDFH)
```

**Figure 3-8. Format of Stack Pointer (SP)**



Figure 3-9. Data Saved to the Stack

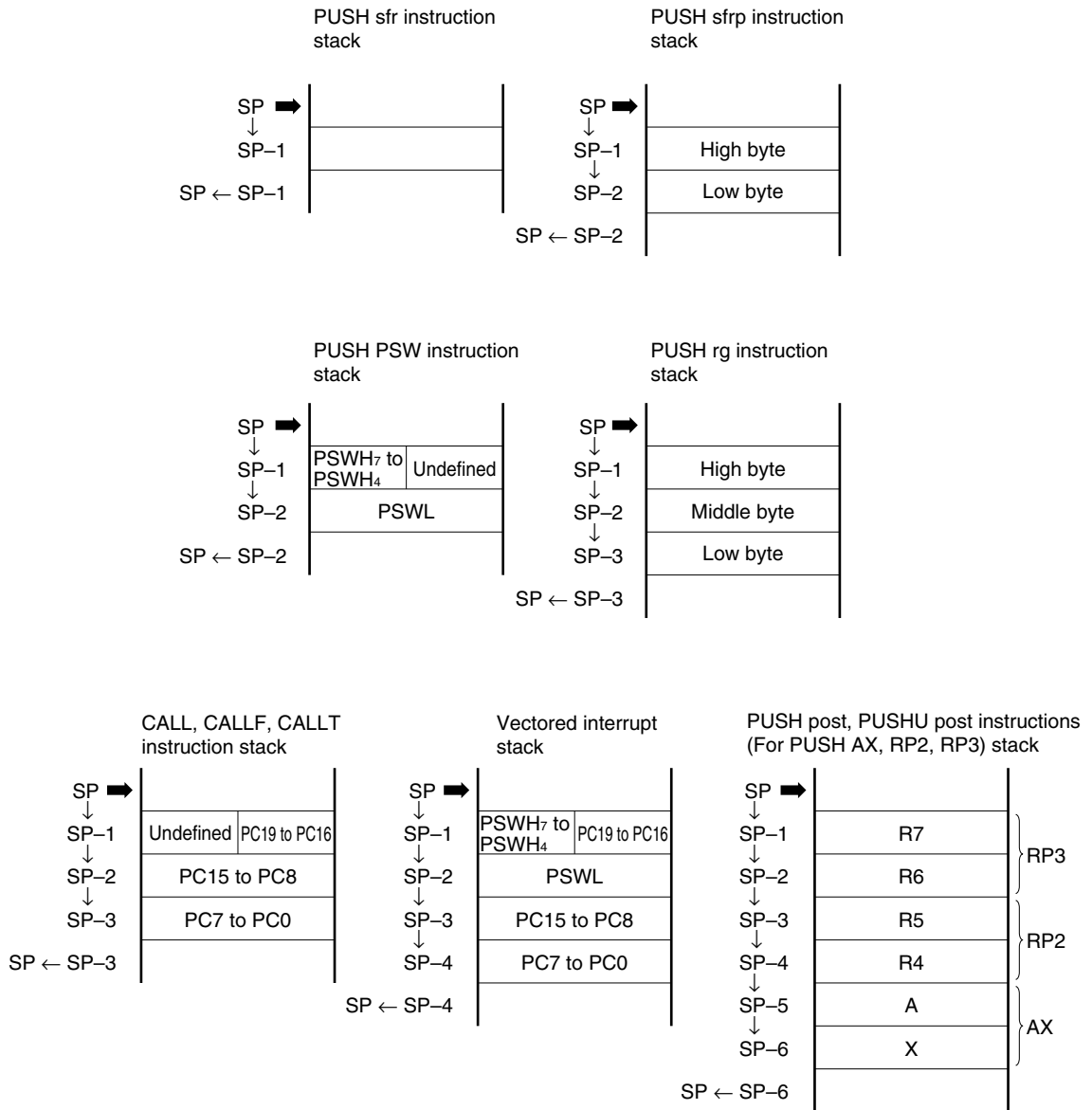
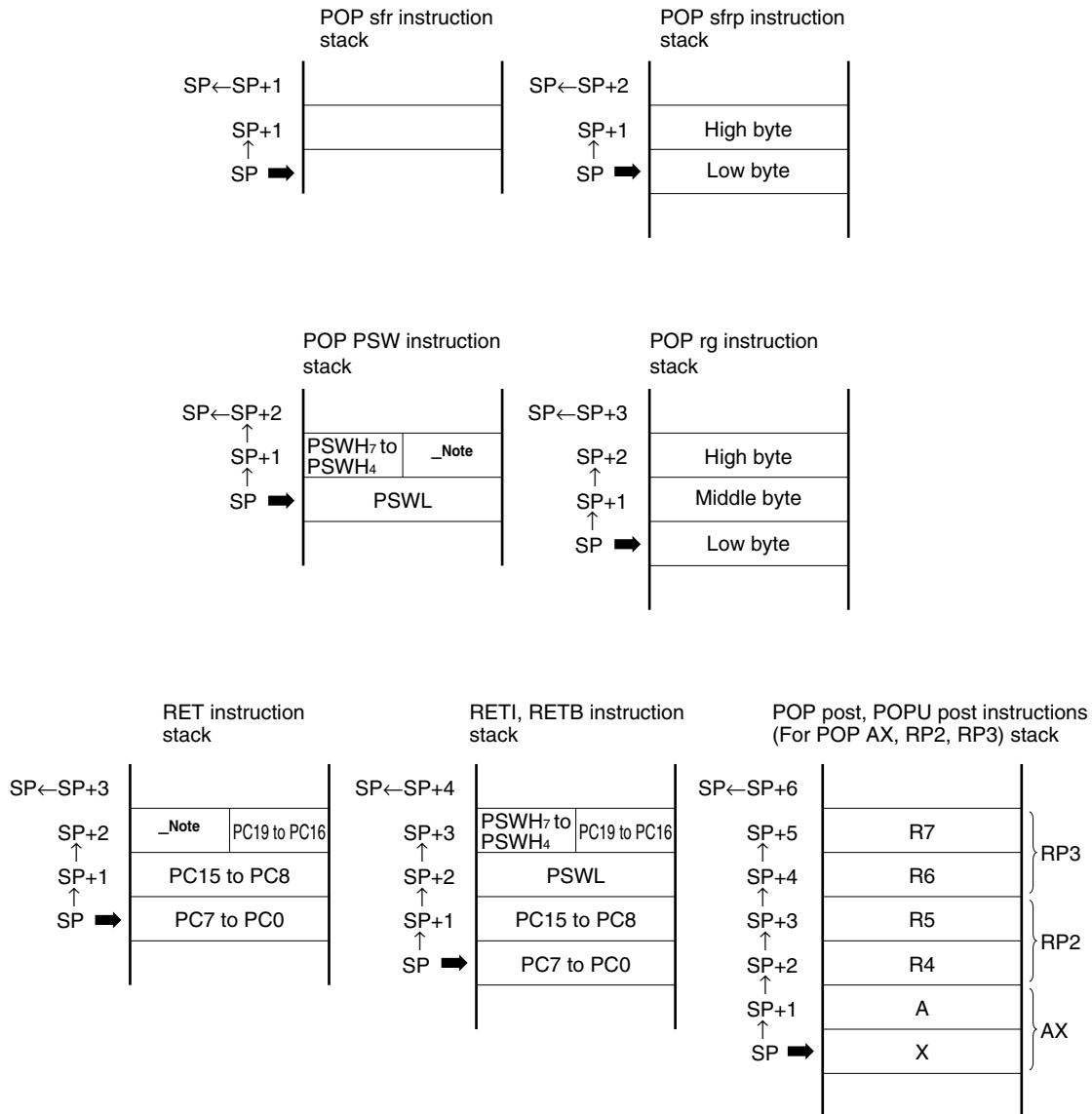


Figure 3-10. Data Restored from the Stack



**Note** This 4-bit data is ignored.



- Cautions**
1. In stack addressing, the entire 1 MB space can be accessed, but the stack cannot be guaranteed in the SFR area and internal ROM area.
  2. The stack pointer (SP) becomes undefined when **RESET** is input. In addition, even when SP is in the undefined state, non-maskable interrupts can be acknowledged. Therefore, when the SP is in the undefined state immediately after the reset is cleared and a request for a non-maskable interrupt is generated, unexpected actions sometimes occur. To avoid this danger, always specify the following in the program after clearing a reset.

```
RSTVCT    CSEG AT 0
          DW    RSTSTRT
          to

INITSEG   CSEG BASE
RSTSTRT:  LOCATION 0H; or LOCATION 0FH
          MOVG SP, #STKBGN
```

### 3.7 General-Purpose Registers

#### 3.7.1 Configuration

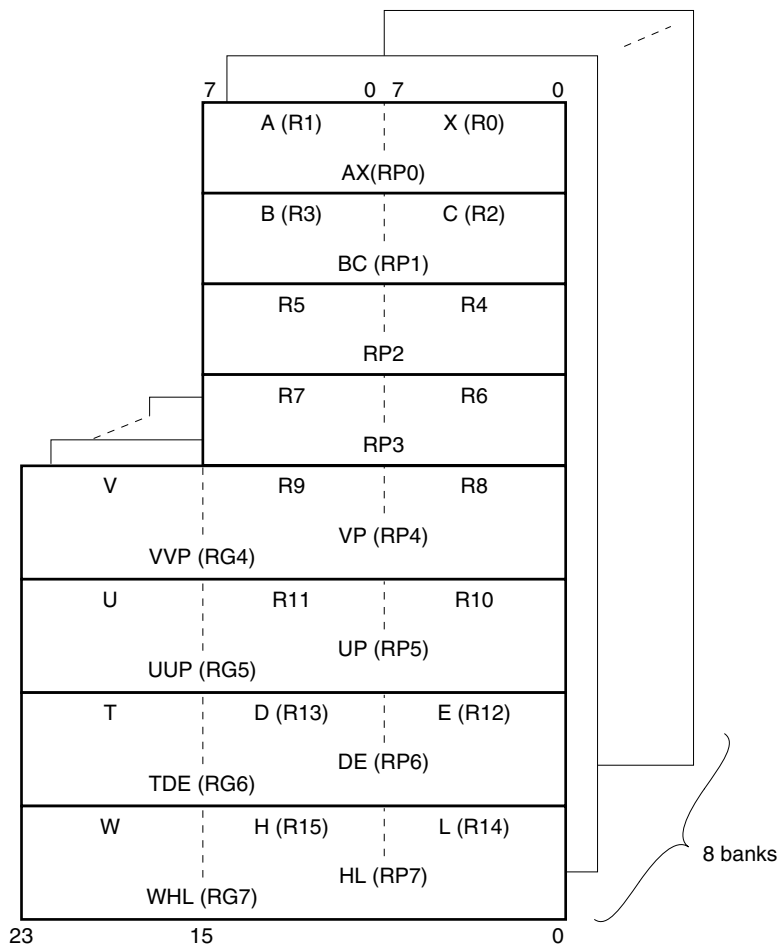
There are sixteen 8-bit general-purpose registers. In addition, two 8-bit general-purpose registers can be combined and used as a 16-bit general-purpose register. Furthermore, four of the 16-bit general-purpose registers are combined with an 8-bit register for address expansion and used as a 24-bit address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

The general-purpose registers except for the V, U, T, and W registers for address expansion are mapped to the internal RAM.

These register sets provide eight banks and can be switched by the software or context switching.

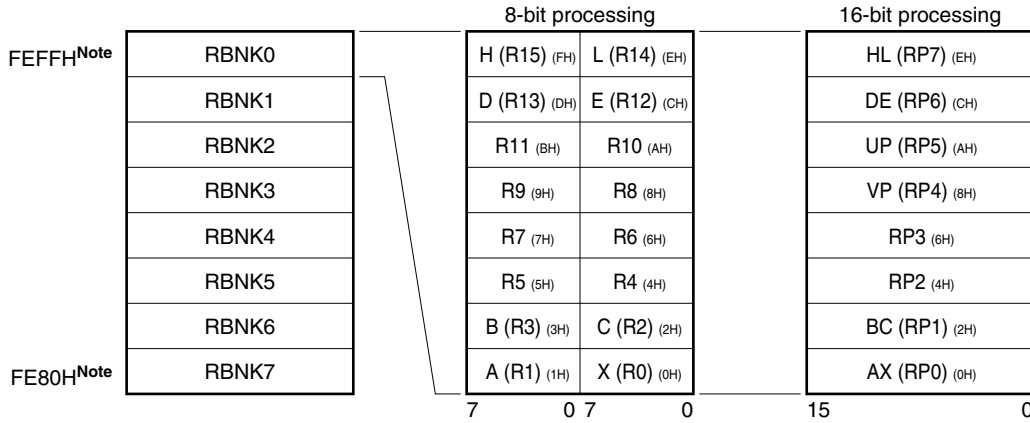
$\overline{\text{RESET}}$  input selects register bank 0. In addition, the register banks that are used in an executing program can be verified by reading the register bank selection flags (RBS0, RBS1, RBS2) in the PSW.

**Figure 3-11. Format of General-Purpose Register**



**Remark** The parentheses enclose the absolute names.

Figure 3-12. General-Purpose Register Addresses

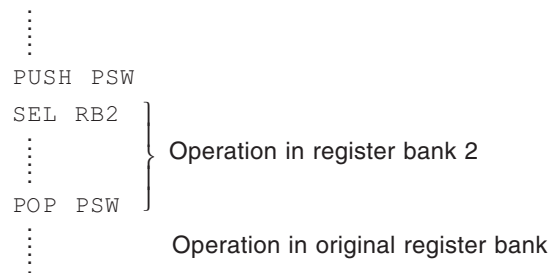


**Note** These are the addresses when the LOCATION 0H instruction is executing. The addresses when the LOCATION 0FH instruction is executed are the sum of the above values and 0F0000H.

**Caution** R4, R5, R6, R7, RP2, and RP3 can be used as the X, A, C, B, AX, and BC registers when the RSS bit in the PSW is set to 1. However, use this function only when using a 78K/III Series program.

**Remark** When changing the register bank and when returning to the original register bank is necessary, execute the SEL RBn instruction after using the PUSH PSW instruction to save the PSW to the stack. If the stack position is not changed when returning to the original state, the POP PSW instruction is used to return. When the register banks in the vectored interrupt processing program are updated, PSW is automatically saved on the stack when an interrupt is accepted and returned by the RETI and RETB instructions. Therefore, when one register bank is used in an interrupt servicing routine, only the SEL RBn instruction is executed, and the PUSH PSW or POP PSW instruction does not have to be executed.

**Example** When register bank 2 is specified



### 3.7.2 Functions

In addition to being manipulatable in 8-bit units, general-purpose registers can be a pair of two 8-bit registers and be manipulated in 16-bit units. Also four of the 16-bit registers can be combined with the 8-bit register for address expansion and manipulated in 24-bit units.

Each register can generally be used as the temporary storage for the operation result or the operand of the operation instruction between registers.

The area from 0FE80H to 0FEFFH (during LOCATION 0H instruction execution, or the 0FFE80H to 0FFEFFH during LOCATION 0FH instruction execution) can be accessed by specifying an address as normal data memory whether or not it is used as the general-purpose register area.

Since there are eight register banks in the 78K/IV Series, efficient programs can be written by suitably using the register banks in normal processing or interrupt servicing.

Each register has the unique functions shown below.

#### **A (R1):**

- This register is primarily for 8-bit data transfers and operation processing. It can be combined with all of the addressing modes for 8-bit data.
- This register can be used to store bit data.
- This register can be used as a register that stores the offset value during indexed addressing or based indexed addressing.

#### **X (R0):**

- This register can store bit data.

#### **AX (RP0):**

- This register is primarily for 16-bit data transfers and operation results. It can be combined with all of the addressing modes for 16-bit data.

#### **AXDE:**

- When a DIVUX, MACW, or MACSW instruction is executing, this register can be used to store 32-bit data.

#### **B (R3):**

- This register functions as a loop counter and can be used by the DBNZ instruction.
- This register can store the offset in indexed addressing and based indexed addressing.
- This register is used as the data pointer in a MACW or MACSW instruction.

#### **C (R2):**

- This register functions as a loop counter and can be used by the DBNZ instruction.
- This register can store the offset in based indexed addressing.
- This register is used as the counter in string and SACW instructions.
- This register is used as the data pointer in a MACW or MACSW instruction.

#### **RP2:**

- When context switching is used, this register saves the lower 16 bits of the program counter (PC).

#### **RP3:**

- When context switching is used, this register saves the higher 4 bits of the program counter (PC) and the program status word (PSW) (except bits 0 to 3 in PSWH).

**VVP (RG4):**

- This register functions as a pointer and specifies the base address in register indirect addressing, based addressing, and based indirect addressing.

**UUP (RG5):**

- This register functions as a user stack pointer and implements another stack separate from the system stack by the PUSHU and POPU instructions.
- This register functions as a pointer and acts as the register that specifies the base address during register indirect addressing and based addressing.

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

- This register stores the offset during indexed addressing and based indexed addressing.

**TDE (RG6):**

- This register functions as a pointer and sets the base address in register indirect addressing and based addressing.
- This register functions as a pointer in string and SACW instructions.

**WHL (RG7):**

- This register primarily performs 24-bit data transfers and operation processing.
- This register functions as a pointer and specifies the base address during register indirect addressing or based addressing.
- This functions as a pointer in string and SACW instructions.

In addition to its function name (X, A, C, B, E, D, L, H, AX, BC, VP, UP, DE, HL, VVP, UUP, TDE, WHL) that emphasizes its unique function, each register can be described by its absolute name (R0 to R15, RP0 to RP7, RG4 to RG7). For the correspondence, refer to **Table 3-5**.

**Table 3-5. Correspondence Between Function Names and Absolute Names**

**(a) 8-bit registers**

Absolute Name	Function Name	
	RSS = 0	RSS = 1 <sup>Note</sup>
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 <sup>Note</sup>
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

**Note** Use RSS = 1 only when a 78K/III Series program is used.

**Remark** R8 to R11 do not have function names.

### 3.8 Special Function Registers (SFRs)

These registers are assigned special functions such as the mode register and control register of the on-chip peripheral hardware and are mapped to the 256-byte area from 0FF00H to 0FFFFH<sup>Note</sup>.

**Note** These are the addresses when the LOCATION 0H instruction is executing. They are FFF00H to FFFFFH when the LOCATION 0FH instruction is executing.

**Caution** In this area, do not access an address that is not allocated by an SFR. If erroneously accessed, the  $\mu$ PD784975A enters the deadlock state. The deadlock state is released only by reset input.

Table 3-6 shows the list of special function registers (SFRs). The meanings of the items are described next.

- Symbol ... This symbol indicates the on-chip SFR. In NEC assembler RA78K4, this is a reserved word. In C compiler CC78K4, it can be used as an sfr variable by a #pragma sfr directive.
- R/W ... Indicates whether the corresponding SFR can be read or written.  
R/W: Can read/write  
R: Read only  
W: Write only
- Bit units for manipulation ... When the corresponding SFR is manipulated, the appropriate bit manipulation unit is indicated. An SFR that can manipulate 16 bits can be described in the sfrp operand. If specified by an address, an even address is described. An SFR that can manipulate one bit can be described in bit manipulation instructions.
- After reset ... Indicates the state of each register when  $\overline{\text{RESET}}$  is input.

Table 3-6. Special Function Register (SFR) List (1/3)

Address Note 1	Name of Special Function Register (SFR)	Symbol		R/W	Bit Units for Manipulation			After Reset		
					1 Bit	8 Bits	16 Bits			
0FF00H	Port 0	P0		R	○	○	—	Undefined		
0FF01H	Port 1	P1			○	○	—			
0FF02H	Port 2	P2		R/W	○	○	—	00H <sup>Note 2</sup>		
0FF04H	Port 4	P4			○	○	—			
0FF05H	Port 5	P5			○	○	—			
0FF06H	Port 6	P6			○	○	—			
0FF07H	Port 7	P7			○	○	—			
0FF08H	Port 8	P8			○	○	—			
0FF09H	Port 9	P9			○	○	—			
0FF0AH	Port 10	P10			○	○	—			
0FF0BH	Port read 7	PLR7			R	○	○		—	Undefined
0FF0CH	Port read 8	PLR8				○	○		—	
0FF0DH	Port read 9	PLR9		○		○	—			
0FF10H	16-bit timer counter 0	TM0		R/W	—	—	○	0000H		
0FF12H	16-bit capture/compare register 00 (16-bit timer/event counter)	CR00			—	—	○			
0FF14H	16-bit capture/compare register 01 (16-bit timer/event counter)	CR01			—	—	○			
0FF16H	Capture/compare control register 0	CRC0			○	○	—		00H	
0FF18H	16-bit timer mode control register 0	TMC0		○	○	—				
0FF1BH	Watch timer clock select register	WTCL		○	○	—				
0FF1CH	Prescaler mode register 0	PRM0		○	○	—				
0FF1EH	Remote controller mode register	REMM		○	○	—				
0FF22H	Port 2 mode register	PM2		○	○	—	FFH			
0FF24H	Port 4 mode register	PM4		○	○	—				
0FF25H	Port 5 mode register	PM5		○	○	—				
0FF26H	Port 6 mode register	PM6		○	○	—				
0FF32H	Pull-up resistor option register 2	PU2		R/W	○	○	—	00H		
0FF4EH	Pull-up resistor option register	PUO			○	○	—			
0FF50H	8-bit timer counter 50	TM50	TM5	R	—	○	○	04H		
0FF51H	8-bit timer counter 51	TM51			—	○	—			
0FF52H	8-bit compare register 50	CR50	CR5	R/W	—	○	○			
0FF53H	8-bit compare register 51	CR51			—	○	—			
0FF54H	8-bit timer mode control register 50	TMC50	TMC5	R/W	○	○	○			
0FF55H	8-bit timer mode control register 51	TMC51			○	○	—			
0FF56H	Timer clock select register 50	TCL50	TCL5	R/W	○	○	○		00H	
0FF57H	Timer clock select register 51	TCL51			○	○	—			

- Notes**
1. These values are when the LOCATION 0H instruction is executing. When the LOCATION 0FH instruction is executing, F0000H is added to these values.
  2. Since each port is initialized in the input mode by a reset, in fact, 00H is not read out. The output latch is initialized to 0.



Table 3-6. Special Function Register (SFR) List (2/3)

Address Note	Name of Special Function Register (SFR)	Symbol		R/W	Bit Units for Manipulation			After Reset
					1 Bit	8 Bits	16 Bits	
0FF70H	Asynchronous serial interface mode register 0	ASIM0		R/W	○	○	—	00H
0FF72H	Asynchronous serial interface status register 0	ASIS0		R	○	○	—	
0FF74H	Transmit shift register 0	TXS0		W	—	○	—	FFH
	Receive buffer register 0	RXB0		R	—	○	—	
0FF76H	Baud rate generator control register 0	BRGC0		R/W	○	○	—	00H
0FF7AH	Oscillation mode select register	CC		R	○	○	—	
0FF80H	A/D converter mode register	ADM		R/W	○	○	—	
0FF81H	A/D converter input select register	ADIS			○	○	—	
0FF83H	A/D conversion result register	ADCR		R	—	○	—	Undefined
0FF90H	Serial operation mode register 0	CSIM0		R/W	○	○	—	00H
0FF91H	Serial operation mode register 1	CSIM1			○	○	—	
0FF92H	Serial operation mode register 2	CSIM2			○	○	—	
0FF94H	Serial I/O shift register 0	SIO0			—	○	—	
0FF95H	Serial I/O shift register 1	SIO1			—	○	—	
0FF96H	Serial I/O shift register 2	SIO2			—	○	—	
0FF9CH	Watch timer mode control register	WTM			○	○	—	
0FFA0H	External interrupt rising edge enable register 0	EGP0			○	○	—	
0FFA2H	External interrupt falling edge enable register 0	EGN0			○	○	—	
0FFA8H	In-service priority register	ISPR			R	○	○	
0FFA9H	Interrupt select control register	SNMI		R/W	○	○	—	80H
0FFAAH	Interrupt mode control register	IMC			○	○	—	
0FFACH	Interrupt mask register 0L	MK0L	MK0		○	○	○	FFH
0FFADH	Interrupt mask register 0H	MK0H			○	○	—	
★ 0FFAEH	Interrupt mask register 1L	MK1L			○	○	—	FFH
0FFB0H	Display mode register 0	DSPM0			○	○	—	10H
0FFB2H	Display mode register 1	DSPM1			○	○	—	01H
0FFB4H	Display mode register 2	DSPM2			○	○	—	00H
0FFC0H	Standby control register	STBC			—	○	—	30H
0FFC2H	Watchdog timer mode register	WDM			—	○	—	00H
0FFC4H	Memory expansion mode register	MM			○	○	—	20H
0FFCFH	Oscillation stabilization time specification register	OSTS			○	○	—	00H
0FFD0H-0FFDFH	External SFR area	—			○	○	—	—
0FFE0H	Interrupt control register (INTWDT)	WDTIC			○	○	—	43H
0FFE1H	Interrupt control register (INTP0)	PIC0			○	○	—	
0FFE2H	Interrupt control register (INTP1)	PIC1			○	○	—	

**Note** These are the values when the LOCATION 0H instruction is executing. When the LOCATION 0FH instruction is executing, F000H is added to this value.

Table 3-6. Special Function Register (SFR) List (3/3)

Address Note	Name of Special Function Register (SFR)	Symbol	R/W	Bit Units for Manipulation			After Reset
				1 Bit	8 Bits	16 Bits	
0FFE3H	Interrupt control register (INTP2)	PIC2	R/W	○	○	—	43H
0FFE4H	Interrupt control register (INTTM00)	TMIC00		○	○	—	
0FFE5H	Interrupt control register (INTTM01)	TMIC01		○	○	—	
0FFE6H	Interrupt control register (INTKS)	KSIC		○	○	—	
0FFE7H	Interrupt control register (INTCSI0)	CSIIC0		○	○	—	
0FFE8H	Interrupt control register (INTCSI1)	CSIIC1		○	○	—	
0FFE9H	Interrupt control register (INTTM50)	TMIC50		○	○	—	
0FFEAH	Interrupt control register (INTTM51)	TMIC51		○	○	—	
0FFEBH	Interrupt control register (INTAD)	ADIC		○	○	—	
0FFECH	Interrupt control register (INTREM)	REMIC		○	○	—	
0FFEDH	Interrupt control register (INTCSI2)	CSIIC2		○	○	—	
0FFEEH	Interrupt control register (INTSER0)	SERIC0		○	○	—	
0FFEFH	Interrupt control register (INTSR0)	SRIC0		○	○	—	
0FFF0H	Interrupt control register (INTST0)	STIC0		○	○	—	
0FFF1H	Interrupt control register (INTWT1)	WTIIC		○	○	—	
0FFF2H	Interrupt control register (INTWT)	WTIC	○	○	—		
0FFFCH	Internal memory size switching register	IMS	W	—	○	—	FFH

**Note** These are the values when the LOCATION 0H instruction is executing. When the LOCATION 0FH instruction is executing, F000H is added to this value.

### 3.9 Cautions

- (1) Program fetches are not possible from the internal high-speed RAM space (when executing the LOCATION 0H instruction: 0FD00H - 0FEFFH, when executing the LOCATION 0FH instruction: FFD00H - FFEFFH)

- (2) Special function register (SFR)

Do not access an address that is allocated to an SFR in the area from 0FF00H to 0FFFFH<sup>Note</sup>. If mistakenly accessed, the  $\mu$ PD784975A enters the deadlock state. The deadlock state is only released by  $\overline{\text{RESET}}$  input.

**Note** These addresses are when the LOCATION 0H instruction is executing. They are FFF00H to FFFFFH when the LOCATION 0FH instruction is executing.

- (3) Stack pointer (SP) operation

Although the entire 1 MB space can be accessed by stack addressing, the stack cannot be guaranteed in the SFR area and the internal ROM space.

- (4) Stack pointer (SP) initialization

The SP becomes undefined when  $\overline{\text{RESET}}$  is input. Even after a reset is cleared, non-maskable interrupts can be accepted. Therefore, the SP enters an undefined state immediately after clearing the reset. When a non-maskable interrupt request is generated, unexpected operations sometimes occur. To minimize these dangers, always describe the following in the program immediately after clearing a reset.

```

RSTVCT   CSEG AT 0
         DW   RSTSTRT

         to

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

```

## CHAPTER 4 PORT FUNCTIONS

### 4.1 Port Functions

The  $\mu$ PD784976A Subseries incorporates 12 input ports, eight output ports and 52 I/O ports. Figure 4-1 shows the port configuration. Every port is capable of 1-bit and 8-bit manipulations and can carry out considerably varied control operations. Besides port functions, the ports can also serve as on-chip hardware I/O pins.

Figure 4-1. Port Types

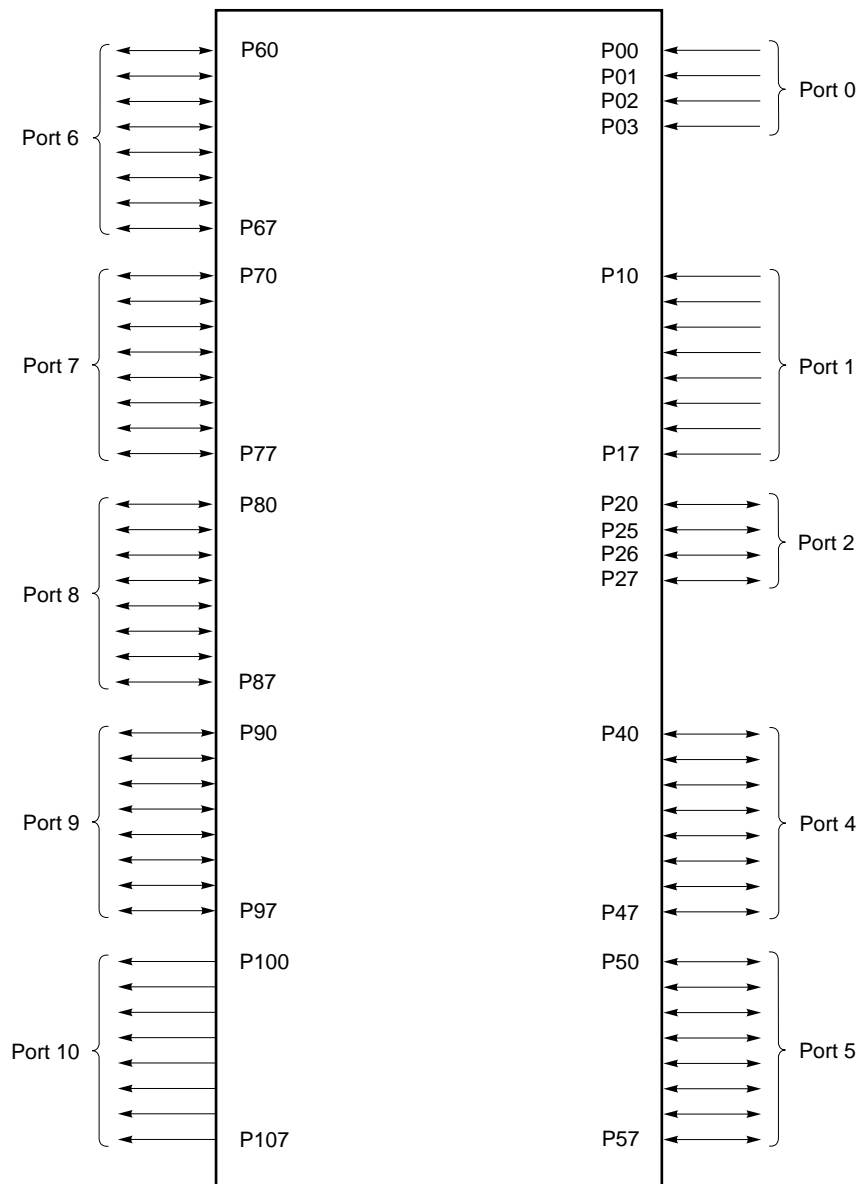


Table 4-1. Port Function

Pin Name	Function	Alternate Function
P00 to P03	Port 0. 4-bit input port.	ANI8 to ANI11
P10 to P17	Port 1. 8-bit input port.	ANI0 to ANI7
P20	Port 2.	TI00
P25	4-bit I/O port.	SI0/RxD0
P26	Input/output can be specified in 1-bit units.	SO0/TxD0
P27	On-chip pull-up resistor can be specified by a software setting in 1-bit or 8-bit units.	SCK0/ASCK0
P40 to P47	Port 4. 8-bit I/O port. Input/output can be specified in 1-bit units. Can directly drive LED. On-chip pull-up resistor can be specified by a software setting in 8-bit units when this port is used as input port.	—
P50 to P54	Port 5. N-ch open-drain 8-bit medium-voltage I/O port. Input/output can be specified in 1-bit units.	—
P55	Can directly drive LED.	SI2
P56	On-chip pull-up resistor can be specified by mask option in 1-bit units (mask ROM versions only). The $\mu$ PD78F4976A does not have pull-up resistors, however.	SO2
P57		SCK2
P60	Port 6.	SI1
P61	8-bit I/O port.	SO1
P62	Input/output can be specified in 1-bit units.	SCK1
P63	On-chip pull-up resistor can be specified by a software setting in 8-bit units when this port is used as input port.	TIO50
P64		INTP0
P65		INTP1
P66		TIO51
P67		INTP2
P70 to P77	Port 7. P-ch open-drain 8-bit high-voltage I/O port. Input/output can be specified in 1-bit units. On-chip pull-down resistor can be specified by mask option in 1-bit units (mask ROM versions only). The $\mu$ PD78F4976A does not have pull-down resistors, however.	FIP16 to FIP23
P80 to P87	Port 8. P-ch open-drain 8-bit high-voltage I/O port. Input/output can be specified in 1-bit units. On-chip pull-down resistor can be specified by mask option in 1-bit units (mask ROM versions only). The $\mu$ PD78F4976A does not have pull-down resistors, however.	FIP24 to FIP31
P90 to P97	Port 9. P-ch open-drain 8-bit high-voltage I/O port. Input/output can be specified in 1-bit units. On-chip pull-down resistor can be specified by mask option in 1-bit units (mask ROM versions only). The $\mu$ PD78F4976A does not have pull-down resistors, however.	FIP32 to FIP39
P100 to P107	Port 10. P-ch open-drain 8-bit high-voltage output port. On-chip pull-down resistor can be specified by mask option in 1-bit units (mask ROM versions only). The $\mu$ PD78F4976A does not have pull-down resistors, however.	FIP40 to FIP47

## 4.2 Port Configuration

A port consists of the following hardware.

**Table 4-2. Port Configuration**

Item	Configuration
Control register	Port mode register (PMm: m = 2, 4 to 6) Pull-up resistor option register (PUO, PU2)
Port	Total: 72 (12 inputs, 8 outputs, 52 inputs/outputs)
Pull-up resistor	<ul style="list-style-type: none"> <li>• Mask ROM version Total: 28 (software control: 20, mask option control: 8)</li> <li>• <math>\mu</math>PD78F4976A Total: 20</li> </ul>
Pull-down resistor	<ul style="list-style-type: none"> <li>• Mask ROM version Total: 32 (mask option control: 32)</li> <li>• <math>\mu</math>PD78F4976A None</li> </ul>

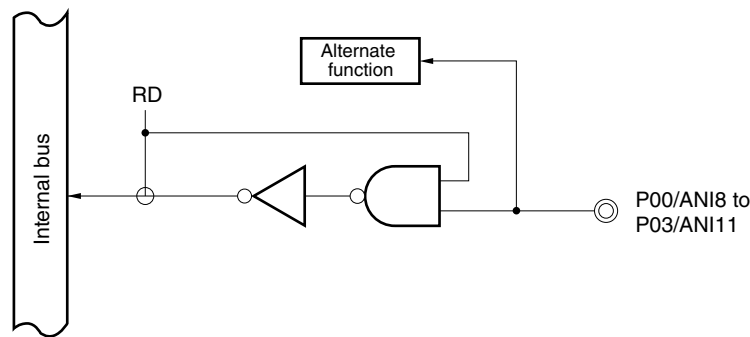
### 4.2.1 Port 0

Port 0 is a 4-bit input port.

Port 0 functions alternately as an A/D converter analog input.

Figure 4-2 shows a block diagram of port 0.

**Figure 4-2. Block Diagram of P00 to P03**



RD: Port 0 read signal

**Caution** Because port 0 also functions as an analog input for the A/D converter, do not issue port read instructions (including bit manipulation instructions) when port 0 is being used as an analog input pin.

When the port is being read, applying an intermediate potential to the analog input pin may impair the reliability of the chip because the intermediate voltage is read out.

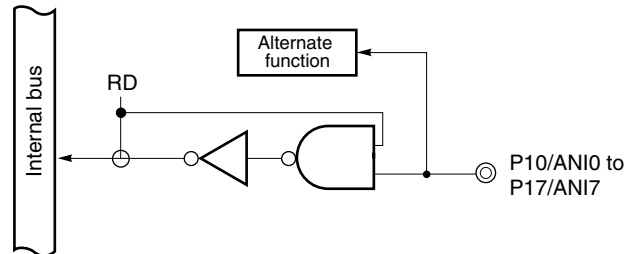
### 4.2.2 Port 1

Port 1 is an 8-bit input port.

Port 1 functions alternately as an A/D converter analog input.

Figure 4-3 shows a block diagram of port 1.

**Figure 4-3. Block Diagram of P10 to P17**



RD: Port 1 read signal

**Caution** Because port 1 also functions as an analog input for the A/D converter, do not issue port read instructions (including bit manipulation instructions) when port 1 is being used as an analog input pin.

When the port is being read, applying an intermediate potential to the analog input pin may impair the reliability of the chip because the intermediate voltage is read out.

### 4.2.3 Port 2

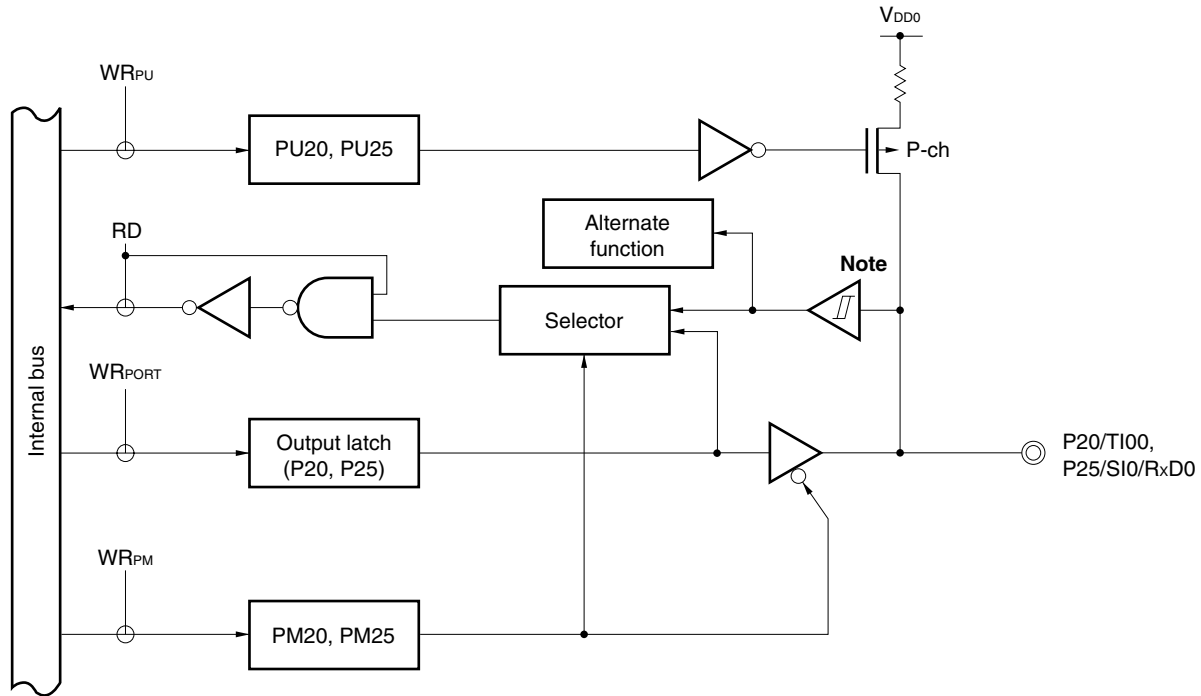
Port 4 is a 4-bit I/O port with output latch. Input/output mode can be specified for the P20 and P25 to P27 pins in 1-bit units using the port 2 mode register (PM2). Use of on-chip pull-up resistors can be specified for the P20 and P25 to P27 pins in 1-bit units using pull-up resistor option register 2 (PU2).

Port 4 functions alternately as a serial interface data input/output, asynchronous serial interface data input/output, serial clock input/output, and timer input.

$\overline{\text{RESET}}$  input sets port 2 to input mode.

Figures 4-4 to 4-6 show block diagrams of port 2.

**Figure 4-4. Block Diagram of P20 and P25**



PU2: Pull-up resistor option register 2

PM2: Port 2 mode register

RD: Port 2 read signal

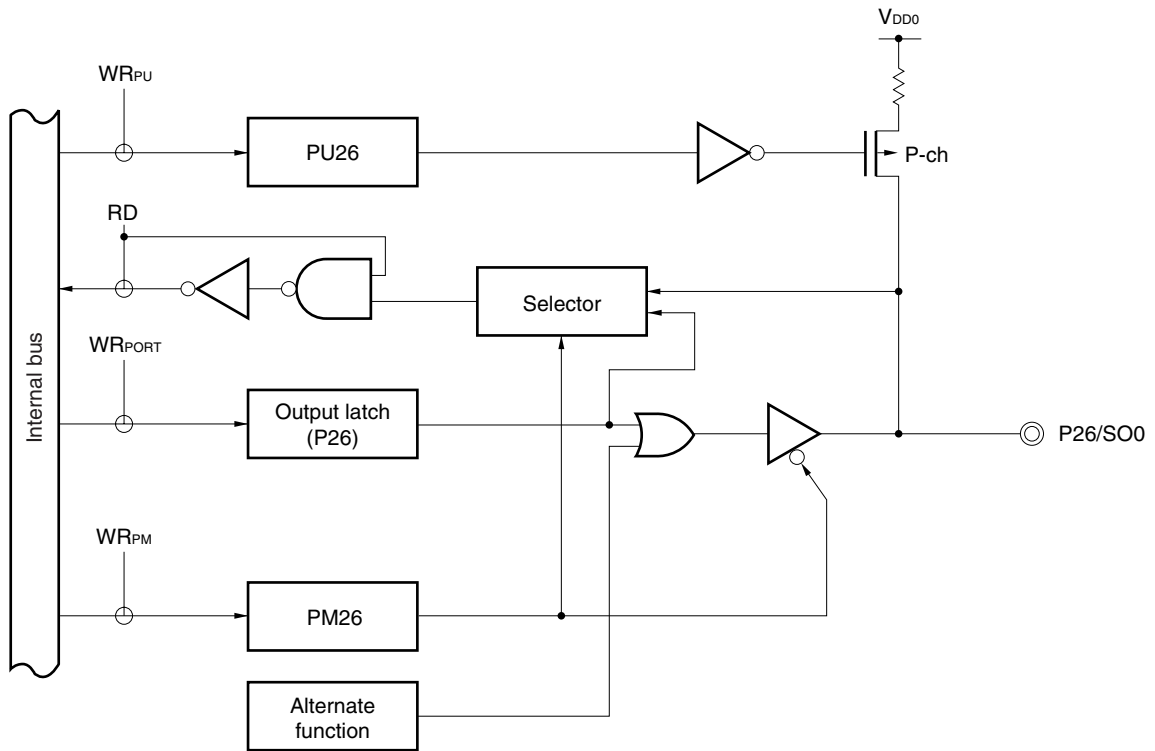
WR: Port 2 write signal

**Note** The Schmitt input buffer of P25 pin is turned off in the STOP and IDLE modes (Schmitt output is fixed to "L".)

**Caution** Do not connect a pull-up resistor to a pin that is specified to be in output mode when port 2 is selected.



Figure 4-5. Block Diagram of P26



PU2: Pull-up resistor option register 2

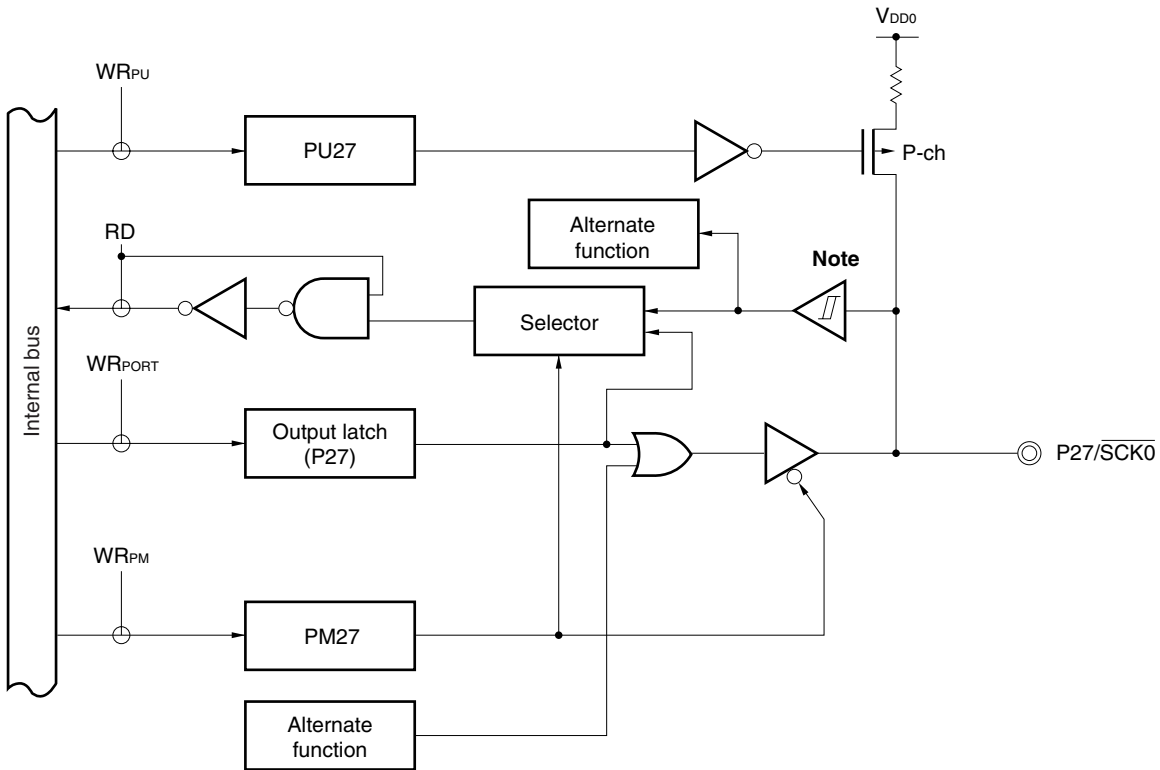
PM2: Port 2 mode register

RD: Port 2 read signal

WR: Port 2 write signal

**Caution** Do not connect a pull-up resistor to a pin that is specified to be in output mode when port 2 is selected.

Figure 4-6. Block Diagram of P27



PU2: Pull-up resistor option register 2

PM2: Port 2 mode register

RD: Port 2 read signal

WR: Port 2 write signal

**Note** The Schmitt input buffer of P27 pin is turned off in the STOP and IDLE modes (Schmitt output is fixed to "L".)

**Caution** Do not connect a pull-up resistor to a pin that is specified to be in output mode when port 2 is selected.

#### 4.2.4 Port 4

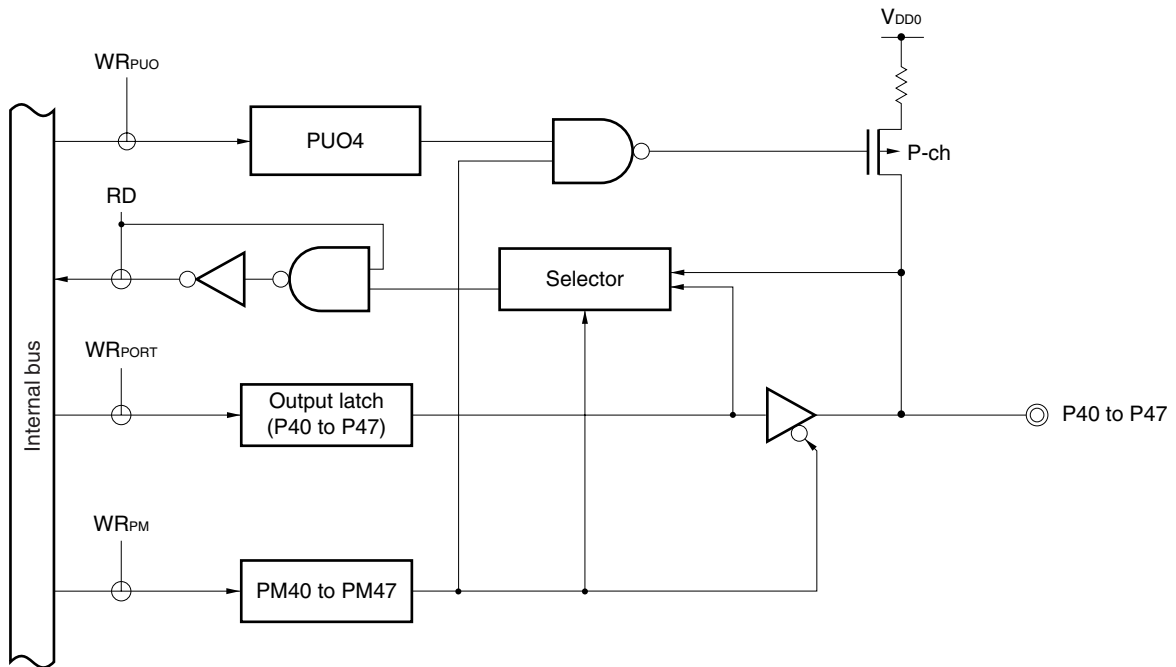
Port 4 is an 8-bit I/O port with output latch. Input/output mode can be specified for the P40 to P47 pins in 1-bit units using port 4 mode register (PM4). Use of the on-chip pull-up resistor can be specified for the P40 to P47 pins per port using bit 4 of pull-up resistor option register 4 (PU4) only when the pins are used as an input port.

Port 4 can drive LEDs directly.

$\overline{\text{RESET}}$  input sets port 4 to input mode.

Figure 4-7 shows a block diagram of port 4.

Figure 4-7. Block Diagram of P40 to P47



PUO: Pull-up resistor option register

PM4: Port 4 mode register

RD: Port 4 read signal

WR: Port 4 write signal

4.2.5 Port 5

Port 5 is an 8-bit I/O port with output latch. Input/output mode can be specified for the P50 to P57 pins in 1-bit units using port 5 mode register (PM5). Use of the on-chip pull-up resistors can be specified in 1-bit units using a mask option in the mask ROM versions. The  $\mu$ PD78F4976A has no pull-up resistor.

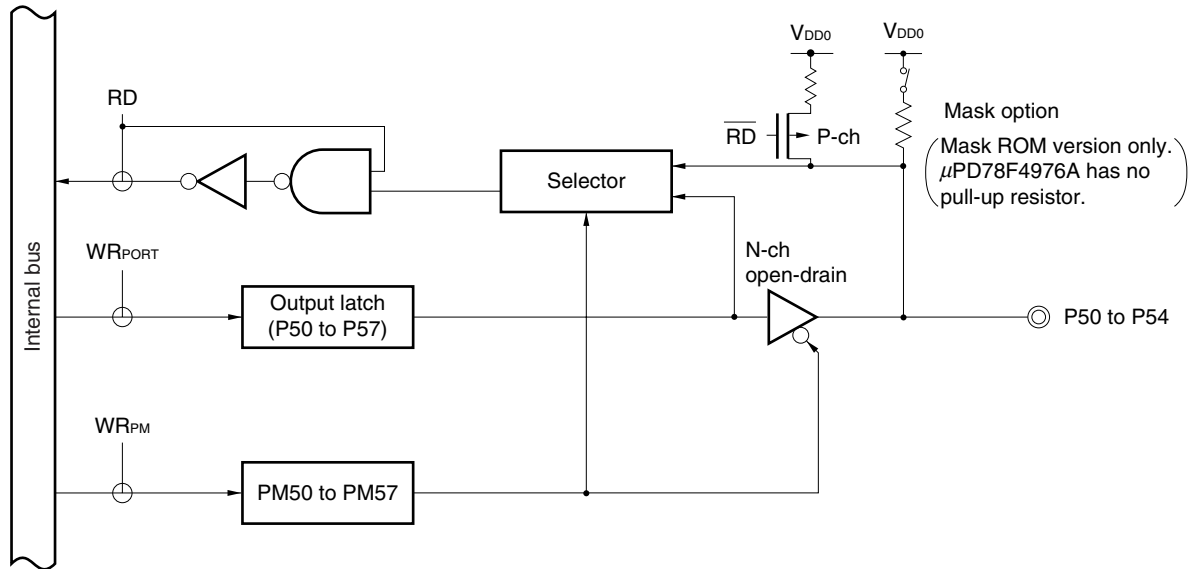
Port 5 can drive LEDs directly.

Port 5 functions alternately as a serial interface data input/output and serial clock input/output.

$\overline{\text{RESET}}$  input sets port 5 to input mode.

Figures 4-8 and 4-9 show block diagrams of port 5.

Figure 4-8. Block Diagram of P50 to P54

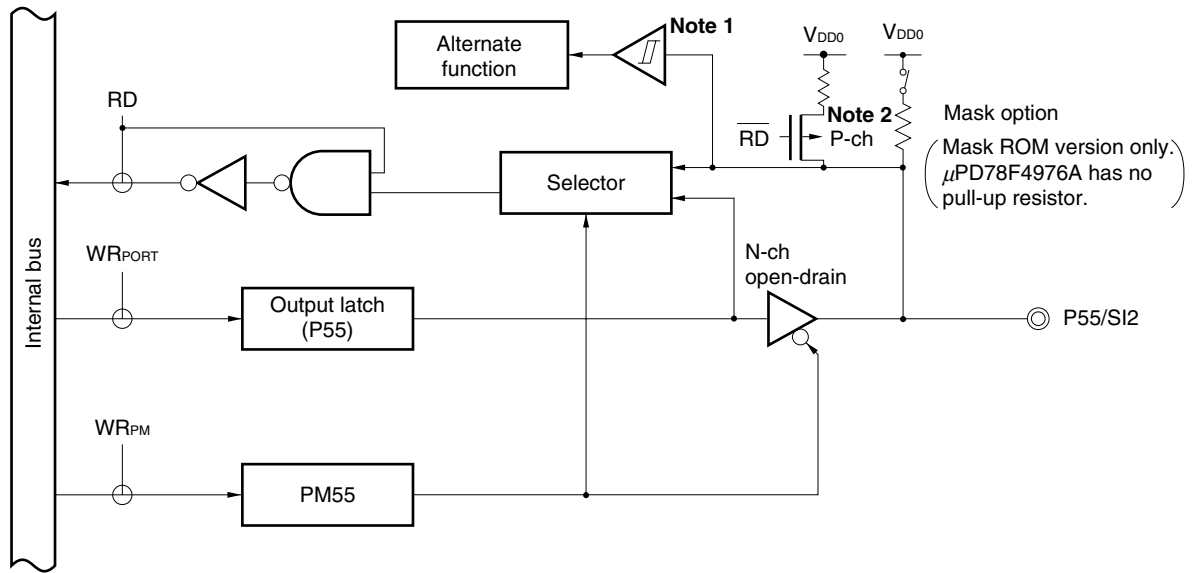


PM5: Port 5 mode register

RD: Port 5 read signal

WR: Port 5 write signal

Figure 4-9. Block Diagram of P55



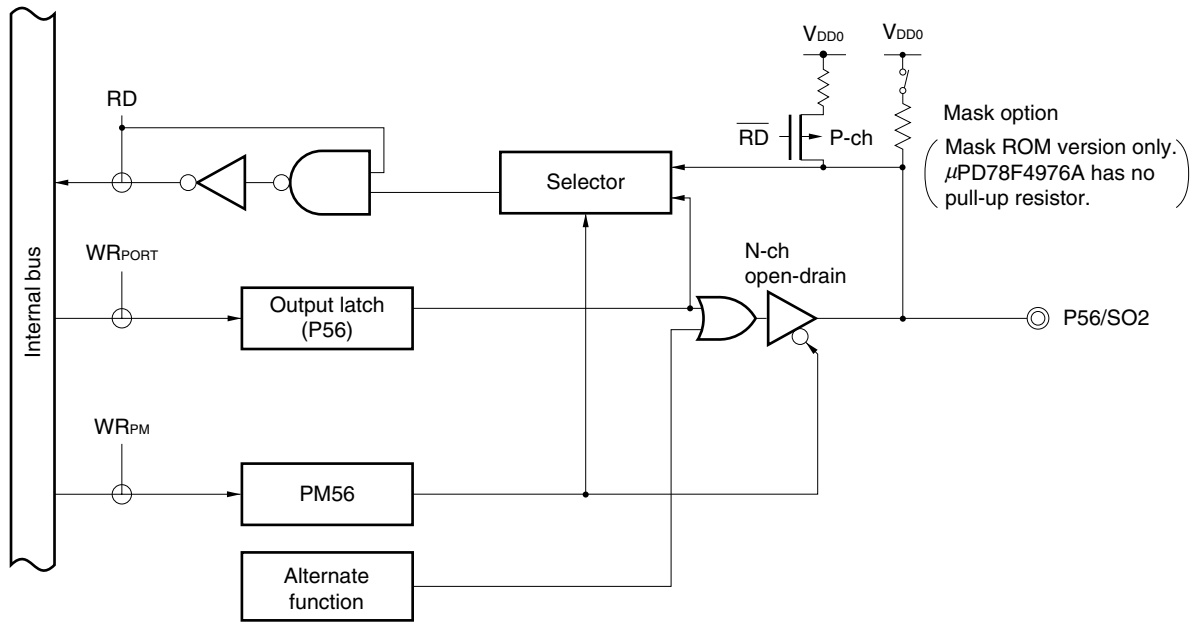
PM5: Port 5 mode register

RD: Port 5 read signal

WR: Port 5 write signal

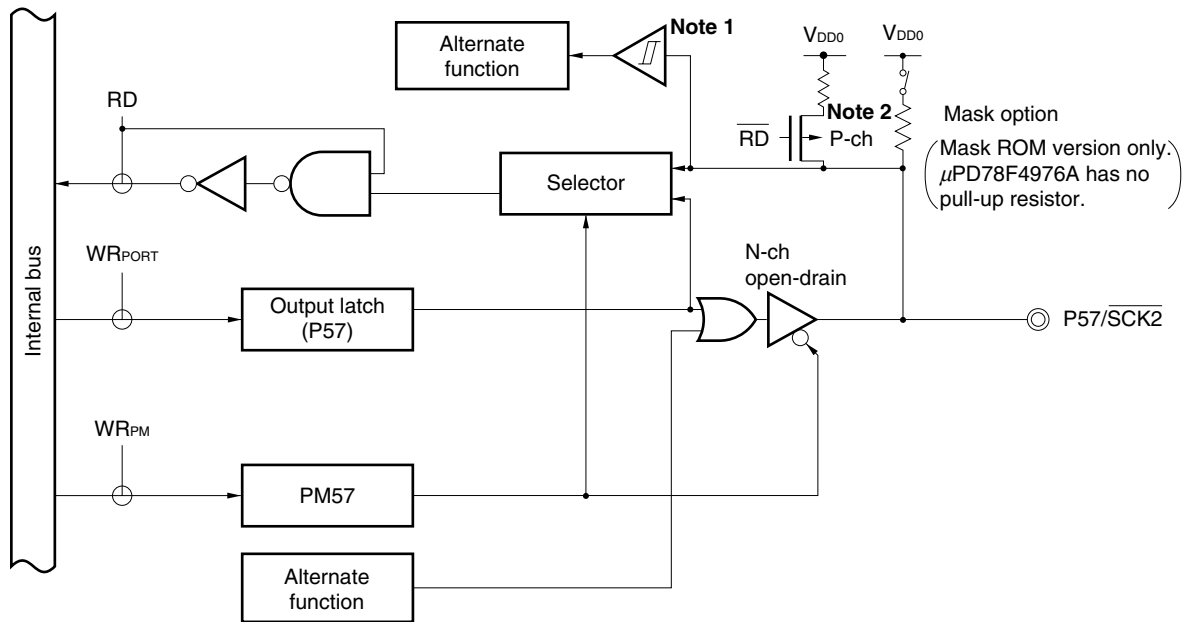
- Notes**
1. The Schmitt input buffer of P55 pin is turned off in the STOP and IDLE modes (Schmitt output is fixed to "L".)
  2. The P-ch transistor is turned off in the STOP and IDLE modes.

Figure 4-10. Block Diagram of P56



PM5: Port 5 mode register  
 RD: Port 5 read signal  
 WR: Port 5 write signal

Figure 4-11. Block Diagram of P57



PM5: Port 5 mode register  
 RD: Port 5 read signal  
 WR: Port 5 write signal

- Notes**
1. The Schmitt input buffer of P57 pin is turned off in the STOP and IDLE modes (Schmitt output is fixed to "L".)
  2. The P-ch transistor is turned off in the STOP and IDLE modes.

#### 4.2.6 Port 6

Port 6 is an 8-bit I/O port with output latch. Input/output mode can be specified for the P60 to P67 pins in 1-bit units using port 6 mode register (PM6). Use of the on-chip pull-up resistor can be specified for the P60 to P67 pins per port using bit 6 (PUO6) of the pull-up option register (PUO) only when the pins are used as an input port.

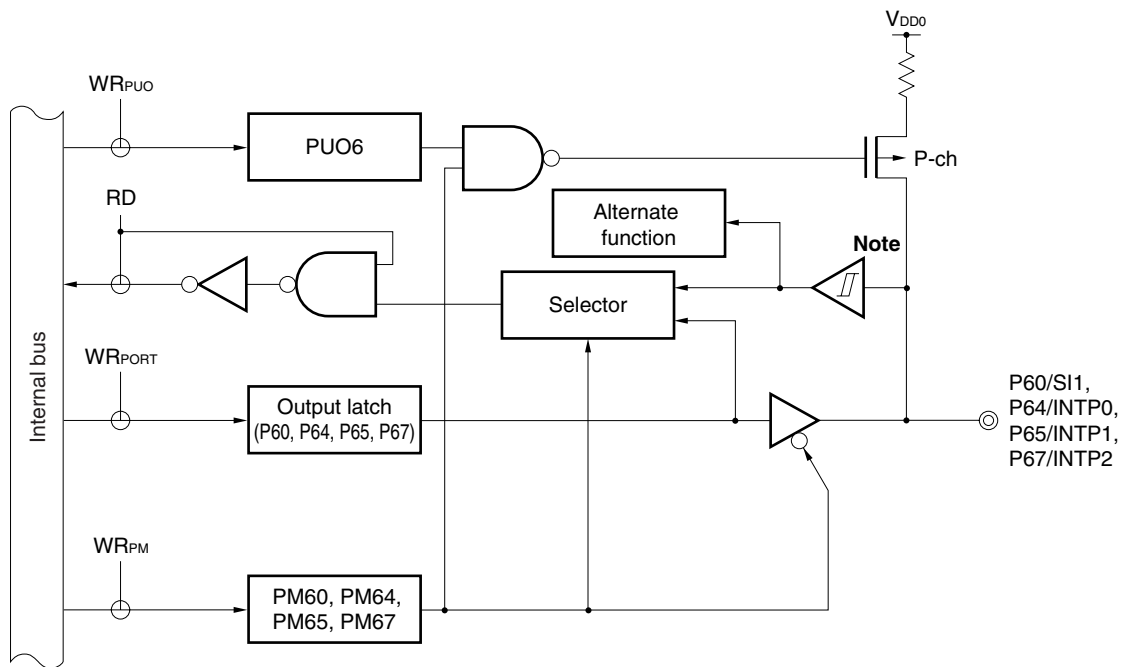
Port 6 functions alternately as a serial interface data input/output, serial clock input/output, timer input/output, and external interrupt request input.

$\overline{\text{RESET}}$  input sets port 6 to input mode.

Figures 4-12 to 4-14 show a block diagram of port 6.

**Caution** Pins P64, P65, and P67 also function as external interrupt request inputs. If they are not used as interrupt input pins, set the external interrupt rising edge enable register (EGP0) and external interrupt falling edge enable register (EGN0) to “Interrupt Disable.” Or, set the interrupt mask flags (PMKn where  $n = 0$  to 2) to 1. Otherwise, the interrupt request flag is set when the port function is placed in output mode and its output level is changed, leading to inadvertent interrupt servicing.

Figure 4-12. Block Diagram of P60, P64, P65, and P67



PUO: Pull-up resistor option register

PM6: Port 6 mode register

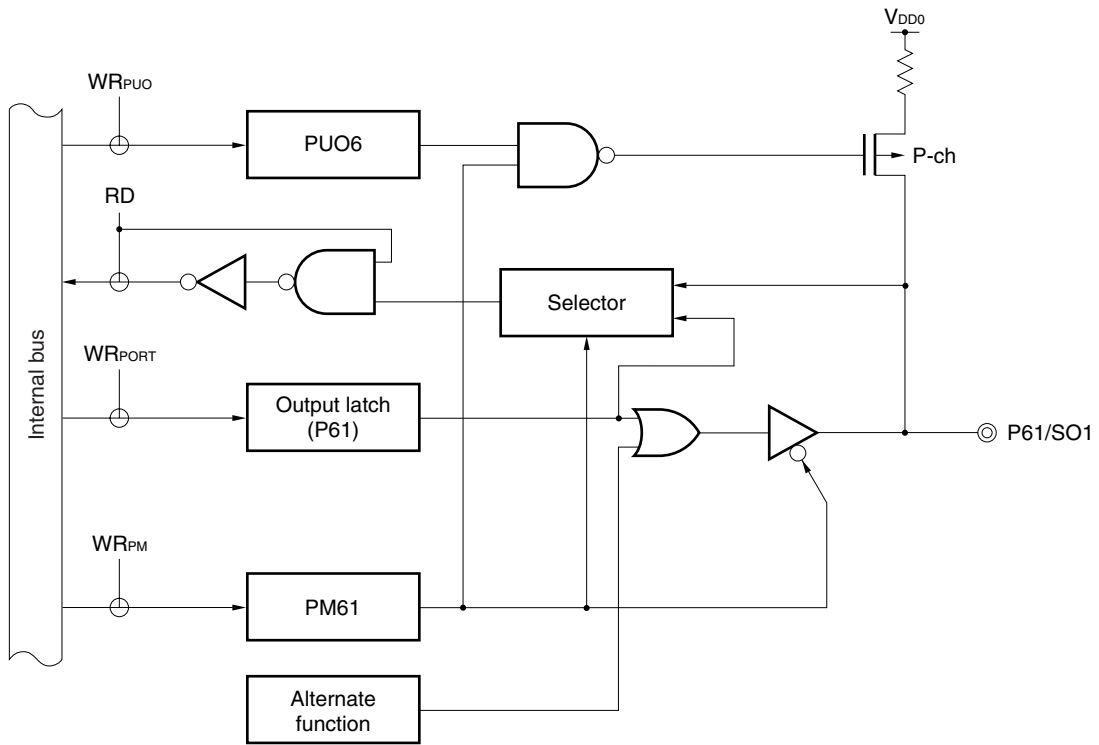
RD: Port 6 read signal

WR: Port 6 write signal

**Note** The Schmitt input buffer of P60, P64, P65 and P67 pins is turned off in the STOP and IDLE modes (Schmitt output is fixed to “L”.)

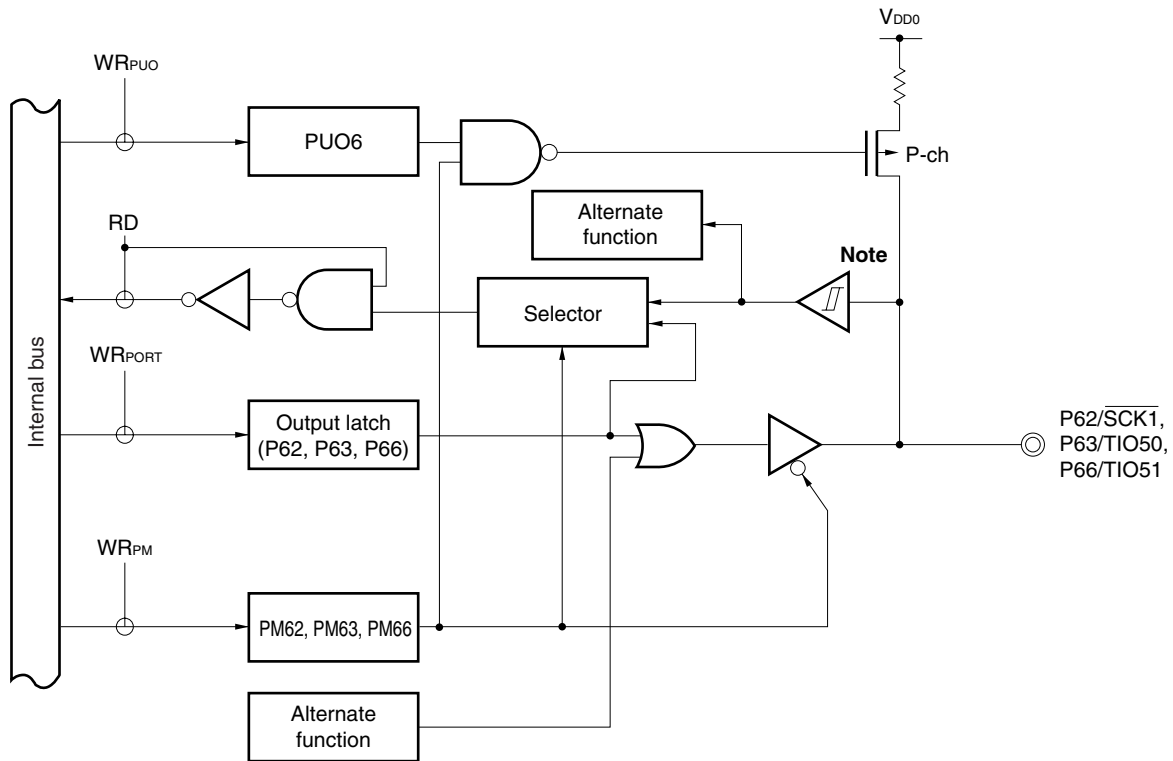


Figure 4-13. Block Diagram of P61



- PUO: Pull-up resistor option register
- PM6: Port 6 mode register
- RD: Port 6 read signal
- WR: Port 6 write signal

Figure 4-14. Block Diagram of P62, P63, and P66



PUO: Pull-up resistor option register

PM6: Port 6 mode register

RD: Port 6 read signal

WR: Port 6 write signal

**Note** The Schmitt input buffer of P62, P63 and P66 pins is turned off in the STOP and IDLE modes (Schmitt output is fixed to “L”.)

4.2.7 Port 7

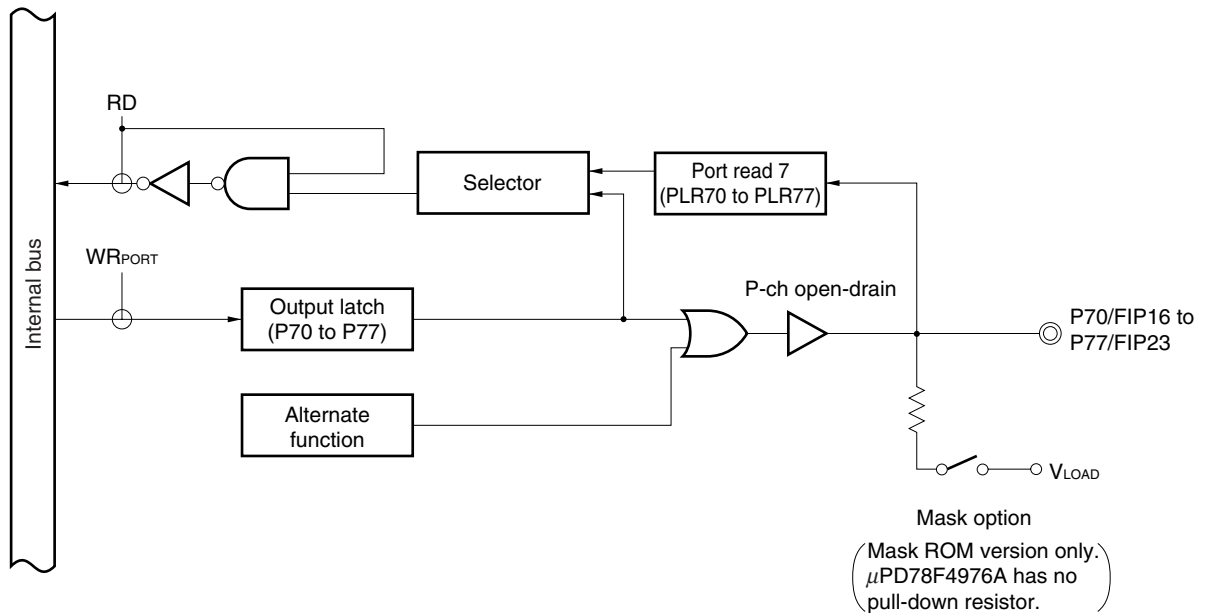
Port 7 is an 8-bit I/O port with output latch. When using this port as an output port, the value assigned to the output latch (P70 to P77) is output. When it is used as an input port, set the output latch (P70 to P77) to 0, and read port read 7 (PLR70 to PLR77). Setting the output latch (P70 to P77) to 1 causes a value to be read from the output latch itself. Use of the on-chip pull-down resistors can be specified in 1-bit units using a mask option in the ROM versions. The  $\mu$ PD78F4976A has no pull-down resistor.

Port 7 functions alternately as a VFD controller/driver output.

RESET input sets port 7 to input mode.

Figure 4-15 shows a block diagram of port 7.

Figure 4-15. Block Diagram of P70 to P77



RD: Port 7 read signal  
 WR: Port 7 write signal

4.2.8 Port 8

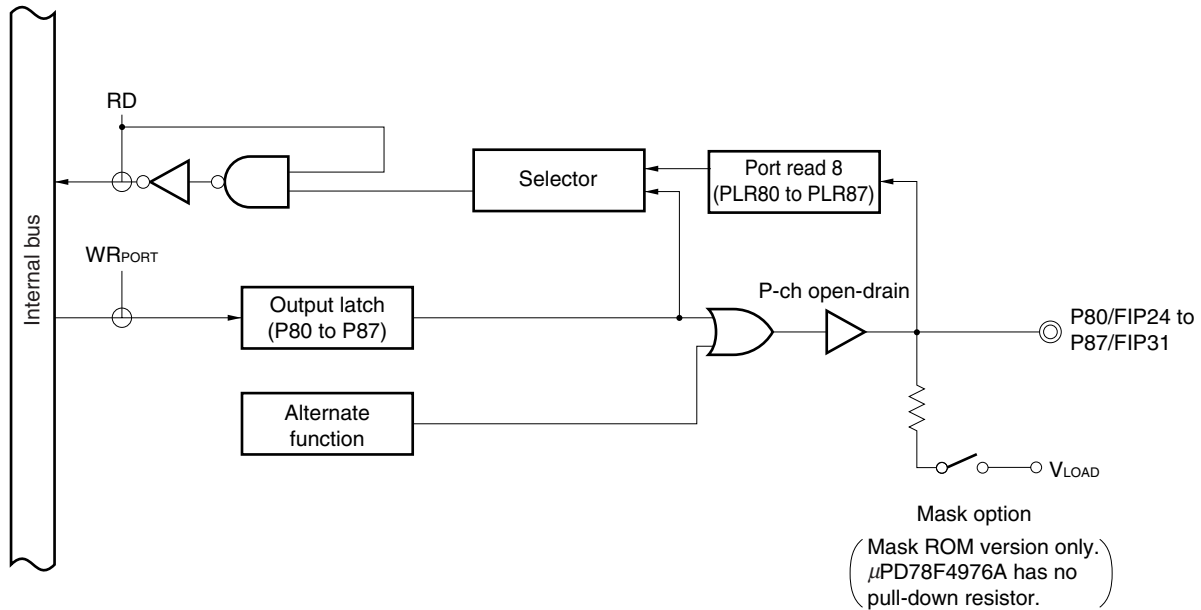
Port 8 is an 8-bit I/O port with output latch. When using this port as an output port, the value assigned to the output latch (P80 to P87) is output. When it is used as an input port, set the output latch (P80 to P87) to 0, and read port read 8 (PLR80 to PLR87). Setting the output latch (P80 to P87) to 1 causes a value to be read from the output latch itself. Use of the on-chip pull-down resistors can be specified in 1-bit units using a mask option in the mask ROM versions. The  $\mu$ PD78F4976A has no pull-down resistor.

Port 8 functions alternately as a VFD controller/driver output.

RESET input sets port 8 to input mode.

Figure 4-16 shows a block diagram of port 8.

Figure 4-16. Block Diagram of P80 to P87



RD: Port 8 read signal  
 WR: Port 8 write signal

**4.2.9 Port 9**

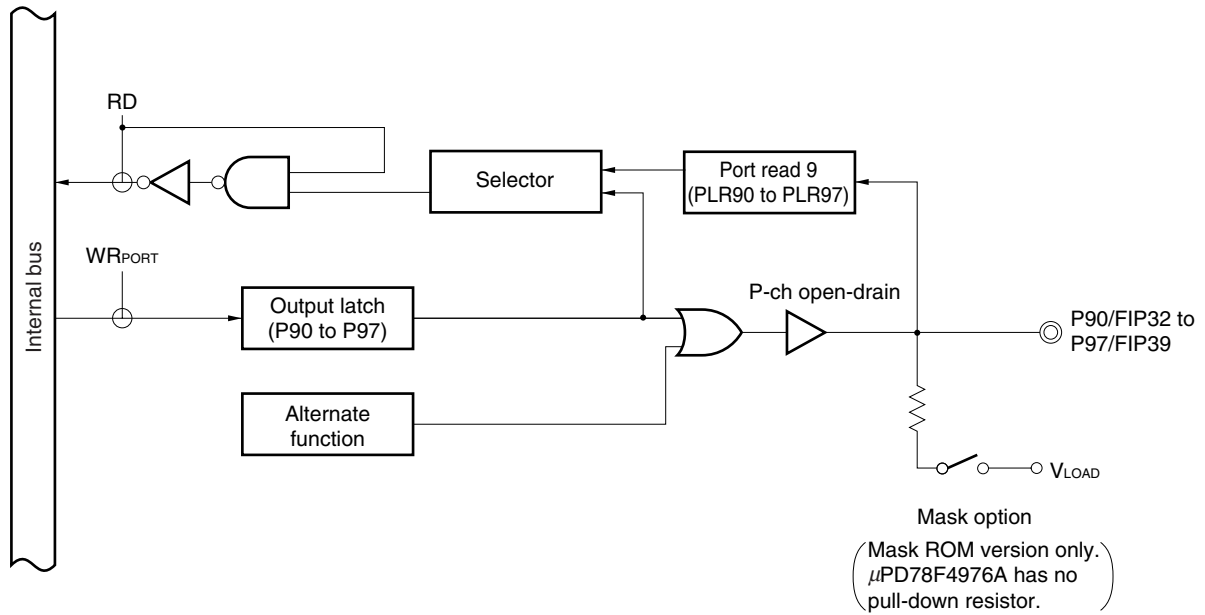
Port 9 is an 8-bit I/O port with output latch. When using this port as an output port, the value assigned to the output latch (P90 to P97) is output. When it is used as an input port, set the output latch (P90 to P97) to 0, and read port read 9 (PLR90 to PLR97). Setting the output latch (P90 to P97) to 1 causes a value to be read from the output latch itself. Use of the on-chip pull-down resistors can be specified in 1-bit units using a mask option in the mask ROM versions. The  $\mu$ PD78F4976A has no pull-down resistor.

Port 9 functions alternately as a VFD controller/driver output.

RESET input sets port 9 to input mode.

Figure 4-17 shows a block diagram of port 9.

**Figure 4-17. Block Diagram of P90 to P97**



RD: Port 9 read signal  
 WR: Port 9 write signal

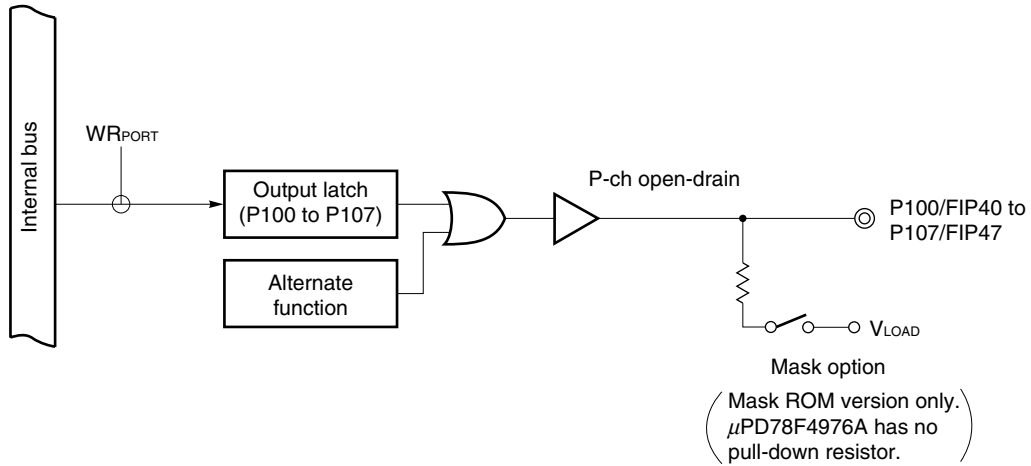
4.2.10 Port 10

Port 10 is an 8-bit output port. Use of the on-chip pull-down resistors can be specified in 1-bit units using a mask option in the mask ROM versions. The  $\mu$ PD78F4976A has no pull-down resistor.

Port 10 functions alternately as a VFD controller/driver output.

Figure 4-18 shows a block diagram of port 10.

Figure 4-18. Block Diagram of P100 to P107



WR: Port 10 write signal

### 4.3 Port Function Control Registers

The following two types of registers control the ports.

- Port mode registers (PM2, PM4 to PM6)
- Pull-up resistor option registers (PUO, PU2)

#### (1) Port mode registers (PM2, PM4 to PM6)

These registers are used to set port input/output in 1-bit units.

PM2 and PM4 to PM6 are independently set with a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$  input sets these registers to FFH.

When a port pin is used as its alternate function pin, set the port mode register and the output latch according to Table 4-3.

- Cautions**
1. Pins P00 to P03 and P10 to P17 are input pins.
  2. Pins P100 to P107 are output pins.
  3. Pins P64, P65, and P67 also function as external interrupt request inputs. If they are not used as interrupt input pins, set the external interrupt rising edge enable register (EGP0) and external interrupt falling edge enable register (EGN0) to “Interrupt Disable.” Or, set the interrupt mask flags (PMKn where n = 0 to 2) to 1. Otherwise, the interrupt request flag is set when the port function is placed in output mode and its output level is changed, leading to inadvertent interrupt servicing.

**Table 4-3. Port Mode Register and Output Latch Setting When Alternate Function Is Used**

Pin Name	Alternate Function		PM <sub>xx</sub>	P <sub>xx</sub>	Pin Name	Alternate Function		PM <sub>xx</sub>	P <sub>xx</sub>
	Function Name	Input/output				Function Name	Input/output		
P20	TIO0	Input	1	×	P60	SI1	Input	1	×
P25	SI0/RxD0	Input	1	×	P61	SO1	Output	0	0
P26	SO0/TxD0	Output	0	0	P62	$\overline{\text{SCK}}1$	Input/output	1/0	×/0
★ P27	$\overline{\text{SCK}}0$	Input/output	1/0	×/0	P63	TIO50	Input/output	1/0	×/0
	ASCK0	Input	1	×	P64	INTP0	Input	1	×
P55	SI2	Input	1	×	P65	INTP1	Input	1	×
P56	SO2	Output	1	0	P66	TIO51	Input/output	1/0	×/0
P57	$\overline{\text{SCK}}2$	Input/output	1	×/0	P67	INTP2	Input	1	×

**Cautions**

1. The setting of PM27 and PM62 varies depending on the clock selected by bits 1 and 0 (SCLn1 and SCLn0) of serial operation mode register n (CSIMn).

Internal clock (SCLn1, SCLn0 ≠ 0, 0): 0

External clock (SCLn1, SCLn0 = 0, 0): 1

Set  $\overline{\text{SCK}}2$  of the  $\mu\text{PD784975A}$  to PM<sub>xx</sub> = 1 (input) regardless of the setting of the internal or external clock.

2. Because the P64, P65, and P67 pins function alternately as external interrupt request inputs, if the output level is changed by setting the port function to output mode, the interrupt request flag is set. To use output mode, therefore, be sure to preset the interrupt mask flag to 1.
3. When the pins of these ports are being used for an alternate function, executing a read instruction for these ports results in undefined data being read.

**Remark**

- ×: don't care
- PM<sub>xx</sub>: Port mode register
- P<sub>xx</sub>: Port output latch

Figure 4-19. Format of Port Mode Register

Symbol	7	6	5	4	3	2	1	0	Address	After reset	R/W
PM2	PM27	PM26	PM25	1	1	1	1	PM20	FF22H	FFH	R/W
PM4	PM47	PM46	PM45	PM44	PM43	PM42	PM41	PM40	FF24H	FFH	R/W
PM5	PM57	PM56	PM55	PM54	PM53	PM52	PM51	PM50	FF25H	FFH	R/W
PM6	PM67	PM66	PM65	PM64	PM63	PM62	PM61	PM60	FF26H	FFH	R/W

PMmn	Pmn pin input/output mode selection (m = 2: n = 0, 5 to 7) (m = 4 to 6: n = 0 to 7)
0	Output mode (output buffer ON)
1	Input mode (output buffer OFF)

(2) Pull-up resistor option register 2 (PU2)

This register is used to set whether or not to use an on-chip pull-up resistor of pins at port 2 in 1-bit units. If PU2 specifies that an on-chip pull-up resistor is to be used for a bit, the pull-up resistor can be used for the bit internally, no matter whether the port is specified to be in input/output mode. Do not specify a pull-up resistor for any pin if port 2 is specified to be in output mode.

PU2 is set using a 1-bit or 8-bit memory manipulation instruction.

RESET input sets PU2 to 00H.

Figure 4-20. Format of Pull-Up Resistor Option Register 2 (PU2)

Symbol	7	6	5	4	3	2	1	0	Address	After reset	R/W
PU2	PU27	PU26	PU25	0	0	0	0	PU20	FF32H	00H	R/W

PU2n	P2n on-chip pull-up resistor selection (n = 0, 5 to 7)
0	On-chip pull-up resistor is not used
1	On-chip pull-up resistor is used



**(3) Pull-up resistor option register (PUO)**

This register specifies whether a pull-up resistor is to be used for ports 4 and 6.

PUO can specify that each of ports 4 and 6 is to be connected to on-chip pull-up resistors.

PUO is set using a 1-bit or 8-bit memory manipulation instruction.

RESET input sets PUO to 00H.

**Figure 4-21. Format of Pull-Up Resistor Option Register (PUO)**

Address: 0FF4EH After reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
PUO	0	PUO6	0	PUO4	0	0	0	0

PUOn	Pn pin on-chip pull-up resistor selection (n = 4, 6)
0	On-chip pull-up resistor is not used
1	On-chip pull-up resistor is used

**Caution** No pull-up resistor can be used for any pins of ports 4 and 6 that have been specified to be in output mode, even when PUOn is set to 1. A pull-up resistor can be used only for a pin that has been specified to be in input mode.

## 4.4 Port Function Operations

Port operations differ depending on whether the input or output mode is set, as shown below.

### 4.4.1 Writing to I/O port

#### (1) Output mode

A value is written to the output latch by a transfer instruction, and the output latch contents are output from the pin.

Once data is written to the output latch, it is retained until data is written to the output latch again.

#### (2) Input mode

A value is written to the output latch by a transfer instruction, but since the output buffer is off, the pin status does not change.

Once data is written to the output latch, it is retained until data is written to the output latch again.

**Caution** In the case of 1-bit memory manipulation instruction, although a single bit is manipulated the port is accessed as an 8-bit unit. Therefore, on a port with a mixture of input and output pins, the output latch contents for pins specified as input are undefined except for the manipulated bit.

### 4.4.2 Reading from I/O port

#### (1) Output mode

The output latch contents are read by a transfer instruction. The output latch contents do not change.

#### (2) Input mode

The pin status is read by a transfer instruction. The output latch contents do not change.

### 4.4.3 Operations on I/O port

#### (1) Output mode

An operation is performed on the output latch contents, and the result is written to the output latch. The output latch contents are output from the pins.

Once data is written to the output latch, it is retained until data is written to the output latch again.

#### (2) Input mode

The output latch contents are undefined, but since the output buffer is off, the pin status does not change.

**Caution** In the case of a 1-bit memory manipulation instruction, although a single bit is manipulated, the port is accessed as an 8-bit unit. Therefore, for a port with a mixture of input and output pins, the output latch contents for pins specified as input are undefined even for bits other than the manipulated bit.

## 4.5 Selecting Mask Option

- ★ The mask ROM versions have the following mask options, which can be used to select resistors of 90 k $\Omega$  (TYP.) or 50 k $\Omega$  (TYP.). The  $\mu$ PD78F4976A does not have mask options.

**Table 4-4. Comparison Between Mask Options of Mask ROM Version and  $\mu$ PD78F4976A**

Pin Name	Mask Option of Mask ROM Version	$\mu$ PD78F4976A
P50 to P57	Pull-up resistor can be connected in 1-bit units.	Pull-up resistor is not provided.
P70/FIP16 to P77/FIP23, P80/FIP24 to P87/FIP31, P90/FIP32 to P97/FIP39, P100/FIP40 to P107/FIP47	Pull-down resistor can be connected in 1-bit units.	Pull-down resistor is not provided.

## CHAPTER 5 CLOCK GENERATOR

### 5.1 Functions of Clock Generator

The clock generator generates clock to be supplied to the CPU and peripheral hardware. The following type of system clock oscillators is available.

- **Main system clock oscillator**

This circuit generates frequencies of 4 to 12.5 MHz. Setting the STOP bit of the standby control register (STBC) to 1 or entering  $\overline{\text{RESET}}$  can stop oscillation.

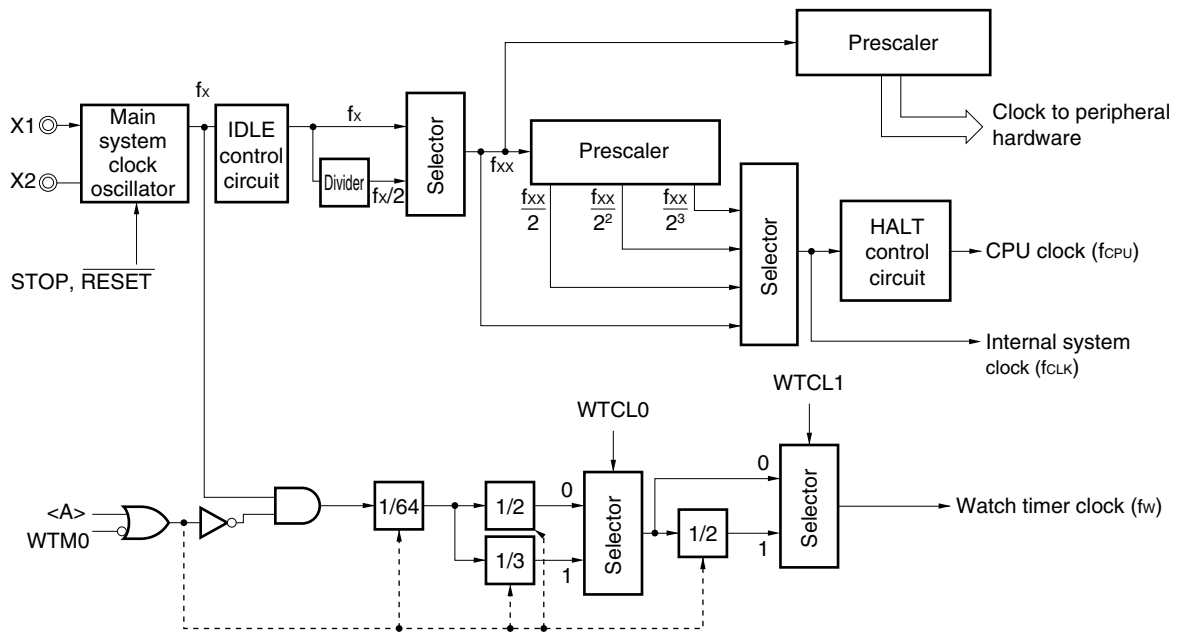
### 5.2 Configuration of Clock Generator

The clock generator includes the following hardware.

**Table 5-1. Configuration of Clock Generator**

Item	Configuration
Control register	Standby control register (STBC) Oscillation mode select register (CC) Oscillation stabilization time specification register (OSTS) Watch timer clock select register (WTCL)
Oscillator	Main system clock oscillator

Figure 5-1. Block Diagram of Clock Generator



- Remarks**
1.  $f_x$ : Main system clock oscillation frequency
  2.  $f_{xx}$ : Main system clock frequency
  3. ----->: Reset
  4. <A>: Status signal that indicates the period in which the oscillation is not stable
  5. WTCL0, WTCL1: Bits 0 and 1 of the watch timer clock select register (WTCL)
  6. WTM0: Bit 0 of the watch timer mode control register (WTM)

### 5.3 Control Register

#### (1) Standby control register (STBC)

This register is used to set the standby mode and select internal system clock. For the details of the standby mode, refer to **CHAPTER 17 STANDBY FUNCTION**.

The write operation can be performed only using the dedicated instruction to avoid entering into the standby mode due to an inadvertent program loop. The dedicated instruction, MOV STBC, #byte, have a special code structure (4 bytes). The write operation is performed only when the OP code of the 3rd byte and 4th byte are mutual 1's complements. When the 3rd byte and 4th byte are not mutual 1's complements, the write operation is not performed and an operand error interrupt is generated. In this case, the return address saved in the stack area indicates the address of the instruction that caused an error. Therefore, the address that caused an error can be determined from the return address that is saved in the stack area.

If a return from an operand error is performed simply with the RETB instruction, an infinite loop will be caused. Because the operand error interrupt occurs only in the case of an inadvertent program loop (if MOV STBC, #byte is described, only the correct dedicated instruction is generated in NEC's RA78K4 assembler), initialize the system for the program that processes an operand error interrupt.

Other write instructions such as MOV STBC, A; AND STBC, #byte; and SET1 STBC.7 are ignored and no operation is performed. In other words, neither is a write operation to STBC performed nor is an interrupt such as an operand error interrupt generated. STBC can be read out any time by means of a data transfer instruction. RESET input sets STBC to 30H.

Figure 5-2 shows the format of STBC.

The standby control register (STBC) is used to select a CPU clock, a frequency division ratio, and a mode (normal operation, HALT, IDLE, or STOP).

STBC is set using an 8-bit memory manipulation instruction.

RESET input sets STBC to 30H.

Figure 5-2. Format of Standby Control Register (STBC)

Address: 0FFC0H After reset: 30H R/W

Symbol	7	6	5	4	3	2	1	0
STBC	0	0	CK1	CK0	0	0	STP	HLT

CK1	CK0	CPU clock selection <sup>Note</sup> (in through-rate clock mode or oscillation division mode)
0	0	$f_{xx}$ ( $f_x$ , $f_x/2$ )
0	1	$f_{xx}/2$ ( $f_x/2$ , $f_x/2^2$ )
1	0	$f_{xx}/2^2$ ( $f_x/2^2$ , $f_x/2^3$ )
1	1	$f_{xx}/2^3$ ( $f_x/2^3$ , $f_x/2^4$ )

STP	HLT	Operation specification flag
0	0	Normal operation mode
0	1	HALT mode (cleared automatically when HALT mode is released)
1	0	STOP mode (cleared automatically when STOP mode is released)
1	1	IDLE mode (cleared automatically when IDLE mode is released)

**Note** A CPU clock can also be selected using the oscillation mode select register (CC).

- 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 (to 1) before setting the STOP mode. If the STOP mode is used with the EXTC bit of the OSTs cleared (to 0) when using external clock input, the  $\mu$ PD784975A may suffer damage or reduced reliability. 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 (refer to 4.3.1).
  2. Execute an NOP instruction three times after the standby instruction (after the standby mode has been released). Otherwise, the standby instruction cannot be executed if execution of the standby instruction and an interrupt request contend, and the interrupt is acknowledged after two or more instructions following the standby instruction have been executed. The instruction that is executed before acknowledging the interrupt is the one that is executed within up to 6 clocks after the standby instruction has been executed.

**Example**

```

MOV STBC #byte
NOP
NOP
NOP
•
•
•
    
```

**Remark**  $f_{xx}$ : Main system frequency ( $f_x$  or  $f_x/2$ )  
 $f_x$ : Main system clock oscillation frequency

**(2) Oscillation mode select register (CC)**

This register specifies whether clock output from the main system clock oscillator with the same frequency as the external clock (through rate clock mode), or clock output that is half of the original frequency is used to operate the internal circuit.

CC is set using a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$  input sets CC to 00H.

**Figure 5-3. Format of Oscillation Mode Select Register (CC)**

Address: 0FF7AH After reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
CC	ENMP	0	0	0	0	0	0	0

ENMP	CPU clock selection
0	Half of original oscillation frequency
1	Through rate clock mode

**Caution** The ENMP bit cannot be reset by software. This bit is reset performing the system reset.



**(3) Oscillation stabilization time specification register (OSTS)**

This register specifies the operation of the oscillator. Either a crystal/ceramic resonator or external clock is set to the EXTC bit in OSTS as the clock used. The STOP mode can be set even during external clock input only when the EXTC bit is set 1.

OSTS is set using a 1-bit or 8-bit transfer instruction.

$\overline{\text{RESET}}$  input sets OSTS to 00H.

**Figure 5-4. Format of Oscillation Stabilization Specification Register (OSTS)**

Address: 0FFCFH After reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
OSTS	EXTC	0	0	0	0	OSTS2	OSTS1	OSTS0

EXTC	External clock selection
0	Crystal/ceramic resonator is used
1	External clock is used

EXTC	OSTS2	OSTS1	OSTS0	Oscillation stabilization time selection
0	0	0	0	$2^{19}/f_{xx}$ (41.9 ms)
0	0	0	1	$2^{18}/f_{xx}$ (21.0 ms)
0	0	1	0	$2^{17}/f_{xx}$ (10.5 ms)
0	0	1	1	$2^{16}/f_{xx}$ (5.2 ms)
0	1	0	0	$2^{15}/f_{xx}$ (2.6 ms)
0	1	0	1	$2^{14}/f_{xx}$ (1.3 ms)
0	1	1	0	$2^{13}/f_{xx}$ (655 $\mu\text{s}$ )
0	1	1	1	$2^{12}/f_{xx}$ (328 $\mu\text{s}$ )
1	×	×	×	$512/f_{xx}$ (41.0 $\mu\text{s}$ )

- Cautions**
1. When a crystal/ceramic resonator is used, make sure to clear the EXTC bit to 0. If the EXTC bit is set to 1, oscillation stops.
  2. When using the STOP mode during external clock input, make sure to set the EXTC bit to 1 before setting the STOP mode. If the STOP mode is used during external clock input when the EXTC bit of OSTS has been cleared to 0, the  $\mu\text{PD784975A}$  may be damaged or its reliability may be impaired.
  3. If the EXTC bit is set to 1 during external clock input, the opposite phase of the clock input to the X1 pin must be input to the X2 pin. If the EXTC bit is set to 1, the  $\mu\text{PD784975A}$  only operates with the clock input to the X2 pin.

- Remarks**
1. The values in parentheses are valid for operation when  $f_{xx}$  is 12.5 MHz.
  2. ×: don't care

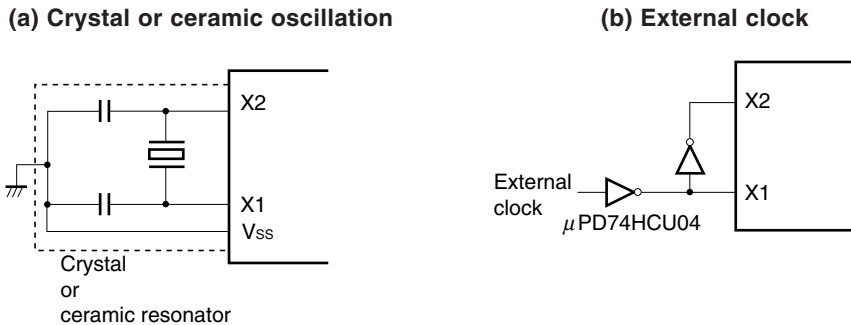
## 5.4 Main System Clock Oscillator

The main system clock oscillator oscillates with a crystal resonator or a ceramic resonator (12.5 MHz TYP.) connected to the X1 and X2 pins.

External clocks can be input to the main system clock oscillator. In this case, input a clock signal to the X1 pin and its inverse clock signal to the X2 pin.

Figure 5-5 shows an external circuit of the main system clock oscillator.

**Figure 5-5. External Circuit of Main System Clock Oscillator**



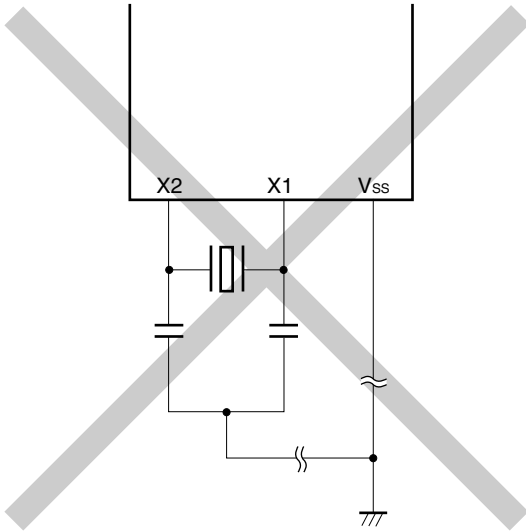
**Caution** When using the main system clock oscillator, wire as follows in the area enclosed by the broken lines in the above figures to avoid an adverse effect from wiring capacitance.

- Keep the wiring length as short as possible.
- Do not cross the wiring with the other signal lines.
- Do not route the wiring near a signal line through which a high fluctuating current flows.
- Always make the ground point of the oscillator capacitor the same potential as  $V_{SS}$ . Do not ground the capacitor to a ground pattern through which a high current flows.
- Do not fetch signals from the oscillator.

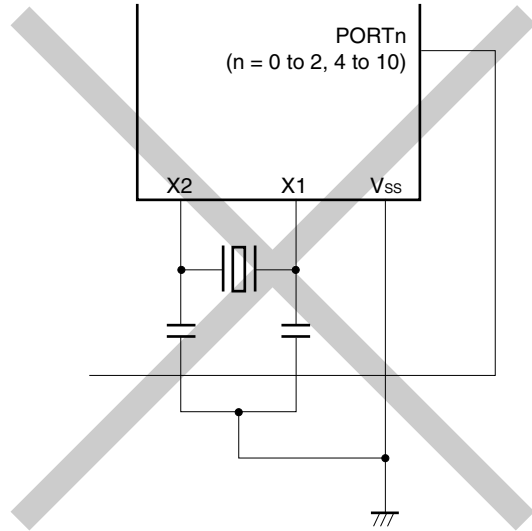
Figure 5-6 shows examples of oscillators that are connected incorrectly.

Figure 5-6. Examples of Oscillator Connected Incorrectly (1/2)

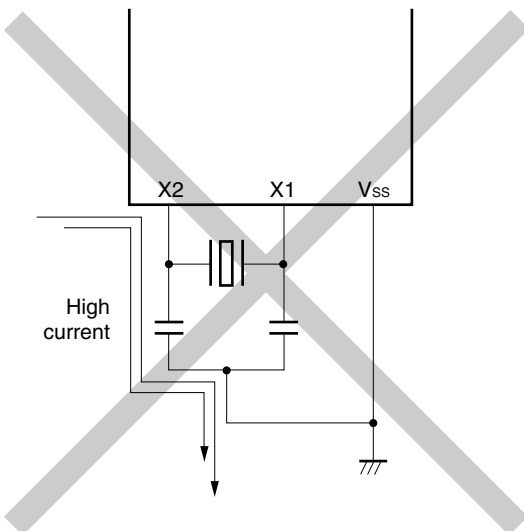
(a) Wiring of connection circuits is too long



(b) Signal conductors intersect each other



(c) Changing high current is too near a signal conductor



(d) Current flows through the ground line of the oscillator (potential at points A, B, and C fluctuate)

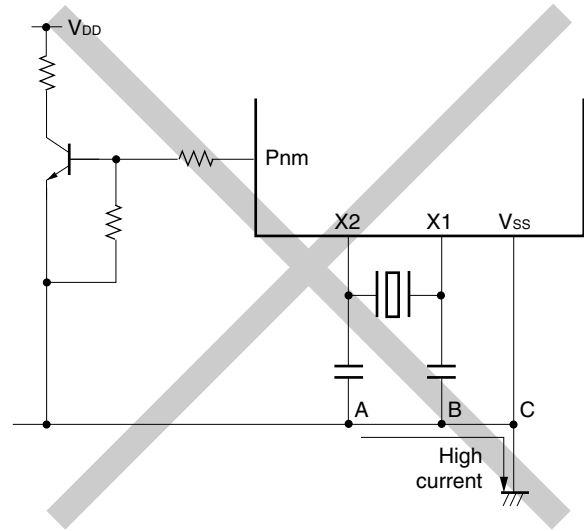
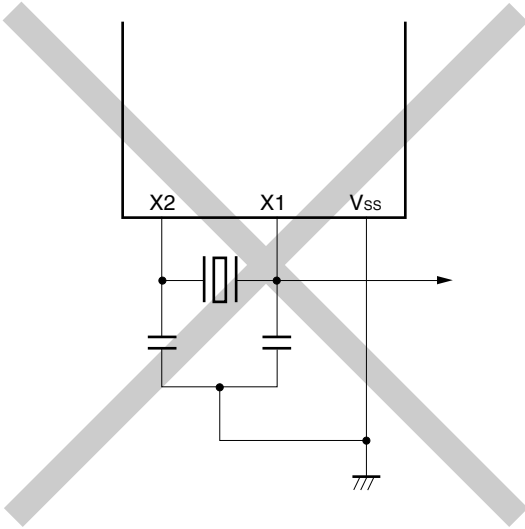


Figure 5-6. Examples of Oscillator Connected Incorrectly (2/2)

(e) Signals are fetched



#### 5.4.1 Divider

The divider divides the output frequency of the main system clock oscillator ( $f_{xx}$ ) to generate different clocks.

## 5.5 Operations of Clock Generator

The clock generator generates the following types of clocks and controls the CPU operating mode including the standby mode.

- Main system clock ( $f_{xx}$ )
- CPU clock ( $f_{CPU}$ )
- Clock to peripheral hardware
- Internal system clock ( $f_{CLK}$ )
- Watch timer clock ( $f_w$ )

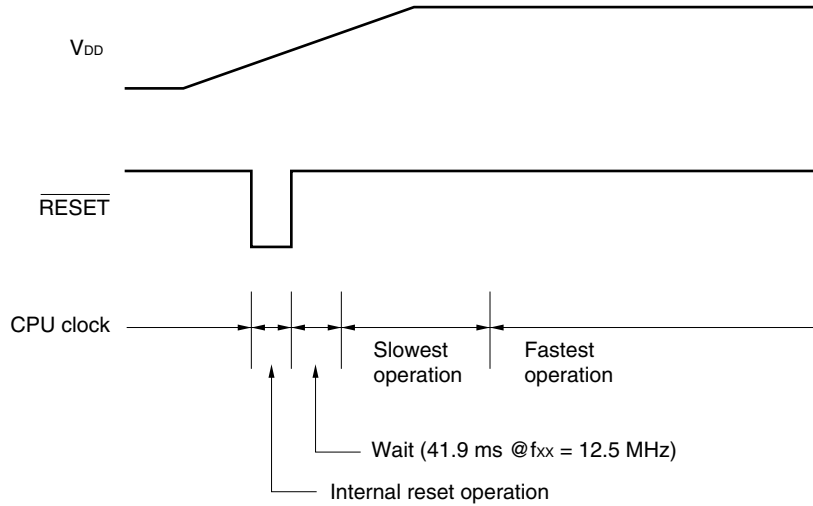
The following clock generator functions and operations are determined with the standby control register (STBC) and the oscillation mode select register (CC).

- Upon generation of the  $\overline{RESET}$  signal, the lowest speed mode of the main system clock (1,280 ns @  $f_{xx} = 12.5$  MHz) is selected (STBC = 30H, CC = 00H). Main system clock oscillation stops while low level is applied to the  $\overline{RESET}$  pin.
- With the main system clock selected, setting STBC and CC appropriately can select any of the five CPU clocks (80 ns, 160 ns, 320 ns, 640 ns, and 1,280 ns @  $f_{xx} = 12.5$  MHz).
- With the main system clock selected, three standby modes, the STOP, HALT, and IDLE modes, are available.
- The main system clock is divided and supplied to the peripheral hardware. Thus, the peripheral hardware (except external input clock operation) also stops if the main system clock is stopped.

## 5.6 Changing CPU Clock Setting

The CPU clock can be switched by means of bits 4 and 5 (CK0 and CK1) of the standby control register (STBC). The CPU clock is changed in the following procedure.

Figure 5-7. Changing CPU Clock



- (1) The CPU is reset if the  $\overline{\text{RESET}}$  pin is made low after power application. The reset is cleared and the main system clock starts oscillating if the  $\overline{\text{RESET}}$  pin is later made high. At this time, it is automatically ensured that oscillation stabilization time ( $2^{19}/f_x$ ) elapses. Subsequently, the CPU starts executing instructions at the lowest speed of the main system clock (1,280 ns @  $f_{xx} = 12.5$  MHz).
- (2) After sufficient time has elapsed during which the  $V_{DD}$  voltage rises to the level at which the CPU can operate at the highest speed, the contents of the standby control register (STBC) and oscillation mode select register (CC) are rewritten, and the CPU operates at the highest speed.

## CHAPTER 6 TIMER COUNTER OVERVIEW

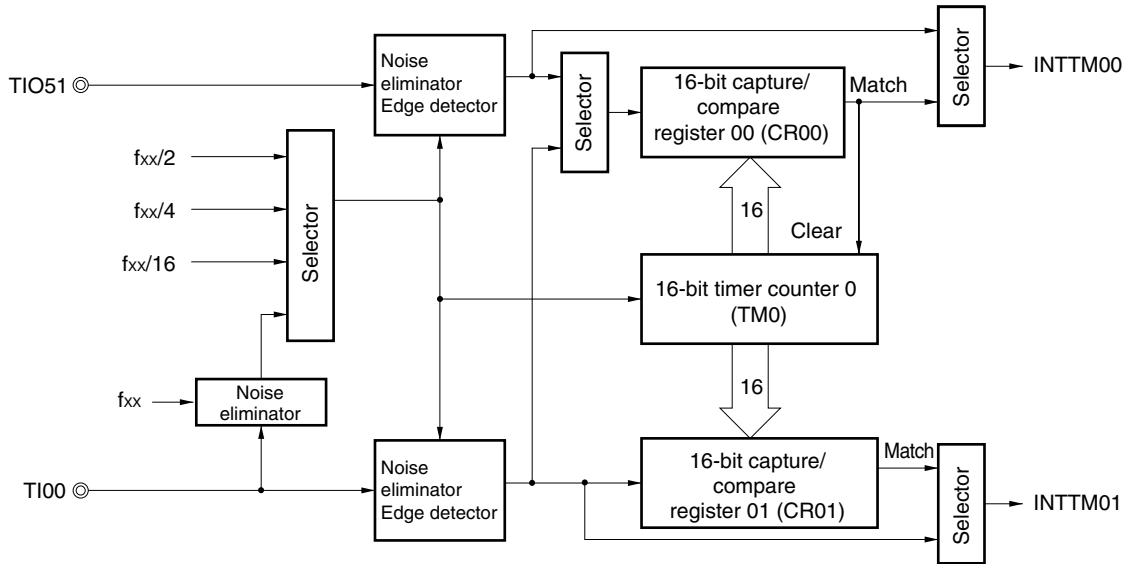
The chip incorporates one 16-bit timer/event counter and two 8-bit PWM timers. Because the chip supports a total of four interrupt requests, four timer counter units can be used.

**Table 6-1. Timer Counter Operation**

Item		Name	16-Bit Timer/Event Counter 0	8-Bit PWM Timer (TM50)	8-Bit PWM Timer (TM51)
Count width	8 bits		—	○	○
	16 bits		○	○	
Operation mode	Interval timer		1 ch	1 ch	1 ch
	External event counter		○	○	○
Function	Timer output		—	1 ch	1 ch
	PWM output		—	○	○
	Square wave output		—	○	○
	Pulse width measurement		Two inputs	—	—
	Number of interrupt requests		2	1	1

**Figure 6-1. Block Diagram of Timer Counter (1/2)**

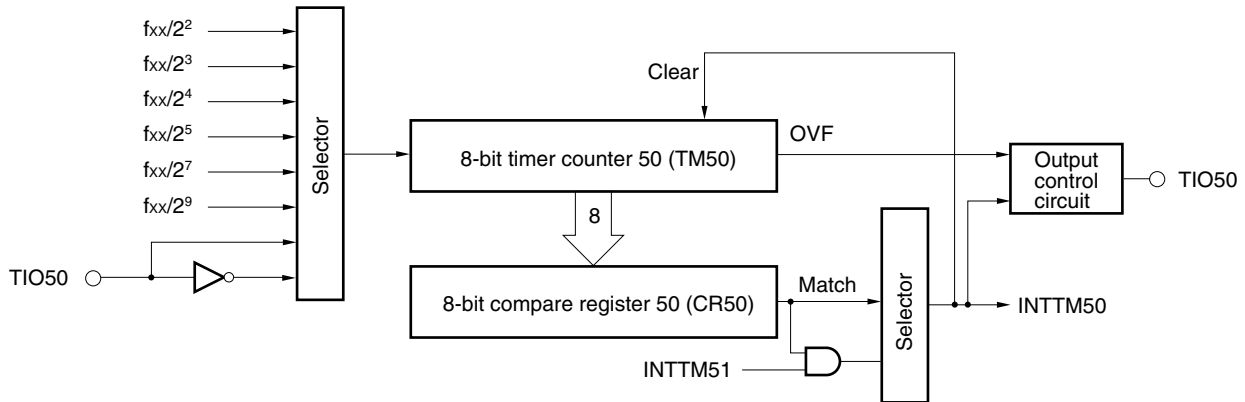
• **16-bit timer/event counter 0**



- Remarks**
1. fxx: Main system clock frequency
  2. Pin TIO51 functions alternately as an external clock input to, and timer output from, TM51.

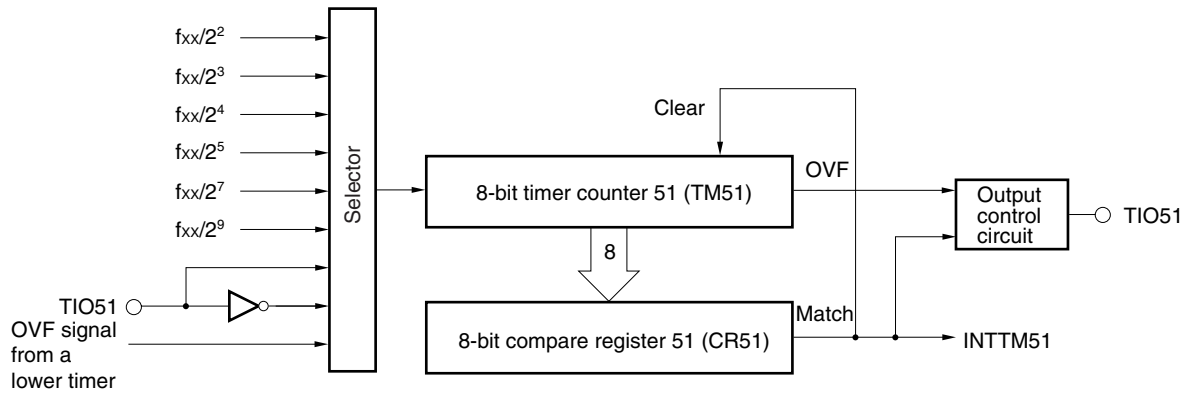
Figure 6-1. Block Diagram of Timer Counter (2/2)

• 8-bit PWM timer 50 (TM50)



- Remarks**
1.  $f_{xx}$ : Main system clock frequency
  2. OVF: Overflow flag

• 8-bit PWM timer 51 (TM51)



- Remarks**
1.  $f_{xx}$ : Main system clock frequency
  2. OVF: Overflow flag
  3. Pin TIO51 functions alternately as a capture input to TM0.



## CHAPTER 7 16-BIT TIMER/EVENT COUNTER

### 7.1 Function

16-bit timer/event counter 0 has the following functions.

- Interval timer
- Pulse width measurement
- External event counter
- Remote controller receive interrupt generation

#### (1) Interval timer

When the 16-bit timer/event counter is used as an interval timer, it generates an interrupt request at predetermined time intervals.

#### (2) Pulse width measurement

The 16-bit timer/event counter can be used to measure the pulse width of a signal input from an external source.

#### (3) External event counter

The 16-bit timer/event counter can be used to measure the number of pulses of a signal input from an external source.

#### (4) Remote controller receive interrupt generation

The 16-bit timer/event counter automatically identifies the pulse width of the signal input from the TI00 pin in accordance with the preset min. and max. values, and generates an interrupt request signal.

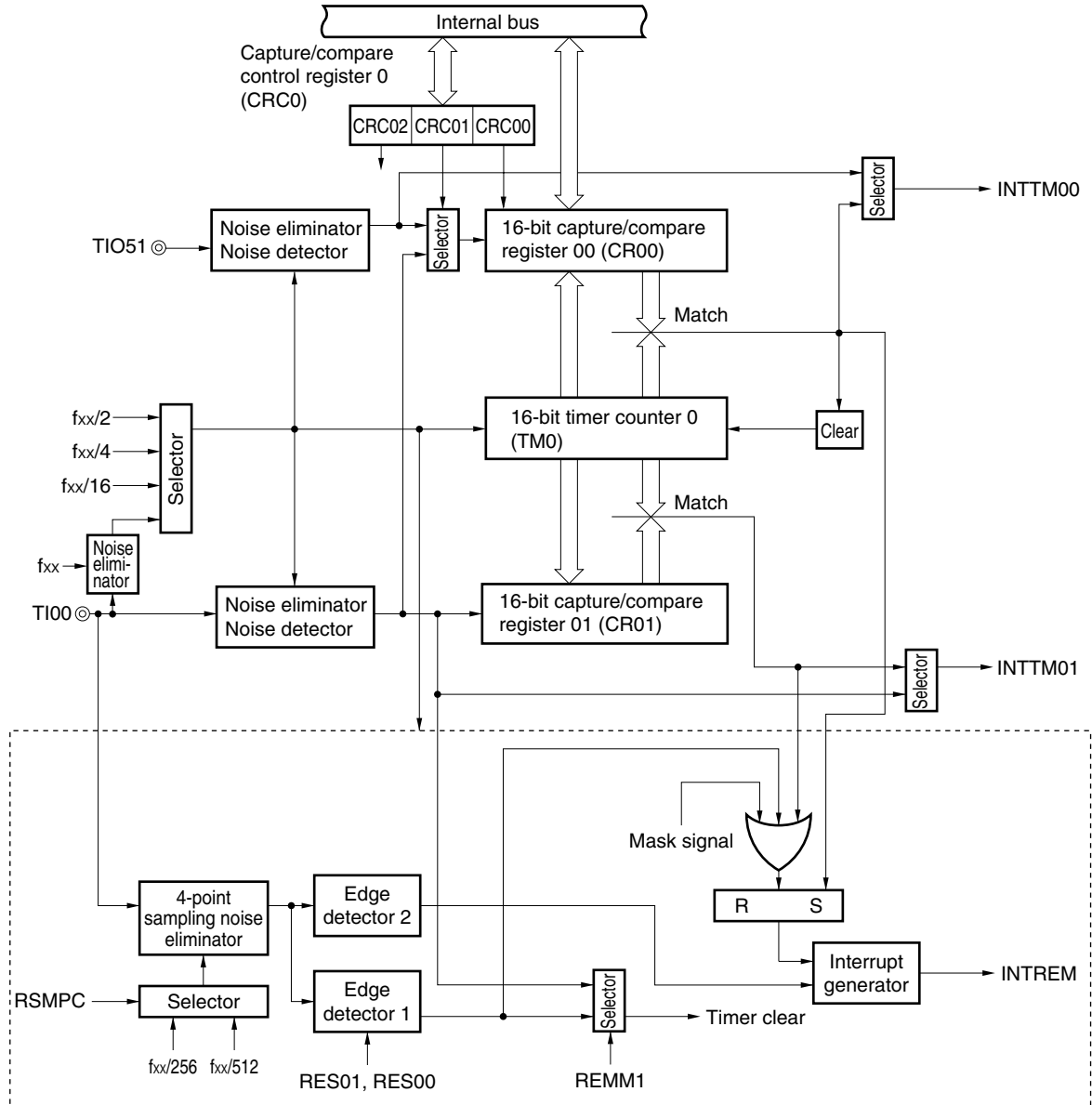
## 7.2 Configuration

16-bit timer/event counter 0 includes of the following hardware.

**Table 7-1. Configuration of 16-Bit Timer/Event Counter 0**

Item	Configuration
Timer register	16 bits × 1 (TM0)
Register	16-bit capture/compare register: 16 bits × 2 (CR00, CR01)
External clock input	1 (TIO0)
Control registers	16-bit timer mode control register 0 (TMC0) Capture/compare control register 0 (CRC0) Prescaler mode register 0 (PRM0) Remote controller receive mode register (REMM)

Figure 7-1. Block Diagram of 16-Bit Timer/Event Counter 0



**Remarks 1.**  $f_{xx}$ : Main system clock frequency

**2.** Pin TIO51 functions alternately as an external clock input to 8-bit PWM timer 51 (TM51) and a timer output.

**3.** [Dashed box]: Remote controller receive interrupt generator block

**4.** REMPC, REMM1, RES00, RES01: Bits 0, 1, 4, and 5 of the remote controller receive mode register (REMM)

**5.** INTREM: Remote controller receive interrupt request signal

**(1) 16-bit timer counter 0 (TM0)**

TM0 is a 16-bit read-only register that counts count pulses.

The counter is incremented in synchronization with the rising edge of an input clock. If the count value is read during operation, input of the count clock is temporarily stopped, and the count value at that point is read. The count value is reset to 0000H in the following cases:

- <1>  $\overline{\text{RESET}}$  is input.
- <2> TMC03 and TMC02 are cleared.
- <3> Valid edge of TI00 is input in the clear & start mode by inputting valid edge of TI00.
- <4> TM0 and CR00 match with each other in the clear & start mode on match between TM0 and CR00.

**(2) 16-bit capture/compare register 00 (CR00)**

CR00 is a 16-bit register that functions as a capture register and as a compare register. Whether this register functions as a capture or compare register is specified by using bit 0 (CRC00) of capture/compare control register 0.

- **When using CR00 as compare register**

The value set to CR00 is always compared with the count value of 16-bit timer counter 0 (TM0). When the values of the two match, an interrupt request (INTTM00) is generated. When TM0 is used as an interval timer, CR00 can also be used as a register that holds the interval time.

- **When using CR00 as capture register**

The valid edge of the TI00 or TIO51 pin can be selected as a capture trigger. The valid edges of TI00 and TIO51 are set by prescaler mode register 0 (PRM0).

Tables 7-2 and 7-3 show the valid edges of the TI00 pin and the valid edges of the TIO51 pin that apply when capture triggers are specified.

**Table 7-2. Valid Edge of TI00 Pin and Valid Edge of Capture Trigger of CR00**

ES01	ES00	Valid Edge of TI00 Pin	Capture Trigger of CR00
0	0	Falling edge	Rising edge
0	1	Rising edge	Falling edge
1	0	Setting prohibited	Setting prohibited
1	1	Both rising and falling edges	No capture operation

**Table 7-3. Valid Edge of TIO51 Pin and Valid Edge of Capture Trigger of CR00**

ES11	ES10	Valid Edge of TIO51 Pin	Capture Trigger of CR00
0	0	Falling edge	Falling edge
0	1	Rising edge	Rising edge
1	0	Setting prohibited	Setting prohibited
1	1	Both rising and falling edges	Both rising and falling edges

CR00 is set using a 16-bit memory manipulation instruction.

RESET input sets CR00 to 0000H.

**Caution** Set CR00 to the value other than 0000H. When using the register as an event counter, a count for one-pulse can not be operated.

### (3) 16-bit capture/compare register 01 (CR01)

This is a 16-bit register that can be used as a capture register and a compare register. Whether it is used as a capture register or compare register is specified by bit 2 (CRC02) of the capture/compare control register 0.

- **When using CR01 as compare register**

The value set to CR01 is always compared with the count value of 16-bit timer counter 0 (TM0). When the values of the two match, an interrupt request (INTTM01) is generated.

- **When using CR01 as capture register**

The valid edge of the TI00 pin can be selected as a capture trigger. The valid edge of TI00 is set by prescaler mode register 0 (PRM0).

Table 7-4 shows the valid edges of the TI00 pin that apply when capture triggers are specified.

**Table 7-4. Valid Edge of Pin TI00 and Valid Edge of Capture Trigger of CR01**

ES01	ES00	Valid Edge of TI00 Pin	Capture Trigger of CR01
0	0	Falling edge	Falling edge
0	1	Rising edge	Rising edge
1	0	Setting prohibited	Setting prohibited
1	1	Both rising and falling edges	Both rising and falling edges

CR01 is set using a 16-bit memory manipulation instruction.

RESET input sets CR01 to 0000H.

**Caution** Set CR01 to the value other than 0000H. When using an event counter, a count for one-pulse can not be operated.

### 7.3 Control Register

The following four types of registers control 16-bit timer/event counter 0.

- 16-bit timer mode control register 0 (TMC0)
- Capture/compare control register 0 (CRC0)
- Prescaler mode register 0 (PRM0)

#### (1) 16-bit timer mode control register 0 (TMC0)

This register specifies the operation mode of the 16-bit timer, and the clear mode and overflow detection of 16-bit timer counter 0.

TMC0 is set using a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$  input sets TMC0 to 00H.

**Caution** 16-bit timer counter 0 (TM0) starts operating when a value other than a combination of 0 and 0 (operation stop mode) is set to TMC02 and TMC03. To stop the operation, set a combination of 0 and 0 to TMC02 and TMC03.

**Figure 7-2. Format of 16-Bit Timer Mode Control Register 0 (TMC0)**

Address: 0FF18H After reset: 00H R/W

Symbol	7	6	5	4	3	2	1	<0>
TMC0	0	0	0	0	TMC03	TMC02	0	OVF0

TMC03	TMC02	TM0 operating mode specification
0	0	Operation stop (TM0 is cleared to 0).
0	1	Free-running mode
1	0	Clears and starts at valid edge input to TI00.
1	1	Clears and starts on match between TM0 and CR00.

OVF0	16-bit timer counter 0 (TM0) overflow detection
0	Does not overflow.
1	Overflows.

- Cautions**
1. To specify the valid edge for pin TI00, use prescaler mode register 0 (PRM0).
  2. When the clear & start mode on match between TM0 and CR00 is selected, the OVF0 flag is set to 1 when the value of TM0 changes from FFFFH to 0000H with CR00 set to FFFFH.

- Remarks**
1. At any timing, writing 0 to OVF0 causes it to be cleared.
  2. TI00: Input pin of 16-bit timer/event counter 0  
 TM: 16-bit timer counter 0  
 CR00: Compare register 00  
 CR01: Compare register 01

**(2) Capture/compare control register 0 (CRC0)**

This register controls the operation of the capture/compare registers (CR00 and CR01).

CRC0 is set using a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$  input sets CRC0 to 00H.

**Figure 7-3. Format of Capture/Compare Control Register 0 (CRC0)**

Address: 0FF16H After reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
CRC0	0	0	0	0	0	CRC02	CRC01	CRC00

CRC02	CR01 operation mode selection
0	Operates as compare register.
1	Operates as capture register.

CRC01	CR00 capture trigger selection
0	Captured at valid edge of TIO51.
1	Captured in reverse phase of valid edge of TI00.

CRC00	CR00 operation mode selection
0	Operates as compare register.
1	Operates as capture register.

- Cautions**
1. Before setting CRC0, be sure to stop the timer operation.
  2. When the clear & start mode on match between TM0 and CR00 is selected by the 16-bit timer mode control register (TMC0), do not specify CRC00 as a capture register.
  3. If both the rising and falling edges are specified as the valid edges for pin TI00, using prescaler mode register 0 (PRM0), CRC00 is disabled from capture operation.
  4. Performing capture securely requires that the capture trigger pulse be longer than two cycles of the count clock selected using PRM0.

**(3) Prescaler mode register 0 (PRM0)**

This register selects a count clock of 16-bit timer/event counter 0 and the valid edges of TI00 and TIO51 inputs. PRM0 is set using a 1-bit or 8-bit memory manipulation instruction.  $\overline{\text{RESET}}$  input sets PRM0 to 00H.

**Figure 7-4. Format of Prescaler Mode Register 0 (PRM0)**

Address: 0FF1CH After reset : 00H R/W

Symbol	7	6	5	4	3	2	1	0
PRM0	ES11	ES10	ES01	ES00	0	0	PRM01	PRM00

ES11	ES10	TIO51 valid edge selection
0	0	Falling edge
0	1	Rising edge
1	0	Setting prohibited
1	1	Both falling and rising edges

ES01	ES00	TI00 valid edge selection
0	0	Falling edge
0	1	Rising edge
1	0	Setting prohibited
1	1	Both falling and rising edges

PRM01	PRM00	Count clock selection
0	0	$f_{xx}/2$ (6.25 MHz)
0	1	$f_{xx}/4$ (3.13 MHz)
1	0	$f_{xx}/16$ (781 kHz)
1	1	Valid edge of TI00

**Caution** When selecting the valid edge of TI00 as the count clock, do not specify the valid edge of TI00 to clear and start the timer and as a capture trigger.

**Remark** The value in parentheses are valid for operation when  $f_{xx}$  is 12.5 MHz.



**(4) Remote controller receive mode register (REMM)**

This register controls the remote controller receive interrupt generator.

REMM is set using a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$  input sets REMM to 00H.

**Figure 7-5. Format of Remote Controller Receive Mode Register (REMM)**

Address: 0FF1EH After reset : 00H R/W

Symbol	7	6	5	4	3	2	1	0
PRMM	RES11	RES10	RES01	RES00	0	0	REMM1	RSMPC

RES11	RES10	TI00 pin valid edge selection 2 (detection timing)
0	0	Edge not detected
0	1	Falling edge
1	0	Rising edge
1	1	Setting prohibited

RES01	RES00	TI00 pin valid edge selection 1 (TM0 clear & start timing)
0	0	Edge not detected
0	1	Falling edge
1	0	Rising edge
1	1	Setting prohibited

REMM1	TM0 clear signal selection when TI00 pin valid edge is input in clear & start mode)
0	Clear signal at TI00 valid edge using bits 4 and 5 (ES00, ES01) of prescaler mode register 0 (PPRM0) (be sure to set this bit to 0 when not using the remote controller receive interrupt generator <sup>Note 2</sup> ; otherwise the INTREM interrupt request signal cannot be generated.)
1	Clear signal at TI00 valid edge using bits 4 and 5 (RES00, RES01) of REMM (be sure to set this bit to 1 when using the remote controller receive interrupt generator <sup>Note 1</sup> .)

RSMPC	Sampling clock selection for 4-point sampling noise eliminator
0	$f_{xx}/256$
1	$f_{xx}/512$

**Notes 1.** When using 16-bit timer/event counter 0 as a remote controller receive interrupt:

- Set bit 1 (REMM1) to 1 and set the operation mode of 16-bit timer/event counter 0 to clear & start mode by inputting the TI00 valid edge (set bits 2 and 3 (TMC02 and TMC03) to “0, 1” using 16-bit timer mode control register 1 (TMC1)). The remote controller receive interrupt operation starts when TMC02 and TMC03 are set. To stop the operation, set TMC02 and TMC03 to “0, 0”.
- Be sure to set as described above when using the counter to generate a remote controller receive interrupt; otherwise the operation is not guaranteed.

**2.** When not using 16-bit timer/event counter 0 as a remote controller receive interrupt:

- Set bit 1 (REMM1) to 0 and bits 4 to 7 (RES00, RES01, RES10, RES11) to 0, and set 16-bit timer/event counter 0 to the operation mode using bits 2 and 3 (TMC02 and TMC03) of TMC1. The operation starts when the operation mode is set. To stop the operation, set TMC02 and TMC03 to “0, 0”.

**Cautions 1.** When writing to REMM, make sure that the timer operation has been stopped.

- 2.** When not using the remote controller receive interrupt generator, set bit 1 (REMM) and bits 4 to 7 (RES00, RES01, RES10, RES11) to 0.



Figure 7-7. Configuration of Interval Timer

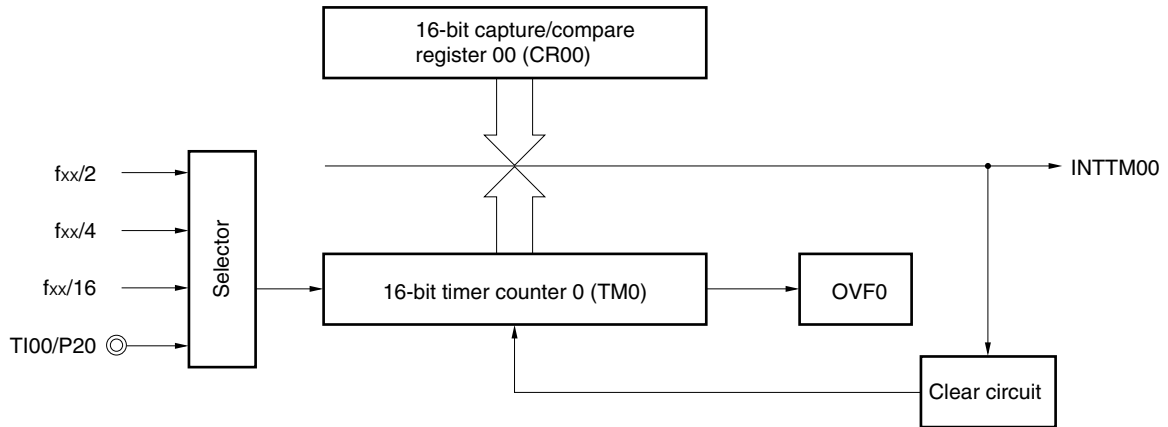
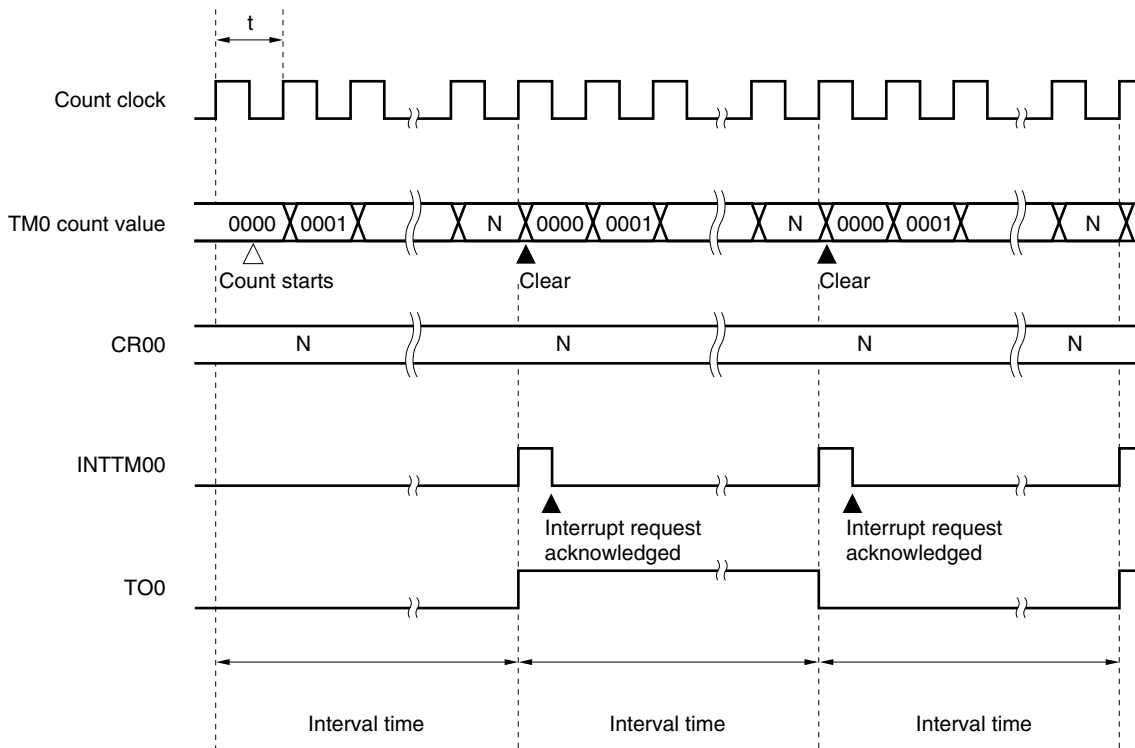


Figure 7-8. Timing of Interval Timer Operation



**Remark** Interval time = (N + 1) × t: N = 0001H to FFFFH

**7.4.2 Pulse width measurement**

16-bit timer counter 0 (TM0) can be used to measure the pulse widths of the signals input to the TI00/P20 and TIO51/P66 pins.

Measurement can be carried out with TM0 used as a free-running counter or by restarting the timer in synchronization with the edge of the signal input to the TI00/P20 pin.

**(1) Pulse width measurement with free running counter and one capture register**

If the edge specified by prescaler mode register 0 (PRM0) is input to the TI00/P20 pin when 16-bit timer counter 0 (TM0) is used as a free-running counter (refer to **Figure 7-9**), the value of TM0 is loaded to 16-bit capture/compare register 01 (CR01), and an external interrupt request signal (INTTM01) is set.

The edge is specified by using bits 4 and 5 (ES00 and ES01) of PRM0. The rising edge, falling edge, or both the rising and falling edges can be selected.

Sampling is performed with the count clock selected by PRM0, and the capture operation is performed when the valid level of the TI00 pin is detected two times. Therefore, noise with a short pulse width can be eliminated.

**Figure 7-9. Control Register Settings for Pulse Width Measurement with Free-Running Counter and One Capture Register**

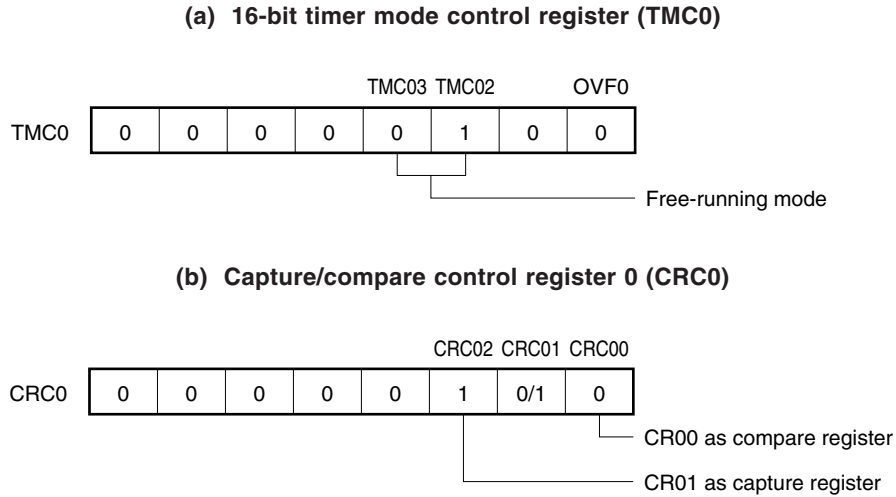


Figure 7-10. Configuration for Pulse Width Measurement with Free-Running Counter

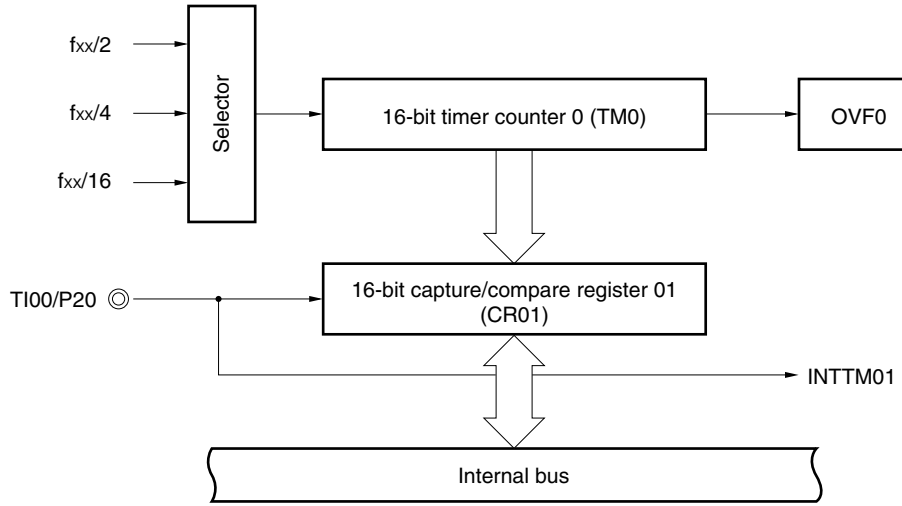
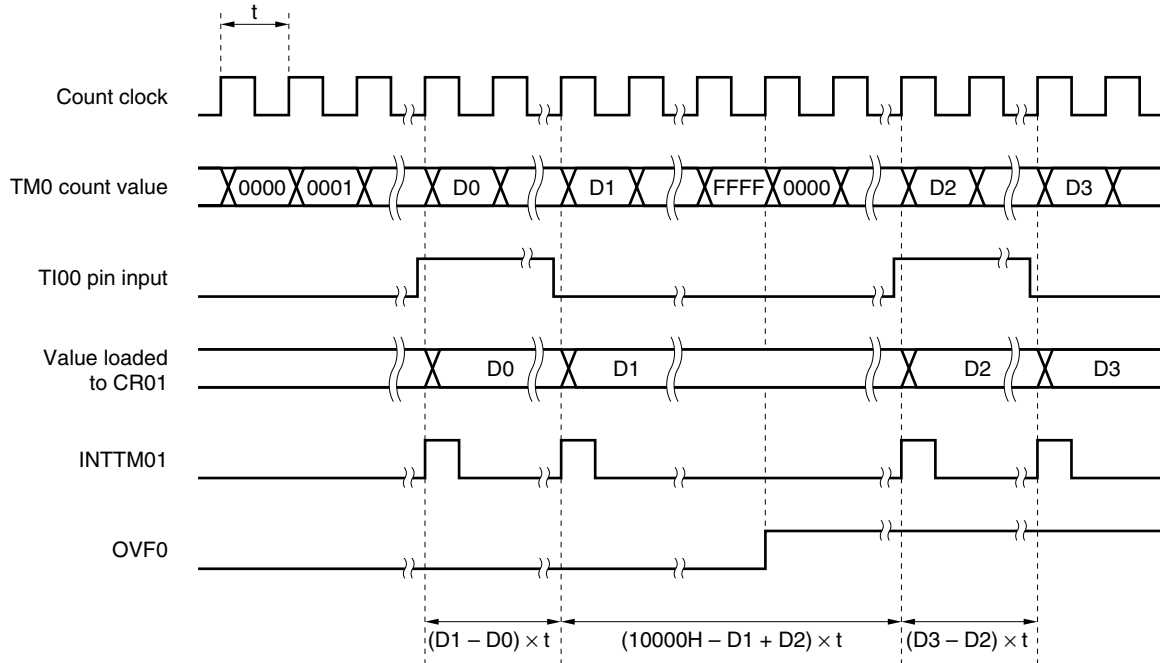


Figure 7-11. Timing of Pulse Width Measurement with Free-Running Counter and One Capture Register (with Both Edges Specified)



**(2) Measurement of two pulse widths with free-running counter**

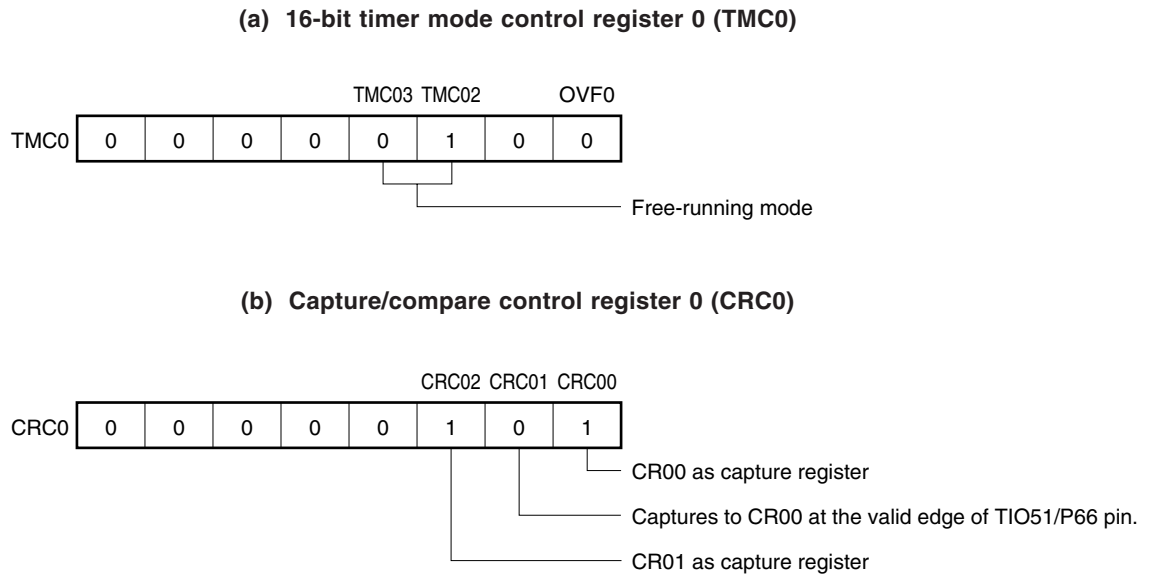
The pulse widths of the two signals respectively input to the TIO0/P20 and TIO51/P66 pins can be measured when 16-bit timer counter 0 (TM0) is used as a free-running counter (refer to the register settings in **Figure 7-12**). When the edge specified by bits 4 and 5 (ES00 and ES01) of prescaler mode register 0 (PRM0) is input to the TIO0/P20 pin, the value of the TM0 is loaded to 16-bit capture/compare register 01 (CR01) and an external interrupt request signal (INTTM01) is set.

When the edge specified by bits 6 and 7 (ES10 and ES11) of PRM0 is input to the TIO51/P66 pin, the value of TM0 is loaded to 16-bit capture/compare register 00 (CR00), and an external interrupt request signal (INTTM00) is set.

For the edges of the TIO0/P20 and TIO51/P66 pins, the rising, falling, or both rising and falling edges can be specified.

Sampling is performed with the count clock selected by prescaler mode register 0 (PRM0), and the capture operation is performed when the valid level of the TIO0/P20 or TIO51/P66 pin is detected two times. Therefore, noise with a short pulse width can be eliminated.

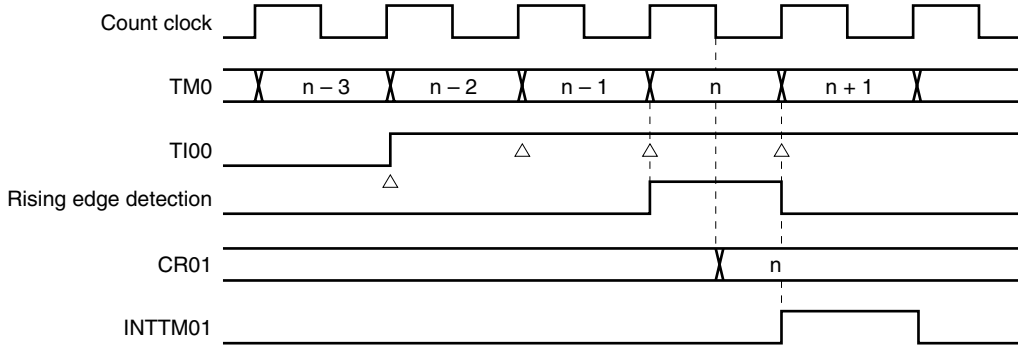
**Figure 7-12. Control Register Settings for Measurement of Two Pulse Widths with Free-Running Counter**



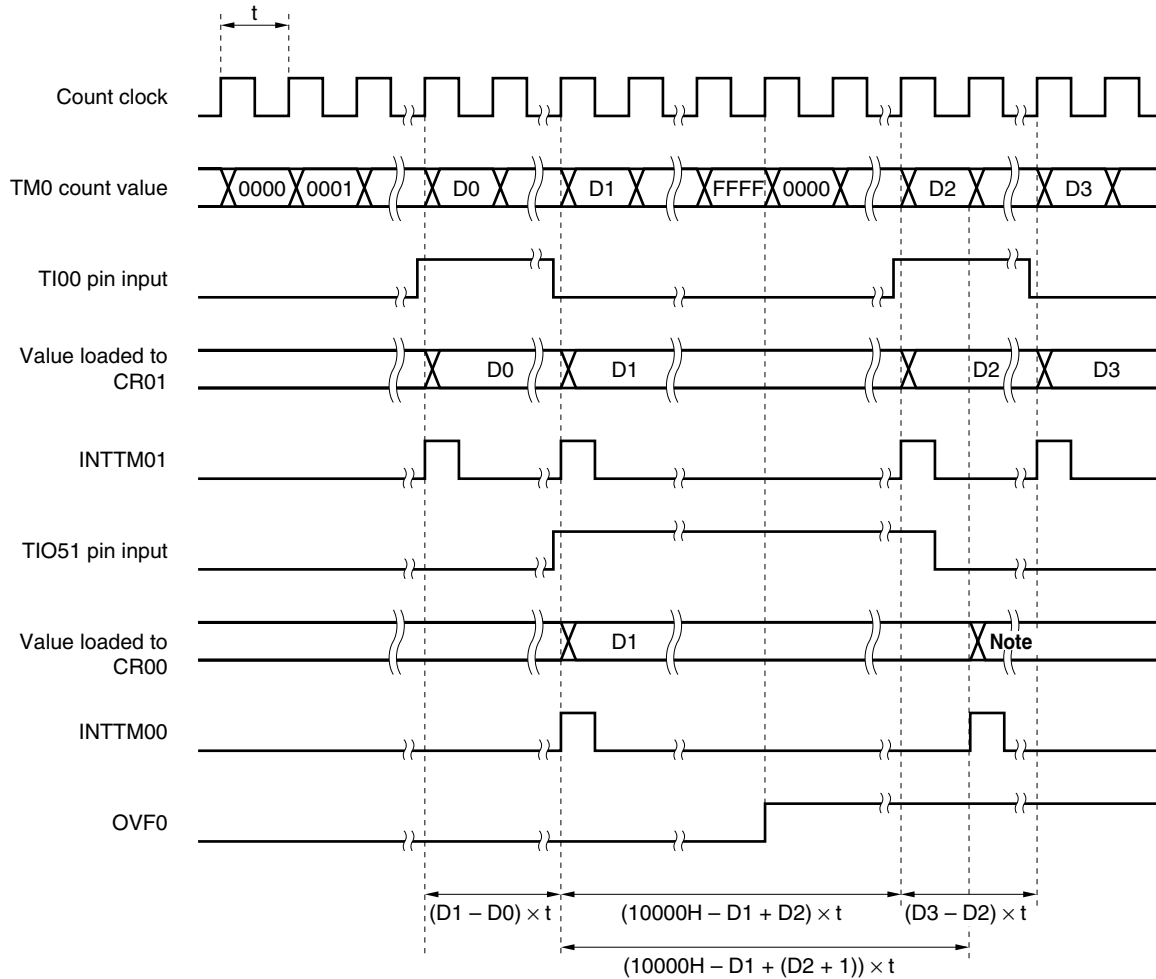
- **Capture operation (free-running mode)**

The following figure illustrates the operation of the capture register when the capture trigger is input.

**Figure 7-13. CR01 Capture Operation with Rising Edge Specified**



**Figure 7-14. Timing of Pulse Width Measurement with Free-Running Counter (with Both Edges Specified)**



**Note**  $D2 + 1$



**Caution** In Figure 7-14, for simplification purposes, consideration of the delay caused by noise elimination has been omitted from the timing of the capture operation based on the inputs to pins TI00 and TI01 and of interrupt request occurrence. For accurate information, refer to Figure 7-13 (which shows the CR01 capture operation with a rising edge specified).

**(3) Pulse width measurement with free-running counter and two capture registers**

When 16-bit timer counter 0 (TM0) is used as a free-running counter (refer to the register settings in **Figure 7-15**), the pulse width of the signal input to the TI00/P20 pin can be measured.

When the edge specified by bits 4 and 5 (ES00 and ES01) of prescaler mode register 0 (PRM0) is input to the TI00/P20 pin, the value of TM0 is loaded to 16-bit capture/compare register 01 (CR01), and an external interrupt request signal (INTTM01) is set.

The value of TM0 is also loaded to 16-bit capture/compare register 00 (CR00) when an edge reverse to the one that triggers capturing to CR01 is input.

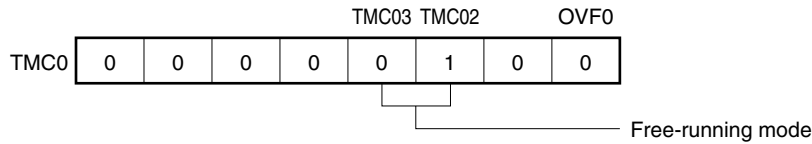
For the edge of the TI00/P20 pin, the rising or falling edge can be specified.

Sampling is performed with the count clock selected by PRM0, and the capture operation is performed when the valid level of the TI00/P20 pin is detected two times. Therefore, noise with a short pulse width can be eliminated.

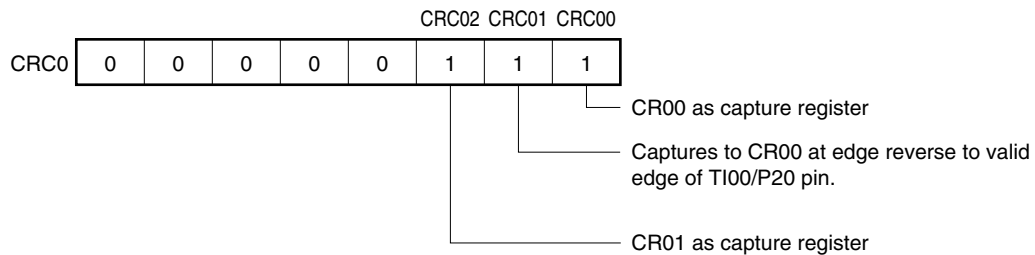
**Caution** If the valid edge of the TI00/P20 pin is specified to be both the rising and falling edges, CR00 cannot perform its capture operation.

**Figure 7-15. Control Register Settings for Pulse Width Measurement with Free-Running Counter and Two Capture Registers**

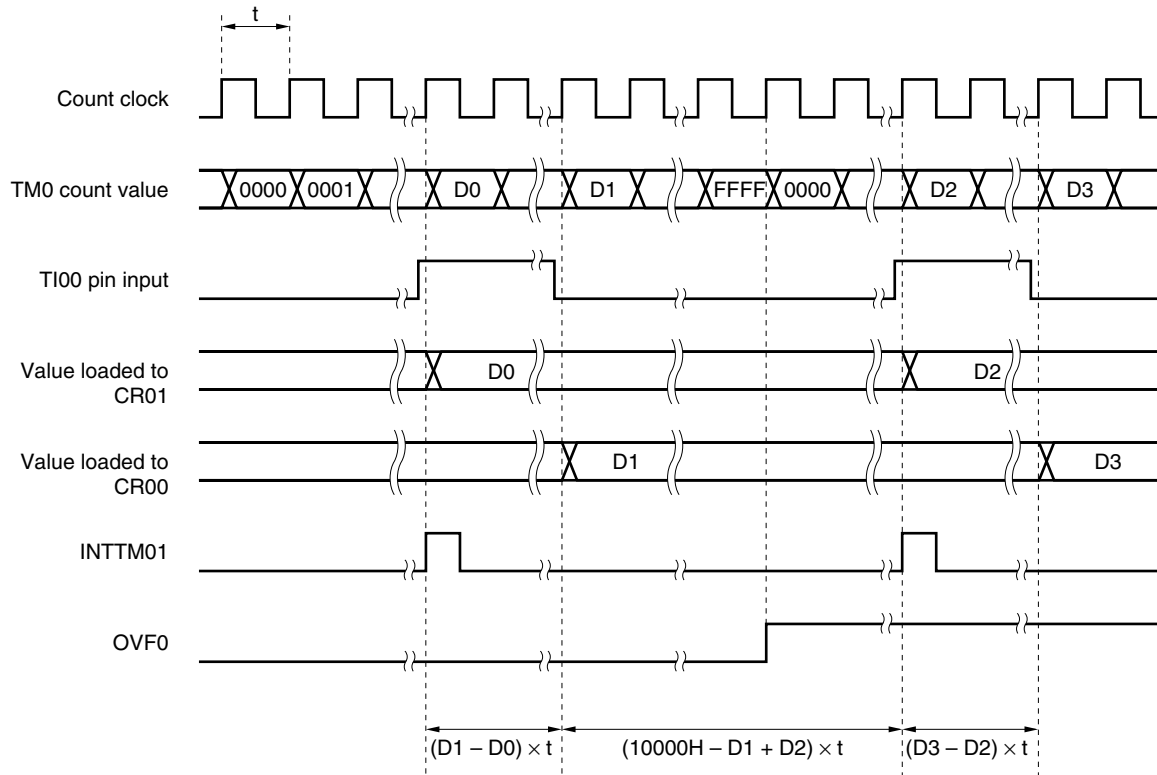
**(a) 16-bit timer mode control register 0 (TMC0)**



**(b) Capture/compare control register 0 (CRC0)**



**Figure 7-16. Timing of Pulse Width Measurement with Free-Running Counter and Two Capture Registers (with Rising Edge Specified)**



**Caution** In Figure 7-16, for simplification purposes, consideration of the delay caused by noise elimination has been omitted from the timing of the capture operation based on the inputs to pin TI00 and of interrupt request occurrence. For accurate information, refer to Figure 7-13 (which shows the CR01 capture operation with a rising edge specified).

**(4) Pulse width measurement by restarting**

When the valid edge of the TI00/P20 pin is detected, the pulse width of the signal input to the TI00/P20 pin can be measured by clearing 16-bit timer counter 0 (TM0) once and then resuming counting after loading the count value of TM0 to 16-bit capture/compare register 01 (CR01). (Refer to the register settings in **Figure 7-17.**)

The edge of the TI00/P20 pin is specified by bits 4 and 5 (ES00 and ES01) of prescaler mode register 0 PRM0. The rising or falling edge can be specified.

Sampling is performed with the count clock selected by PRM0, and the capture operation is performed when the valid level of the TI00/P20 pin is detected two times. Therefore, noise with a short pulse width can be eliminated.

**Caution** If the valid edge of the TI00/P20 pin is specified to be both the rising and falling edges, capture/compare register 00 (CR00) cannot perform its capture operation.

**Figure 7-17. Control Register Settings for Pulse Width Measurement by Restarting**

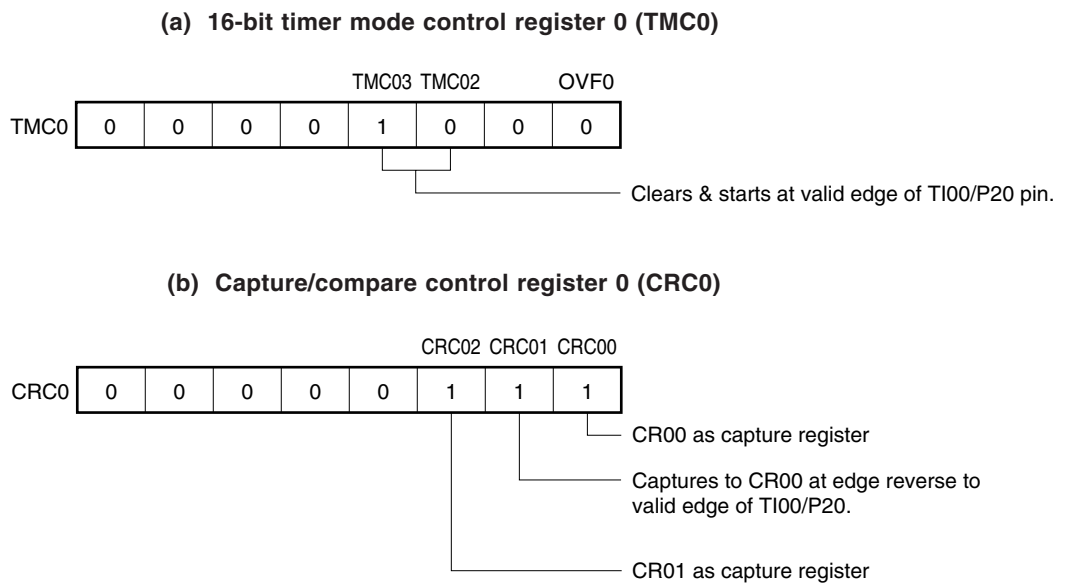
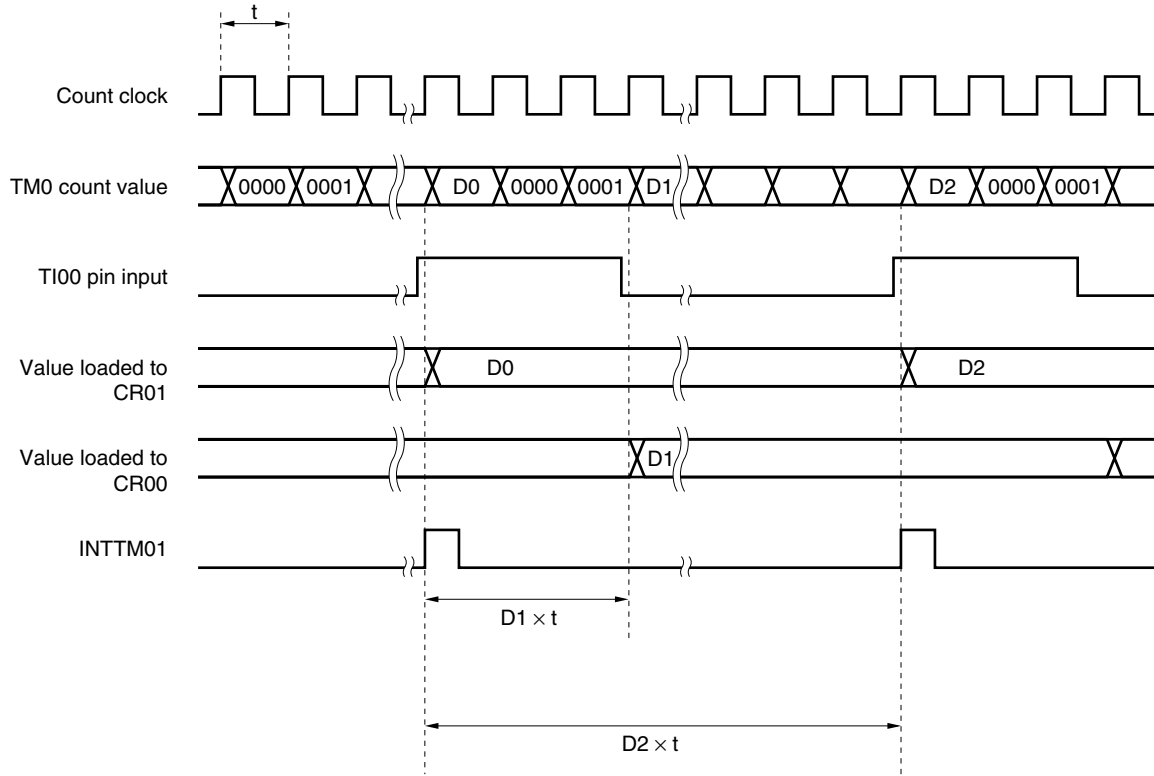


Figure 7-18. Timing of Pulse Width Measurement by Restarting (with Rising Edge Specified)



**Caution** In Figure 7-18, for simplification purposes, consideration of the delay caused by noise elimination has been omitted from the timing of the capture operation based on the inputs to pin TI00 and of interrupt request occurrence. For accurate information, refer to Figure 7-13 (which shows the CR01 capture operation with a rising edge specified).

### 7.4.3 Operation as external event counter

16-bit time/event counter can be used as an external event counter which counts the number of clock pulses input to the TI00/P20 pin from an external source by using 16-bit timer counter 0 (TM0).

Each time the valid edge specified by prescaler mode register 0 (PRM0) has been input to the TI00/P20 pin, TM0 is incremented.

When the count value of TM0 matches the value of 16-bit capture/compare register 00 (CR00), TM0 is cleared to 0, and an interrupt request signal (INTTM00) is generated.

Set CR00 to the value other than 0000H. (A 1-pulse counter can not be operated.)

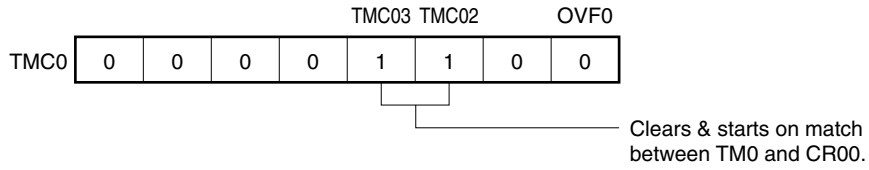
To perform counting with clock pulses input to pin TI00/P20, specify the valid edge for TI00 using bits 0 and 1 (PRM00 and PRM01) of PRM0.

The edge of the TI00/20 pin is specified by bits 4 and 5 (ES00 and ES01) of PRM0. The rising, falling, or both the rising and falling edges can be specified.

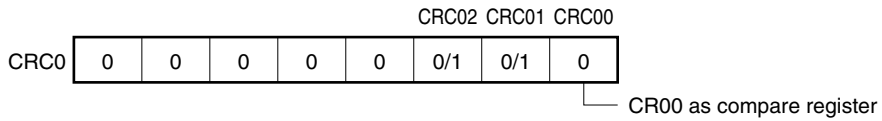
For the capture operation is not performed until the valid level of pin TI00 is detected two times by sampling with the count clock selected by PRM0. Therefore, noise with a small pulse width can be eliminated.

Figure 7-19. Control Register Settings in External Event Counter Mode

(a) 16-bit timer mode control register 0 (TMC0)

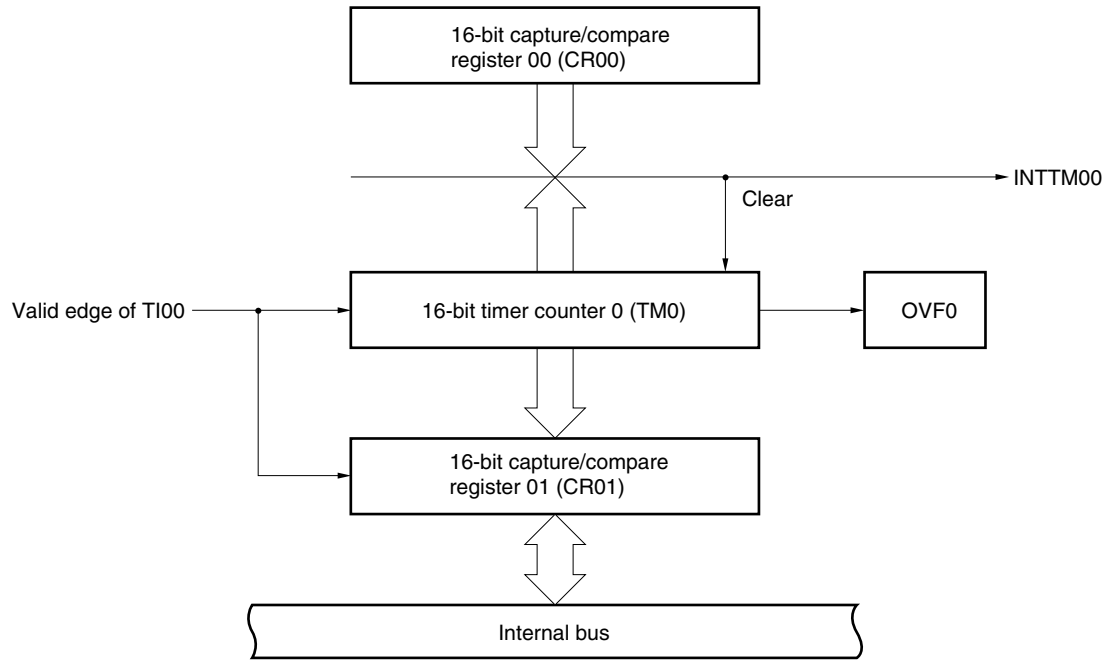


(b) Capture/compare control register 0 (CRC0)

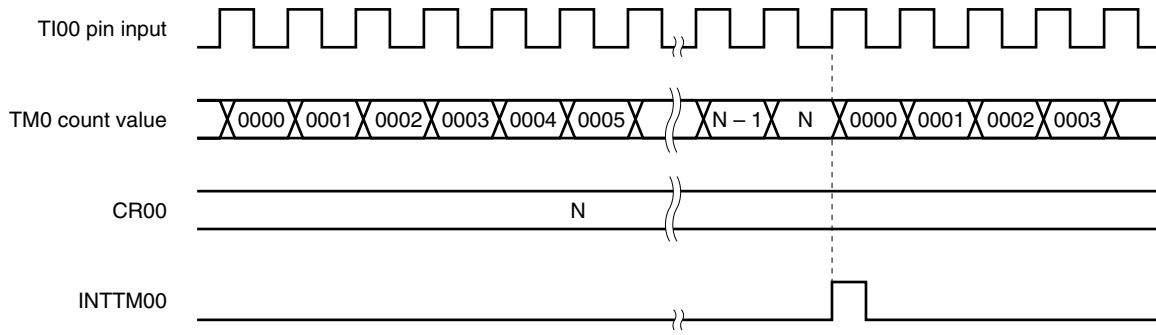


**Remark** 0/1 : When these bits are reset to 0 or set to 1, the other functions can be used along with the external event counter function. For details, refer to **Figures 7-2** and **7-3**.

Figure 7-20. Configuration of External Event Counter



**Figure 7-21. Timing of External Event Counter Operation (with Rising Edge Specified)**


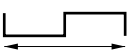




**Caution** Read TM0 when reading the count value of the external event counter.

## 7.5 Generation of Remote Controller Receive Interrupt

If the pulse interval of the signal input next to the TI00 pin is between the minimum and maximum values preset when 16-bit timer/event counter 0 is used, an interrupt request signal is generated as a remote controller signal. This signal can be identified by the pulse interval, high width, and low width, depending on the setting of bits 4 to 7 (RES00, RES01, RES10, and RES11) of the remote controller receive mode register (REMM).

**Table 7-5. Selection of TI00 Pin Valid Edge and Signal Identifier**

RES01 (TM0 Clear Start Timing)	RES00	RES11 (Detection Timing)	RES10	Signal Identified by:
1 (Rising edge)	0	1 (Rising edge)	0	Pulse interval 
0 (Falling edge)	1	0 (Falling edge)	1	Pulse interval 
0 (Falling edge)	1	1 (Rising edge)	0	Low width 
1 (Rising edge)	0	0 (Falling edge)	1	High width 

### 7.5.1 Operating procedure

- (1) Set 16-bit capture compare register 00 (CR00) and 16-bit capture compare register 01 (CR01) in compare mode (by clearing bits 0 and 2 (CRC00 and CRC02) of capture/compare control register 0 (CRC0) to 0).
- (2) Set the minimum value of the criteria to CR00 and the maximum value to CR01.
- (3) Set valid edges 2 and 1 of the TI00 pin by referring to Table 7-5 Selection of TI00 Pin Valid Edge and Signal Identifier (and by using bits 4 to 7 (RES00, RES01, RES10, and RES11) of REMM).  
Set the clear signal of TM0 in clear & start mode to “clear signal at valid TI00 edge with bits 4 and 5 (RES00 and RES01) of REMM” (bit 1 (REMM1) of REMM is 1) by inputting the valid edge to the TI00 pin.
- (4) Set the operation mode of TM0 to clear & start mode (set bits 2 and 3 (TMC02 and TMC03) of 16-bit timer mode control register 0 (TMC0) to “0, 1”) by inputting the valid edge to the TI00 pin. The count operation is started using the count clock (setting the valid edge of TI00 as the count clock is prohibited) specified by bits 0 and 1 (PRM00 and PRM01) of prescaler mode register 0 (PRM0). The timer is cleared when the edge specified by bits 4 and 5 (RES00 and RES01) of REMM is input to the TI00 pin.
- (5) If the value of TM0 matches the value of CR00 (minimum value), the flip-flop (F/F) is set. This F/F is reset when the value of TM0 matches the value of CR01 (maximum value).
- (6) An interrupt request signal (INTREM) is generated if the edge specified by bits 6 and 7 (RES10 and RES11) is input to the TI00 pin while the F/F is set.

**Remark** The valid edge is detected and the clock cycle selected by bit 0 (RSMPC) of REMM is sampled. When the valid edge has been detected four times, the level is reported to the internal circuit (refer to **7.6 Noise Eliminator of Remote Controller Receive Interrupt Generator**).

**Table 7-6. Setting Range of CR00 (Min) and CR01 (Max), and Generation of INTREM**

Setting Range of CR00	Setting Range of CR01	Generation of INTREM (N: CR00 Set Value, M: CR01 Set Value)	
		00001H to FFFE H (7,637 ns to 250 ms)	0002H to FFFF H (11,456 ns to 250 ms)
		Other than above	Not generated

**Remarks 1.** Tpw: Pulse width, high width, or low width (selected by bits 4 to 7 (RES00, RES01, RES10, and RES11) of REMM) after the signal has passed through the 4-point sampling noise eliminator.

**2.** The value of the setting range of CR00 and CR01 in parentheses is (set value + 1) × (count rate) with a count clock of  $f_{xx}/16$ .

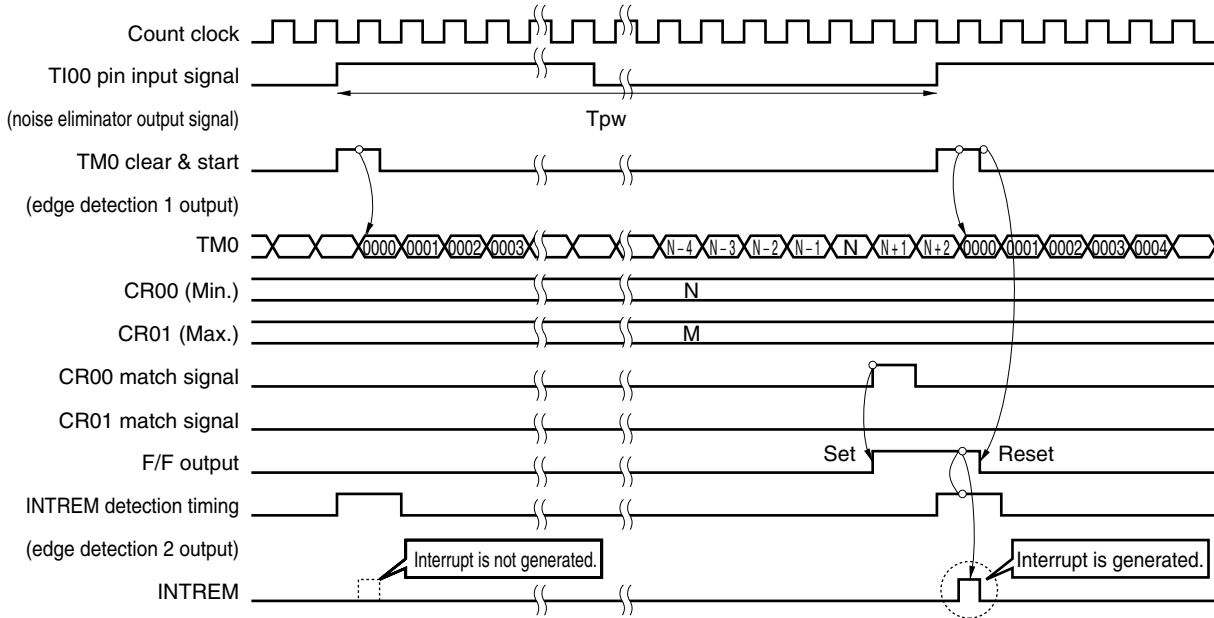
**Cautions 1.** Set CR00 to a value of 0001H to FFFE H, and CR01 to 0002H to FFFF H. Be sure to set a value greater than that of CR00 to CR01; otherwise the operation will not be guaranteed.

**2.** Because Tpw is a signal that has passed through the 4-point sampling noise eliminator, it has an error of the sampling clock ±1 clock or less with respect to the signal input to the TI00 pin (refer to Figure 7-24 Sampling Timing Chart). Set the values of CR00 and CR01 taking this error into consideration.

**Figure 7-22. Operation Timing When Remote Controller Receive Interrupt Is Generated (1/2)**

**(a) When INTREM interrupt request signal is generated  
(count rate of  $(N + 1) \times TM0 \leq Tpw \leq$  count rate of  $M \times TM0$ )**

**Example** When the interrupt signal is identified by the pulse interval  
(when RES11, RES10, RES01, and RES00 are set to 1, 0, 1, and 0.)



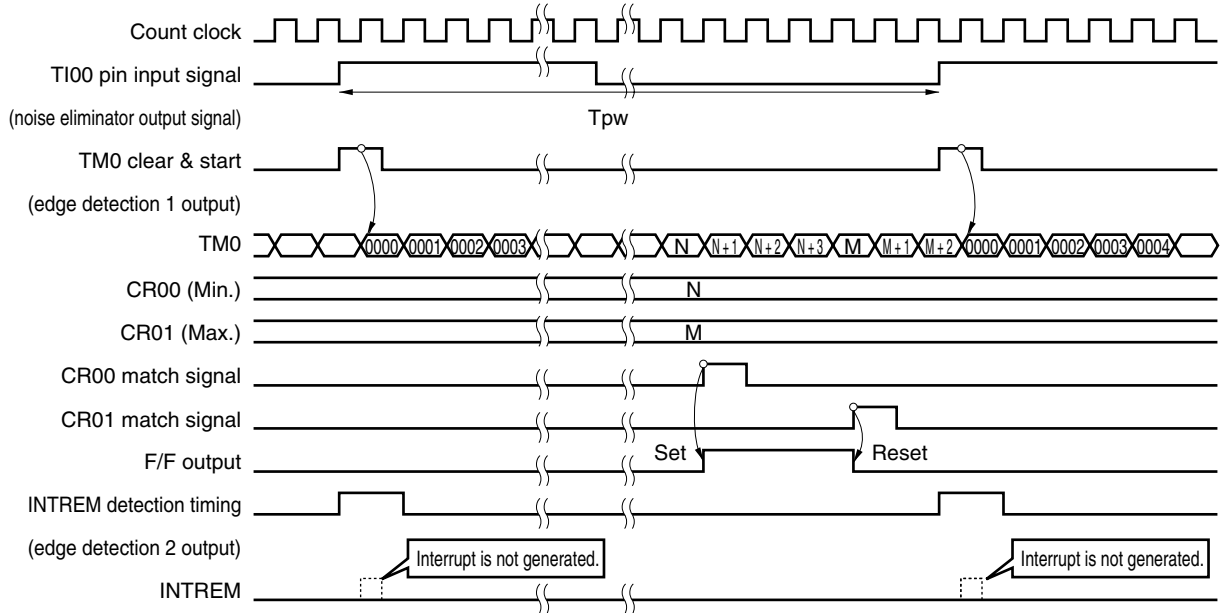
**Caution** If the INTREM detection timing occurs while the output of the F/F (flip-flop) is set (period of “H”), an interrupt request signal (INTREM) is generated.



Figure 7-22. Operation Timing When Remote Controller Receive Interrupt Is Generated (2/2)

(b) When INTREM interrupt request signal is not generated  
(count rate of  $(M + 1) \times TM0 \leq Tpw$ )

**Example** When the interrupt signal is identified by the pulse interval  
(when RES11, RES10, RES01, and RES00 are set to 1, 0, 1, and 0.)



**Caution** If the INTREM detection timing occurs while the output of the F/F (flip-flop) is set (period of “L”), the interrupt request signal (INTREM) is not generated.

**Remark**  $Tpw$ : Pulse width after the signal has passed through the 4-point sampling noise eliminator.

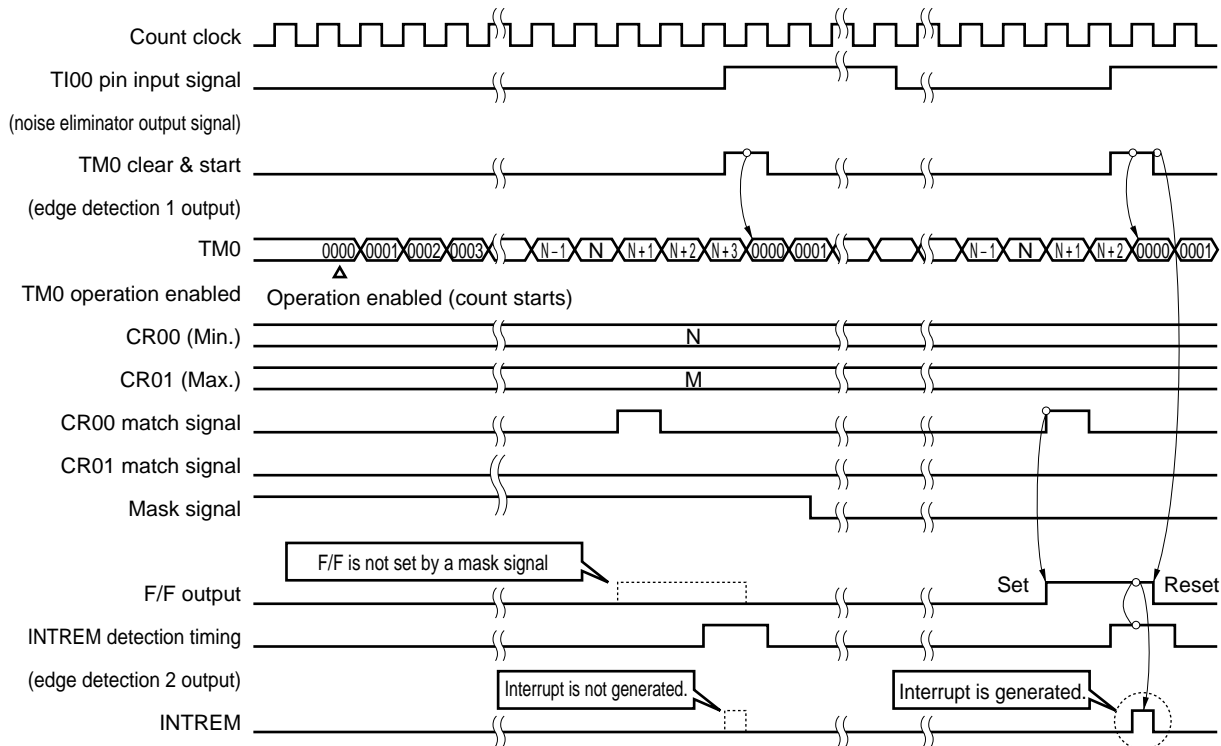
7.5.2 Cautions on generation of remote controller interrupt

(1) When the interrupt signal is identified by the pulse interval

The generation of INTREM is prohibited until the first valid edge of the TM0 clear & start signal is input after the timer has started operation.

INTREM generation is prevented by a mask signal before TM0 is cleared by the first valid edge of the TM0 clear & start signal after the timer has started operation (this is to prevent the erroneous generation of INTREM by a signal other than the remote controller signal). The mask signal is input to the reset line of the F/F and becomes active from when the timer has stopped to when the first valid edge of the TM0 clear & start signal is input after the timer has started operation. While the mask signal is active, the F/F is in the reset state and is not set (refer to the figure below).

**Example** When the interrupt signal is identified by the pulse interval (when RES11, RES10, RES01, and RES00 are set to 1, 0, 1, and 0.)

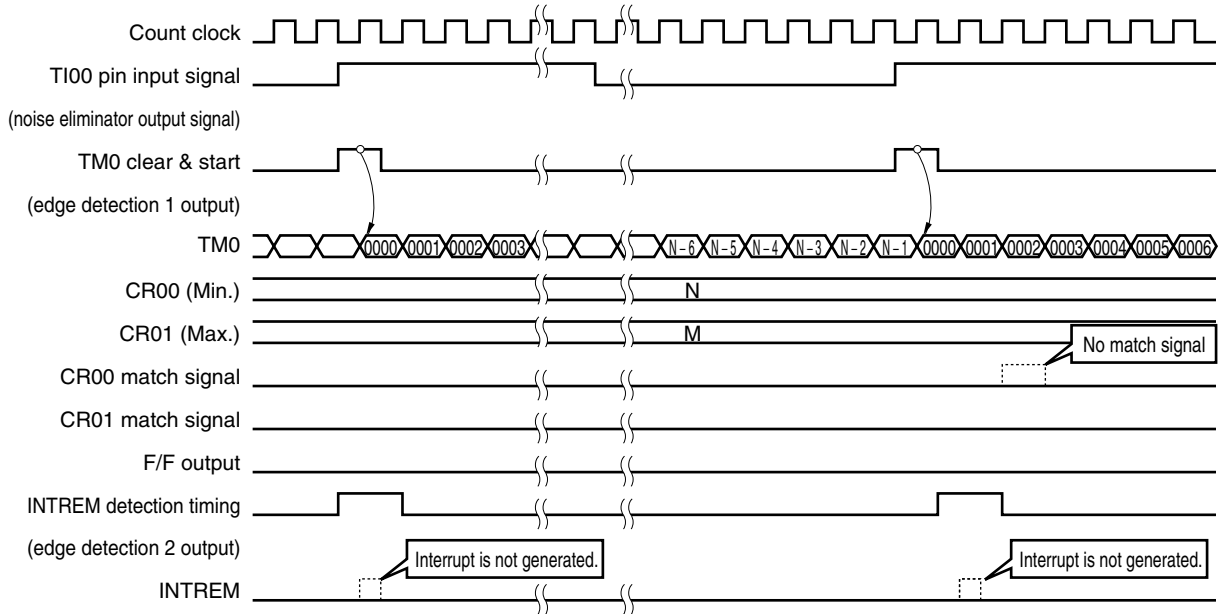


(2) Conflict of TM0 clear & start signal, CR00 match signal, and INTREM detection timing

**Example** When the interrupt signal is identified by the pulse interval (when RES11, RES10, RES01, and RES00 are set to 1, 0, 1, and 0)

<1> When INTREM is not generated

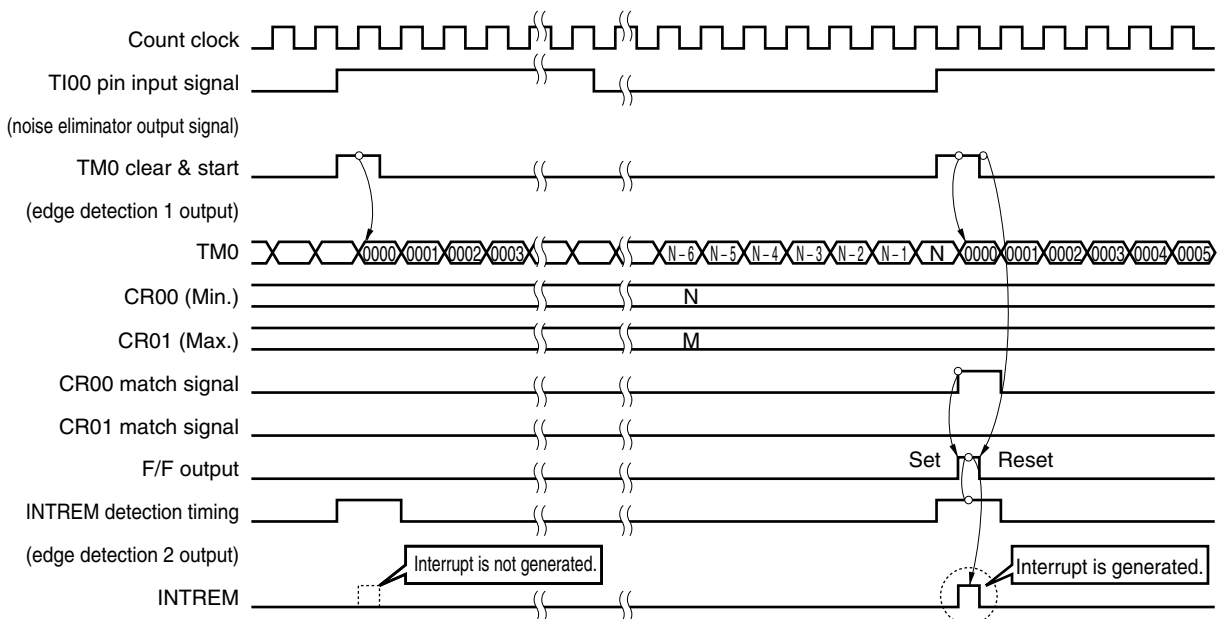
The remote controller receive interrupt signal is not generated if the TM0 clear & start signal is generated before the value of TM0 matches the value of CR00.



<2> When INTREM is generated

If the TM0 clear & start signal, CR00 match signal, and INTREM detection timing conflict, the remote controller receive interrupt is generated.

The output of the F/F is set by the CR00 match signal and INTREM is generated. The F/F output is reset by the TM0 clear & start signal.

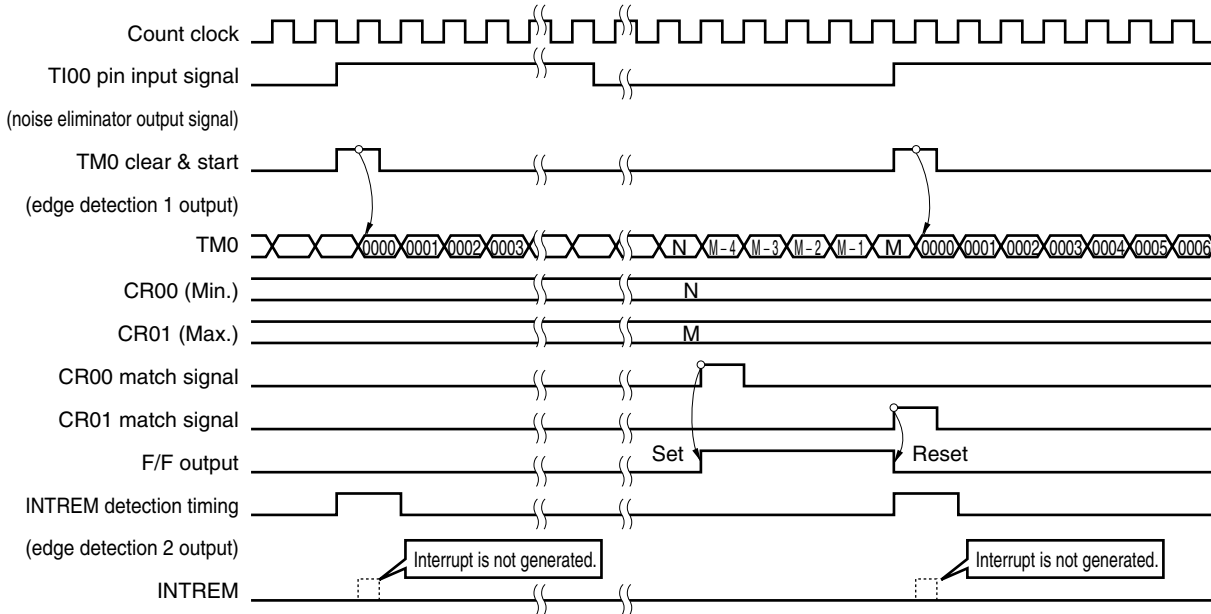


(3) Conflict of TM0 clear & start signal, CR01 match signal, and INTREM detection timing

**Example** When the interrupt signal is identified by the pulse interval (when RES11, RES10, RES01, and RES00 are set to 1, 0, 1, and 0)

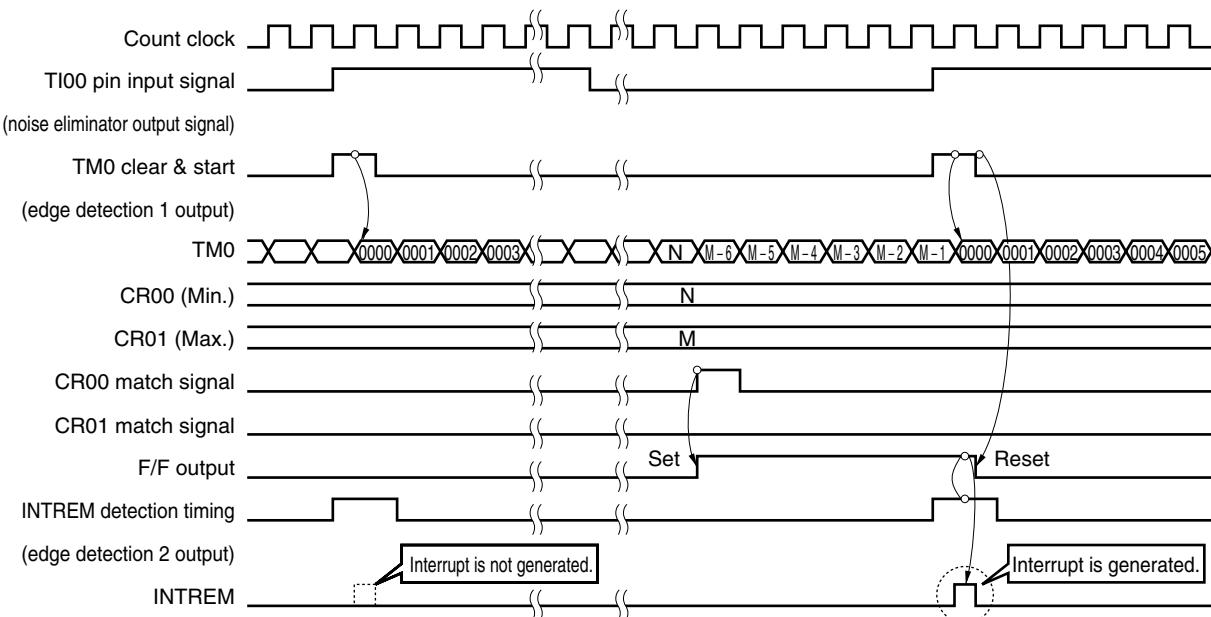
<1> When INTREM is not generated

If the TM0 clear & start signal, CR01 match signal, and INTREM detection timing conflict, the remote controller receive interrupt is not generated.



<2> When INTREM is generated

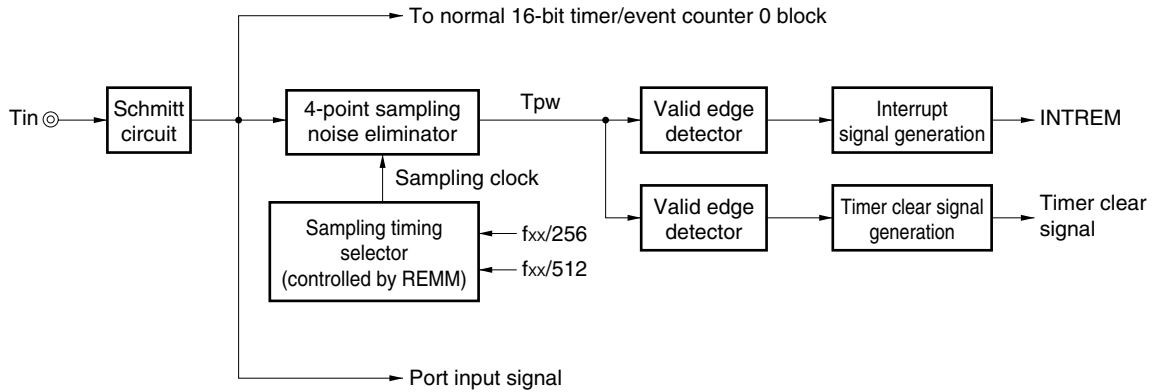
The remote controller receive interrupt signal is generated if the INTREM detection timing occurs (F/F is set) before the value of TM0 matches the value of CR01.



## 7.6 Noise Eliminator of Remote Controller Receive Interrupt Generator

The noise eliminator of the remote controller receive interrupt circuit performs 4-point sampling at the timing specified by bit 0 of the remote controller receive mode register (REMM). It loads the level of a signal if it is the same four times in a row.

Figure 7-23. Block Diagram of INTREM



- **Sampling timing**

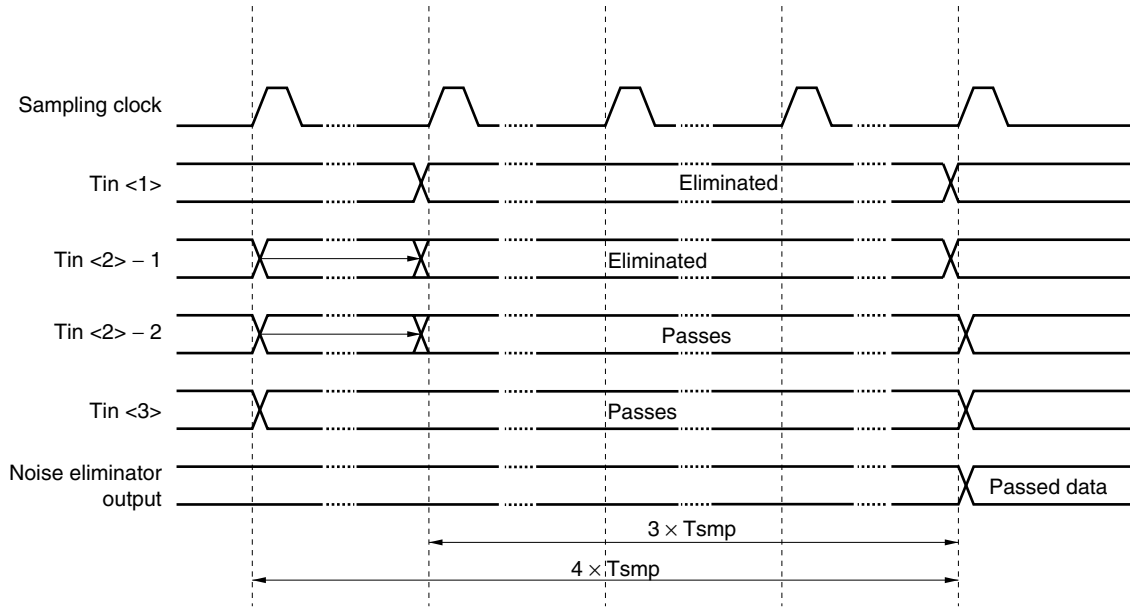
$T_{in}$ : Width of T100 pin input signal,  $T_{smp}$ : Sampling clock rate

- <1>  $T_{in} \leq (3 \times T_{smp})$  ... Eliminated as noise
- <2>  $(3 \times T_{smp}) < T_{in} < (4 \times T_{smp})$  ... May be eliminated as noise or may pass as valid signal (depending on timing)
- <3>  $T_{in} \geq (4 \times T_{smp})$  ... Passes as valid signal

Therefore, a signal with a width of  $(3 \times T_{smp})$  to  $(4 \times T_{smp})$  is eliminated as noise. To accurately pass a signal as a valid signal, the signal width must be  $4 \times T_{smp}$  or more.

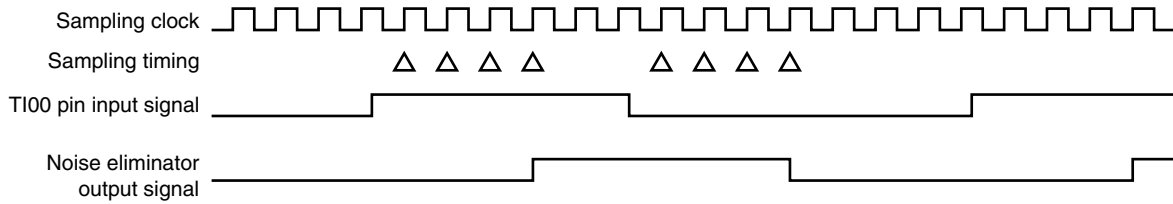
**Caution** Because a digital sampling circuit is used, if a pulse with a narrow width is input successively, it may pass through the noise eliminator.

Figure 7-24. Sampling Timing Chart



**Caution** The pin level ( $T_{in}$ ) has a delay time of  $(3 \times T_{smp})$  to  $(4 \times T_{smp})$  before and after the signal passes the noise eliminator. The delay time at which the pin level ( $T_{in}$ ) passes through the sampling circuit is  $(3 \times T_{smp})$  to  $(4 \times T_{smp})$  with a variation of  $1 T_{smp}$ .

Figure 7-25. Noise Eliminator Output Signal

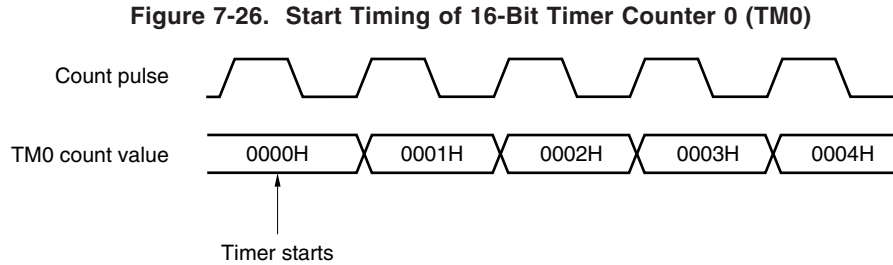


**Caution** A time delay of sampling clock  $\pm 1$  clock occurs before and after the signal passes through the noise eliminator.

7.7 Cautions

(1) Error on starting timer

An error of up to 1 clock occurs before the match signal is generated after the timer has been started. This is because 16-bit timer counter 0 (TM0) is started asynchronously in respect to the count pulse.

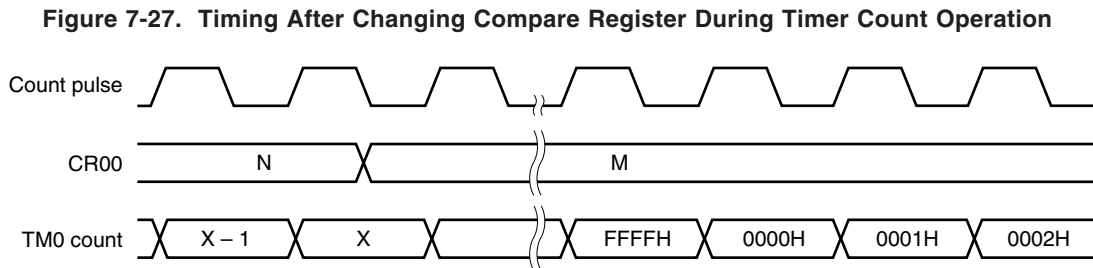


(2) Setting 16-bit capture/compare register

Set 16-bit capture/compare registers 00 and 01 (CR00 and CR01) to the value other than 0000H. When using this register as an event counter, a count for one-pulse can not be operated.

(3) Setting compare register during timer count operation

If the value to which the current value of 16-bit capture/compare register 00 (CR00) has been changed is less than the value of 16-bit timer counter 0 (TM0), TM0 continues counting, overflows, and starts counting again from 0. If the new value of CR00 (M) is less than the old value (N), the timer must be restarted after the value of CR00 has been changed.

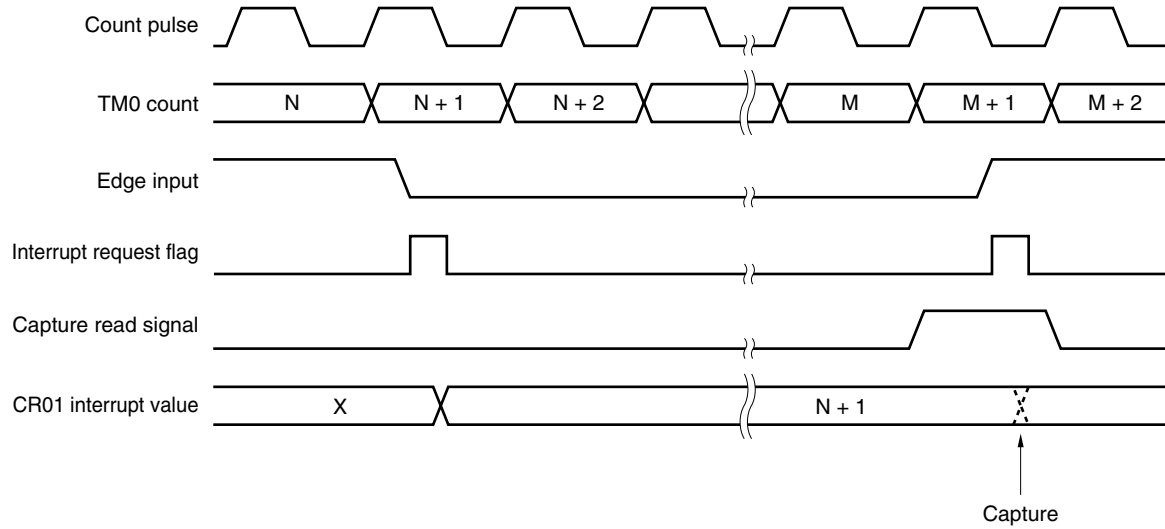


**Remark**  $N > X > M$

**(4) Data hold timing of capture register**

If the valid edge is input to the TI00/P20 pin while the 16-bit capture/compare register 01 (CR01) is read, CR01 performs the capture operation, but this capture value is not guaranteed. However, the interrupt request flag (INTTM01) is set as a result of detection of the valid edge

**Figure 7-28. Data Hold Timing of Capture Register**

**(5) Setting valid edge**

Before setting the valid edge of the TI00/P20 pin, stop the timer operation by resetting bits 2 and 3 (TMC02 and TMC03) of the 16-bit timer mode control register (TMC0) to a combination of 0 and 0. Set the valid edge by using bits 4 and 5 (ES00 and ES01) of prescaler mode register 0.



**(6) Operation of OVF0 flag**

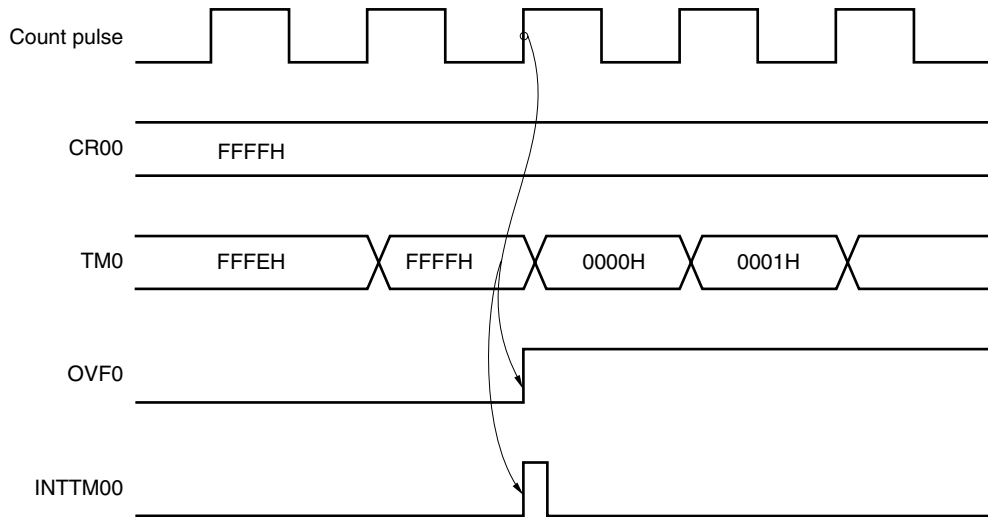
The OVF0 flag is set to 1 in the following case:

Select mode in which 16-bit timer/event counter is cleared and started on match between TM0 and CR00

↓  
Set CR00 to FFFFH.  
↓

When TM0 counts up from FFFFH to 0000H

**Figure 7-29. Operation Timing of OVF0 Flag**



**(7) Contention operation**

<1> Contention between the read period of the 16-bit capture/compare registers (CR00 and CR01) and the capture trigger input (CR00 and CR01 are used as capture registers.)

The capture trigger input is preceded. The read data of CR00 and CR01 is undefined.

<2> Match timing contention between the write period of the 16-bit capture/compare registers (CR00 and CR01) and 16-bit timer counter 0 (TM0). (CR00 and CR01 are used as compare registers.)

A match discrimination is not normally performed. Do not perform the write operation of CR00 and CR01 around the match timing.

**(8) Interrupt request signals (INTTM00 and INTTM01)**

Even when 16-bit timer/event counter 0 is used as a remote controller receive interrupt generator, interrupt requests (INTTM00 and INTTM01) are generated. To suppress these interrupt requests, disable them (by clearing bit 6 (TMMK00) of interrupt control register 0 (TMIC00) and bit 6 (TMMK01) of interrupt control register 1 (TMIC01)).

**(9) Valid edges of TI00 and TI01 pins**

If the TI00/TIO51 pin is high immediately after system reset, the pin is detected as having a rising edge immediately after TM0 operation is first enabled. This must be taken into consideration especially when the pin is pulled up.

**(10) Edge detection by noise eliminator**

If the TI00 pin is high immediately after system reset when the 4-point sampling noise eliminator of the remote controller receive interrupt generator detects an edge, and if TM0 is enabled before the high-level signal input to the TI00 pin passes the 4-point sampling noise eliminator after the system reset signal has been cleared, the signal is detected as having a rising edge immediately after TM0 operation is first enabled. This must be taken into consideration especially when the signal is pulled up.

## CHAPTER 8 8-BIT PWM TIMERS

### 8.1 Functions of 8-Bit PWM Timers

The 8-bit PWM timers have the following two operation modes.

- Mode in which only an 8-bit timer counter (TM5n: n = 0 or 1) is used (single mode)
- Mode in which the two 8-bit PWM timers are cascaded (16-bit resolution: cascade mode)

These two modes are explained next.

#### (1) Mode in which only a TM5n (n = 0 or 1) is used (single mode)

In this mode, the 8-bit PWM timer operates as an 8-bit timer/event counter. In this mode, the following functions can be used.

- Interval timer
- External event counter
- Square wave output
- PWM output

#### (2) Mode in which two timers are cascaded (16-bit resolution: cascade mode)

When the two PWM timers are cascaded, they operate as a 16-bit timer/event counter. In this mode, the following functions can be used.

- Interval timer with 16-bit resolution
- External event counter with 16-bit resolution
- Square output with 16-bit resolution

## 8.2 Configuration of 8-Bit PWM Timers

The 8-bit PWM timers include the following hardware.

**Table 8-1. Configuration of 8-Bit PWM Timers**

Item	Configuration
Timer counter	8-bit timer counter 5n (TM5n)
Register	8-bit compare register 5n (CR5n)
Timer output/ external clock input	TIO5n
Control registers	Timer clock select register 5n (TCL5n) 8-bit timer mode control register 5n (TMC5n)

n = 0, 1

**Figure 8-1. Block Diagram of 8-Bit PWM Timer 50**

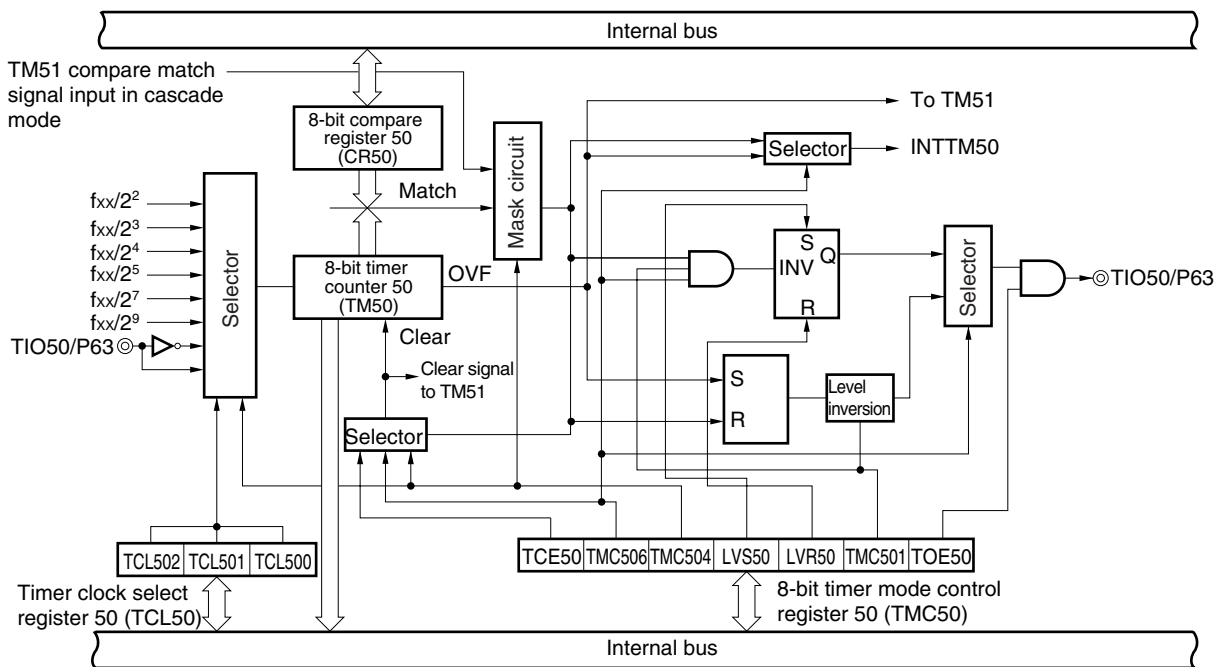
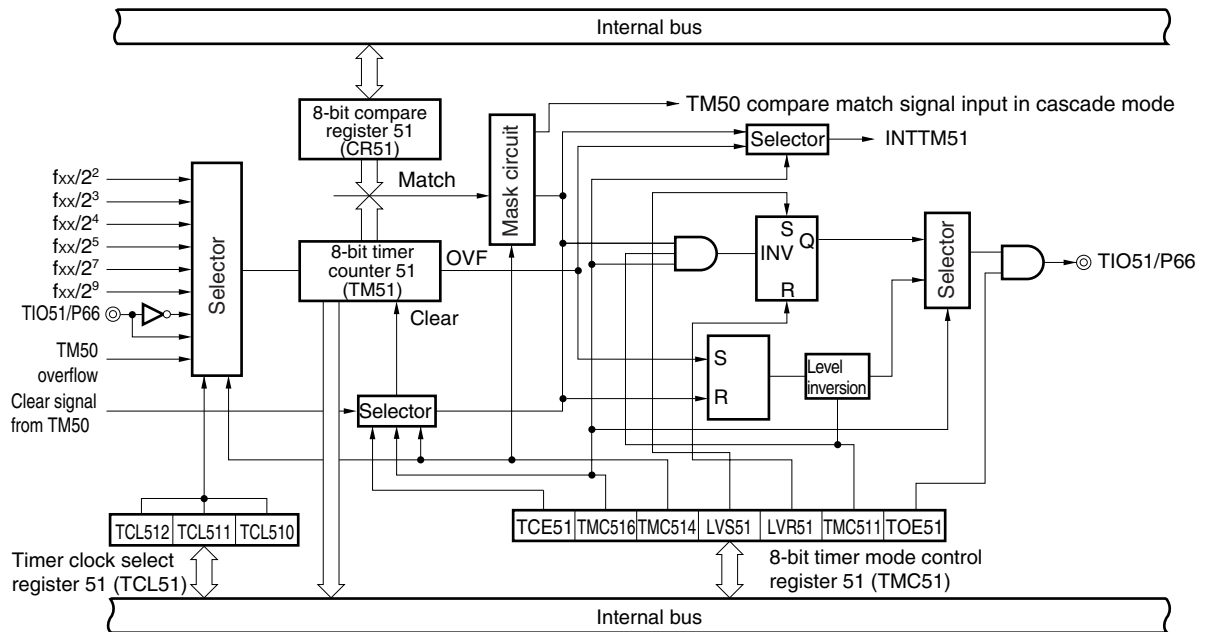


Figure 8-2. Block Diagram of 8-Bit PWM Timer 51



**(1) 8-bit timer counter 5n (TM5n: n = 0 or 1)**

TM5n is an 8-bit read-only register that counts the count pulse.

The value of this counter is incremented in synchronization with the rising edge of the count clock. When the count value is read during operation, input of the count clock is temporarily stopped, and the count value at that point is read. The count value is cleared to 00H in the following cases.

- <1>  $\overline{\text{RESET}}$  input
- <2> Clearing TCE5n
- <3> Match between TM5n and CR5n in clear & start mode

**Caution** In the cascade mode, TCE50 of TMC50 is 00H even if it is cleared.

**Remark** n = 0 or 1

**(2) 8-bit compare register 5n (CR5n: n = 0 or 1)**

The value set in this register is constantly compared with the count value of 8-bit timer counter 5n (TM5n). When the two values match, an interrupt request (INTTM5n) is generated (in the modes other than PWM mode).

The value of CR5n can be set in a range of 00H to FFH, and can be rewritten during count operation.

**Caution** When setting data to this register in the cascade mode, be sure to stop the timer operation. The timer is stopped by clearing both bit 7 (TCE50) of 8-bit timer mode control register 50 (TMC50) and bit 7 (TCE51) of 8-bit timer mode control register 51 (TMC51).

**Remark** n = 0 or 1

### 8.3 8-Bit PWM Timer Control Registers

The following two types of registers control the 8-bit PWM timers.

- Timer clock select register 5n (TCL5n: n = 0 or 1)
- 8-bit timer mode control register 5n (TMC5n: n = 0 or 1)

**(1) Timer clock select register 5n (TCL5n: n = 0 or 1)**

This register sets the count clock and valid edge of the TIO5n input of 8-bit timer counter 5n (TM5n: n = 0 or 1).

TCL5n is set using a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$  input sets TCL5n to 00H.

**Figure 8-3. Format of Timer Clock Select Register 5n (TCL5n)**

Symbol	7	6	5	4	3	2	1	0	Address	After reset	R/W
TCL5n	0	0	0	0	0	TCL5n2	TCL5n1	TCL5n0	FF56H (TCL50), FF57H (TCL51)	00H	R/W

TCL5n2	TCL5n1	TCL5n0	Count clock selection
0	0	0	Falling edge of TIO5n
0	0	1	Rising edge of TIO5n
0	1	0	$f_{xx}/2^2$ (3.125 MHz)
0	1	1	$f_{xx}/2^3$ (1.56 MHz)
1	0	0	$f_{xx}/2^4$ (781 kHz)
1	0	1	$f_{xx}/2^5$ (391 kHz)
1	1	0	$f_{xx}/2^7$ (98 kHz)
1	1	1	$f_{xx}/2^9$ (24 kHz)

- Cautions**
1. When rewriting the data of TCL5n, keep the timer stopped.
  2. Be sure to set bits 3 to 7 to 0.

- Remarks**
1. In cascade mode, only the setting of TCL502 to TCL500 is valid.
  2. n = 0 or 1
  3.  $f_{xx}$ : Main system clock frequency
  4. The values in parentheses are valid when  $f_{xx}$  is 12.5 MHz.

**(2) 8-bit timer mode control register 5n (TMC5n: n = 0 or 1)**

TMC5n sets has the following six functions.

- <1> Controls count operation of 8-bit timer counter 5n (TM5n: n = 0 or 1)
- <2> Selects operation mode of 8-bit timer counter 5n (TM5n: n = 0 or 1)
- <3> Selects single or cascade mode (TMC51 only)
- <4> Sets status of timer output
- <5> Controls timer or selects active level in PWM (free-running) mode
- <6> Controls timer output

TMC5n is set using a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$  input sets TMC5n to 04H.



**Figure 8-4. Format of 8-Bit Timer Control Register 5n (TMC5n)**

Symbol	<7>	6	5	4	<3>	<2>	1	0	Address	After reset	R/W
TMC5n	TCE5n	TMC5n6	0	TMC5n4	LVS5n	LVR5n	TMC5n1	TOE5n	FF54H (TMC50), FF55H (TMC51)	04H	R/W

TCE5n	TM5n count operation control
0	Clears counter to 0 and stops counting (prescaler disabled)
1	Starts counting

TMC5n6	TM5n operation mode selection
0	Clear & start on match between TM5n and CR5n
1	PWM (free-running) mode

TMC5n4 Notes 1, 2	Single/cascade mode selection
0	Single mode (used as 8-bit timer)
1	Cascade mode (connected to TM50 and used as 16-bit timer)

LVS5n	LVR5n	Timer output status setting
0	0	Does not affect
0	1	Resets timer output to 0
1	0	Sets timer output to 1
1	1	Setting prohibited

TMC5n1	Other than PWM mode (TMC5n6 = 0)	PWM mode (TMC5n6 = 1)
	Timer control	Active level selection
0	Disables inversion	High active
1	Enables inversion	Low active

TOE5n	Timer output control
0	Disables output (port mode)
1	Enables output

- Notes**
1. Be sure to set TMC504 to 0.
  2. Set TMC514 according to its format.

**Caution** When selecting the operation mode of TM5n according to TMC5n6 and selecting the concatenation mode (single mode or cascade mode) according to TMC514, keep the timers stopped.

- Remarks**
1. In the PWM mode, the PWM output is at the inactive level if TCE5n = 0.
  2. When LVS5n and LVR5n are read after data has been set, they are 0.
  3. n = 0 or 1

## 8.4 Operations of 8-Bit PWM Timers

### 8.4.1 Operation as interval timer (8-bit operation)

An 8-bit PWM timer operates as an interval timer that generates an interrupt request at intervals specified by the count value preset to 8-bit compare register 5n (CR5n).

When the count value of 8-bit timer counter 5n (TM5n) matches the set value of CR5n, the value of TM5n is cleared to 0 and TM5n continues counting. At the same time, an interrupt request signal (INTTM5n) is generated.

The count clock of TM5n can be selected by using the bits 0 to 2 (TCL5n0 to TCL5n2) of timer clock select register 5n (TCL5n).

**Remark** n = 0 or 1

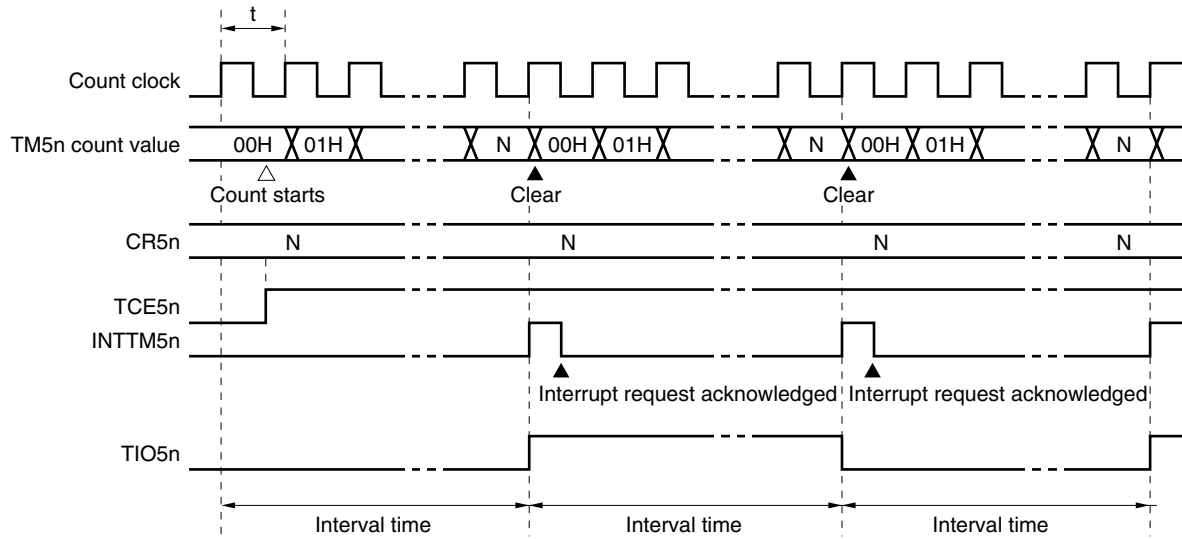
#### [Setting]

- (1) Set the registers.
  - TCL5n: Selects count clock.
  - CR5n: Compare value
  - TMC5n: Selects clear & start mode in which TM5n is cleared and started when its value matches CR5n.  
(TMC5n = 0000×××0B × = don't care)
- (2) The count operation is started when TCE5n = 1.
- (3) When the values of TM5n and CR5n match, INTTM5n is generated (TM5n is cleared to 00H).
- (4) After that, INTTM5n is generated at fixed intervals. To stop the count operation, clear TCE5n = 0.

**Remark** n = 0 or 1

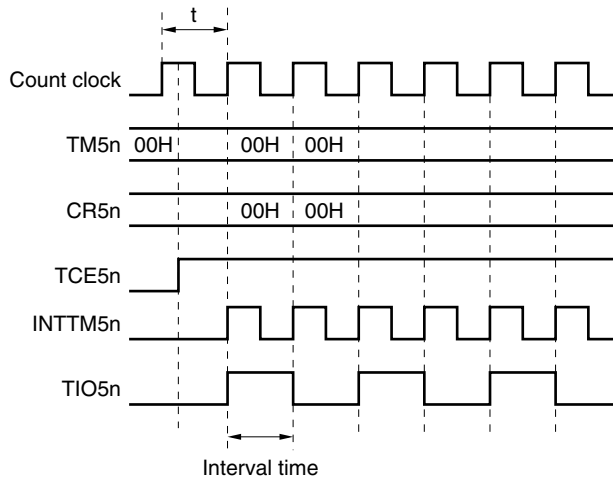
Figure 8-5. Timing of Interval Timer Operation (1/3)

(a) Basic operation



- Remarks**
1. Interval time =  $(n + 1) \times t$ ; N = 00H to FFH
  2. n = 0 or 1

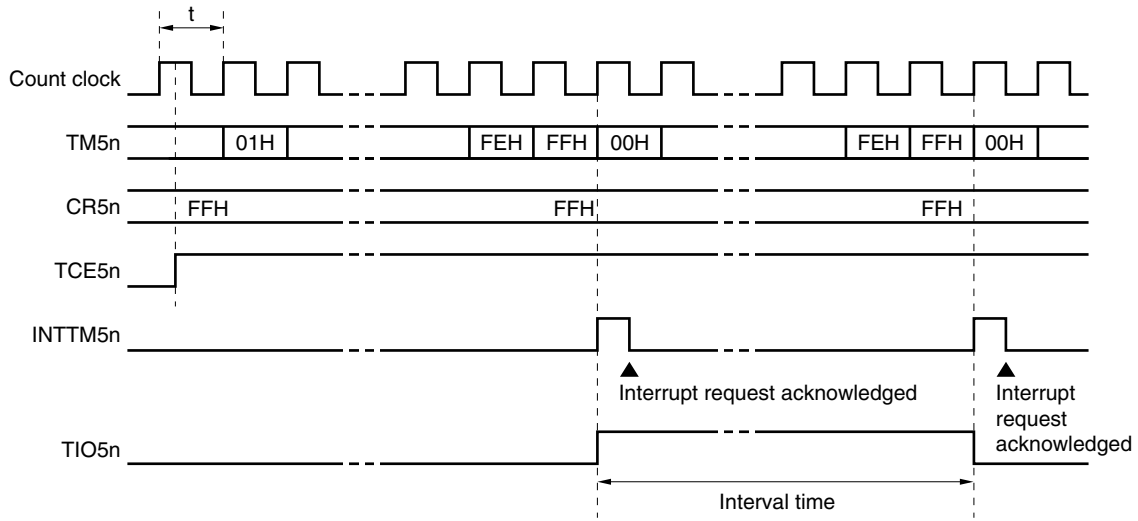
(b) When CR5n = 00H



n = 0 or 1

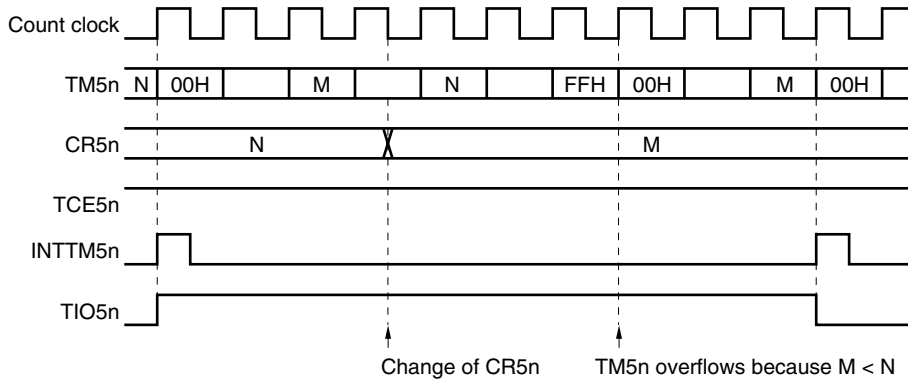
Figure 8-5. Timing of Interval Timer Operation (2/3)

(c) When CR5n = FFH



n = 0 or 1

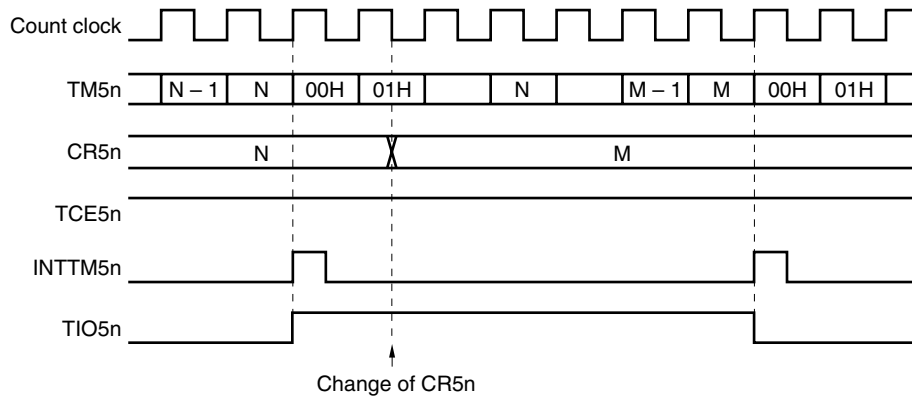
(d) Operation when CR5n is changed (M < N)



n = 0 or 1

Figure 8-5. Timing of Interval Timer Operation (3/3)

(e) Operation when CR5n is changed ( $M > N$ )



n = 0 or 1

8.4.2 Operation as external event counter

The external event counter counts the number of count clock pulses input to TIO5n from an external source.

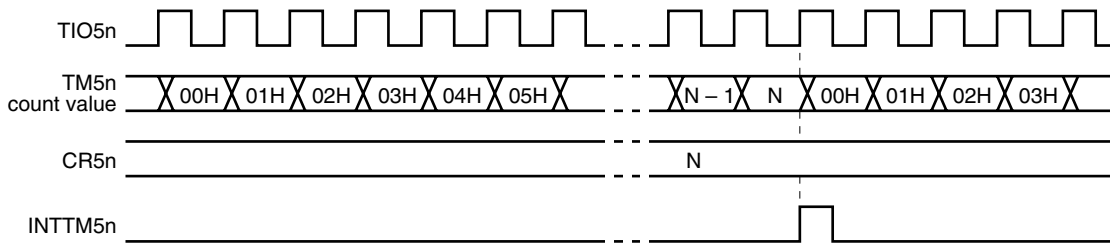
Each time the valid edge specified by timer clock select register 5n (TCL5n) has been input to TIO5n, the value of TM5n is incremented. The edge can be selected from rising or falling.

If the measured value of TM5n matches the value of 8-bit compare register 5n (CR5n), TM5n is cleared to 0 and an interrupt request signal (INTTM5n) is generated.

After that, INTTM5n is generated each time the value of TM5n coincides with the value of CR5n.

**Remark** n = 0 or 1

Figure 8-6. Timing of External Event Counter Operation (with Rising Edge Specified)



n = 0 or 1

### 8.4.3 Square-wave (8-bit resolution) output operation

A square wave of any frequency can be output at intervals specified by the preset value to 8-bit compare register 5n (CR5n).

If the bit 0 (TOE5n) of 8-bit timer mode control register 5n (TMC5n) is set to 1, the output status of TIO5n is inverted at intervals specified by the count value preset to CR5n. In this way, a square wave of any frequency (duty = 50%) can be output.

**Remark** n = 0 or 1

#### [Setting]

(1) Set each register.

- Write "0" to the port latch and port mode register for a port that also functions as a timer output pin.
- TCL5n: Selects count clock.
- CR5n: Compare value
- TMC5n: Clear & start mode in which TM5n is cleared and started when its value matches that of CR5n.

LVS5n	LVR5n	Status Setting of Timer Output
1	0	High-level output
0	1	Low-level output

Inversion of the timer output is enabled.

Timer output enable → TOE5n = 1

- (2) The count operation is started when TCE5n = 1.
- (3) The timer output is inverted when the values of TM5n and CR5n match. Moreover, INTTM5n is generated, and TM5n is cleared to 00H.
- (4) After that, the timer output is inverted at fixed intervals, and TIO5n outputs a square wave.

**Remark** n = 0 or 1

#### 8.4.4 8-bit PWM output operation

The PWM timer performs 8-bit PWM output operation when bit 6 (TMC5n6) of 8-bit timer mode control register 5n (TMC5n) is set to 1, and outputs a pulse with a duty factor determined by the value set to 8-bit compare register 5n (CR5n) from the TIO5n pin.

Set the width of the active level of the PWM pulse to CR5n. The active level can be selected by bit 1 (TMC5n1) of TMC5n.

The count clock can be selected by bits 0 to 2 (TCL5n0 to TCL5n2) of timer clock select register 5n (TCL5n). PWM output can be enabled or disabled by bit 0 (TOE5n) of TMC5n.

**Caution** CR5n can be rewritten only once in one cycle in the PWM mode.

**Remark** n = 0 or 1

##### (1) Basic PWM output operation

###### [Setting]

- (1) Write 0 to the port latch and port mode register for a port that also functions as a timer output pin.
- (2) Set the active level width by using 8-bit compare register 5n (CR5n).
- (3) Select the count clock by using timer clock select register 5n (TCL5n).
- (4) Select the active level by using bit 1 (TMC5n1) of TMC5n.
- (5) Set bit 0 (TOEn) of TMC5n to 1 to enable timer output.
- (6) The timer starts counting when bit 7 (TCE5n) of TMC5n is set to 1.  
To stop the counting, set 0 to TCE5n.

**Remark** n = 0 or 1

###### [PWM output operation]

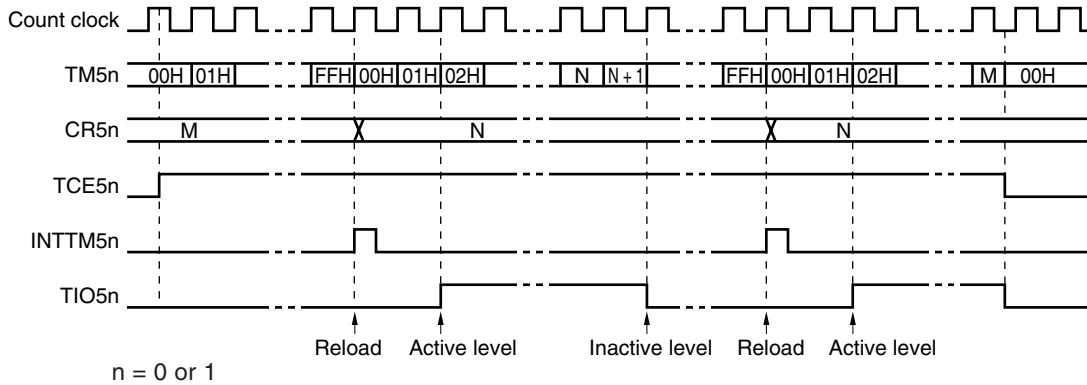
- (1) When the timer starts counting, an inactive level is output from TIO5n as PWM output, until the timer overflows.
- (2) When the overflow occurs, the active level is output. The active level is continuously output until the CR5n and the count value of 8-bit timer counter 5n (TM5n) match.
- (3) The inactive level is output after CR5n and the count value have matched, until an overflow occurs again.
- (4) After that, (2) and (3) are repeated, until the counting operation is stopped.
- (5) PWM output is deasserted inactive when the counting operation is stopped by clearing TCE5n to 0.

**Remark** n = 0 or 1

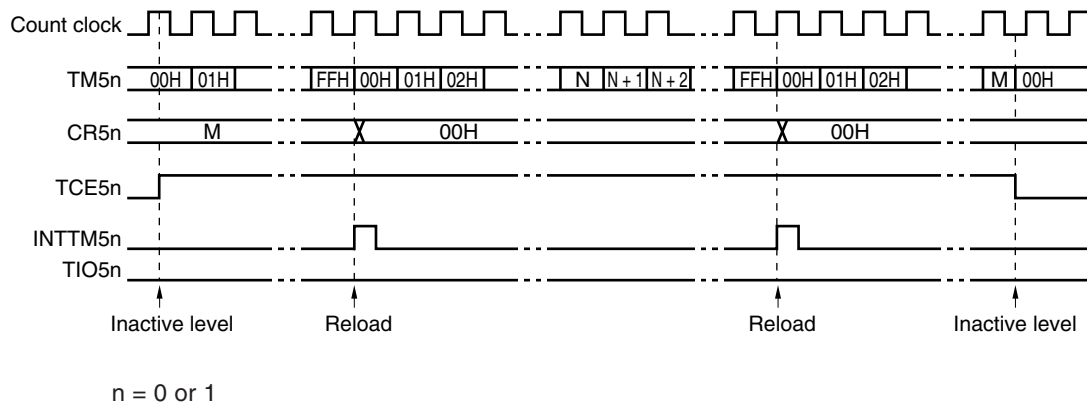
(a) Basic PWM output operation

Figure 8-7. PWM Output Operation Timing

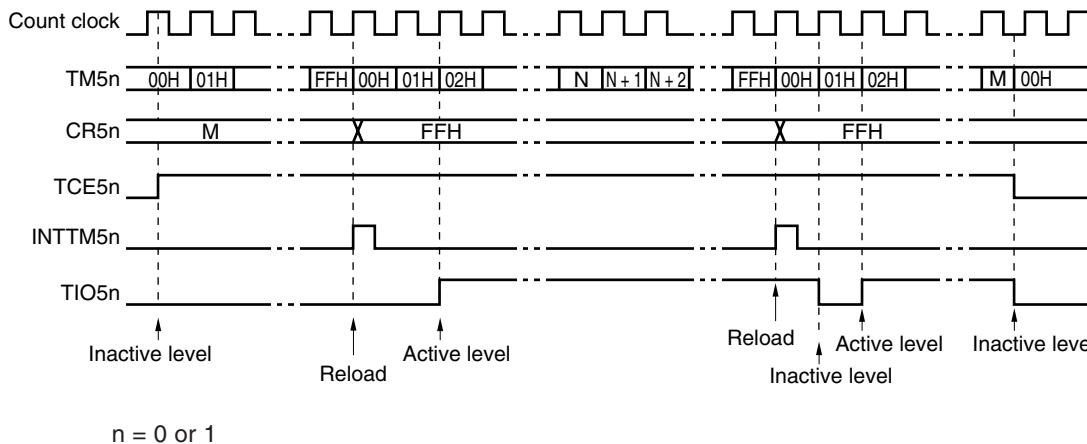
(i) Basic operation (when active level = H)



(ii) When CR5n = 0



(iii) When CR5n = FFH

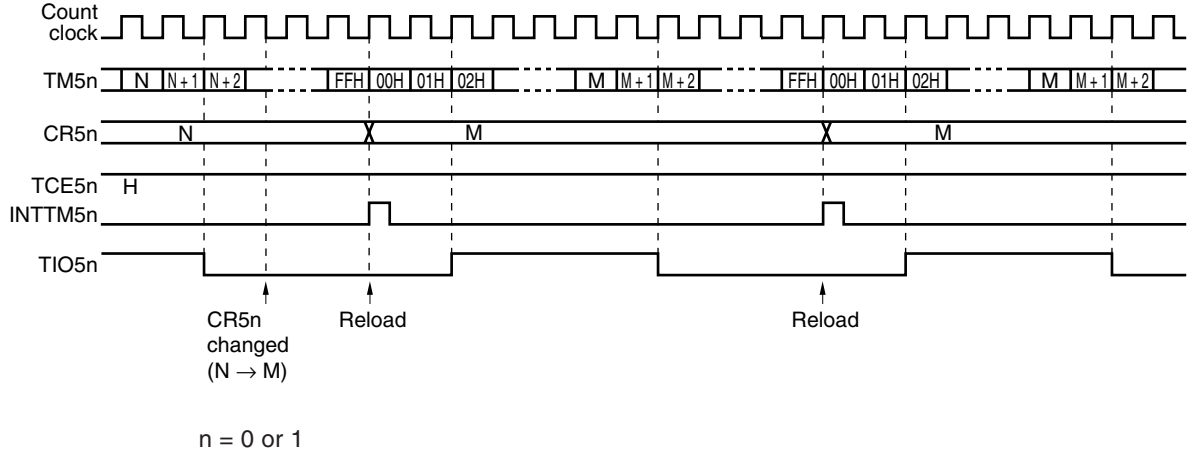




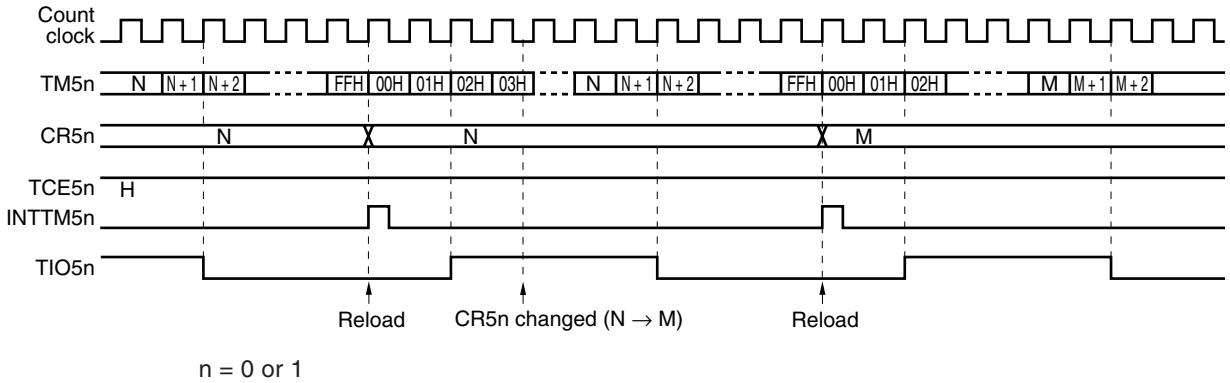
(b) Operation when CR5n is changed

Figure 8-8. Operation Timing When CR5n Is Changed

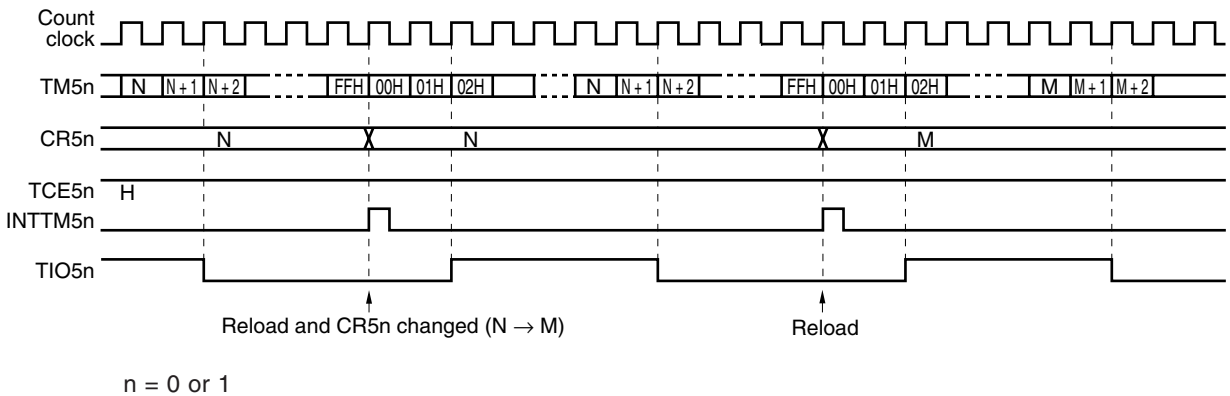
(i) If CR5n value is changed from N to M before overflow of TM5n



(ii) If CR5n value is changed from N to M after overflow of TM5n



(iii) If CR5n value is changed from N to M for duration of 2 clocks (00H and 01H) immediately after overflow of TM5n



**(2) Cascade (16-bit timer) mode**

- Operation as interval timer (with 16-bit resolution)

The two PWM timers can be used as a timer counter with 16-bit resolution by setting bit 4 (TMC514) of 8-bit timer mode control register 51 (TMC51) to 1.

In this case, the 16-bit timer counter operates as an interval timer that repeatedly generates an interrupt request at intervals specified by the count value preset to 8-bit compare register 51 (CR51).

**[Setting]**

(1) Set each register.

- TCL50: TM50 selects the count clock.  
The setting of TM51 cascaded timer is not necessary.
- CR5n: Compare value (Each compare value can be set in a range of 00H to FFH.)
- TMC5n: Selects the clear & start mode in which the timers are cleared and started on match between TM5n and CR5n.

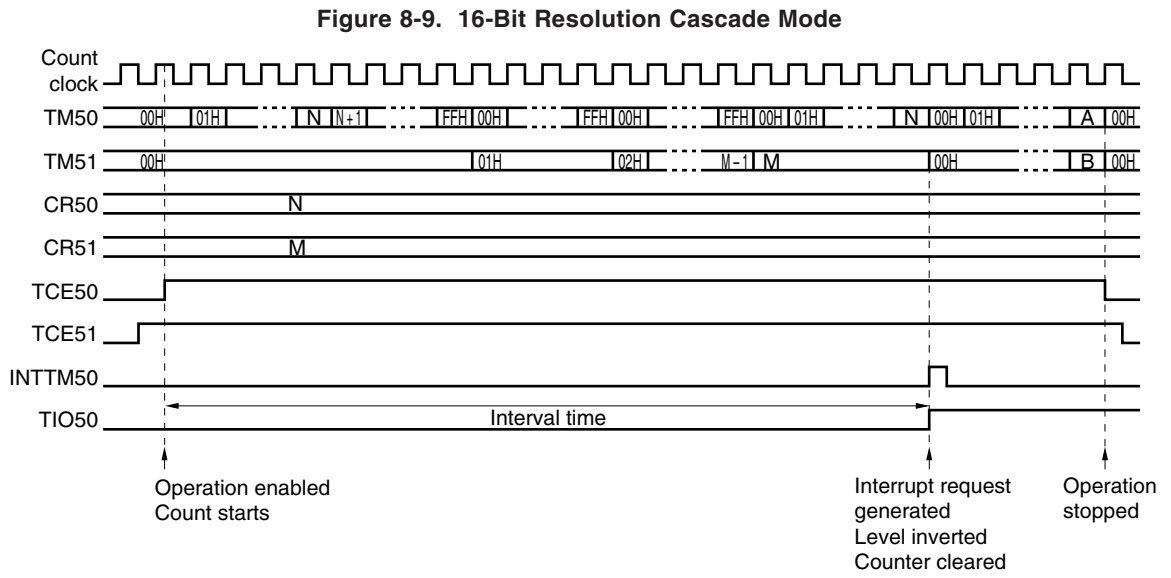
$$\left( \begin{array}{l} \text{TM50} \rightarrow \text{TMC50} = 0000\text{x}\text{x}\text{x}0\text{B} \quad \text{x: don't care} \\ \text{TM51} \rightarrow \text{TMC51} = 0001\text{x}\text{x}\text{x}0\text{B} \quad \text{x: don't care} \end{array} \right)$$

- (2) The counting is started when TCE51 of TMC51 is set to 1 followed by setting of TCE50 of TMC50 to 1.
- (3) When the values of TM5n and CR5n of the cascaded timers cascade match, INTTM50 is generated by TM50 (all the TM5n's are cleared to 00H).
- (4) After that, INTTM50 is repeatedly generated at the same interval.

- Cautions**
1. Before setting 8-bit compare register 5n (CR5n), be sure to stop the timer operation.
  2. Even when the timers are cascaded, if the count value of TM51 matches the value of CR51, INTT51 of TM51 is generated, unless masked. Be sure to mask and disable the interrupt of TM51.
  3. Set TCE51 of TMC51 first, and then TCE50 of TMC50.
  4. The counting can be restarted or stopped by setting 1 or 0 to TCE50 of only TMC50. When setting 8-bit compare register 5n (CR5n), be sure to clear bit 7 (TCE50) of TMC50 and bit 7 (TCE51) of TMC51.

**Remark** n = 0 or 1

Figure 8-9 shows an example of the timing in the 16-bit resolution cascade mode.

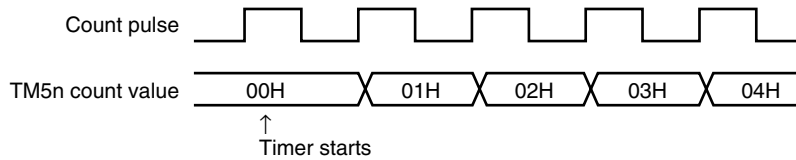


### 8.5 Cautions on 8-Bit PWM Timers

**(1) Error on starting timer**

The time until the match signal is generated after the timer has been started includes an error of up to 1 clock, because 8-bit timer counter 5n (TM5n: n = 0 or 1) is started in asynchronization with the count pulse.

**Figure 8-10. Start Timing of 8-Bit Timer Counter 5n (TM5n)**



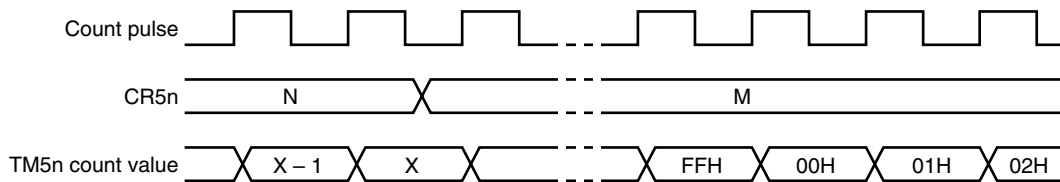
n = 0 or 1

**(2) Operation after changing compare register during timer count operation**

If the value to which the current value of 8-bit compare register 5n (CR5n) is changed is less than the value of 8-bit timer counter 5n (TM5n), the timer continues counting, overflows, and restarts counting from 0. If the new value of CR5n (M) is less than the old value (N), the timer must be restarted after CR5n has been changed.

**Remark** n = 0 or 1

**Figure 8-11. Timing After Changing Compare Register Value During Timer Count Operation**



N > X > M  
n = 0 or 1

**Caution** Except when TIO5n input is selected, be sure to clear TCE5n to 0 to set the stop status (n = 0 or 1).

**(3) Reading TM5n during timer operation**

Because the selected clock is temporarily stopped when TM5n (n = 0 or 1) is read during operation, select a clock with a long high/low level.

When reading TM5n (n = 0 or 1) in cascade mode, provide for changes in the count during a read, for example, by reading counts twice, comparing them, and using one of them only when they match.

## CHAPTER 9 WATCHDOG TIMER

The watchdog timer detects runaway programs.

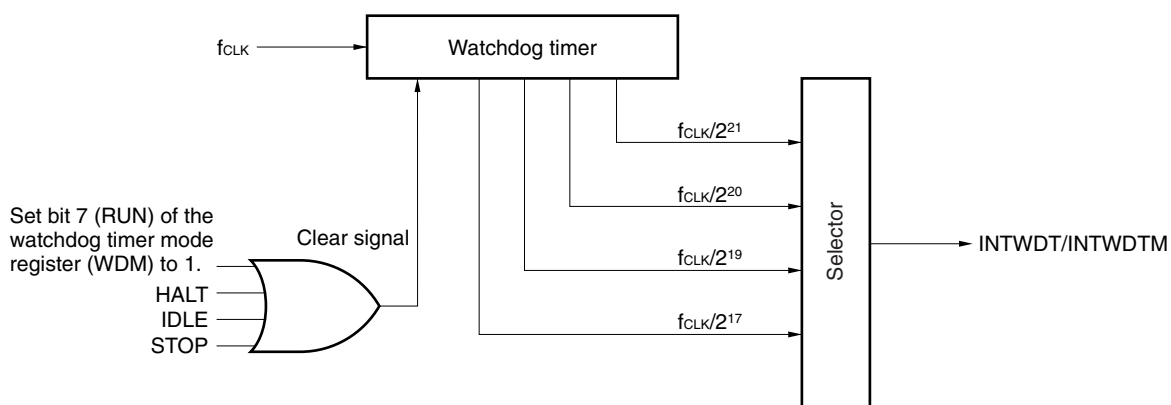
Program or system errors are detected by the generation of watchdog timer interrupts. Therefore, at each location in the program, the instruction that clears the watchdog timer (starts the count) within a constant time is input.

If the watchdog timer overflows without executing the instruction that clears the watchdog timer within the set period, a watchdog timer interrupt (INTWDT) is generated to signal a program error.

### 9.1 Configuration

Figure 9-1 shows a block diagram of the watchdog timer.

Figure 9-1. Block Diagram of Watchdog Timer



- Cautions**
1. When a standby mode (HALT/STOP/IDLE) is selected during operation of the watchdog timer, the watchdog timer is cleared and stopped. If a request is made to clear the HALT/IDLE standby mode, the watchdog timer starts operating immediately after the request is issued. If a request is made to clear the STOP standby mode, the watchdog timer starts operating once the oscillation stabilization time elapses after the request is issued.
  2. INTWDT is a non-maskable interrupt, while INTWDTM is a maskable interrupt. Whether to use the watchdog timer interrupt as non-maskable or maskable can be specified using bit 1 (SWDT) of the interrupt select control register (SNMI). For an explanation of this register, refer to Section 16.3.6 of CHAPTER 16 INTERRUPT FUNCTION.

**Remark**  $f_{CLK}$ : Internal system clock ( $f_{xx}$  to  $f_{xx}/8$ )

## 9.2 Control Register

- **Watchdog timer mode register (WDM)**

WDM is the 8-bit register that controls watchdog timer operation.

To prevent the watchdog timer from erroneously clearing this register due to a runaway program, this register is only written by a special instruction. This special instruction has a special code format (4 bytes) in MOV WDM, #byte instruction. Writing takes place only when the third and fourth op codes are mutual 1's complements. If the third and fourth op codes are not mutual 1's complements and not written, the operand error interrupt is generated. In this case, the return address saved in the stack is the address of the instruction that caused the error. Therefore, the address that caused the error can be identified from the return address saved in the stack. If returning by simply using the RETB instruction from the operand error, an infinite loop results.

Since an operand error interrupt is generated only when the program is running wild (the correct special instruction is only generated when MOV WDM, #byte is described in the RA78K4 NEC assembler), make the program initialize the system.

Other write instructions (MOV WDM, A; AND WDM, #byte; SET1 WDM7, etc.) are ignored and nothing happens. In other words, WDM is not written and interrupts such as operand error interrupts are not generated.

After a system reset ( $\overline{\text{RESET}}$  input), when the watchdog timer starts (when the RUN bit is set to 1), WDM contents cannot change. Only a reset can stop the watchdog timer. The watchdog timer can be cleared by a special instruction.

WDM can be read by 8-bit data transfer instructions.

$\overline{\text{RESET}}$  input sets WDM to 00H.

Figure 9-2 shows the format of WDM.

Figure 9-2. Format of Watchdog Timer Mode Register (WDM)

Address: 0FFC2H After reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
WDM	RUN	0	0	0	0	WDT2	WDT1	0

RUN	Watchdog timer operation setting
0	Stops the watchdog timer.
1	Clears the watchdog timer and starts counting.

WDT2	WDT1	Count clock	Overflow time [ms] ( $f_{CLK} = 12.5 \text{ MHz}$ )
0	0	$f_{CLK}/2^{17}$	10.5
0	1	$f_{CLK}/2^{19}$	41.9
1	0	$f_{CLK}/2^{20}$	83.9
1	1	$f_{CLK}/2^{21}$	167.8

- Cautions**
1. Only the dedicated instruction (MOV WDM, #byte) can write to the watchdog timer mode register (WDM).
  2. When writing to WDM to set the RUN bit to 1, write the same value every time. Even if different values are written, the contents written the first time cannot be updated.
  3. When the RUN bit is set to 1, it cannot be reset to 0 by the software.

**Remark**  $f_{CLK}$ : Internal system clock ( $f_{xx}$  to  $f_{xx}/8$ )  
 $f_{xx}$ : Main system clock frequency

### 9.3 Operations

The watchdog timer is cleared by setting the RUN bit of the watchdog timer mode register (WDM) to 1 to start counting. After the RUN bit is set to 1, when the overflow time set by bits WDT2 and the WDT1 in WDM has elapsed, a non-maskable interrupt (INTWDT) is generated.

If the RUN bit is reset to 1 before the overflow time elapses, the watchdog timer is cleared, and counting restarts.

### 9.4 Cautions

#### 9.4.1 General cautions when using the watchdog timer

(1) The watchdog timer is one way to detect runaway operation, but all runaway operations cannot be detected. Therefore, in a device that particularly demands reliability, the runaway operation must be detected early not only by the on-chip watchdog timer but by an externally attached circuit; and when returning to the normal state or while in the stable state, processing like stopping the operation must be possible.

(2) The watchdog timer cannot detect runaway operation in the following cases.

- <1> When the watchdog timer is cleared in a timer interrupt servicing program
- <2> When there are successive temporary stores of interrupt requests and macro services (refer to **16.9 When Interrupt Requests and Macro Service Are Temporarily Held Pending**)
- <3> When runaway operation is caused by logical errors in the program (when each module in the program operates normally, but the entire system does not operate properly), and when the watchdog timer is periodically cleared
- <4> When the watchdog timer is periodically cleared by an instruction group that is executed during runaway operation
- <5> When the STOP mode and HALT mode or IDLE mode is the result of runaway operation
- <6> When the watchdog timer also runs wild when the CPU runs wild because of introduced noise

In cases <1>, <2>, and <3>, detection becomes possible by correcting the program.

In case <4>, the watchdog timer can be cleared only by the 4-byte special instruction. Similarly in <5>, if there is no 4-byte special instruction, the STOP mode and HALT mode or IDLE mode cannot be set. Since the result of the runaway operation is to enter state <2>, three or more bytes of consecutive data must be a specific pattern (example, BT PSWL.bit, \$\$). Therefore, the results of <4>, <5>, and the runaway operation are believed to very rarely enter state <2>.

#### 9.4.2 Cautions about the $\mu$ PD784976A Subseries watchdog timer

- (1) Only the special instruction (MOV WDM, #byte) can write to the watchdog timer mode register (WDM).
- (2) If the RUN bit is set to 1 by writing to the watchdog timer mode register (WDM), write the same value every time. Even when different values are written, the contents written the first time cannot be changed.
- (3) If the RUN bit is set to 1, it cannot be reset to 0 by the software.



## CHAPTER 10 WATCH TIMER

### 10.1 Functions

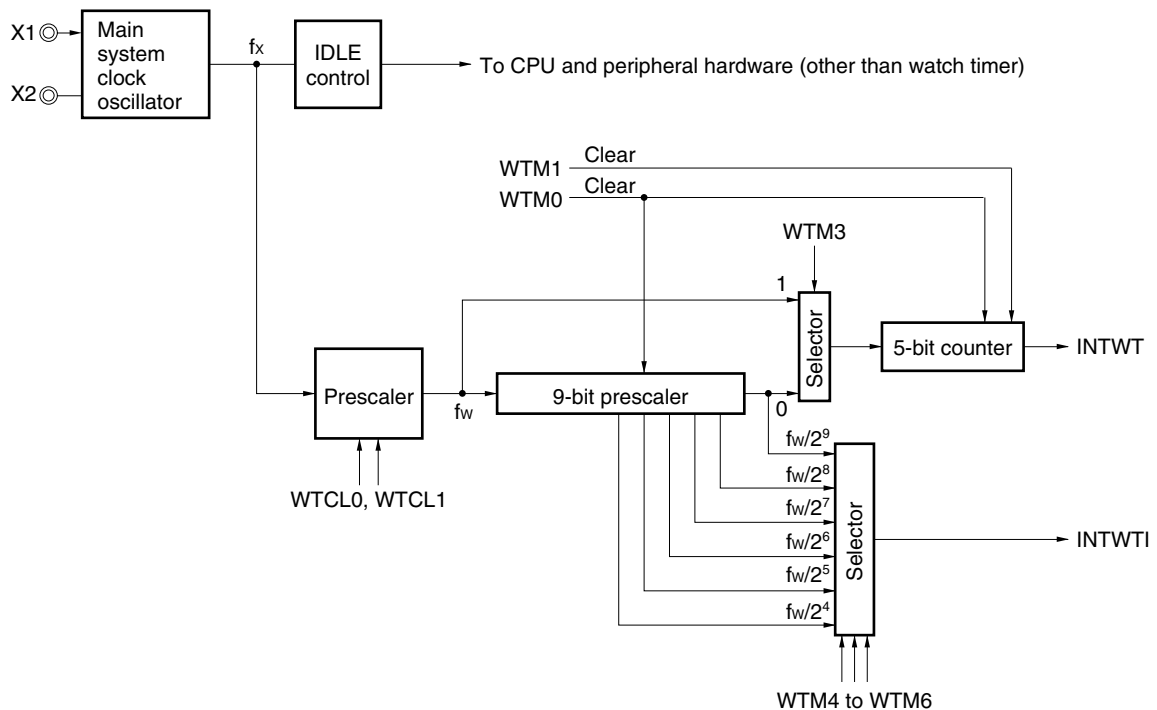
The watch timer has the following functions.

- Watch timer
- Interval timer

The watch timer and interval timer functions can be used at the same time.

Figure 10-1 shows the block diagram of the watch timer.

**Figure 10-1. Block Diagram of Watch Timer**



- Cautions**
1. The interrupt of the watch timer (INTWT) can be used to release IDLE mode (IDLE mode cannot be released by the interval timer interrupt (INTWTI).)
  2. The watch timer can operate in the IDLE mode using the main system clock oscillation frequency ( $f_x$ ).
  3. The watch timer clock frequency ( $f_w$ ) is not generated when timer operation is disabled (bit 0 (WTM0) of WTM = 0) or during the period between the STOP status and the end of the oscillation stabilization time.

- Remarks**
1.  $f_x$ : Main system clock oscillation frequency
  2.  $f_w$ : Watch timer clock frequency
  3. WTM0, WTM1, WTM4 to WTM6: Bits 0, 1, 4 to 6 of the watch timer mode control register (WTM)
  4. WTCL0, WTCL1: Bits 0 and 1 of the watch timer clock select register (WTCL)

**(1) Watch timer**

The watch timer generates an interrupt request (INTWT) at time intervals of 0.5 seconds using the main system clock oscillation frequency ( $f_x$ ).

The watch timer can operate in IDLE mode using the main system clock oscillation frequency ( $f_x$ ).

**Caution** To set the 0.5-second interval, use the following oscillation frequencies: 4.194 MHz, 6.291 MHz, 8.388 MHz, 12.582 MHz

**(2) Interval timer**

The watch timer generates an interval interrupt request signal (INTWTI) at time intervals specified in advance ( $f_w/2^4$  to  $f_w/2^9$ ) based on the main system clock oscillation frequency ( $f_x$ ).

**Caution** The interrupt of the watch timer (INTWT) can be used to release IDLE mode. IDLE mode cannot be released by the interval timer interrupt (INTWTI).

**10.2 Configuration**

The watch timer includes the following hardware.

**Table 10-1. Configuration of Watch Timer**

Item	Configuration
Counter	5 bits × 1
Prescaler	9 bits × 1
Control registers	Watch timer mode control register (WTM) Watch timer clock select register (WTCL)

### 10.3 Watch Timer Control Registers

The watch timer mode control register (WTM) and watch timer clock select register (WTCS) control the watch timer.

#### (1) Watch timer mode control register (WTM)

This register enables or disables the count clock and operation of the watch timer, sets the interval time of the prescaler, and controls the operation of the 5-bit counter.

WTM is set using a 1-bit or 8-bit memory manipulation instruction.

RESET input sets WTM to 00H.

**Figure 10-2. Watch Timer Mode Control Register (WTM)**

Symbol	7	6	5	4	3	2	<1>	<0>	Address	After reset	R/W
WTM	0	WTM6	WTM5	WTM4	WTM3	0	WTM1	WTM0	FF9CH	00H	R/W

WTM6	WTM5	WTM4	Selection of prescaler interval time
0	0	0	$2^4/f_w$ (488 $\mu$ s)
0	0	1	$2^5/f_w$ (977 $\mu$ s)
0	1	0	$2^6/f_w$ (1.95 ms)
0	1	1	$2^7/f_w$ (3.91 ms)
1	0	0	$2^8/f_w$ (7.81 ms)
1	0	1	$2^9/f_w$ (15.6 ms)
Other than above			Setting prohibited

WTM3	Selection of watch timer interrupt time
0	$2^{14}/f_w$ (0.5 s)
1	$2^5/f_w$ (977 $\mu$ s)

WTM1	5-bit counter operation control
0	Clears after operation stops
1	Starts

WTM0	Enables operation of watch timer
0	Stops operation (clears both prescaler and timer)
1	Enables operation

**Remarks 1.**  $f_w$ : Watch timer clock frequency

**2.** Values in parentheses apply when  $f_w = 32.768$  kHz.

- Cautions**
1. The time until the first watch timer interrupt (INTWT) and interval timer interrupt (INTWTI) requests are generated after operation is enabled is not exactly the same as the set interval. Interrupts are generated at the preset interval from the second time onward. Similarly, the time until the first INTWT and INTWTI interrupt requests are generated when STOP mode is set and released while operation is enabled is not exactly the same as the set interval.
  2. Changing the time (setting bits 3 to 6 (WTM3 to WTM6) of the watch timer mode control register (WTM)) is prohibited during timer operation. Change the time when the watch timer is stopped (bit 0 (WTM0) of WTM = 0).
  3. The value of the timer cannot be read or written.
  4. Be sure to set bits 2 and 7 of WTM to 0.

**(2) Watch timer clock select register (WTCL)**

This register selects the source clock of the watch timer.

WTCL is set using a 1-bit memory manipulation instruction.

RESET input sets WTCL to 00H.

**Figure 10-3. Watch Timer Clock Select Register (WTCL)**

Symbol	7	6	5	4	3	2	<1>	<0>	Address	After reset	R/W
WTCL	0	0	0	0	0	0	WTCL1	WTCL0	FF1BH	00H	R/W

WTCL1	WTCL0	Selection of clock supplied to watch timer
0	0	f <sub>x</sub> /128
0	1	f <sub>x</sub> /192
1	0	f <sub>x</sub> /256
1	1	f <sub>x</sub> /384

**Caution** WTCL cannot be written during timer operation. Set WTCL after the watch timer operation is stopped (bit 0 (WTM0) of the watch timer mode control register (WTM) = 0).

- Setting of the watch timer interrupt request

Set the watch timer mode control register (WTM) and WTCL as follows to generate a watch timer interrupt at intervals of 0.5 seconds or 977 μs.

**Table 10-2. Setting of Watch Timer Interrupt Request**

Main System Clock Oscillation Frequency (f <sub>x</sub> )	WTCL1	WTCL0	Watch Timer Interrupt Interval	
			WTM3 = 0	WTM3 = 1
4.194 MHz	0	0	f <sub>x</sub> /2 <sup>21</sup> (0.5 s)	f <sub>x</sub> /2 <sup>12</sup> (977 μs)
6.291 MHz	0	1	f <sub>x</sub> /(2 <sup>20</sup> × 3) (0.5 s)	f <sub>x</sub> /(2 <sup>11</sup> × 3) (977 μs)
8.388 MHz	1	0	f <sub>x</sub> /2 <sup>22</sup> (0.5 s)	f <sub>x</sub> /2 <sup>13</sup> (977 μs)
12.582 MHz	1	1	f <sub>x</sub> /(2 <sup>21</sup> × 3) (0.5 s)	f <sub>x</sub> /(2 <sup>12</sup> × 3) (977 μs)

## CHAPTER 11 A/D CONVERTER

### 11.1 Function of A/D Converter

The A/D converter converts analog input signals into digital values with a resolution of 8 bits. Twelve analog input channels (ANI0 to ANI11) can be controlled.

The A/D conversion operation can be started only by software.

One of the analog input channels, ANI0 to ANI11, is selected for A/D conversion. The A/D conversion operation is repeatedly performed, and each time it has been completed once, an interrupt request (INTAD) is generated.

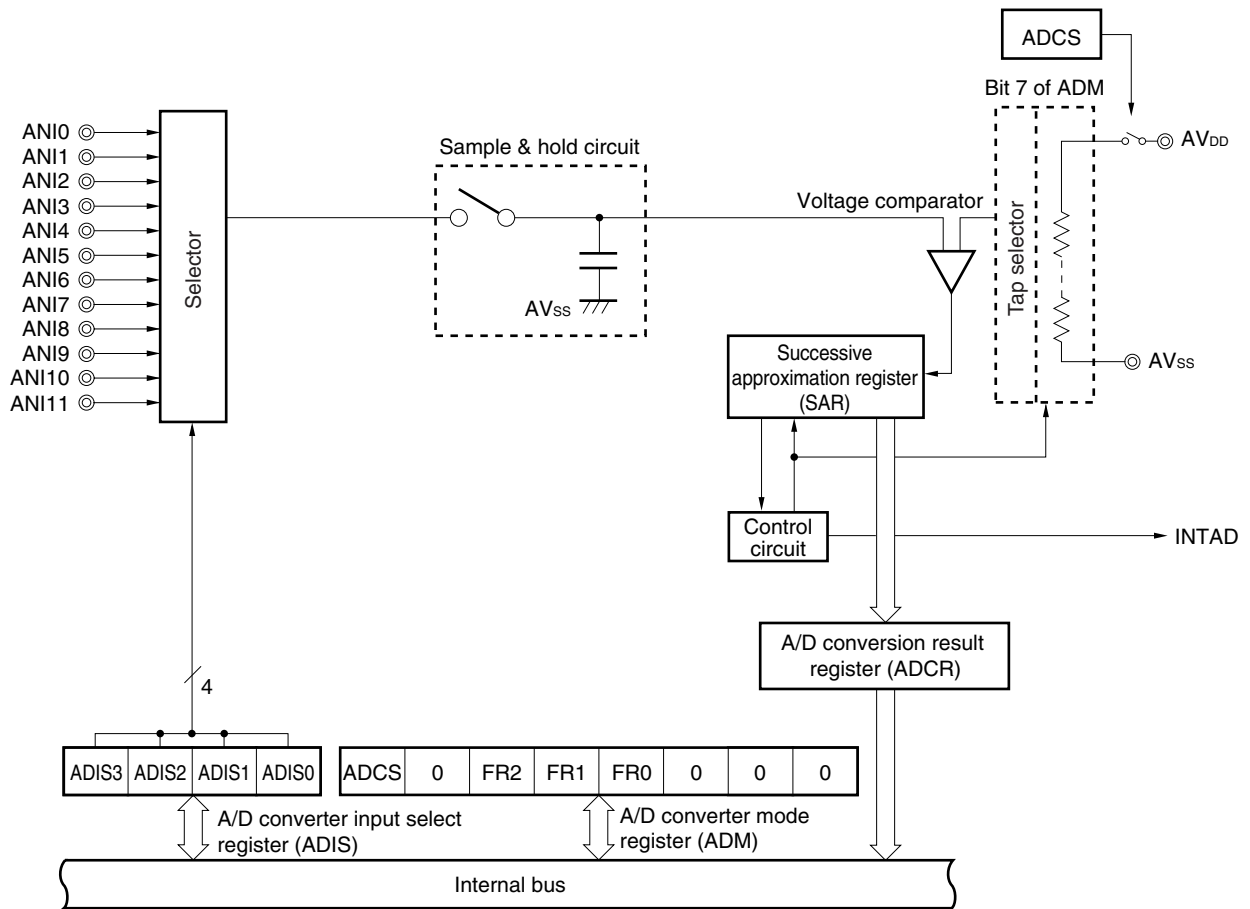
### 11.2 Configuration of A/D Converter

The A/D converter includes the following hardware.

**Table 11-1. Configuration of A/D Converter**

Item	Configuration
Analog input	12 channels (ANI0 to ANI11)
Register	Successive approximation register (SAR) A/D conversion result register (ADCR)
Control register	A/D converter mode register (ADM) A/D converter input select register (ADIS)

Figure 11-1. Block Diagram of A/D Converter

**(1) Successive approximation register (SAR)**

This register compares the voltage value of the input analog signal with the value of the voltage tap (compare voltage) from the series resistor string, and holds the result of the comparison, starting from the most significant bit (MSB).

When the comparison result has been retained to this register up to the least significant bit (LSB) (i.e., when the A/D conversion has been completed), the contents of this register are transferred to the A/D conversion result register (ADCR).

**(2) A/D conversion result register (ADCR)**

This register holds the result of the A/D conversion. Each time the A/D conversion has been completed, the conversion result is loaded to this register from the successive approximation register (SAR).

ADCR is read using an 8-bit memory manipulation instruction.

The value of this register is undefined when  $\overline{\text{RESET}}$  signal is input.

**(3) Sample & hold circuit**

The sample & hold circuit samples the analog input signals sent from the input circuit one by one, and sends them to the voltage comparator. It also holds the voltage value of the sampled analog input signal during A/D conversion.

**(4) Voltage comparator**

The voltage comparator compares the analog input signal with the output voltage of the series resistor string.

**(5) Series resistor string**

The series resistor string is connected between the  $AV_{DD}$  and  $AV_{SS}$  pins, and generates a voltage to be compared with the input analog signal.

**(6) ANI0 to ANI11 pins**

These are 12 channels of analog input pins of the A/D converter, and input analog signals to be converted.

**Caution** Make sure that the input voltages of ANI0 to ANI11 are within the rated range. If a voltage greater than  $AV_{DD}$  or less than  $AV_{SS}$  is input a channel (even if it is within the absolute maximum rating range), the converted value of the channel is undefined, and, in the worst case, the converted values of the other channels are affected.

**(7)  $AV_{SS}$  pin**

This is the ground potential pin of the A/D converter. Make sure this pin is always at the same potential as the  $V_{SS1}$  pin even when the A/D converter is not used.

**(8)  $AV_{DD}$  pin**

★ This is the analog power supply pin of the A/D converter. When the A/D converter is used, use the  $AV_{DD}$  pin with the same potential as  $V_{DD1}$ . When the A/D converter is not used, the  $AV_{DD}$  pin can be used with the same potential as  $V_{SS1}$ .

In the standby mode, the current flowing to the series resistor string can be lowered by stopping the conversion operation (by clearing bit 7 (ADCS) of the A/D converter mode register (ADM)).



### 11.3 A/D Converter Control Registers

The following two types of registers control the A/D converter.

- A/D converter mode register (ADM)
- A/D converter input select register (ADIS)

#### (1) A/D converter mode register (ADM)

This register specifies the conversion time of the input analog signal to be converted, and starts or stops the conversion operation.

ADM is set using a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$  input sets ADM to 00H.

**Figure 11-2. Format of A/D Converter Mode Register**

Symbol	<7>	6	5	4	3	2	1	0	Address	After reset	R/W
ADM	ADCS	0	FR2	FR1	FR0	0	0	0	FF80H	00H	R/W

ADCS	A/D conversion operation control
0	Stops conversion
1	Enables conversion

FR2	FR1	FR0	A/D conversion time selection <sup>Note 1</sup>		
			Number of clocks	$f_{xx} = 12.5 \text{ MHz}$	$f_{xx} = 6.2 \text{ MHz}$
0	0	0	144/ $f_{xx}$	Setting prohibited	23.0 $\mu\text{s}$
0	0	1	120/ $f_{xx}$		19.2 $\mu\text{s}$
0	1	0	96/ $f_{xx}$		15.4 $\mu\text{s}$
1	0	0	288/ $f_{xx}$	23.0 $\mu\text{s}$	46.1 $\mu\text{s}$
1	0	1	240/ $f_{xx}$	19.2 $\mu\text{s}$	38.4 $\mu\text{s}$
1	1	0	192/ $f_{xx}$	15.4 $\mu\text{s}$	30.7 $\mu\text{s}$
Other than above			—	Setting prohibited	

- Notes**
1. Make sure that the A/D conversion time is 14  $\mu\text{s}$  or longer.
  2. When rewriting the data of FR0 to FR2, keep the A/D converter stopped.

**Caution** The conversion result is undefined immediately after bit 7 (ADCS) has been set.

**Remark**  $f_{xx}$ : Main system clock frequency

**(2) A/D converter input select register (ADIS)**

This register specifies a port that inputs the analog voltage to be converted.

ADIS is set using a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$  input sets ADIS to 00H.

**Figure 11-3. Format of A/D Converter Input Select Register**

Symbol	7	6	5	4	3	2	1	0	Address	After reset	R/W
ADIS	0	0	0	0	ADIS3	ADIS2	ADIS1	ADIS0	FF81H	00H	R/W

ADIS3	ADIS2	ADIS1	ADIS0	Analog input channel specification
0	0	0	0	ANI0
0	0	0	1	ANI1
0	0	1	0	ANI2
0	0	1	1	ANI3
0	1	0	0	ANI4
0	1	0	1	ANI5
0	1	1	0	ANI6
0	1	1	1	ANI7
1	0	0	0	ANI8
1	0	0	1	ANI9
1	0	1	0	ANI10
1	0	1	1	ANI11
Other than above				Setting prohibited

## 11.4 Operation of A/D Converter

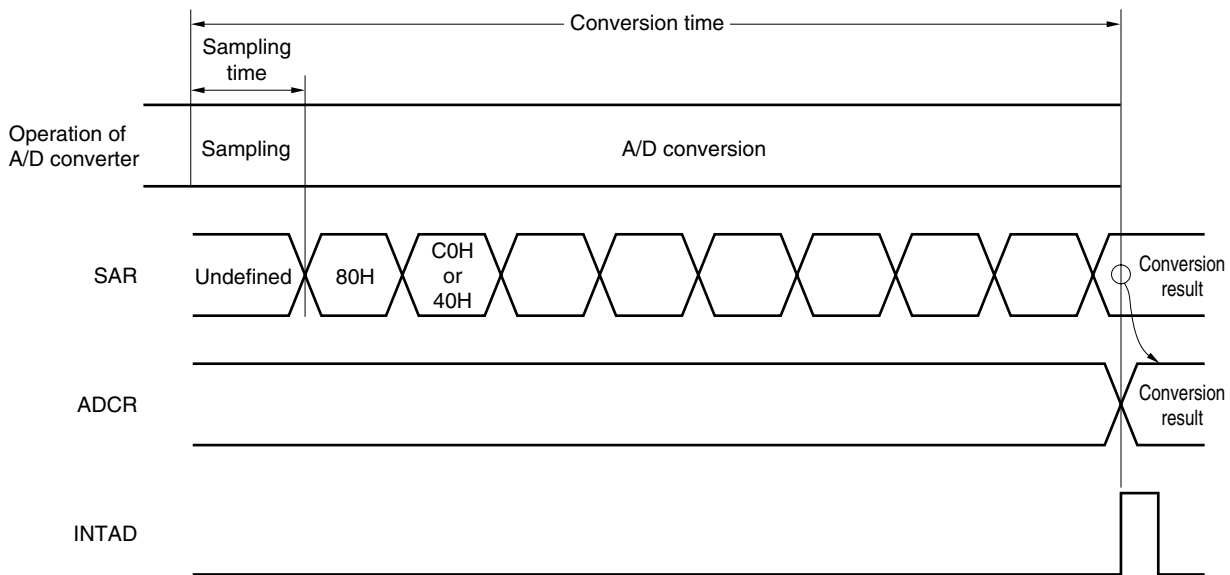
### 11.4.1 Basic operation of A/D converter

- (1) Select one channel for A/D conversion by using the A/D converter input select register (ADIS).
- (2) The sample & hold circuit samples the voltage input to the selected analog input channel.
- (3) The sample & hold circuit enters the hold status after it has performed sampling for fixed time, and holds the input analog voltage until the A/D conversion is completed.
- (4) Bit 7 of the successive approximation register (SAR) is set. The tap selector sets the voltage tap of the series resistor string to  $(1/2) AV_{DD}$ .
- (5) The voltage comparator compares the voltage difference between the voltage of the series resistor string and voltage tap. If the input analog voltage is greater than  $(1/2) AV_{DD}$ , the MSB of the SAR remains set. If it is less than  $(1/2) AV_{DD}$ , MSB is reset.
- (6) Bit 6 of the SAR is automatically set, and the next comparison is performed. The voltage tap of the series resistor string is selected as follows, depending on the value of bit 7 to which the result has been already set.
  - Bit 7 = 1:  $(3/4) AV_{DD}$
  - Bit 7 = 0:  $(1/4) AV_{DD}$This voltage tap is compared with the input analog voltage. Depending on this result, bit 6 of the SAR is manipulated as follows:
  - If input analog voltage  $\geq$  voltage tap: Bit 6 = 1
  - If input analog voltage  $\leq$  voltage tap: Bit 6 = 0
- (7) Comparison continues like this up to bit 0 of the SAR.
- (8) When comparison of 8 bits has been completed, the valid digital result remains in the SAR, and its value is transferred and latched to the A/D conversion result register (ADCR).

At the same time, an A/D conversion end interrupt request (INTAD) is generated.

**Caution** The first A/D conversion result obtained immediately after setting bit 7 (ADCS) of the A/D converter mode register (ADM) to 1 is undefined and should be discarded by polling the A/D conversion end interrupt request (INTAD).

Figure 11-4. Basic Operation of A/D Converter



The A/D conversion operation is performed successively until bit 7 (ADCS) of the A/D converter mode register (ADM) is reset to 0 by software.

If an attempt is made to write data to the ADM or A/D converter input select register (ADIS) during A/D conversion operation, the conversion operation is initialized, and conversion is started from the beginning if ADCS is set to 1. The value of the A/D conversion result register (ADCR) is undefined when the  $\overline{\text{RESET}}$  signal is input.

**Caution** The first A/D conversion result obtained immediately after setting bit 7 (ADCS) of the A/D converter mode register (ADM) to 1 is undefined and should be discarded by polling the A/D conversion end interrupt request (INTAD).

**11.4.2 Input voltage and conversion result**

The analog voltage input to an analog input pin (ANI0 to ANI11) and the result of A/D conversion (A/D conversion result register (ADCR)) have the following relation:

$$ADCR = \text{INT} \left( \frac{V_{IN}}{AV_{DD}} \times 256 + 0.5 \right)$$

or,

$$(ADCR - 0.5) \times \frac{AV_{DD}}{256} \leq V_{IN} < (ADCR + 0.5) \times \frac{AV_{DD}}{256}$$

INT( ): Function that returns integer of value in ( )

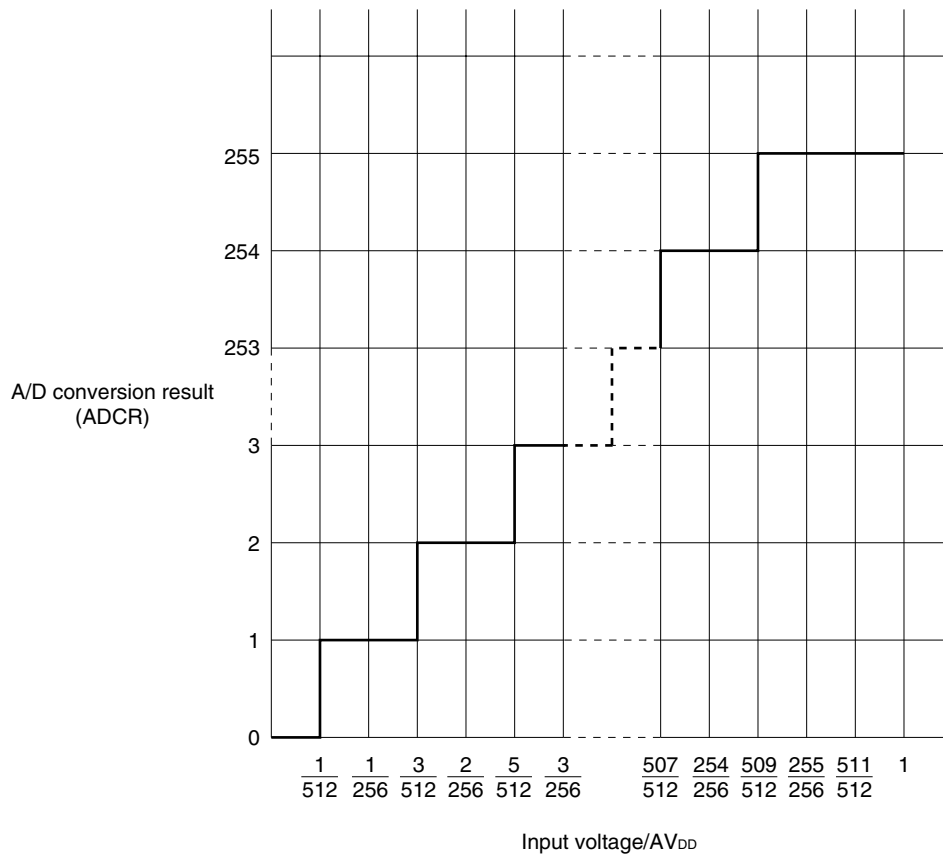
V<sub>IN</sub>: Analog input voltage

AV<sub>DD</sub>: Supply voltage to A/D converter

ADCR: Value of A/D conversion result register (ADCR)

Figure 11-5 shows the relation between the analog input voltage and A/D conversion result.

**Figure 11-5. Relation Between Analog Input Voltage and A/D Conversion Result**



### 11.4.3 Operation mode of A/D converter

Select one analog input channel from ANI0 to ANI11 by the A/D converter input select register (ADIS) to start A/D conversion.

The A/D conversion operation can be started only by software (by setting the A/D converter mode register (ADM)).

The A/D conversion result is stored in the A/D conversion result register (ADCR), and an interrupt request signal (INTAD) is generated.

- A/D conversion by software start

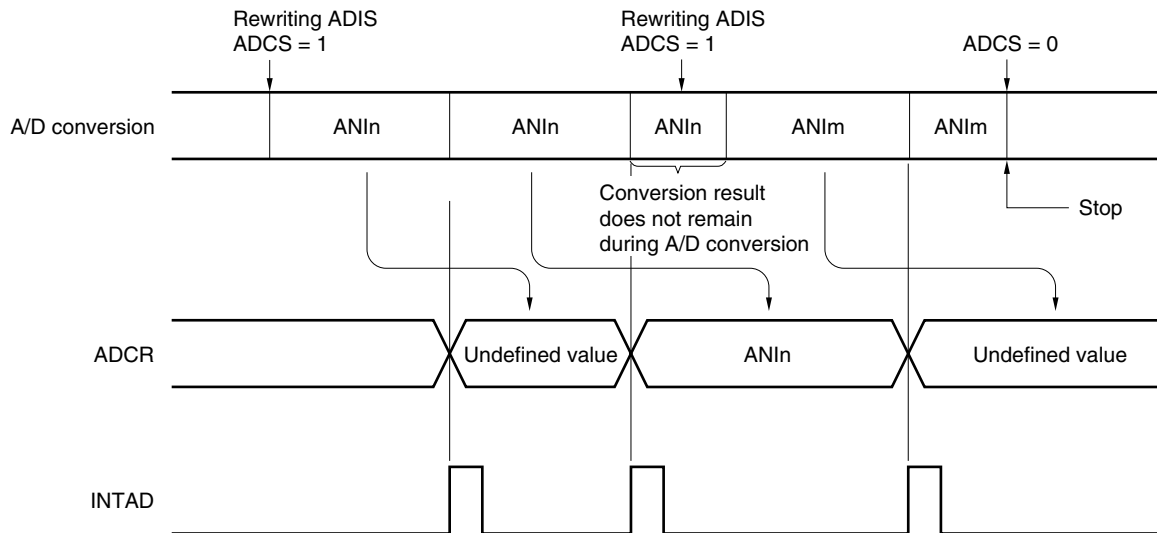
Converting the voltage applied to the analog input pin specified by the A/D converter input select register (ADIS) is started when bit 7 (ADCS) of the A/D converter mode register (ADM) is set to 1.

When the A/D conversion has been completed, the result of the conversion is stored in the A/D conversion result register (ADCR), and an interrupt request (INTAD) is generated. When the A/D conversion has been started and completed once, the next conversion operation is immediately started. This is repeated until new data is written to ADIS.

If ADIS is rewritten during A/D conversion, the conversion under execution is stopped, and conversion of the selected analog input channel is started.

If data with ADCS being 0 is written to the ADM during A/D conversion, the conversion is immediately stopped.

**Figure 11-6. A/D Conversion by Software Start**



**Remark**  $n = 0, 1, \dots, 11$   
 $m = 0, 1, \dots, 11$

**Caution** The first A/D conversion result obtained immediately after setting bit 7 (ADCS) of the A/D converter mode register (ADM) to 1 is undefined and should be discarded by polling the A/D conversion end interrupt request (INTAD).

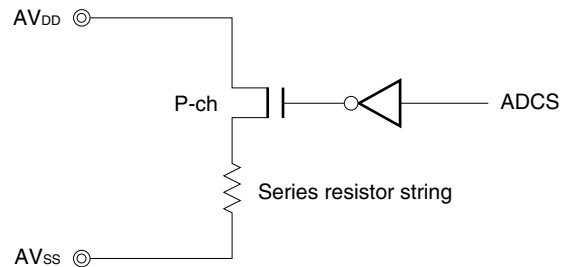
## 11.5 Notes on A/D Converter

### (1) Current consumption in standby mode

The A/D converter is stopped in the standby mode. At this time, the current consumption can be reduced by stopping the conversion (by clearing bit 7 (ADCS) of the A/D converter mode register (ADM) to 0).

Figure 11-7 shows how the current consumption can be reduced in the standby mode.

**Figure 11-7. Example of Reducing Current Consumption in Standby Mode**



### (2) Input range of ANI0 to ANI11

Make sure that the input voltages of ANI0 to ANI11 are within the rated range. If a voltage greater than AVDD or less than AVSS is input to a channel (even if it is within the absolute maximum rating range), the converted value of the channel is undefined, and, in the worst case, the converted values of the other channels are affected.

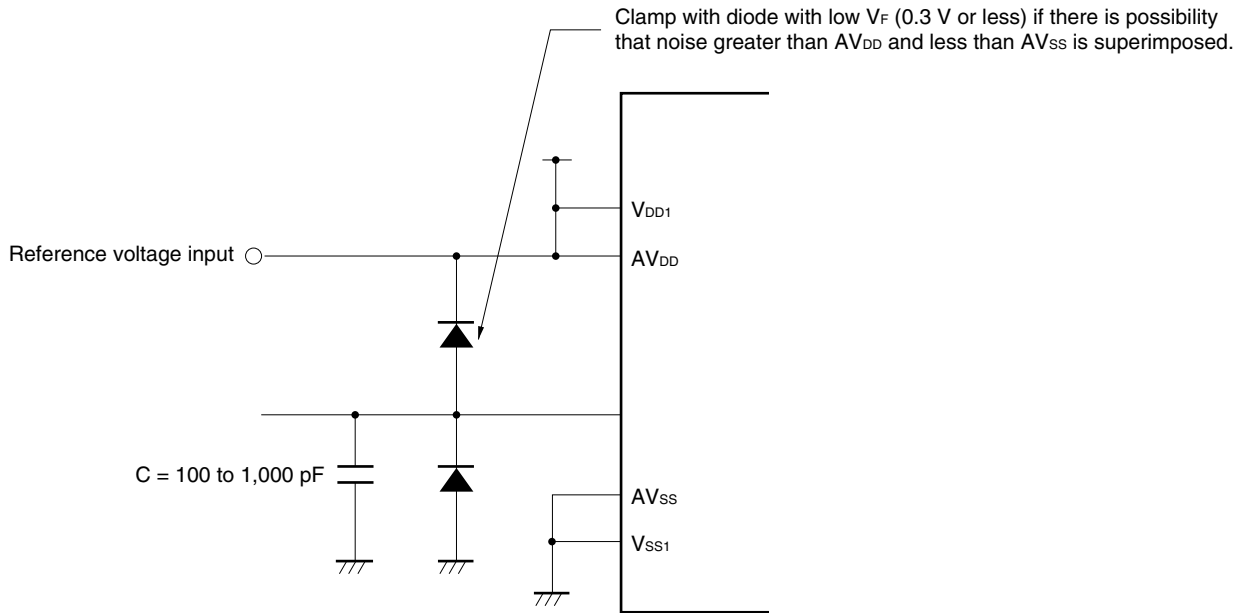
### (3) Conflicting operation

- <1> Conflict between writing and reading A/D conversion result register (ADCR) on completion of conversion  
Reading ADCR takes precedence. After ADCR has been read, a new conversion result is written to ADCR.
- <2> Conflict between writing ADCR and input of external trigger signal on completion of conversion  
An external trigger signal is not accepted during A/D conversion. Therefore, the external trigger signal is not accepted while ADCR is written.
- <3> Conflict between writing ADCR and writing the A/D converter mode register (ADM) or writing the A/D converter input select register (ADIS) on completion of conversion  
Writing ADM or ADIS takes precedence. ADCR is not written. Nor is the conversion end interrupt request signal (INTAD) generated.

### (4) Noise measures

To maintain the 8-bit resolution, care must be exercised that no noise is superimposed on the AVDD and ANI0 to ANI11 pins. The higher the output impedance of the analog input source, the heavier the influence of noise. To suppress noise, connecting external C as shown in Figure 11-8 is recommended.

Figure 11-8. Processing of Analog Input Pin

**(5) ANI0 through ANI11**

The analog input pins (ANI0 to ANI11) are multiplexed with port pins (P00 to P03 and P11 to P17). When one of ANI0 to ANI11 is selected for A/D conversion, do not execute an instruction that inputs data to the port during the conversion. If such an instruction is executed, the conversion resolution may drop. When a digital pulse is applied to a pin adjacent to the analog input pins during A/D conversion, the expected A/D conversion value may not be obtained because of coupling noise. Therefore, do not apply a pulse to the pins adjacent to the analog input pins during A/D conversion.

**(6) Input impedance of AV<sub>DD</sub> pin**

The reference voltage source pin is also used as the AV<sub>DD</sub> pin. A series resistor string with a resistance of about 21.4 k $\Omega$  is connected between the AV<sub>DD</sub> and AV<sub>SS</sub> pins. If the output impedance of the reference voltage source is high, therefore, the impedance is virtually connected in series with the resistor string between the AV<sub>DD</sub> and AV<sub>SS</sub> pins, increasing the error of the reference voltage.

**(7) Interrupt request flag (ADIF)**

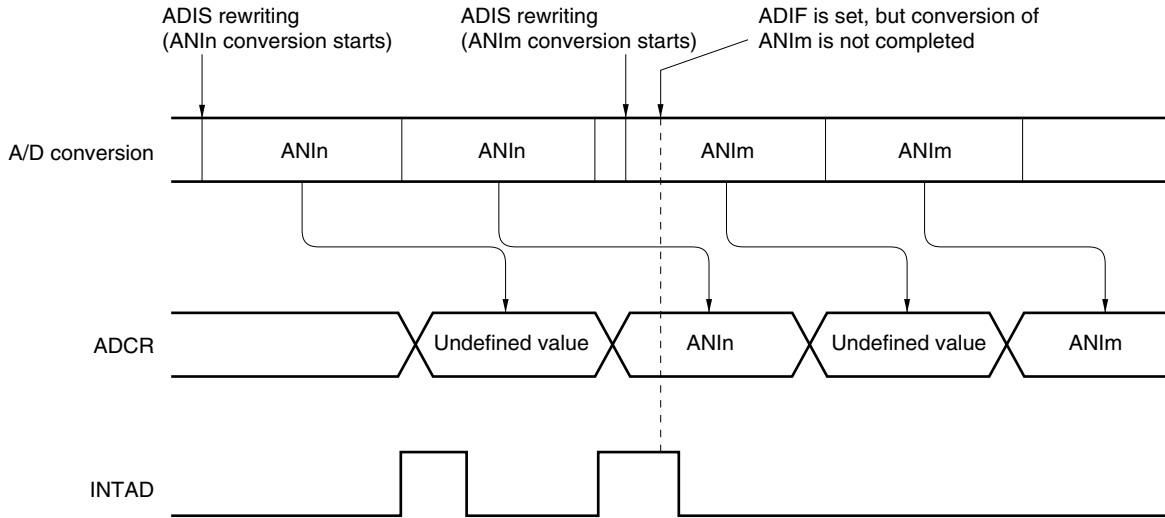
The interrupt request flag (ADIF) is not cleared even if the contents of the A/D converter input select register (ADIS) are changed.

If the analog input pin is changed during A/D conversion, therefore, the A/D conversion result of the old analog input may be written to ADIS immediately before ADIS is rewritten, and consequently, the conversion end interrupt flag may be set. If the ADIF is read immediately after ADIS has been rewritten, ADIF may be set despite that the A/D conversion of the new analog input has not been completed.

Before resuming A/D conversion that has been stopped, clear ADIF.



**Figure 11-9. Timing of A/D Conversion End Interrupt Request Generation**



★

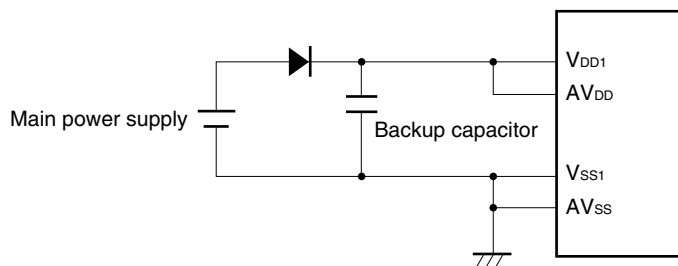
**Remark**  $n = 0, 1, \dots, 11$   
 $m = 0, 1, \dots, 11$

**Caution** The first A/D conversion result obtained immediately after setting bit 7 (ADCS) of the A/D converter mode register (ADM) to 1 is undefined and should be discarded by polling the A/D conversion end interrupt request (INTAD).

**(8) AV<sub>DD</sub> pin**

The AV<sub>DD</sub> pin supplies power to the analog circuit. It also supplies power to the input circuit of ANI0 to ANI11. Therefore, apply the same potential as that of the V<sub>DD1</sub> pin to this pin, as shown in Figure 11-10, in an application where a backup power supply is used.

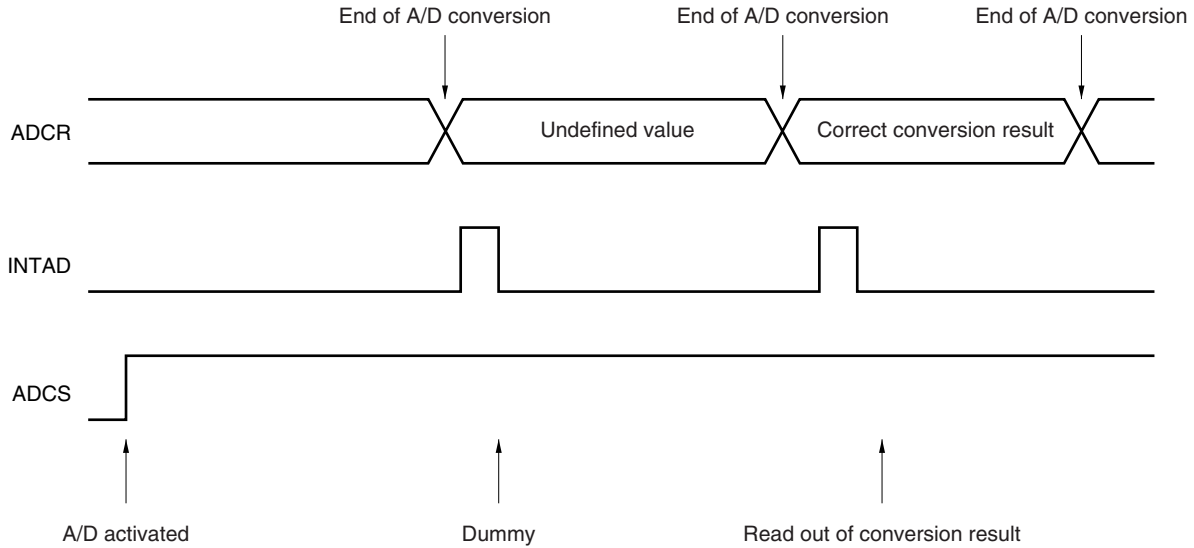
**Figure 11-10. Processing of AV<sub>DD</sub> Pin**



**(9) Result of conversion immediately after A/D conversion is started**

The first A/D conversion result obtained after the start of A/D conversion is undefined and should be discarded by polling the A/D conversion end interrupt request (INTAD) or using other such means.

**Figure 11-11. Result of Conversion Immediately After A/D Conversion Is Started**



★ **(10) Timing that makes the A/D conversion result undefined**

If the timing of the end of A/D conversion and the timing of the stop of operation of the A/C converter conflict, the A/D conversion value may be undefined. Because of this, be sure to read the A/D conversion result while the A/D converter is in operation. Furthermore, when reading an A/D conversion result after the A/D converter operation has stopped, be sure to have done so by the time the next conversion result is complete. The conversion result read timing is shown in Figures 11-12 and 11-13 below.

**Figure 11-12. Conversion Result Read Timing (When Conversion Result Is Undefined)**

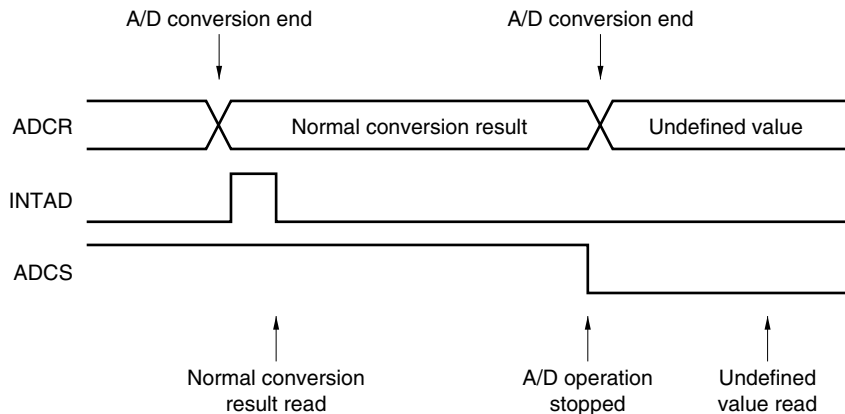
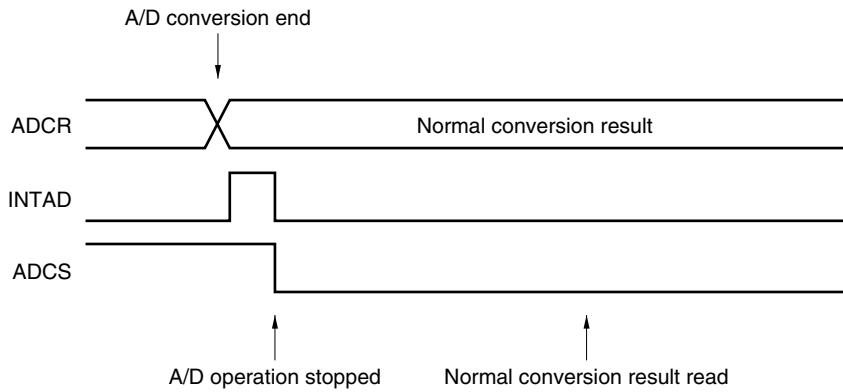


Figure 11-13. Conversion Result Read Timing (When Conversion Result Is Normal)



★ **(11) Cautions on board design**

In order to avoid negative effects from digital circuit noise on the board, analog circuits must be placed as far away as possible from digital circuits. It is particularly important to prevent analog and digital signal lines from crossing or coming into close proximity, as A/D conversion characteristics are vulnerable to degradation from the induction of noise or other such factors.

★ **(12) Reading A/D conversion result register (ADCR)**

If the conversion result register (ADCR) is read after stopping the A/D conversion operation, the conversion result may be undefined. Therefore, be sure to read ADCR before stopping operation of the A/D converter.

## CHAPTER 12 SERIAL INTERFACE

### 12.1 Function of Serial Interface

The serial interface has the following two modes.

- Operation stop mode
- 3-wire serial I/O mode

#### (1) Operation stop mode

This mode is used when serial transfer is not performed.

#### (2) 3-wire serial I/O mode (with MSB first)

In this mode, 8-bit data is transferred by using three lines: serial clock ( $\overline{SCK}$ ), serial output (SO), and serial input (SI).

Because simultaneous transmit/receive operation can be performed in the 3-wire serial I/O mode, the processing time of data transfer can be shortened.

The first bit of the 8-bit data to be transferred is fixed to the MSB.

The 3-wire serial I/O mode is useful when connecting peripheral I/Os or display controller having a clocked serial interface.

### 12.2 Configuration of Serial Interface

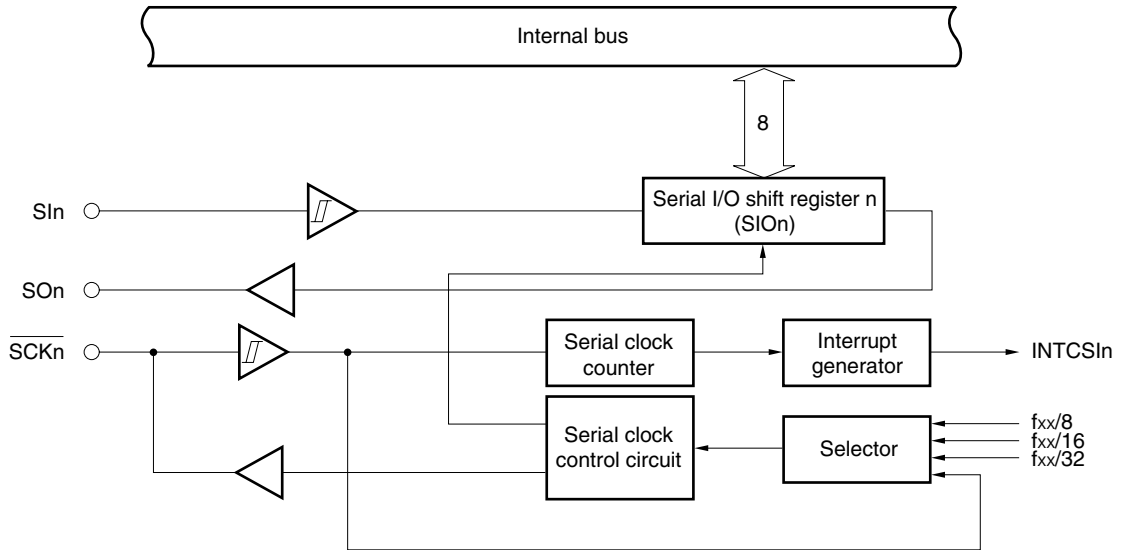
The serial interface includes the following hardware.

**Table 12-1. Configuration of Serial Interface**

Item	Configuration
Register	Serial I/O shift register n (SION)
Control register	Serial operation mode register n (CSIMn)

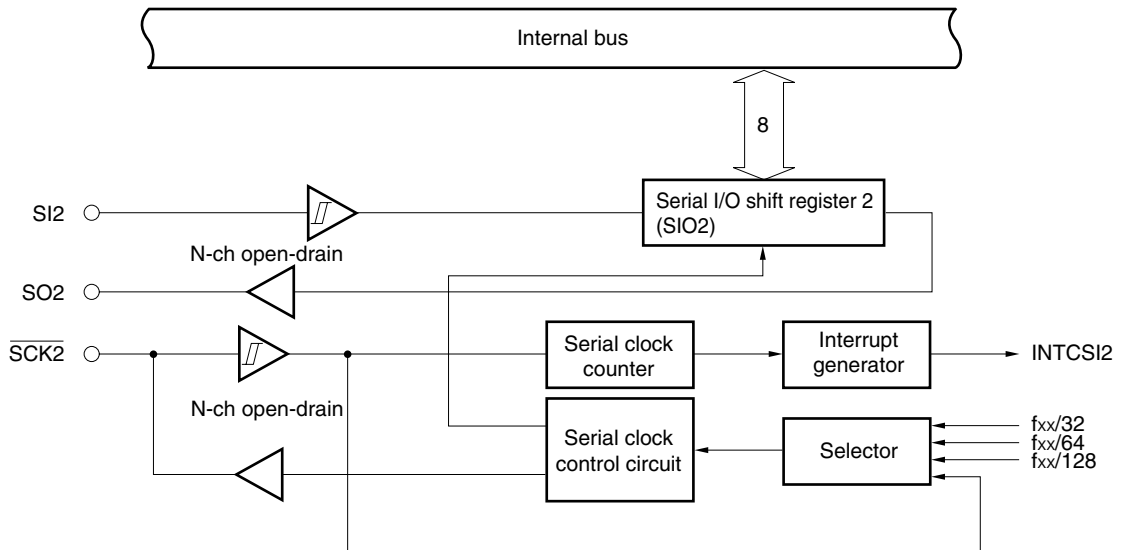
**Remark** n = 0 to 2

Figure 12-1. Block Diagram of Serial Interface 0, 1



Remark n = 0 or 1

Figure 12-2. Block Diagram of Serial Interface 2



**(1) Serial I/O shift register n (SIO<sub>n</sub>)**

This 8-bit register converts parallel data to serial data to perform serial transmission/reception (shift operation) in synchronization with the serial clock.

SIO<sub>n</sub> is set using an 8-bit memory manipulation instruction.

The serial operation is started by writing data to or reading data from SIO<sub>n</sub> when bit 7 (CSIE<sub>n</sub>) of serial operation mode register n (CSIM<sub>n</sub>) is 1.

During transmission, the data written to SIO<sub>n</sub> is output to the serial output line (SO<sub>n</sub>).

During reception, the data is read to SIO<sub>n</sub> from the serial input line (SI<sub>n</sub>).

The contents of this register are undefined when the  $\overline{\text{RESET}}$  signal is input.

**Caution** Do not access SIO<sub>n</sub> during transmission, except when triggering the transmission (reading SIO<sub>n</sub> is prohibited when bit 2 (MODE<sub>n</sub>) of CSIM<sub>n</sub> = 0, and writing is prohibited when MODE<sub>n</sub> = 1).

**Remark** n = 0 to 2

**(2) Serial clock counter**

This counter counts the serial clock output or input during transmission/reception to check that 8-bit data has been transmitted/received.

### 12.3 Serial Interface Control Registers

The serial interface is controlled by serial operation mode register n (CSIMn).

- **Serial operation mode register n (CSIMn)**

This register selects the serial clock and operation mode of the serial interface, and enables or disables the operation.

CSIMn is set using a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$  input sets CSIMn to 00H.

**Remark** n = 0 to 2

**Figure 12-3. Format of Serial Operation Mode Register n**

Symbol	<7>	6	5	4	3	2	1	0	Address	After reset	R/W
CSIMn	CSIE <sub>n</sub>	0	0	0	0	MODE <sub>n</sub>	SCL <sub>n1</sub>	SCL <sub>n0</sub>	FF90H, FF91H	00H	R/W

CSIE <sub>n</sub>	Enables or disables operation of SIO <sub>n</sub>		
	Shift register operation	Serial counter	Port
0	Stopped	Cleared	Port function <sup>Note</sup>
1	Enabled	Count operation enabled	Serial function + port function

MODE <sub>n</sub>	Transfer operation mode flag		
	Operation mode	Transfer start trigger	SOn output
0	Transmit or transmit/receive mode	SIO <sub>n</sub> write	Normal output
1	Receive mode	SIO <sub>n</sub> read	Fixed to low level

SCL <sub>n1</sub>	SCL <sub>n0</sub>	Clock selection
0	0	External clock input to $\overline{\text{SCKn}}$ pin
0	1	$f_{xx}/8$ (1.56 MHz)
1	0	$f_{xx}/16$ (781 kHz)
1	1	$f_{xx}/32$ (391 kHz)

**Note** The pins connected to SIn, SOn, and  $\overline{\text{SCKn}}$  can be used as port pins when CSIE<sub>n</sub> = 0 (when the SIO<sub>n</sub> operation is stopped).

- Remarks**
1.  $f_{xx}$ : Main system clock frequency
  2. The values in parentheses are valid for operation when  $f_{xx}$  is 12.5 MHz.
  3. n = 0 or 1

Figure 12-4. Format of Serial Operation Mode Register 2

Symbol	<7>	6	5	4	3	2	1	0	Address	After reset	R/W
CSIM2	CSIE2	0	0	0	0	MODE2	SCL21	SCL20	FF92H	00H	R/W

CSIE2	Enables or disables operation of SIO2		
	Shift register operation	Serial counter	Port
0	Stopped	Cleared	Port function <sup>Note</sup>
1	Enabled	Count operation enabled	Serial function + port function

MODE2	Transfer operation mode flag		
	Operation mode	Transfer start trigger	SO2 output
0	Transmit or transmit/receive mode	SIO2 write	Normal output
1	Receive mode	SIO2 read	Fixed to low level

SCL21	SCL20	Clock selection
0	0	External clock input to $\overline{\text{SCK2}}$ pin
0	1	$f_{xx}/32$ (131 MHz)
1	0	$f_{xx}/64$ (65.5 kHz)
1	1	$f_{xx}/128$ (32.7 kHz)

**Note** The pins connected to SI2, SO2, and  $\overline{\text{SCK2}}$  can be used as port pins when CSIE2 = 0 (when the SIO2 operation is stopped).

**Caution** Because the  $\overline{\text{SCK2}}$  pin of serial interface 2 (SIO2) is an N-ch open-drain pin, the clock output from this pin does not have a duty factor of 50% if the internal clock is selected.

The set values listed above are for clocks that can be used when a pull-up resistor of 10 kΩ is connected at  $f_{xx} = 4.194$  MHz.

Under any other conditions, or if the wiring capacitance of the board differs even when the above conditions are satisfied, the operation may not be performed correctly even if the above clock is selected. Be sure to perform evaluation before selecting a clock.

**Remarks** 1.  $f_{xx}$ : Main system clock frequency

★ 2. The values in parentheses are valid for operation when  $f_{xx}$  is 4.194 MHz.



★

Table 12-2. Serial Interface Operation Mode Settings

(1) Operation stopped mode

(a) P25 to P27

ASIM0		CSIM0			PM25	P25	PM26	P26	PM27	P27	First Bit	Shift Clock	P25/SI0/RxD0 Pin Function	P26/SO0/TxD0 Pin Function	P27/ $\overline{\text{SCK0}}$ /ASCK0 Pin Function
TXE0	RXE0	CSIE0	SCL01	SCL00											
0	0	0	×	×	× <sup>Note 1</sup>	× <sup>Note 1</sup>	× <sup>Note 1</sup>	× <sup>Note 1</sup>	× <sup>Note 1</sup>	× <sup>Note 1</sup>	–	–	P25	P26	P27
Other than above													Setting prohibited		

(b) P60 to P62, P55 to P57

ASIM0		CSIM1 CSIM2			PM60 PM55	P60 P55	PM61 PM56	P61 P56	PM62 PM57	P62 P57	First Bit	Shift Clock	P60/SI1 P55/SI2 Pin Function	P61/SO1 P56/SO2 Pin Function	P62/ $\overline{\text{SCK1}}$ P57/ $\overline{\text{SCK2}}$ Pin Function
TXE0	RXE0	CSIE1	SCL11	SCL10 SCL21 SCL20											
×	×	0	×	×	× <sup>Note 1</sup>	× <sup>Note 1</sup>	× <sup>Note 1</sup>	× <sup>Note 1</sup>	× <sup>Note 1</sup>	× <sup>Note 1</sup>	–	–	P60 P55	P61 P56	P62 P57
Other than above													Setting prohibited		

(2) 3-wire serial I/O mode

(a) SI0, SO0,  $\overline{\text{SCK0}}$

ASIM0		CSIM0			PM25	P25	PM26	P26	PM27	P27	First Bit	Shift Clock	P25/SI0/RxD0 Pin Function	P26/SO0/TxD0 Pin Function	P27/ $\overline{\text{SCK0}}$ /ASCK0 Pin Function
TXE0	RXE0	CSIE0	SCL01	SCL00											
0	0	1	0	0	1 <sup>Note 2</sup>	× <sup>Note 2</sup>	0	0	1	×	MSB	External clock	SI0 <sup>Note 2</sup>	SO0 (CMOS output)	$\overline{\text{SCK0}}$ input
			<sup>Note 3</sup>	<sup>Note 3</sup>				0	0		Internal clock	$\overline{\text{SCK0}}$ output			
Other than above													Setting prohibited		

(b) SI1, SO1,  $\overline{\text{SCK1}}$

ASIM0		CSIM1			PM60	P60	PM61	P61	PM62	P62	First Bit	Shift Clock	P60/SI1 Pin Function	P60/SO1 Pin Function	P62/ $\overline{\text{SCK1}}$ Pin Function
TXE0	RXE0	CSIE1	SCL11	SCL10											
×	×	1	0	0	1 <sup>Note 2</sup>	× <sup>Note 2</sup>	0	0	1	×	MSB	External clock	SI1 <sup>Note 2</sup>	SO1 (CMOS output)	$\overline{\text{SCK1}}$ input
			<sup>Note 3</sup>	<sup>Note 3</sup>				0	0		Internal clock	$\overline{\text{SCK1}}$ output			
Other than above													Setting prohibited		

(c) SI2, SO2,  $\overline{\text{SCK2}}$

ASIM0		CSIM2			PM55	P55	PM56	P56	PM57	P57	First Bit	Shift Clock	P55/SI2 Pin Function	P56/SO2 Pin Function	P57/ $\overline{\text{SCK2}}$ Pin Function
TXE0	RXE0	CSIE2	SCL21	SCL20											
×	×	1	0	0	1 <sup>Note 2</sup>	× <sup>Note 2</sup>	1	0	1	×	MSB	External clock	SI2 <sup>Note 2</sup>	SO2 (CMOS output)	$\overline{\text{SCK2}}$ input
			<sup>Note 3</sup>	<sup>Note 3</sup>				0			Internal clock	$\overline{\text{SCK2}}$ output			
Other than above													Setting prohibited		

**Notes** 1. These pins can be used for port functions.

2. When only transmission is used, these pins can be used as P25, P60, P55 (CMOS I/O).

3. Refer to serial operation mode registers 0, 1, and 2 (CSIM0, CSIM1, and CSIM2).

**Remark** ×: don't care

## 12.4 Operation of Serial Interface

The serial interface operates in the following two modes.

- Operation stop mode
- 3-wire serial I/O mode

### 12.4.1 Operation stop mode

In the operation stop mode, the power consumption can be reduced because serial transfer is not executed.

Because serial I/O shift register n (SIO<sub>n</sub>) does not perform the shift operation, this register can be used as a normal 8-bit register.

In this mode, the SIn, SO<sub>n</sub>, and  $\overline{\text{SCKn}}$  pins can be used as normal I/O port pins.

#### (1) Register setting

The operation stop mode is set by serial operation mode register n (CSIM<sub>n</sub>).

CSIM<sub>n</sub> is set using a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$  input sets CSIM<sub>n</sub> to 00H.

#### (a) Format of serial operation mode register n (CSIM<sub>n</sub>)

Symbol	<7>	6	5	4	3	2	1	0	Address	After reset	R/W
CSIM <sub>n</sub>	CSIE <sub>n</sub>	0	0	0	0	MODE <sub>n</sub>	SCL <sub>n1</sub>	SCL <sub>n0</sub>	FF90H, FF91H	00H	R/W

CSIE <sub>n</sub>	Enables or disables operation of SIO <sub>n</sub>		
	Shift register operation	Serial counter	Port
0	Stopped	Cleared	Port function <sup>Note</sup>
1	Enabled	Count operation enabled	Serial function + port function

**Note** The pins connected to SIn, SO<sub>n</sub>, and  $\overline{\text{SCKn}}$  can be used as port pins when CSIE<sub>n</sub> = 0 (when the SIO<sub>n</sub> operation is stopped).

**Remark** n = 0 to 2

### 12.4.2 3-wire serial I/O mode

The 3-wire serial I/O mode is useful for connecting a peripheral I/O or display controller having a clocked serial interface.

Communication is established by using three lines: serial clock ( $\overline{SCKn}$ ), serial output (SOn), and serial input (SIn).

#### (1) Register setting

The 3-wire serial I/O mode is set by serial operation mode register n (CSIMn).

CSIMn is set using a 1-bit or 8-bit memory manipulation instruction.

$\overline{RESET}$  input sets CSIMn to 00H.

#### (a) Format of serial operation mode register n

Symbol	<7>	6	5	4	3	2	1	0	Address	After reset	R/W
CSIMn	CSIEn	0	0	0	0	MODEn	SCLn1	SCLn0	FF90H, FF91H	00H	R/W

CSIEn	Enables or disables operation of SION		
	Shift register operation	Serial counter	Port
0	Stopped	Cleared	Port function <sup>Note</sup>
1	Enabled	Count operation enabled	Serial function + port function

MODEn	Transfer operation mode flag		
	Operation mode	Transfer start trigger	SOn output
0	Transmit or transmit/receive mode	SION write	Normal output
1	Receive mode	SION read	Fixed to low level

SCLn1	SCLn0	Clock selection
0	0	External clock input to $\overline{SCKn}$ pin
0	1	$f_{xx}/8$ (1.56 MHz)
1	0	$f_{xx}/16$ (781 kHz)
1	1	$f_{xx}/32$ (391 kHz)

**Note** The pins connected to SIn, SOn, and  $\overline{SCKn}$  can be used as port pins when CSIEn = 0 (when the SION operation is stopped).

- Remarks**
1.  $f_{xx}$ : Main system clock frequency
  2. The values in parentheses are valid for operation when  $f_{xx}$  is 12.5 MHz.
  3.  $n = 0$  or  $1$

**(b) Format of serial operation mode register 2**

Symbol	<7>	6	5	4	3	2	1	0	Address	After reset	R/W
CSIM2	CSIE2	0	0	0	0	MODE2	SCL21	SCL20	FF90H	00H	R/W

CSIE2	Enables or disables operation of SIO2		
	Shift register operation	Serial counter	Port
0	Stopped	Cleared	Port function <sup>Note</sup>
1	Enabled	Count operation enabled	Serial function + port function

MODE2	Transfer operation mode flag		
	Operation mode	Transfer start trigger	SO2 output
0	Transmit or transmit/receive mode	SIO2 write	Normal output
1	Receive mode	SIO2 read	Fixed to low level

SCL21	SCL20	Clock selection
0	0	External clock input to $\overline{\text{SCK2}}$ pin
0	1	$f_{xx}/32$ (131 MHz)
1	0	$f_{xx}/64$ (65.5 kHz)
1	1	$f_{xx}/128$ (32.7 kHz)

**Note** The pins connected to SI2, SO2, and  $\overline{\text{SCK2}}$  can be used as port pins when CSIE2 = 0 (when the SIO2 operation is stopped).

**Caution** Because the  $\overline{\text{SCK2}}$  pin of serial interface 2 (SIO2) is an N-ch open-drain pin, the clock output from this pin does not have a duty factor of 50% if the internal clock is selected.

The set values listed above are for clocks that can be used when a pull-up resistor of 10 kΩ is connected at  $f_{xx} = 4.194$  MHz.

Under any other conditions, or if the wiring capacitance of the board differs even when the above conditions are satisfied, the operation may not be performed correctly even if the above clock is selected. Be sure to perform evaluation before selecting a clock.

**Remarks** 1.  $f_{xx}$ : Main system clock frequency

★ 2. The values in parentheses are valid for operation when  $f_{xx}$  is 4.194 MHz.

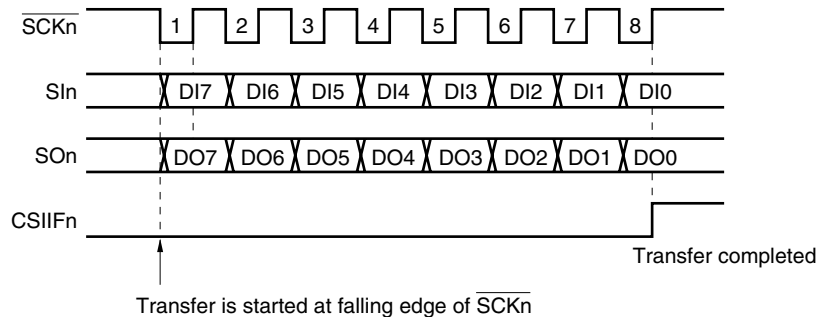
**(2) Communication operation**

In the three-wire serial I/O mode, data is transmitted or received in 8-bit units. Each bit of the data is transmitted or received in synchronization with the serial clock.

The shift operation of serial I/O shift register n (SIO<sub>n</sub>) is performed in synchronization with the falling of the serial clock ( $\overline{\text{SCKn}}$ ). The transmit data is retained by the SIO<sub>n</sub> latch and output from the SIO<sub>n</sub> pin. At the rising edge of  $\overline{\text{SCKn}}$ , the receive data input to the SIO<sub>n</sub> pin is latched to SIO<sub>n</sub>.

When transfer of 8-bit data has been completed, SIO<sub>n</sub> automatically stops its operation, and an interrupt request flag (CSIF<sub>n</sub>) is set.

**Figure 12-5. Timing in 3-Wire Serial I/O Mode**



**Remark** n = 0 to 2

**(3) Transfer start**

Serial transfer is started when data is assigned to (or read from) serial I/O shift register n (SIO<sub>n</sub>) if the following two conditions are satisfied.

- Operation control bit of SIO<sub>n</sub> (CSIE<sub>n</sub>) = 1
- If the internal serial clock is stopped or  $\overline{\text{SCKn}}$  is high after 8-bit serial transfer
- Transmit or transmit/receive mode  
Transfer is started if SIO<sub>n</sub> is written when CSIE<sub>n</sub> = 1 and MODE<sub>n</sub> = 0.
- Receive mode  
Transfer is started if SIO<sub>n</sub> is read when CSIE<sub>n</sub> = 1 and MODE<sub>n</sub> = 1.

**Caution** Transfer is not started even if CSIE<sub>n</sub> is set to 1 after data has been written to SIO<sub>n</sub>.

Serial transfer is automatically stopped and an interrupt request flag (CSIF<sub>n</sub>) is set when 8-bit transfer has been completed.

**Remark** n = 0 to 2

## 12.5 Functions of Serial Interface 2 (SIO2)

- (1) The SO2 and  $\overline{\text{SCK2}}$  pins of serial interface 2 (SIO2) are N-ch open-drain.
- (2) The internal serial clock can be selected from fxx/32, fxx/64, and fxx/128.
- (3) In the  $\mu\text{PD784975A}$ , use of a pull-up resistor can be specified for the SI2, SO2, and  $\overline{\text{SCK2}}$  pins in 1-bit units using a mask option. In the  $\mu\text{PD784976A}$ , pull-up resistors are not provided (refer to **CHAPTER 4 4.2.5 Port 5**).
- (4) When using the P55 to P57 pins as serial pins, set the port 5 mode register (PM5) to input mode (set bits 5 to 7 (PM55 to PM57) of PM5 to 1).
  - For serial interface 0, 1 (SIO0, SIO1)  
When using the SIO0, SIO1,  $\overline{\text{SCK0}}$ , and  $\overline{\text{SCK1}}$  pins as output pins, set the port 2 mode register (PM2) and port 6 mode register (PM6) to output mode (refer to **Table 4-2** in **CHAPTER 4**).

## 12.6 Cautions on Using Serial Interface 2 (SIO2)

- (1) When using the P55 to P57 pins as serial pins, set the port 5 mode register (PM5) to input mode (set bits 5 to 7 (PM55 to PM57) of PM5 to 1).

- For serial interface 0, 1 (SIO0, SIO1)  
When using the SIO0, SIO1,  $\overline{\text{SCK0}}$ , and  $\overline{\text{SCK1}}$  pins as output pins, set PM2 and PM6 to output mode (refer to **CHAPTER 4 Table 4-2**).

If PM5 is set to output mode when using the SO2 and  $\overline{\text{SCK2}}$  pins as output pins, output signals from the peripheral evaluation chip and CPU evaluation chip conflict during emulation. Consequently, the reliability of the chip may be degraded.

- (2) When using the SO2 and  $\overline{\text{SCK2}}$  pins as output pins when operation is started or stopped, set the pins using the following procedure.

<1> When operation is started

- Set the port 5 mode register (PM5) to input mode (set bits 6 and 7 (PM56 and PM57) of PM5 to 1).
- Write 0 to the output latch.
- Set bits 0 and 1 (SCL20 and SCL21) of serial operation mode register 2 (CSIM2) to other than "0, 0" (in the case of the  $\overline{\text{SCK2}}$  pin)
- Enable serial operation (set bit 7 of CSIM2 (CSIE2) to 1).

At this time, the output buffer is turned on, and serial data (in case of the SO2 pin) and the serial clock (in the case of the  $\overline{\text{SCK2}}$  pin) enter a wait state.

<2> When operation is stopped

- Disable serial operation (set bit 7 of CSIM2 (CSIE2) to 0).  
At this time, the output buffer is turned off and is in input port mode.
- Set bits 6 and 7 (PM56 and PM57) of PM5 to 0, 0 to set output mode.

## CHAPTER 13 ASYNCHRONOUS SERIAL INTERFACE

### 13.1 Functions of Asynchronous Serial Interface

The asynchronous serial interface (UART) offers the following two modes.

- Operation stop mode
- Asynchronous serial interface (UART) mode (with pin switching function)

#### (1) Operation stop mode

This mode is used when serial transfer is not performed to reduce the power consumption.

In operation stop mode, P25/RxD0/SI0, P26/TxD0/SO0, and P27/ASCK0/ $\overline{\text{SCK0}}$  can be used as a general-purpose input port or 3-wire serial interface 0 (SIO0).

#### (2) Asynchronous serial interface (UART) mode (with pin switching function)

This mode is used to send and receive 1-byte data that follows the start bit, and supports full-duplex transmission. A UART-dedicated baud rate generator is provided on-chip, enabling transmission at any baud rate within a broad range.

The baud rate can also be defined by dividing the input clock to the ASCK0 pin.

The asynchronous serial interface (UART) and 3-wire serial interface 0 (SIO0) cannot be used at the same time because they share pins.

These modes can be switched by setting asynchronous serial interface mode register 0 (ASIM0) and serial operation mode register 0 (CSIM0) (refer to **Table 13-1**).

**Table 13-1. Switching Asynchronous Serial Interface Mode and 3-Wire Serial I/O Mode**

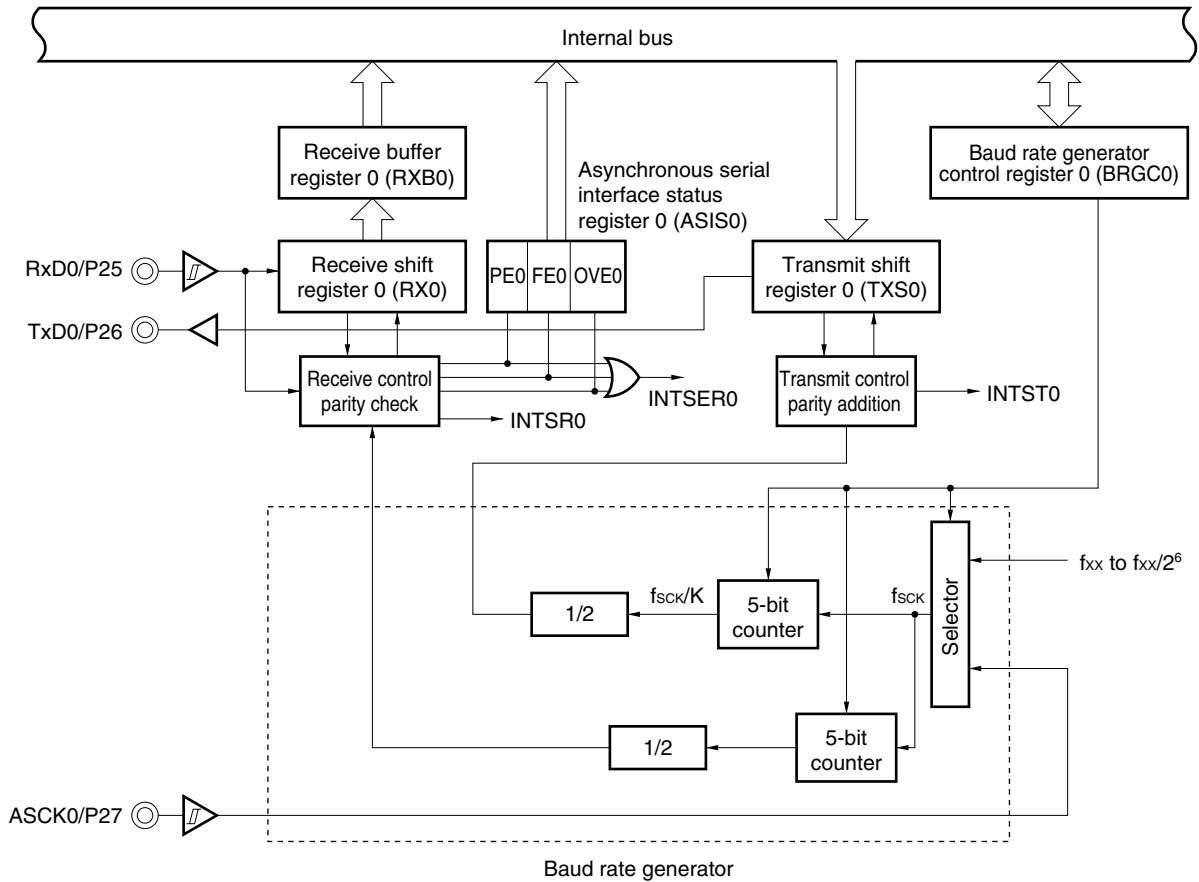
ASIM0		CSIM0	Operation Mode Selection
Bit 7 (TXE0)	Bit 6 (RXE0)	Bit 7 (CSIE0)	
0	0	0	Operation stop mode
0	0	1	3-wire serial I/O 0 (SIO0) mode
0	1	0	Asynchronous serial interface (UART) mode
1	0	0	
1	1	0	
Other than above			Setting prohibited

**Caution** Pins that are not switched can be used as port pins.



Figure 13-1 shows a block diagram of the asynchronous serial interface (UART).

Figure 13-1. Block Diagram of Asynchronous Serial Interface (UART)



## 13.2 Configuration of Asynchronous Serial Interface

The asynchronous serial interface includes the following hardware.

**Table 13-2. Configuration of Asynchronous Serial Interface**

Item	Configuration
Registers	Transmit shift register 0 (TXS0) Receive shift register 0 (RX0) Receive buffer register 0 (RXB0)
Control registers	Asynchronous serial interface mode register 0 (ASIM0) Asynchronous serial interface status register 0 (ASIS0) Baud rate generator control register 0 (BRGC0)

### (1) Transmit shift register 0 (TXS0)

This register is used to set transmit data. Data written to TXS0 is sent as serial data. If a data length of 7 bits is specified, bits 0 to 6 of the data written to TXS0 are transferred as transmit data. Transmission is started by writing data to TXS0. TX0 can be written with an 8-bit memory manipulation instruction, but cannot be read. RESET input sets TXS0 to FFH.

**Caution Do not write to TXS0 during transmission. TXS0 and receive buffer register 0 (RXB0) are allocated to the same address. Therefore, attempting to read TXS0 will result in reading the values of RXB0.**

### (2) Receive shift register 0 (RX0)

This register is used to convert serial data input to the RxD0 pin to parallel data. Receive data is transferred to the receive buffer register 0 (RXB0) one byte at a time as it is received. RX0 cannot be directly manipulated by program.

### (3) Receive buffer register 0 (RXB0)

This register is used to hold receive data. Each time one byte of data is received, new receive data is transferred from the receive shift register 0 (RX0). If a data length of 7 bits is specified, receive data is transferred to bits 0 to 6 of RXB0, and the MSB of RXB0 always becomes 0. RXB0 can be read by an 8-bit memory manipulation instruction, but cannot be written. RESET input sets RXB0 to FFH.

**Caution Be sure to read receive buffer register 0 (RXB0) even when a receive error occurs; otherwise an overrun error occurs next time data is received causing a receive error.**

### (4) Transmission control circuit

This circuit controls transmit operations such as the addition of a start bit, parity bit, and stop bit(s) to data written to transmit shift register 0 (TXS0), according to the contents set to the asynchronous serial interface mode register 0 (ASIM0).

**(5) Reception control circuit**

This circuit controls reception according to the contents set to the asynchronous serial interface mode register 0 (ASIM0). It also performs error check for parity errors, etc., during reception and transmission. If it detects an error, it sets a value corresponding to the nature of the error in the asynchronous serial interface status register 0 (ASIS0).

**13.3 Asynchronous Serial Interface Control Registers**

The following three types of registers control the asynchronous serial interface (UART).

- Asynchronous serial interface mode register 0 (ASIM0)
- Asynchronous serial interface status register 0 (ASIS0)
- Baud rate generator control register 0 (BRGC0)

**(1) Asynchronous serial interface mode register 0 (ASIM0)**

ASIM0 is an 8-bit register that controls serial transfer using the asynchronous serial interface (UART).

ASIM0 is set using a 1-bit or 8-bit memory manipulation operation.

$\overline{\text{RESET}}$  input sets ASIM0 to 00H.

Figure 13-2 shows the format of ASIM0.

**Figure 13-2. Format of Asynchronous Serial Interface Mode Register 0 (ASIM0)**

Address: 0FF70H After reset: 00H R/W

Symbol	<7>	<6>	5	4	3	2	1	0
ASIM0	TXE0	RXE0	PS01	PS00	CL0	SL0	ISRM0	0

TXE0	RXE0	Operation mode	RxD0/P25/SI0 pin function	TxD0/P26/SO0 pin function
0	0	Operation stop	Port function (P25)/ serial function (RxD0)	Port function (P26)/ serial function (TxD0)
0	1	UART mode (Receive only)	Serial function (RxD0)	Port function (P26)/ serial function (TxD0)
1	0	UART mode (Transmit only)	Port function (P25)/ Serial function (RxD0)	Serial function (TxD0)
1	1	UART mode (Transmit/Receive)	Serial function (RxD0)	Serial function (TxD0)

PS01	PS00	Parity bit specification
0	0	No parity
0	1	Transmit: 0 parity Receive: Parity error not generated
1	0	Odd parity
1	1	Even parity

CL0	Transmit data character length specification
0	7 bits
1	8 bits

SL0	Transmit data stop bit length specification
0	1 bit
1	2 bits

ISRM0	Receive completion interrupt control at error occurrence
0	Generate receive completion interrupt when error occurs
1	Do not generate receive completion interrupt when error occurs

- Cautions**
1. Be sure to set bit 0 of ASIM0 to 0.
  2. Do not switch the operation mode until the current serial transmit/receive operation has stopped.

**(2) Asynchronous serial interface status register 0 (ASIS0)**

ASIS0 is a register used to display the type of error when a receive error occurs.

ASIS0 can be read by a 1-bit or 8-bit memory manipulation instructions.

$\overline{\text{RESET}}$  input sets ASIS0 to 00H.

**Figure 13-3. Format of Asynchronous Serial Interface Status Register 0 (ASIS0)**

Address: 0FF72H After reset: 00H R

Symbol	7	6	5	4	3	<2>	<1>	<0>
ASIS0	0	0	0	0	0	PE0 <sup>Note 1</sup>	FE0 <sup>Note 2</sup>	OVE0 <sup>Note 3</sup>

PE0	Parity error flag
0	Parity error not generated
1	Parity error generated (when data is read from the receive buffer, or when a 1-byte data is received)

FE0	Framing error flag
0	Framing error not generated
1	Framing error generated <sup>Note 4</sup> (when stop bit(s) is not detected)

OVE0	Overrun error flag
0	Overrun error not generated (when the next receive operation is completed before the CPU reads the receive data from RXB0)
1	Overrun error generated <sup>Note 5</sup> (when data is read from receive buffer register)

- Notes**
1. The parity error flag is cleared if the subsequent parity bit detection is correctly performed.
  2. Only the first stop bit in the receive data is detected, regardless of the number of stop bits.
  3. The contents of receive shift register 0 (RX0) are transferred to receive buffer register 0 (RXB0) each time one character is received. When an overrun error occurs, the subsequent receive data is overwritten to RXB0. Consequently, the data read from RXB0 is the one received after the overwritten data.
  4. Even if the stop bit length has been set to 2 bits with bit 2 (SL0) of the asynchronous serial interface mode register 0 (ASIM0), stop bit detection during reception is only 1 bit.
  5. Be sure to read RXB0 when an overrun error occurs.  
An overrun error is generated each time data is received until RXB0 is read.

**Caution** Be sure to set bits 3 to 7 of ASIS0 to 0.

**(3) Baud rate generator control register 0 (BRGC0)**

BRGC0 is a register used to set the serial clock of the asynchronous serial interface.

BRGC0 is set using an 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$  input sets BRGC0 to 00H.

**Figure 13-4. Format of Baud Rate Generator Control Register 0 (BRGC0)**

Address: 0FF76H After reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
BRGC0	0	TPS02	TPS01	TPS00	MDL03	MDL02	MDL01	MDL00

TPS02	TPS01	TPS00	5-bit counter source clock selection	m
0	0	0	P27/ASCK0	–
0	0	1	$f_{xx}$ (12.5 MHz)	0
0	1	0	$f_{xx}/2$ (6.25 MHz)	1
0	1	1	$f_{xx}/4$ (3.13 MHz)	2
1	0	0	$f_{xx}/8$ (1.56 MHz)	3
1	0	1	$f_{xx}/16$ (781 kHz)	4
1	1	0	$f_{xx}/32$ (391 kHz)	5
1	1	1	$f_{xx}/64$ (195 kHz)	6

MDL03	MDL02	MDL01	TPS00	Baud rate generator input clock selection	k
0	0	0	0	$f_{sck}/16$	0
0	0	0	1	$f_{sck}/17$	1
0	0	1	0	$f_{sck}/18$	2
0	0	1	1	$f_{sck}/19$	3
0	1	0	0	$f_{sck}/20$	4
0	1	0	1	$f_{sck}/21$	5
0	1	1	0	$f_{sck}/22$	6
0	1	1	1	$f_{sck}/23$	7
1	0	0	0	$f_{sck}/24$	8
1	0	0	1	$f_{sck}/25$	9
1	0	1	0	$f_{sck}/26$	10
1	0	1	1	$f_{sck}/27$	11
1	1	0	0	$f_{sck}/28$	12
1	1	0	1	$f_{sck}/29$	13
1	1	1	0	$f_{sck}/30$	14
1	1	1	1	Setting prohibited	–

**Caution** If a write operation to BRGC0 is performed during communication, the baud rate generator output will become garbled and normal communication will not be achieved. Therefore, do not perform write operations to BRGC0 during communication.

- Remarks**
1. f<sub>sck</sub>: Source clock of 5-bit counter
  2. m: Value set by TPS00 to TPS02 (0 ≤ m ≤ 6)
  3. k: Value set by MDL00 to MDL03 (0 ≤ k ≤ 14)

★

**Table 13-3. Serial Interface Operation Mode Settings**

**(1) Operation stopped mode**

ASIM0		CSIM0			PM25	P25	PM26	P26	PM27	P27	First Bit	Shift Clock	P25/SI0/RxD0 Pin Function	P26/SO0/TxD0 Pin Function	P27/ $\overline{\text{SCK0}}$ /ASCK0 Pin Function
TXE0	RXE0	CSIE0	SCL01	SCL00											
0	0	0	×	×	×	×	×	×	×	×	–	–	P25	P26	P27
Other than above													Setting prohibited		

**(2) Asynchronous serial interface mode**

ASIM0		CSIM0			PM25	P25	PM26	P26	PM27	P27	First Bit	Shift Clock	P25/SI0/RxD0 Pin Function	P26/SO0/TxD0 Pin Function	P27/ $\overline{\text{SCK0}}$ /ASCK0 Pin Function
TXE0	RXE0	CSIE0	SCL01	SCL00											
1	0	0	×	×	×	×	0	0	1	×	LSB	External clock	P25	TxD0 (CMOS output)	ASCK0 input
									×	×		Internal clock			P27
0	1				1	×	×	×	1	×		External clock	RxD	P26	ASCK0 input
									×	×		Internal clock			P27
1	1						0	0	1	×		External clock	TxD0 (CMOS output)	ASCK0 input	
									×	×		Internal clock			P27
Other than above													Setting prohibited		

- Notes**
1. These pins can be used for port functions.
  2. Refer to 13.4.2 Asynchronous serial interface (UART) mode (2) Communication operation (c) transmission.

### 13.4 Operation of Asynchronous Serial Interface

The three types of operation modes of asynchronous serial interface (UART) are explained below.

#### 13.4.1 Operation stop mode

Serial transfer cannot be performed in the operation stop mode, resulting in reduced power consumption. Moreover, in the operation stop mode, pins can be used as regular ports.

##### (1) Register setting

Setting of the operation stop mode is done with asynchronous serial interface mode register 0 (ASIM0). ASIM0 is set using a 1-bit or 8-bit memory manipulation instruction.  $\overline{\text{RESET}}$  input sets ASIM0 to 00H.

Address: 0FF70H After reset: 00H R/W

Symbol	<7>	<6>	5	4	3	2	1	0
ASIM0	TXE0	RXE0	PS01	PS00	CL0	SL0	ISRM0	0

TXE0	RXE0	Operation mode	RxD0/P25/SI0 pin function	TxD0/P26/SO0 pin function
0	0	Operation stop	Port function (P25)/ serial function (RxD0)	Port function (P26)/ serial function (TxD0)
0	1	UART mode (Receive only)	Serial function (RxD0)	Port function (P26)/ serial function (TxD0)
1	0	UART mode (Transmit only)	Port function (P25)/ Serial function (RxD0)	Serial function (TxD0)
1	1	UART mode (Transmit/Receive)	Serial function (RxD0)	Serial function (TxD0)

- Cautions**
1. Be sure to set bit 0 of ASIM0 to 0.
  2. Do not switch the operation mode until the current serial transmit/receive operation has stopped.



**13.4.2 Asynchronous serial interface (UART) mode**

This mode is used to transmit and receive the 1-byte data following the start bit. It supports full-duplex operation.

A UART-dedicated baud rate generator is incorporated enabling communication using any baud rate within a large range.

The MIDI standard's baud rate (31.25 kbps) can be used utilizing the UART-dedicated baud rate generator.

**(1) Register setting**

The UART mode is set with asynchronous serial interface mode register 0 (ASIM0), asynchronous serial interface status register 0 (ASIS0), and baud rate generator control register 0 (BRGC0).

**(a) Asynchronous serial interface mode register 0 (ASIM0)**

ASIM0 can be set using a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$  input sets ASIM0 to 00H.

Address: 0FF70H After reset: 00H R/W

Symbol	<7>	<6>	5	4	3	2	1	0
ASIM0	TXE0	RXE0	PS01	PS00	CL0	SL0	ISRM0	0

TXE0	RXE0	Operation mode	RxD0/P25/SI0 pin function	TxD0/P26/SO0 pin function
0	0	Operation stop	Port function (P25)/ serial function (RxD0)	Port function (P26)/ serial function (TxD0)
0	1	UART mode (Receive only)	Serial function (RxD0)	Port function (P26)/ serial function (TxD0)
1	0	UART mode (Transmit only)	Port function (P25)/ Serial function (RxD0)	Serial function (TxD0)
1	1	UART mode (Transmit/Receive)	Serial function (RxD0)	Serial function (TxD0)

PS01	PS00	Parity bit specification
0	0	No parity
0	1	Transmit: 0 parity Receive: Parity error not generated
1	0	Odd parity
1	1	Even parity

CL0	Transmit data character length specification
0	7 bits
1	8 bits

SL0	Transmit data stop bit length specification
0	1 bit
1	2 bits

ISRM0	Receive completion interrupt control at error occurrence
0	Generate receive completion interrupt when error occurs
1	Do not generate receive completion interrupt when error occurs

- Cautions**
1. Be sure to set bit 0 of ASIM0 to 0.
  2. Do not switch the operation mode until the current serial transmit/receive operation has stopped.

**(b) Asynchronous serial interface status register 0 (ASIS0)**

ASIS0 is a register used to display the type of error when a receive error occurs.

ASIS0 can be read using a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$  input sets ASIS0 to 00H.

Address: 0FF72H After reset: 00H R

Symbol	7	6	5	4	3	<2>	<1>	<0>
ASIS0	0	0	0	0	0	PE0 <sup>Note 1</sup>	FE0 <sup>Note 2</sup>	OVE0 <sup>Note 3</sup>

PE0	Parity error flag
0	Parity error not generated
1	Parity error generated (when data is read from the receive buffer, or when a 1-byte data is received)

FE0	Framing error flag
0	Framing error not generated
1	Framing error generated <sup>Note 4</sup> (when stop bit(s) is not detected)

OVE0	Overrun error flag
0	Overrun error not generated (When the next receive operation is completed before the CPU reads the receive data from RXB0)
1	Overrun error generated <sup>Note 5</sup> (when data is read from receive buffer register)

- Notes**
1. The parity error flag is cleared if the subsequent parity bit detection is correctly performed.
  2. Only the first stop bit in the receive data is detected, regardless of the number of stop bits.
  3. The contents of receive shift register 0 (RX0) are transferred to the receive buffer register 0 (RXB0) each time one character is received. When an overrun error occurs, the subsequent receive data is overwritten to RXB0. Consequently, the data read from RXB0 is the one received after the overwritten data.
  4. Even if the stop bit length has been set to 2 bits with bit 2 (SL0) of the asynchronous serial interface mode register 0 (ASIM0), stop bit detection during reception is only 1 bit.
  5. Be sure to read RXB0 when an overrun error occurs.  
An overrun error is generated each time data is received until RXB0 is read.

**Caution** Be sure to set bits 3 to 7 of ASIS0 to 0.

**(c) Baud rate generator control register 0 (BRGC0)**

BRGC0 is set using an 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$  input sets BRGC0 to 00H.

Address: 0FF76H After reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
BRGC0	0	TPS02	TPS01	TPS00	MDL03	MDL02	MDL01	MDL00

TPS02	TPS01	TPS00	5-bit counter source clock selection	m
0	0	0	P27/ASCK0	–
0	0	1	$f_{xx}$ (12.5 MHz)	0
0	1	0	$f_{xx}/2$ (6.25 MHz)	1
0	1	1	$f_{xx}/4$ (3.13 MHz)	2
1	0	0	$f_{xx}/8$ (1.56 MHz)	3
1	0	1	$f_{xx}/16$ (781 kHz)	4
1	1	0	$f_{xx}/32$ (391 kHz)	5
1	1	1	$f_{xx}/64$ (195 kHz)	6

MDL03	MDL02	MDL01	TPS00	Baud rate generator input clock selection	k
0	0	0	0	$f_{sck}/16$	0
0	0	0	1	$f_{sck}/17$	1
0	0	1	0	$f_{sck}/18$	2
0	0	1	1	$f_{sck}/19$	3
0	1	0	0	$f_{sck}/20$	4
0	1	0	1	$f_{sck}/21$	5
0	1	1	0	$f_{sck}/22$	6
0	1	1	1	$f_{sck}/23$	7
1	0	0	0	$f_{sck}/24$	8
1	0	0	1	$f_{sck}/25$	9
1	0	1	0	$f_{sck}/26$	10
1	0	1	1	$f_{sck}/27$	11
1	1	0	0	$f_{sck}/28$	12
1	1	0	1	$f_{sck}/29$	13
1	1	1	0	$f_{sck}/30$	14
1	1	1	1	Setting prohibited	–

**Caution** If a write operation to BRGC0 is performed during communication, the baud rate generator output will become garbled and normal communication will not be achieved. Therefore, do not perform write operations to BRGC0 during communication.

- Remarks**
1.  $f_{sck}$ : Source clock of 5-bit counter
  2. m: Value set by TPS00 to TPS02 ( $0 \leq m \leq 6$ )
  3. k: Value set by MDL00 to MDL03 ( $0 \leq k \leq 14$ )

- **Generation of clock for baud rate generator**

**<1> When the values of bits 4 to 6 (TPS00 to TPS02) are set to “001B to 111B”**

The baud rate to generate the internal clock (f<sub>xx</sub>) is obtained from the following expression.

$$[\text{Baud rate}] = \frac{f_{xx}}{2^{m+1} \times (k + 16)}$$

f<sub>xx</sub>: Main system clock oscillation frequency

m: Value set in bits 4 to 6 (TPS00 to TPS02) of BRGC0 (0 ≤ m ≤ 6)

k: Value set in bits 0 to 3 (MDL00 to MDL03) of BRGC0 (0 ≤ k ≤ 14)

**<2> When the values of TPS00 to TPS02 are set to “000B”**

The baud rate generated from the external clock (ASCK0) is obtained from the following expression.

$$[\text{Baud rate}] = \frac{[\text{Frequency of ASCK0}]}{2(k + 16)}$$

k: Value set in bits 0 to 3 (MDL00 to MDL03) of BRGC0 (0 ≤ k ≤ 14)

The relation between the source clock of the 5-bit counter and the m value is shown in Table 13-4.

**Table 13-4. Relation Between 5-Bit Counter Source Clock and m Value**

TPS02	TPS01	TPS00	5-Bit Counter Source Clock Selection	m
0	0	0	P27/ASCK0	—
0	0	1	f <sub>xx</sub> (12.5 MHz)	0
0	1	0	f <sub>xx</sub> /2 (6.25 MHz)	1
0	1	1	f <sub>xx</sub> /4 (3.13 MHz)	2
1	0	0	f <sub>xx</sub> /8 (1.56 MHz)	3
1	0	1	f <sub>xx</sub> /16 (781 kHz)	4
1	1	0	f <sub>xx</sub> /32 (391 kHz)	5
1	1	1	f <sub>xx</sub> /64 (195 kHz)	6

• **Baud rate capacity error range**

The baud rate capacity range depends on the number of bits per frame and the counter division ratio [2 (k + 16)].

Table 13-5 shows the relation between the selection clock of the baud rate generator control register 0 (BRGC0) and the baud rate.

**Table 13-5. Relation Between BRCR0 Selection Clock and Baud Rate**

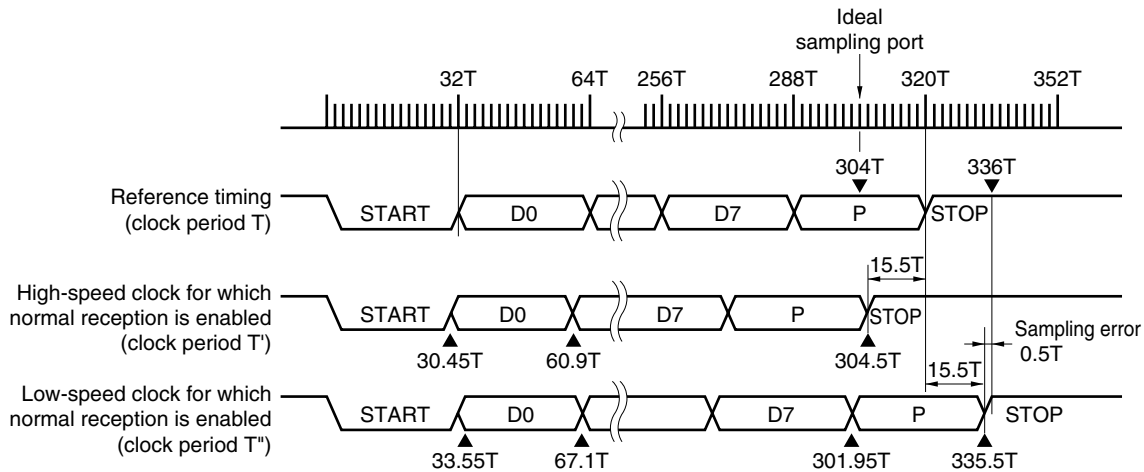
Baud Rate (bps)	f <sub>xx</sub> = 12.5 MHz		f <sub>xx</sub> = 6.00 MHz		f <sub>xx</sub> = 4.00 MHz	
	BRCR0 value	Error (%)	BRCR0 value	Error (%)	BRCR0 value	Error (%)
1,200	—	—	—	—	7AH	0.16
2,400	—	—	7AH	0.16	6AH	0.16
4,800	74H	1.73	6AH	0.16	5AH	0.16
9,600	64H	1.73	5AH	0.16	4AH	0.16
19,200	54H	1.73	4AH	0.16	3AH	0.16
31,250	49H	0.00	40H	0.00	30H	0.00
38,400	44H	1.73	3AH	0.16	2AH	0.16
76,800	34H	1.73	2AH	0.16	1AH	0.16
150 K	24H	1.73	1AH	0.16	—	—
300 K	14H	1.73	—	—	—	—

**Remark** f<sub>xx</sub>: Internal clock frequency

m: Value set in bits 4 to 6 (TPS00 to TPS02) of BRGC0 (0 ≤ m ≤ 6)

k: Value set in bits 0 to 3 (MDL00 to MDL03) of BRGC0 (0 ≤ k ≤ 14)

**Figure 13-5. Baud Rate Capacity Error Considering Sampling Errors (When k = 0)**



**Remark** T: 5-bit counter source clock period

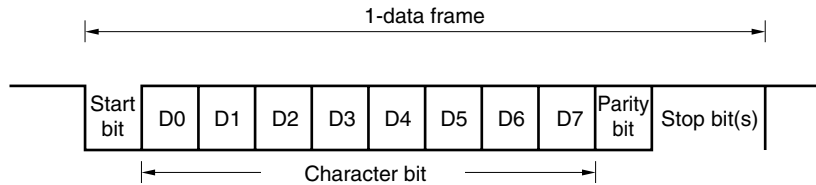
$$\text{Baud rate capacity error (k = 0)} = \frac{\pm 15.5}{320} \times 100 = 4.8438 (\%)$$

**(2) Communication operation****(a) Data format**

The transmit/receive data format consists of a start bit, character bits, and stop bit(s) forming character frames, as shown in Figure 13-6.

Specification of the character bit length inside data frames, selection of the parity, and selection of the stop bit length, are performed with the asynchronous serial interface mode register 0 (ASIM0).

**Figure 13-6. Format of Asynchronous Serial Interface Transmit/Receive Data**



- 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

If 7 bits has been selected as the number of character bits, only the lower 7 bits (bits 0 to 6) are valid. In the case of transmission, the highest bit (bit 7) is ignored. In the case of reception, the highest bit (bit 7) always becomes 0.

The setting of the serial transfer rate is performed with the asynchronous serial interface mode register 0 (ASIM0) and the baud rate generator control register 0 (BRGC0).

If a serial data reception error occurs, it is possible to determine the contents of the reception error by reading the status of the asynchronous serial interface status register 0 (ASIS0).

**(b) Parity types and operations**

Parity bits serve to detect bit errors in transmit data. Normally, the parity bit used on the transmit side and the receive side are of the same type. In the case of even parity and odd parity, it is possible to detect 1 bit (odd number) errors. In the case of 0 parity and no parity, errors cannot be detected.

**(i) Even parity**

- During transmission

Makes the number of “1”s in transmit data that includes the parity bit even. The value of the parity bit changes as follows.

If the number of “1” bits in transmit data is odd: 1  
if the number of “1” bits in transmit data is even: 0

- During reception

The number of “1” bits in receive data that includes the parity bit is counted, and if it is odd, a parity error occurs.

**(ii) Odd parity**

- During transmission

Odd parity is the reverse of even parity. It makes the number of “1”s in transmit data that includes the parity bit odd. The value of the parity bit changes as follows.

If the number of “1” bits in transmit data is odd: 0  
if the number of “1” bits in transmit data is even: 1

- During reception

The number of “1” bits in receive data is counted, and if it is even, a parity error occurs.

**(iii) 0 Parity**

During transmission, makes the parity bit “0”, regardless of the transmit data.

Parity bit check is not performed during reception. Therefore, no parity error occurs, regardless of whether the parity bit value is “0” or “1”.

**(iv) No parity**

No parity is appended to transmit data.

Transmit data is received assuming that it has no parity bit. No parity error can occur because there is no parity bit.



**(c) Transmission**

Transmission is begun by writing transmit data to the transmission shift register 0 (TXS0). The start bit, parity bit, and stop bit(s) are automatically added.

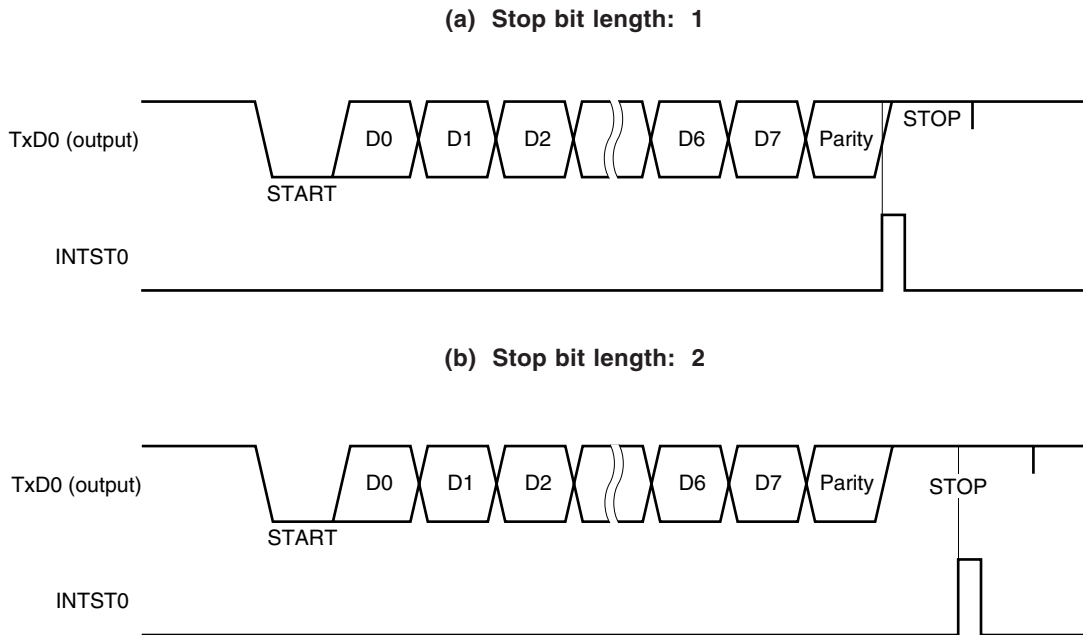
The contents of the transmit shift register 0 (TXS0) are shifted out upon transmission start, and when the transmit shift register 0 (TXS0) becomes empty, a transmit interrupt (INTST0) is generated.

**Caution** In the case of UART transmission, follow the procedure below for the first byte.

- <1> Set the port to input mode (PM26 = 1), and write 0 to the port latch.
- <2> Set bit 7 (TXE0) of the asynchronous serial interface mode register 0 (ASIM0) to 1 so as to enable transmission.
- <3> Set the port to output mode (PM26 = 0).
- <4> Write transmission data to TXS0 and start the transmit operation.

If the port is set to the output mode first, 0 will be output from the pins, which may cause malfunction.

**Figure 13-7. Asynchronous Serial Interface Transmit Completion Interrupt Timing**



**Caution** Do not write to the asynchronous serial interface mode register 0 (ASIM0) during transmission. If you write to the ASIM0 register during transmission, further transmission operations may become impossible (in this case, input  $\overline{\text{RESET}}$  to return to normal). Whether transmission is in progress or not can be judged by software, using the transmit completion interrupt (INTST0) or the interrupt request flag (STIF0) set by INTST0.

**(d) Reception**

When the RXE0 bit of the asynchronous serial interface mode register 0 (ASIM0) is set to 1, reception is enabled and sampling of the RxD0 pin input is performed.

Sampling of the RxD0 pin input is performed by the serial clock set in ASIM0.

When the RxD0 pin input becomes low level, the 5-bit counter of the port rate generator starts counting, and outputs the data sampling start timing signal when half the time of the set baud rate has elapsed. If the result of re-sampling the RxD0 pin input with this start timing signal is low level, the RxD0 pin input is perceived as the start bit, the 5-bit counter is initialized and begins counting, and data sampling is performed. When, following the start bit, character data, the parity bit, and one stop bit are detected, reception of one frame of data is completed.

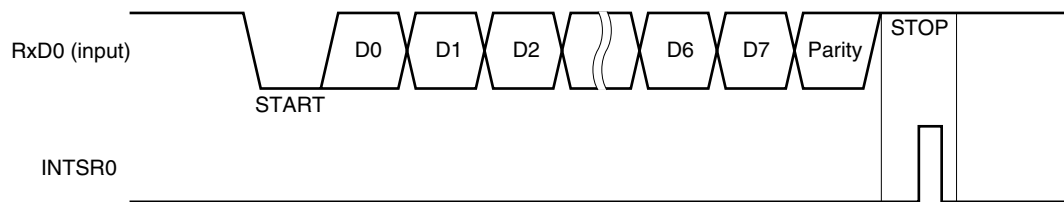
When reception of one frame of data is completed, the receive data in the shift register is transferred to the receive shift register (RXB0), and a receive completion interrupt (INTSR0) is generated.

Moreover, even if an error occurs, the receive data for which the error occurred is transferred to RXB0. If an error occurs, when bit 1 (ISRM0) of ASIM0 is cleared to 0, INTSR0 is generated. (refer to **Figure 13-9**).

When bit ISRM0 is set to 1, INTSR0 is not generated.

When bit RXE0 is reset to 0 during a receive operation, the receive operation is immediately stopped. At this time, the contents of RXB0 and ASIS0 remain unchanged, and INTSR0 and INTSER0 are not generated.

**Figure 13-8. Asynchronous Serial Interface Receive Completion Interrupt Timing**



**Caution** Even when a receive error occurs, be sure to read the receive buffer register 0 (RXB0). If RXB0 is not read, an overrun error will occur during reception of the next data, and the reception error status will continue indefinitely.

**(e) Receive error**

Errors that occur during reception are of three types: parity errors, framing errors, and overrun errors. As the data reception result error flag is set inside the asynchronous serial interface status register 0 (ASIS0), the receive error interrupt (INTSER0) is generated. A receive error interruption is generated before a receive end interrupt (INTSR0). Receive error causes are shown in Table 13-6.

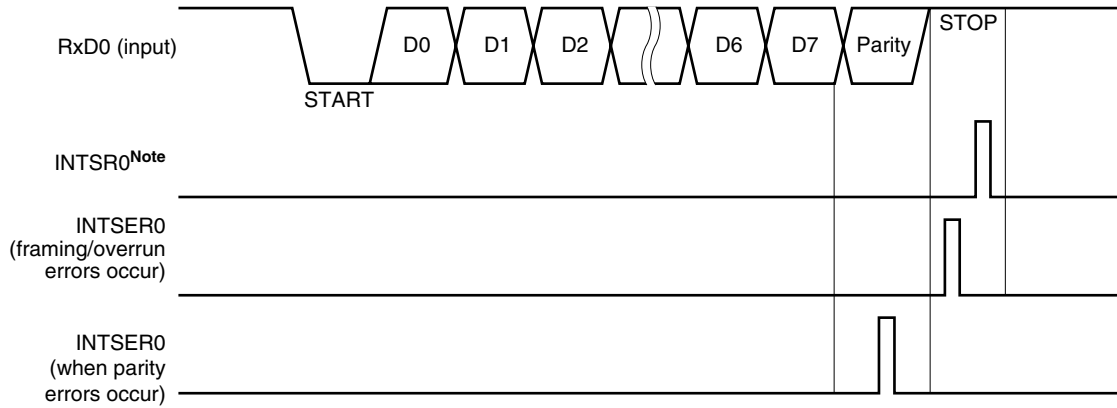
What type of error has occurred during reception can be detected by reading the contents of the asynchronous serial interface status register 0 (ASIS0) during processing of the receive error interrupt (INTSER0) (refer to **Table 13-6** and **Figure 13-9**).

The contents of ASISn are reset to 0 either when the receive buffer register 0 (RXB0) is read or when the next data is received (If the next data has an error, this error flag is set).

**Table 13-6. Receive Error Causes**

Receive Error	Cause	ASIS0
Parity error	Parity specified for transmission and parity of receive data don't match	04H
Framing error	Stop bit was not detected	02H
Overrun error	Next data reception was completed before data was read from the receive buffer register	01H

**Figure 13-9. Receive Error Timing**



**Note** If a receive error occurs, when bit ISRM0 is set (1), INTSR0 is not generated.

**Cautions 1.** The contents of ASIS0 are reset to 0 either when the receive buffer register 0 (RXB0) is read or when the next data is received. To find out the contents of the error, be sure to read ASIS0 before reading RXB0.

**2.** Be sure to read the receive buffer register 0 (RXB0) even when a receive error occurs. If RXB0 is not read, an overrun error will occur at reception of the next data, and the receive error status will continue indefinitely.

## CHAPTER 14 VFD CONTROLLER/DRIVER

### 14.1 Function of VFD Controller/Driver

The VFD controller/driver of the  $\mu$ PD784976A Subseries has the following functions.

- (1) Can output display signals (DMA operation) by automatically reading display data.
- (2) The pins not used for VFD display can be used as I/O port or output port pins (FIP16 to FIP47 pins only).
- (3) Luminance can be adjusted in 8 steps by display mode register 1 (DSPM1).
- (4) Hardware for key scan application
  - Generates an interrupt signal (INTKS) indicating key scan timing
  - Timing in which key scan data is output can be detected by key scan flag (KSF).
  - Whether key scan timing is inserted or not can be selected.
- (5) High-voltage output buffer that can directly drive VFD
- (6) FIP0 to FIP15 pins are connected to pull-down resistors. FIP16 to FIP47 pins can be connected to pull-down resistors using a mask option (mask ROM version only). The  $\mu$ PD78F4976A does not have pull-down resistors).

Of the 48 VFD output pins of the  $\mu$ PD784976A Subseries, FIP16 to FIP47 are multiplexed with port pins. FIP0 to FIP15 are dedicated VFD output pins.

FIP16 to FIP47 can be used as port pins when VFD display is disabled by bit 7 (DSPEN) of display mode register 0 (DSPM0). Even when VFD display is enabled, the VFD output pins not used for display signal output can be used as port pins.

**Table 14-1. VFD Output Pins and Multiplexed Port Pins**

VFD Pin Name	Multiplexed Port Name	I/O
FIP16 to FIP23	P70 to P77	I/O port
FIP24 to FIP31	P80 to P87	I/O port
FIP32 to FIP39	P90 to P97	I/O port
FIP40 to FIP47	P100 to P107	Output port

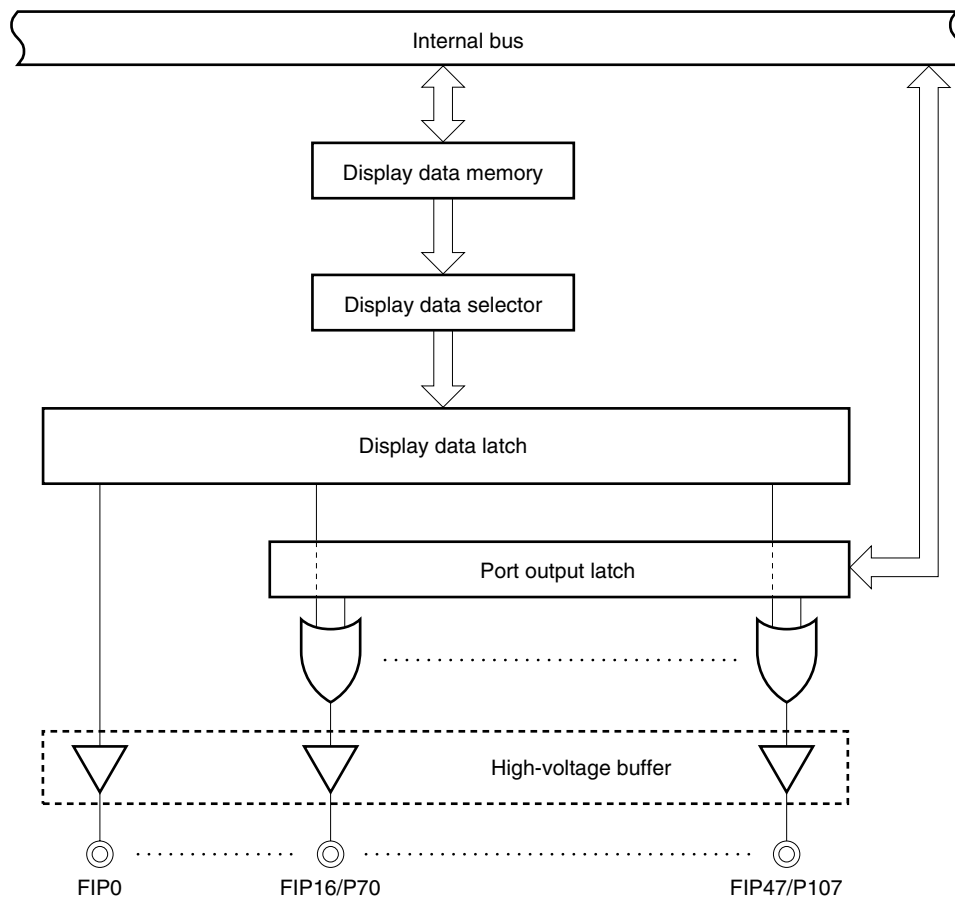
## 14.2 Configuration of VFD Controller/Driver

The VFD controller/driver includes the following hardware.

**Table 14-2. Configuration of VFD Controller/Driver**

Item	Configuration
Display	48
Control register	Display mode register 0 (DSPM0) Display mode register 1 (DSPM1) Display mode register 2 (DSPM2)

**Figure 14-1. Block Diagram of VFD Controller/Driver**



### 14.3 VFD Controller/Driver Control Registers

#### 14.3.1 Control registers

The following three types of registers control the VFD controller/driver.

- Display mode register 0 (DSPM0)
- Display mode register 1 (DSPM1)
- Display mode register 2 (DSPM2)

##### (1) Display mode register 0 (DSPM0)

DSPM0 performs the following setting.

- Enables or disables display
- Number of VFD output pins

DSPM0 is set using a 1-bit or 8-bit memory manipulation instruction.

RESET input sets DSPM0 to 10H.

**Figure 14-2. Format of Display Mode Register 0**

Symbol	<7>	6	5	4	3	2	1	0	Address	After reset	R/W
DSPM0	DSPEN	0	FOUT5	FOUT4	FOUT3	FOUT2	FOUT1	FOUT0	FFB0H	10H	R/W

DSPEN	Enables or disables VFD
0	Disables
1	Enables

FOUT5	FOUT4	FOUT3	FOUT2	FOUT1	FOUT0	Number of VFD output pins
0	1	0	1	1	1	17 to 24
0	1	1	1	1	1	25 to 32
1	0	0	1	1	1	33 to 40
1	0	1	1	1	1	41 to 48
Other than above						Setting prohibited

- Cautions**
1. Be sure to set bit 6 to 0.
  2. Do not write data to the bits other than DSPEN when bit 7 (DSPEN) is 1.
  3. Be sure to set the output latch of the multiplexed port of a pin used for VFD output to 0.

**(2) Display mode register 1 (DSPM1)**

DSPM1 performs the following setting.

- Blanking width of VFD output signal
- Number of display patterns

DSPM1 is set using a 1-bit or 8-bit memory manipulation instruction.

RESET input sets DSPM1 to 01H.

**Figure 14-3. Format of Display Mode Register 1 (DSPM1)**

Symbol	7	6	5	4	3	2	1	0	Address	After reset	R/W
DSPM1	FBLK2	FBLK1	FBLK0	FPAT4	FPAT3	FPAT2	FPAT1	FPAT0	FFB2H	01H	R/W

FBLK2	FBLK1	FBLK0	Blanking width of VFD output signal
0	0	0	1/16
0	0	1	2/16
0	1	0	4/16
0	1	1	6/16
1	0	0	8/16
1	0	1	10/16
1	1	0	12/16
1	1	1	14/16

FPAT4	FPAT3	FPAT2	FPAT1	FPAT0	Number of display patterns
0	0	0	0	1	2
0	0	0	1	0	3
0	0	0	1	1	4
0	0	1	0	0	5
0	0	1	0	1	6
0	0	1	1	0	7
0	0	1	1	1	8
0	1	0	0	0	9
0	1	0	0	1	10
0	1	0	1	0	11
0	1	0	1	1	12
0	1	1	0	0	13
0	1	1	0	1	14
0	1	1	1	0	15
0	1	1	1	1	16
Other than above					Setting prohibited

**Caution** Do not write data to display mode register 1 (DSPM1) when bit 7 (DSPEN) of display mode register 0 (DSPM0) is 1.

**(3) Display mode register 2 (DSPM2)**

DSPM2 performs the following setting. It also indicates the status of the display timing/key scan.

- Insertion of key scan timing
- Display cycle ( $T_{DSP}$ )

DSPM2 is set using a 1-bit or 8-bit memory manipulation instruction. However, only bit 7 (KSF) can be read by a 1-bit memory manipulation instruction.

$\overline{RESET}$  input sets DSPM2 to 00H.

**Figure 14-4. Format of Display Mode Register 2 (DSPM2)**

Symbol	< 7 >	6	5	4	3	2	1	0	Address	After reset	R/W
DSPM2	KSF	KSM	0	0	0	0	FCYC1	FCYC0	FFB4H	00H	R/W

KSF	Status of key scan cycle
0	Other than key scan cycle
1	Key scan cycle

KSM	Selects insertion of key scan cycle
0	Not inserted
1	Inserted

FCYC1	FCYC0	Display cycle
0	0	$16 \times 2^9 / f_{xx}$ (655.36 $\mu$ s)
0	1	$16 \times 2^8 / f_{xx}$ (327.68 $\mu$ s)
1	0	$16 \times 2^7 / f_{xx}$ (163.84 $\mu$ s)
1	1	$16 \times 2^6 / f_{xx}$ (81.92 $\mu$ s)

- Cautions**
1. Be sure to set bits 2 to 5 to 0.
  2. Do not write data to display mode register 2 (DSPM2) when bit 7 (DSPEN) of display mode register 0 (DSPM0) is 1.

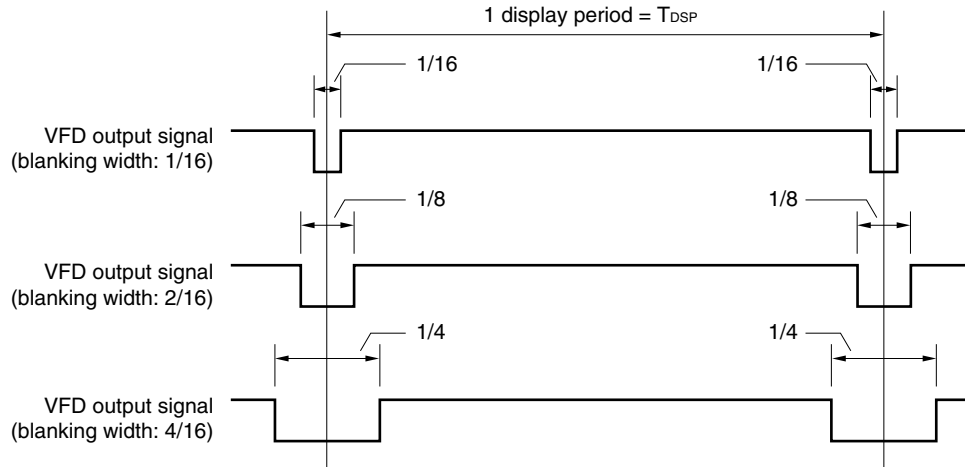
- Remarks**
1.  $f_{xx}$ : Main system clock frequency
  2. The values in parentheses are valid for operation when  $f_{xx}$  is 12.5 MHz.



### 14.3.2 One display period and blanking width

The VFD output signals are blanked equally at the beginning and end of the display period by the blanking width set by bits 5 to 7 (FBLK0 to FBLK2) of display mode register 1 (DSPM1).

Figure 14-5. Blanking Width of VFD Output Signal



### 14.4 Display Data Memory

The display data memory is a 96-byte RAM area that stores data to be displayed, and is mapped to addresses EA00H to EA5FH.

The VFD controller reads the data stored in the display data memory independently of the CPU operation for VFD display (DMA operation).

The area of the display data memory not used for display can be used as a normal RAM area.

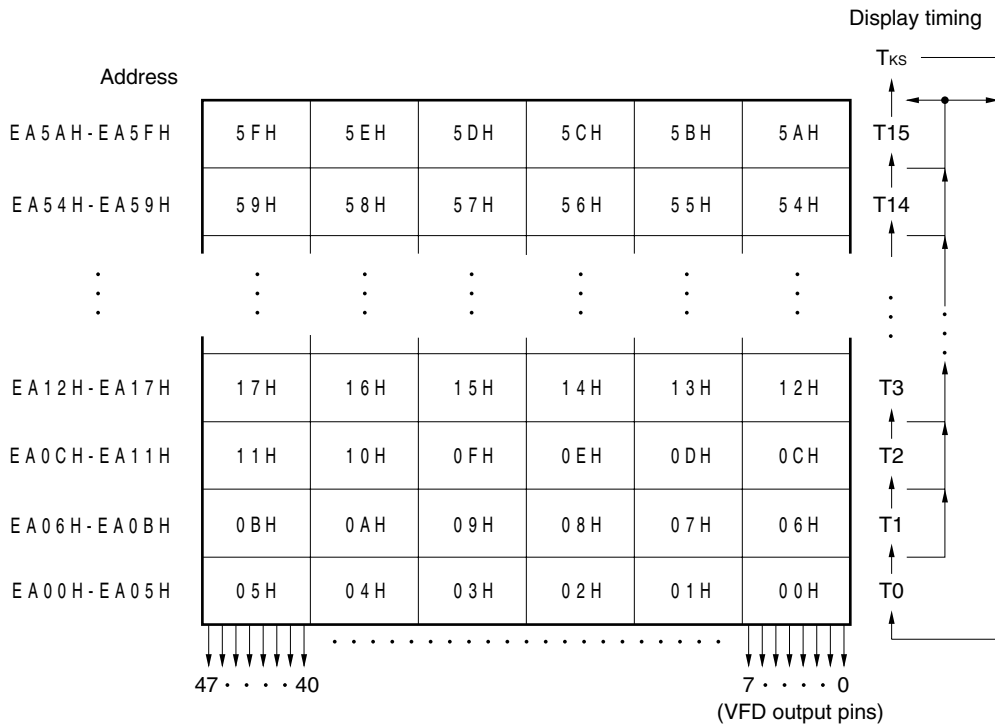
At key scan timing, all the VFD output pins are cleared to 0, and the data of the output latches of ports 7 to 10 are output to FIP16/P70 to FIP47/P107.

The address location of the display data memory is as follows:

- **With 48 VFD output pins and 16 patterns**

The addresses of the display data memory corresponding to the data output at each display timing (T0 to T15) are as shown in Figure 14-6 (for example, T0 = EA00H to EA05H, and T1 = EA06H to EA0BH). When 48 VFD output pins (FIP0 to FIP47) are used, one block of display data consists of 6 bytes. VFD output pins 0 (FIP0) to 47 (FIP47) correspond to one block of display data sequentially, starting from the least significant bit toward the most significant bit.

**Figure 14-6. Relation Between Address Location of Display Data Memory and VFD Output (with 48 VFD Output Pins and 16 Patterns)**



## 14.5 Key Scan Flag and Key Scan Data

### 14.5.1 Key scan flag

The key scan flag (KSF) is set to 1 during key scan timing, and is automatically reset to 0 at display timing.

KSF is mapped to bit 7 of display mode register 2 (DSPM2) and can be tested in 1-bit units. It cannot be written, however.

By testing KSF, it can be determined whether key scan timing is in progress, and whether key input data is correct can be checked.

Whether key scan timing is inserted or not can be selected by using the key scan timing insertion specification flag (KSM) (bit 6 of display mode register 2 (DSPM2)).

### 14.5.2 Key scan data

Data stored to ports 7 to 10 are output from the FIP16 to FIP47 pins during key scan timing.

**Caution** If scanning is performed in such a manner that both a segment and a digit turn on during key scan timing, the display may flicker.

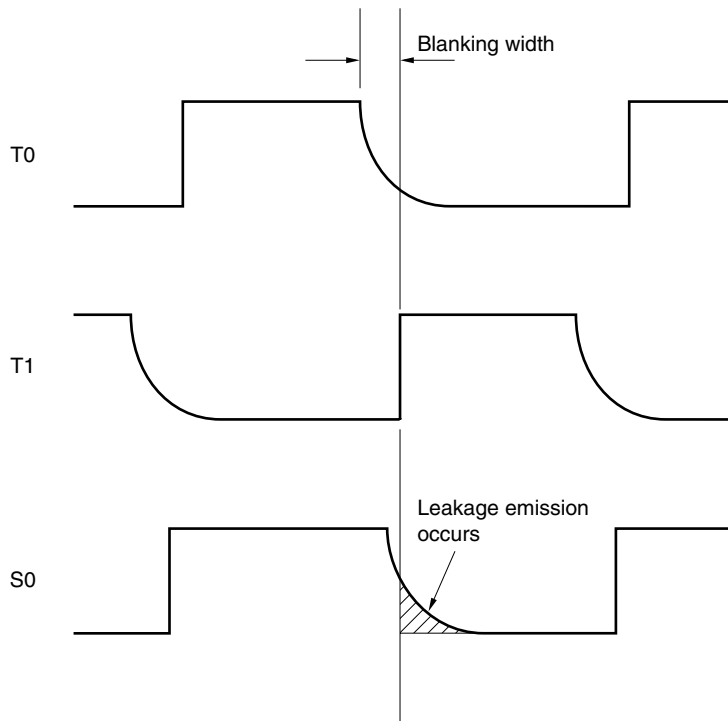
## 14.6 Leakage Emission of Fluorescent Indicator Panel

Leakage emission may take place when a fluorescent indicator panel is driven by the  $\mu$ PD784976A Subseries. The possible causes of this leakage emission are as follows:

### (1) Short blanking time

Figure 14-7 shows the signal waveforms of a 2-digit display where the first digit T0 lights and the second digit remains dark. If the blanking time is too short as shown in this figure, the T1 signal rises before the segment signal is deasserted, causing leakage emission. Generally, the blanking time must be about  $20\ \mu\text{s}$ . Determine the set value of display mode register 1 (DSPM1), taking this into consideration.

**Figure 14-7. Leakage Emission Because of Short Blanking Time**



**(2) Segment-grid capacitance of fluorescent indicator panel**

Even if a sufficiently long blanking time is ensured as shown in Figure 14-9, leakage emission may still occur. This is because the fluorescent indicator panel has a capacitance between the grid and segment, as indicated by  $C_{SG}$  in Figure 14-8, and the timing signal pin is raised via  $C_{SG}$ . If the voltage of the timing signal rises beyond the cutoff voltage ( $E_K$ ) as shown in Figure 14-9, leakage emission occurs.

This whisker-like voltage changes with the values of  $C_{SG}$  and on-chip pull-down resistor ( $R_L$ ). The greater the value of  $C_{SG}$ , and the greater the value of  $R_L$ , the higher this voltage, increasing the possibility of the occurrence of leakage emission.

The value of  $C_{SG}$  differs depending on the display area of the fluorescent indicator panel. The larger the area, the higher the  $C_{SG}$ .

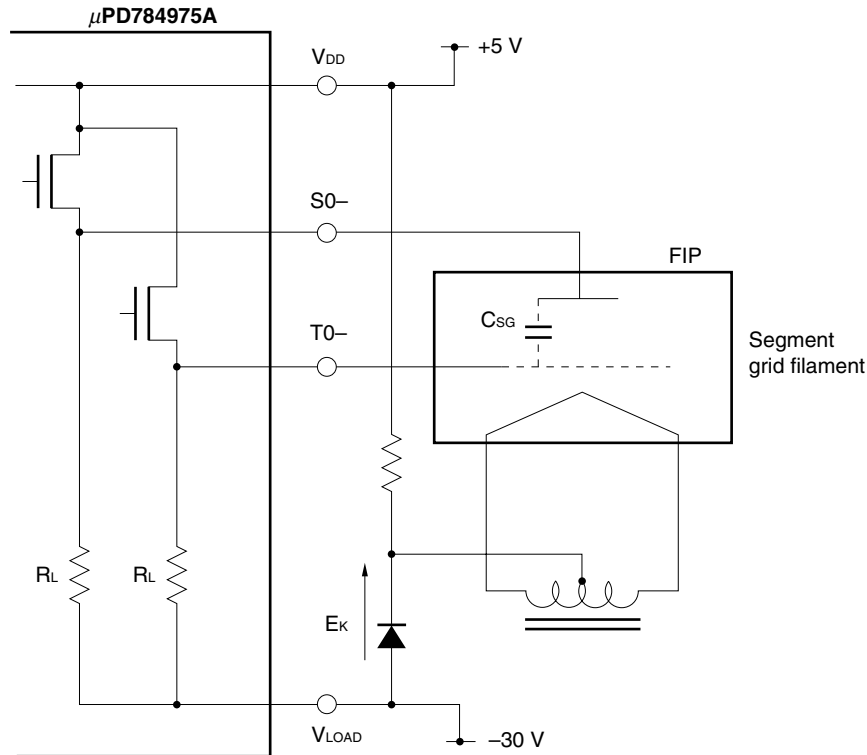
Therefore, the value of the pull-down resistor differs depending on the size of the fluorescent indicator panel, in order to prevent leakage emission.

Because the value of the pull-down resistor that can be connected by mask option is relatively high, the leakage emission may not be suppressed by the on-chip pull-down resistor alone.

In case sufficient display quality cannot be obtained, deepen the back bias (increase  $E_K$ ), attach a filter to the fluorescent indicator panel, or connect an external pull-down resistor of several 10 k $\Omega$  to the timing signal pin. The likelihood of leakage emission caused by  $C_{SG}$  occurring changes depending on the duty cycle of the whisker voltage vis-a-vis the total display cycle. The fewer the number of display digits, the less likelihood of occurrence of leakage emission.

Lowering the display luminance also has an effect of suppressing the leakage emission.

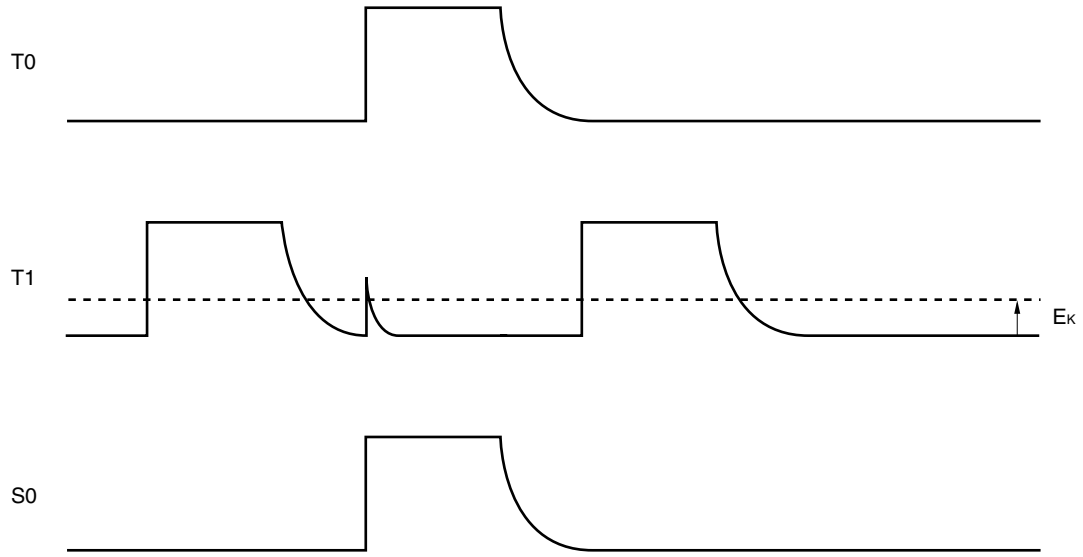
**Figure 14-8. Leakage Emission Caused by  $C_{SG}$**



$E_K$ : Cutoff voltage

$R_L$ : On-chip pull-down resistor

Figure 14-9. Leakage Emission Caused by  $C_{SG}$

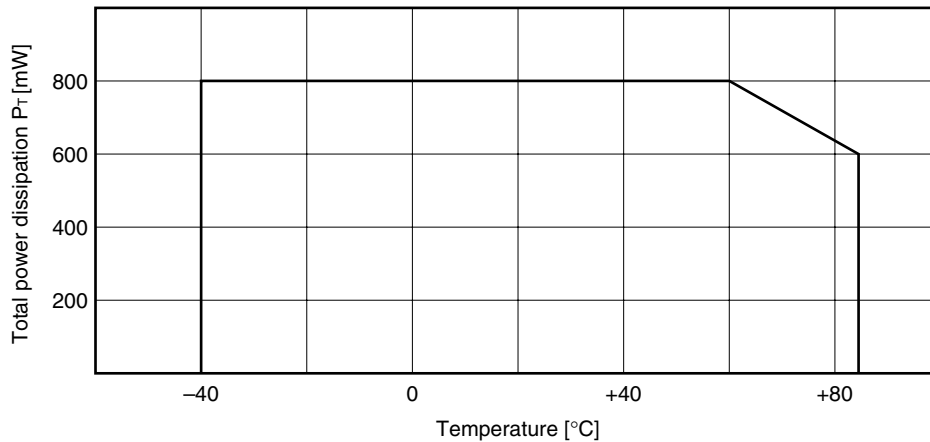


## 14.7 Calculation of Total Power Dissipation

The following three power dissipation are available for the  $\mu$ PD784976A Subseries. The sum of the three power dissipation should be less than the total power dissipation  $P_T$  (refer to **Figure 14-10**) (80% or less of ratings is recommended).

- <1> CPU power dissipation: Calculate  $V_{DD} (MAX.) \times I_{DD} (MAX.)$ .
- <2> Output pin power dissipation: Power dissipation when maximum current flows into each VFD output pin.
- <3> Pull-down resistor power dissipation: Power dissipation by pull-down resistor incorporated in VFD output pin.

**Figure 14-10. Total Power Dissipation  $P_T$  ( $T_A = -40^\circ\text{C}$  to  $+85^\circ\text{C}$ )**



The following is how to calculate total power dissipation for the example in Figure 14-11.

**Example** Assume the following conditions:

$V_{DD} = 5.5 \text{ V}$ , 12.5 MHz oscillation

Supply current ( $I_{DD}$ ) = 40.0 mA

VFD output: 11 grids  $\times$  10 segments (Blanking width = 1/16: when FBLK0 to FBLK2 = 000B)

Maximum current at the grid pin is 10 mA.

Maximum current at the segment pin is 3 mA.

At the key scan timing, VFD output pin is OFF.

VFD output voltage: grid  $V_{OD} = V_{DD} - 2 \text{ V}$  (voltage drop of 2 V)

segments  $V_{OD} = V_{DD} - 0.5 \text{ V}$  (voltage drop of 0.5 V)

Fluorescent display control voltage ( $V_{LOAD}$ ) = -30 V

Mask option pull-down resistor = 30 k $\Omega$

By placing the above conditions in calculation <1> to <3>, the total dissipation can be worked out.

<1> CPU power dissipation:  $5.5 \text{ V} \times 40.0 \text{ mA} = 220.0 \text{ mW}$

<2> Output pin power dissipation:

$$\begin{aligned} \text{Grid} \quad & (V_{DD} - V_{OD}) \times \frac{\text{Total current value of each grid}}{\text{The no. of grids} + 1} \times (1 - \text{Blanking width}) = \\ & 2 \text{ V} \times \frac{10 \text{ mA} \times 11 \text{ Grids}}{11 \text{ Grids} + 1} \times \left(1 - \frac{1}{16}\right) = 17.2 \text{ mW} \\ \text{Segment} \quad & (V_{DD} - V_{OD}) \times \frac{\text{Total segment current value of illuminated dots}}{\text{The no. of grids} + 1} \times (1 - \text{Blanking width}) = \\ & 0.5 \text{ V} \times \frac{3 \text{ mA} \times 31 \text{ Dots}}{11 \text{ Grids} + 1} \times \left(1 - \frac{1}{16}\right) = 3.6 \text{ mW} \end{aligned}$$

<3> Pull-down resistor power dissipation:

$$\begin{aligned} \text{Grid} \quad & \frac{(V_{DD} - V_{LOAD})^2}{\text{Pull-down resistor value}} \times \frac{\text{The no. of grids}}{\text{The no. grids} + 1} \times (1 - \text{Blanking width}) = \\ & \frac{(5.5 \text{ V} - 2 \text{ V} - (-30 \text{ V}))^2}{30 \text{ k}\Omega} \times \frac{11 \text{ Grids}}{11 \text{ Grids} + 1} \times \left(1 - \frac{1}{16}\right) = 32.1 \text{ mW} \\ \text{Segment} \quad & \frac{(V_{OD} - V_{LOAD})^2}{\text{Pull-down resistor value}} \times \frac{\text{The no. of illuminated dots}}{\text{The no. of grids} + 1} \times (1 - \text{Blanking width}) = \\ & \frac{(5.5 \text{ V} - 0.5 \text{ V} - (-30 \text{ V}))^2}{30 \text{ k}\Omega} \times \frac{31 \text{ dots}}{11 \text{ Grids} + 1} \times \left(1 - \frac{1}{16}\right) = 98.9 \text{ mW} \end{aligned}$$

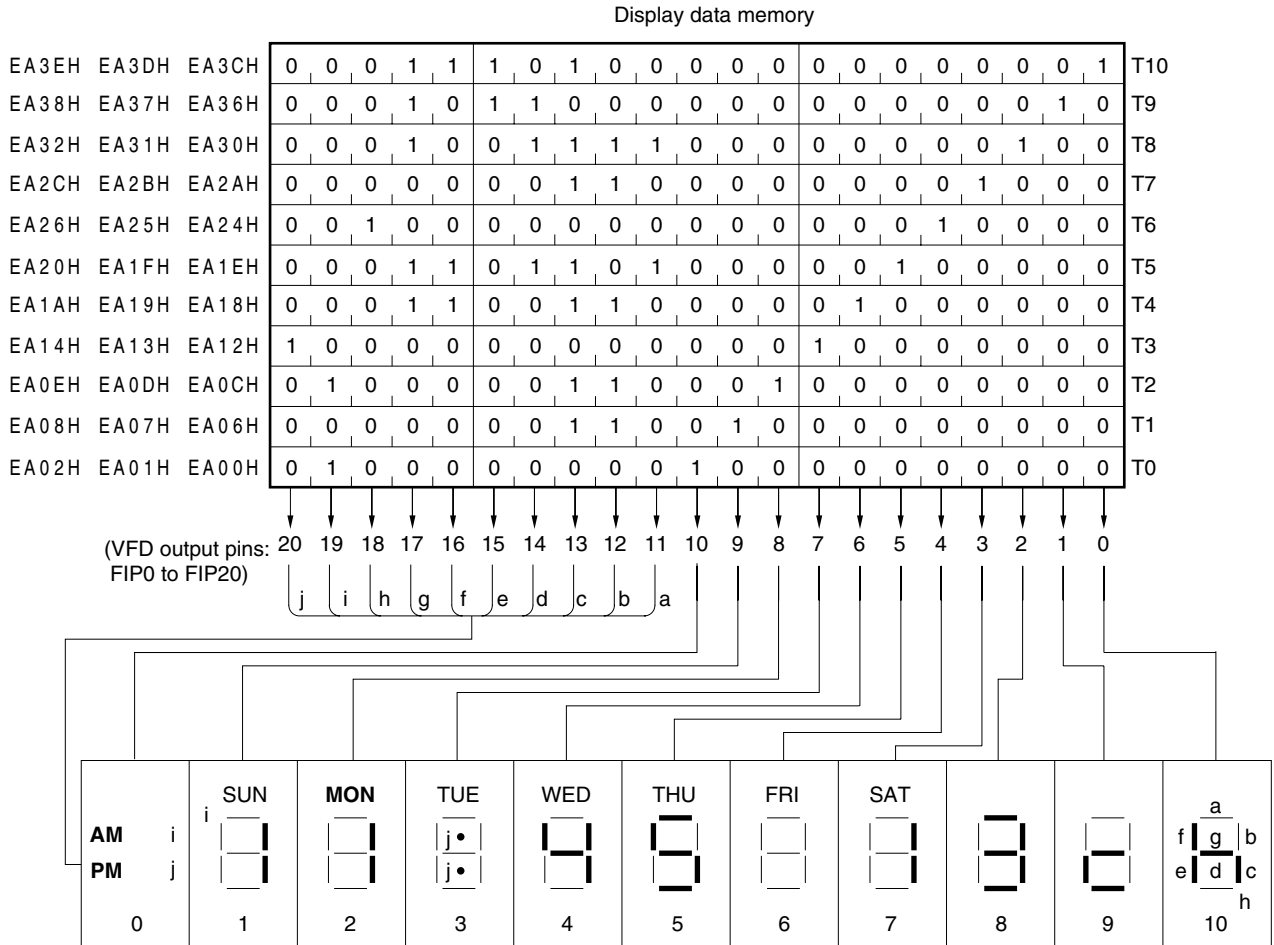
$$\text{Total power dissipation} = \text{<1>} + \text{<2>} + \text{<3>} = 220.0 + 17.2 + 3.6 + 32.1 + 98.9 = 371.8 \text{ mW}$$

In this example, the total power dissipation do not exceed the rating of the total power dissipation shown in Figure 14-10, so there is no problem in power dissipation.

However, when the total power dissipation exceeds the rating of the total power dissipation, it is necessary to lower the power dissipation. To reduce power dissipation, reduce the number of pull-down resistor.



Figure 14-11. Relationship Between Display Data Memory and VFD Output with 10 Segments × 11 Digits Displayed



## CHAPTER 15 EDGE DETECTION FUNCTION

The P64, P65, and P67 pins have an edge detection function that can be programmed to detect the rising edge or falling edge and sends the detected edge to on-chip hardware components.

The edge detection function is always functioning, even in STOP mode and IDLE mode.

### 15.1 Control Registers

- **External Interrupt Rising Edge Enable Register (EGP0), External Interrupt Falling Edge Enable Register (EGN0)**

The EGP0 and EGN0 registers specify the valid edge to be detected by the P64, P65, and P67 pins.

EGP0 and EGN0 are read/written using a 1-bit or 8-bit manipulation instruction.

RESET input sets EGP0 and EGN0 to 00H.

**Figure 15-1. Format of External Interrupt Rising Edge Enable Register (EGP0) and External Interrupt Falling Edge Enable Register (EGN0)**

Address: 0FFA0H After reset: 00H R/W

	7	6	5	4	3	2	1	0
EGP0	EGP7	0	EGP5	EGP4	0	0	0	0

Address: 0FFA2H After reset: 00H R/W

	7	6	5	4	3	2	1	0
EGN0	EGN7	0	EGN5	EGN4	0	0	0	0

EGPn	EGNn	INTPm pin valid edge (n = 4, 5, 7, m = 0 to 2)
0	0	Interrupt disable
0	1	Falling edge
1	0	Rising edge
1	1	Both rising and falling edges

**Remark** Bits 4, 5, and 7 of EGP0 and EGN0 control pins INTP0 to INTP2, respectively.

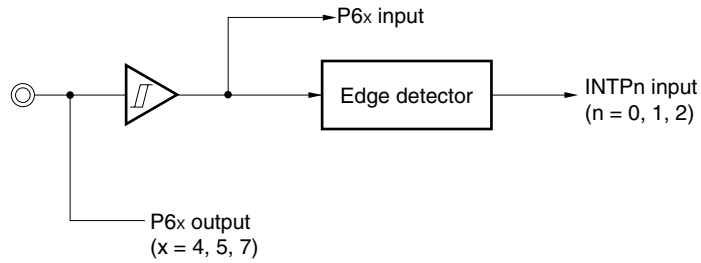
Controlled Pins	INTP0	INTP1	INTP2
Bits of EGP0	EGP4	EGP5	EGP7
Bits of EGN0	EGN4	EGN5	EGN7

## 15.2 Edge Detection of P64, P65, and P67 Pins

Figure 15-2 shows the configuration of an edge detector for pins P64, P65, and P67.

These pins are not provided with a noise eliminator resulting from analog delay. So, edge detection (recognition) begins immediately after a valid edge input to the pins passes the input buffer, which is of hysteresis type.

**Figure 15-2. Edge Detection of P64, P65, and P67 Pins**



## CHAPTER 16 INTERRUPT FUNCTION

The  $\mu$ PD784975A is provided with the following three interrupt request service modes: vectored interrupt, context switching, and macro service modes (refer to **Table 16-1**). These three service modes can be set as required in the program. However interrupt service by macro service can only be selected for interrupt request sources provided with the macro service processing mode shown in Table 16-2. 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 16-1. Interrupt Request Service Modes**

Interrupt Request Service Mode	Servicing Performed	PC & PSW Contents	Service
Vectored interrupts	Software	Saving to & restoration from stack	Executed by branching to service program at address <sup>Note</sup> 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 to service program at address <sup>Note</sup> specified by and branching fixed area in register bank
Macro service	Hardware (firmware)	Retained	Execution of pre-set service such as data transfers between memory and I/O

**Note** 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.

16.1 Interrupt Request Sources

The  $\mu$ PD784975A has the 23 interrupt request sources shown in Table 16-2, with a vector table allocated to each.

Table 16-2. Interrupt Request Sources (1/2)

Interrupt Request	Default Priority	Interrupt Request Generating Source	Generating Unit	Interrupt Control	Context Switching Register Name	Macro Service	Macro Service Control Word Address	Vector Table Address
Software	None	BRK instruction execution	—	—	Not possible	Not possible	—	003EH
		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	—	003CH
Non-maskable	None	INTWDT (watchdog timer overflow)	Watchdog timer	—	Not possible	Not possible	—	0004H
Maskable	0	INTWDTM (watchdog timer overflow)		WDTIC	Possible	Possible	0FE06H	0006H
	1	INTP0 (pin input edge detection)	Edge detection	PIC0	Possible	Possible	0FE08H	0008H
	2	INTP1 (pin input edge detection)		PIC1			0FE0AH	000AH
	3	INTP2 (pin input edge detection)		PIC2			0FE0CH	000CH
	4	INTTM00 (occurrence of signal indicating a match between the 16-bit timer counter (TM0) and capture compare register (CR00))	16-bit timer/event counter 0	TMIC00			0FE0EH	000EH
	5	INTTM01 (occurrence of signal indicating a match between the 16-bit timer counter (TM0) and capture compare register (CR01))		TMIC01			0FE10H	0010H
	6	INTKS (timing of key scanning from VFD controller/driver)	VFD controller/driver	KSIC			0FE12H	0012H
	7	INTCSI0 (end of 3-wire transfer of CSI0)	Serial interface	CSIIC0			0FE14H	0014H
	8	INTCSI1 (end of 3-wire transfer of CSI1)		CSIIC1			0FE16H	0016H
	9	INTTM50 (match between the 8-bit timer counter (TM50) and 8-bit compare register (CR50))	8-bit PWM timer (TM50)	TMIC50			0FE18H	0018H
	10	INTTM51 (match between the 8-bit timer counter (TM51) and 8-bit compare register (CR51))	8-bit PWM timer (TM51)	TMIC51			0FE1AH	001AH
11	INTAD (end of A/D conversion)	A/D converter	ADIC	0FE1CH			001CH	

- 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.** CSI: Clocked synchronous serial interface

Table 16-2. Interrupt Request Sources (2/2)

Interrupt Request	Default Priority	Interrupt Request Generating Source	Generating Unit	Interrupt Control	Context Switching Register Name	Macro Service	Macro Service Control Word Address	Vector Table Address
Maskable	12	INTREM (generation of remote control receive interrupt by 16-bit timer/event counter 0)	16-bit timer/event counter 0	REMEC	Possible	Possible	0FE1EH	001EH
	13	INTCSI2 (end of CSI2 3-wire transfer)	Serial interface (SIO2)	CSIIC2			0FE20H	0020H
	14	INTSER0 (occurrence of UART receive error)	Asynchronous serial interface (UART)	SERIC0			0FE22H	0022H
	15	INTSR0 (end of reception by UART)		SRIC0			0FE24H	0024H
	16	INTST0 (end of transmission by UART)		STIC0			0FE26H	0026H
	17	INTWT1 (reference interval time signal from watch timer)	Watch timer	WTIIC			0FE28H	0028H
	18	INTWT (watch timer overflow)		WTIC			0FE2AH	002AH

**Remark** The default priority is a fixed number. This indicates the order of priority when two or more interrupt requests specified as having the same priority are generated simultaneously.

### 16.1.1 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.

### 16.1.2 Operand error interrupts

These interrupts are generated if there is an illegal operand in an MOV STBC, #byte instruction or MOV WDM, #byte instruction, and LOCATION instruction.

Operand error interrupts are acknowledged even in the interrupt disabled state, and are not subject to priority control.

### 16.1.3 Non-maskable interrupts

A non-maskable interrupt is generated by the watchdog timer.

Non-maskable interrupts are acknowledged unconditionally<sup>Note</sup>, even in the interrupt disabled state. They are not subject to interrupt priority control, and are of higher priority than any other interrupt.

**Note** 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

### 16.1.4 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: refer to **Table 16-2**).

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 16-2. 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.

## 16.2 Interrupt Service Modes

There are three  $\mu$ PD784975A interrupt service modes, as follows.

- Vectored interrupt service
- Macro service
- Context switching

### 16.2.1 Vectored interrupt service

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 service routine is executed.

### 16.2.2 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 (refer to **16.8 Macro Service Function**).

### 16.2.3 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 (refer to **16.4.2 BRKCS instruction software interrupt (software context switching) acknowledgment operation** and **16.7.2 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).



### 16.3 Interrupt Servicing Control Registers

$\mu$ PD784975A interrupt servicing is controlled for each interrupt request by various control registers that perform interrupt servicing specification. The interrupt control registers are listed in Table 16-3.

**Table 16-3. Control Registers**

Register Name	Symbol	Function
Interrupt control registers	WDTIC, PIC0, PIC1, PIC2, CSIIC0, CSIIC1, TMIC00, TMC01, KSIC, TMIC50, TMIC51, ADIC, REMIC, CSIIC2, SERIC0, SRIC0, STIC0, WTIIC, WTIC	Record generation of interrupt request, control masking, specify vectored interrupt servicing or macro service processing, enable or disable context switching function, and specify priority.
★ Interrupt mask register	MK0 (MK0L, MK0H), MK1L	Controls masking of maskable interrupt request. Associated with mask control flag in interrupt control register. MK0 can be accessed in word or byte units. MK1L can be accessed in byte units.
In-service priority register	ISPR	Records priority of interrupt request currently accepted.
Interrupt mode control register	IMC	Controls nesting of maskable interrupt with priority specified to lowest level (level 3).
Interrupt select control register	SNMI	Selects whether to use interrupt signal from watchdog timer as maskable or non-maskable interrupt.
Watchdog timer mode register	WDM	Controls operation of watchdog timer.

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 16-4.

**Table 16-4. Flag List of Interrupt Control Registers for Interrupt Request Sources**

Default Priority	Interrupt Request Signal	Interrupt Control Register					
			Interrupt Request Flag	Interrupt Mask Flag	Macro Service Enable Flag	Priority Specification Flag	Context Switching Enable Flag
0	INTWDT	WDTIC	WDTIF	WDTMK	WDTISM	WDTPR0 WDTPR1	WDCSE
1	INTP0	PIC0	PIF0	PMK0	PISM0	PPR00 PPR01	PCSE0
2	INTP1	PIC1	PIF1	PMK1	PISM1	PPR10 PPR11	PCSE1
3	INTP2	PIC2	PIF2	PMK2	PISM2	PPR20 PPR21	PCSE2
4	INTTM00	TMIC00	TMIF00	TMMK00	TMISM00	TMPR000 TMPR001	TMCSE00
5	INTTM01	TMIC01	TMIF01	TMMK01	TMISM01	TMPR010 TMPR011	TMCSE01
6	INTKS	KSIC	KSIF	KSMK	KSISM	KSPR0 KSPR1	KSCSE
7	INTCSI0	CSIIC0	CSIIF0	CSIMK0	CSIISM0	CSIPR00 CSIPR01	CSICSE0
8	INTCSI1	CSIIC1	CSIIF1	CSIMK1	CSIISM1	CSIPR10 CSIPR11	CSICSE1
9	INTTM50	TMIC50	TMIF50	TMMK50	TMISM50	TMPR500 TMPR501	TMCSE50
10	INTTM51	TMIC51	TMIF51	TMMK51	TMISM51	TMPR510 TMPR511	TMCSE51
11	INTAD	ADIC	ADIF	ADMK	ADISM	ADPR0 ADPR1	ADCSE
12	INTREM	REMIC	REMIF	REMMK	REMISM	REMPR0 REMPR1	REMCSE
13	INTCSI2	CSIIC2	CSIIF2	CSIMK2	CSIISM2	CSIPR20 CSIPR21	CSICSE2
14	INTSER0	SERIC0	SERIF0	SERMK0	SERISM0	SERPR00 SERPR01	SERCSE0
15	INTSR0	SRIC0	SRIF0	SRMK0	SRISM0	SRPR00 SRPR01	SRCSE0
16	INTST0	STIC0	STIF0	STMK0	STISM0	STPR00 STPR01	STCSE0
17	INTWTI	WTIIC	WTIIF	WTIMK	WTIISM	WTIPR0 WTIPR1	WTICSE
18	INTWT	WTIC	WTIF	WTMK	WTISM	WTPR0 WTPR1	WTCSE

### 16.3.1 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 16-1.

#### (1) Priority specification flags (xxPR1/xxPR0)

The priority specification flags specify the priority on an individual interrupt source basis for the 19 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 by software.

$\overline{\text{RESET}}$  input sets all bits to 1.

#### (2) Context switching enable flag (xxCSE)

The context switching enable flag specifies that a maskable interrupt request is to be serviced 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 servicing can be started faster than with normal vectored interrupt servicing.

This flag can be manipulated bit-wise by software.

$\overline{\text{RESET}}$  input sets all bits to 0.

#### (3) Macro service enable flag (xxISM)

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 service/context switching service).

This flag can be manipulated bit-wise by software.

$\overline{\text{RESET}}$  input sets all bits to 0.

#### (4) Interrupt mask flag (xxMK)

The interrupt mask flag specifies enabling/disabling of vectored interrupt servicing and macro service processing for the interrupt request corresponding to that flag.

The interrupt mask contents are not changed by the start of interrupt service, etc., and are the same as the interrupt mask register contents (refer to **16.3.2 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 by software.

$\overline{\text{RESET}}$  input sets all bits to 1.

#### (5) Interrupt request flag (xxIF)

The 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 by software.

$\overline{\text{RESET}}$  input sets all bits to 0.

Figure 16-1. Interrupt Control Register (××ICn) (1/2)

Address: 0FFE0H to 0FFE8H	After reset : 43H				R/W			
Symbol	<7>	<6>	<5>	<4>	3	2	<1>	<0>
WDTIC	WDTIF	WDTMK	WDTISM	WDCSE	0	0	WDTPR1	WDTPR0
PIC0	PIF0	PMK0	PISM0	PCSE0	0	0	PPR01	PPR00
PIC1	PIF1	PMK1	PISM1	PCSE1	0	0	PPR11	PPR10
PIC2	PIF2	PMK2	PISM2	PCSE2	0	0	PPR21	PPR20
TMIC00	TMIF00	TMMK00	TMISM00	TMCSE00	0	0	TMPR001	TMPR000
TMIC01	TMIF01	TMMK01	TMISM01	TMCSE01	0	0	TMPR011	TMPR010
KSIC	KSIF	KSMK	KSISM	KSCSE	0	0	KSPR1	KSPR0
CSIIC0	CSIF0	CSIMK0	CSIISM0	CSICSE0	0	0	CSIPR01	CSIPR00
CSIIC1	CSIF1	CSIMK1	CSIISM1	CSICSE1	0	0	CSIPR11	CSIPR10

××IFn	Interrupt request generation
0	No interrupt request (Interrupt signal is not generated.)
1	Interrupt request (Interrupt signal is generated.)

××MKn	Interrupt servicing enable/disable
0	Interrupt servicing enable
1	Interrupt servicing disable

××ISMn	Interrupt servicing mode specification
0	Vectored interrupt servicing/context switching processing
1	Macro service processing

××CSEn	Context switching processing specification
0	Processing with vectored interrupt
1	Processing with context switching

××PRn1	××PRn0	Interrupt request priority specification
0	0	Priority 0 (Highest priority)
0	1	Priority 1
1	0	Priority 2
1	1	Priority 3

Figure 16-1. Interrupt Control Register (××ICn) (2/2)

Address: 0FFE9H to 0FFEBH	After reset : 43H				R/W			
Symbol	<7>	<6>	<5>	<4>	3	2	<1>	<0>
TMIC50	TMIF50	TMMK50	TMISM50	TMCSE50	0	0	TMPR501	TMPR500
TMIC51	TMIF51	TMMK51	TMISM51	TMCSE51	0	0	TMPR511	TMPR510
ADIC	ADIF	ADMK	ADISM	ADCSE	0	0	ADPR1	ADPR0
REMIC	REMIF	REMMK	REMISM	REMCSE	0	0	REMPR1	REMPR0
CSIC2	CSIF2	CSIMK2	CSISM2	CSICSE2	0	0	CSIPR21	CSIPR20
SERIC0	SERIF0	SERMK0	SERISM0	SERCSE0	0	0	SERPR01	SERPR00
SRIC0	SRIF0	SRMK0	SRISM0	SRCSE0	0	0	SRPR01	SRPR00
STIC0	STIF0	STMK0	STISM0	STCSE0	0	0	STPR01	STPR00
WTIC	WTIF	WTMK	WTISM	WTCSE	0	0	WTIPR1	WTIPR0
WTIC	WTIF	WTMK	WTISM	WTCSE	0	0	WTPR1	WTPR0

××IFn	Interrupt request generation
0	No interrupt request (Interrupt signal is not generated.)
1	Interrupt request (Interrupt signal is generated.)

××MKn	Interrupt servicing enable/disable
0	Interrupt servicing enable
1	Interrupt servicing disable

××ISMn	Interrupt servicing mode specification
0	Vectored interrupt servicing/context switching processing
1	Macro service processing

××CSEn	Context switching processing specification
0	Processing with vectored interrupt
1	Processing with context switching

××PRn1	××PRn0	Interrupt request priority specification
0	0	Priority 0 (Highest priority)
0	1	Priority 1
1	0	Priority 2
1	1	Priority 3

**★ 16.3.2 Interrupt mask registers (MK0, MK1L)**

MK0 and MK1L are composed of interrupt mask flags. MK0 is a 16-bit register which can be manipulated in 16-bit units. MK0 can also be manipulated in 8-bit units using MK0L and MK0H. MK1L can be manipulated in 8-bit units.

In addition, each bit of MK0 and MK1L can be manipulated individually with a 1-bit manipulation instruction. Each interrupt mask flag controls enabling/disabling of the corresponding interrupt request.

When an interrupt mask flag is set to 1, acknowledgment of the corresponding interrupt request is disabled.

When an interrupt mask flag is cleared to 0, the corresponding interrupt request can be acknowledged as a vectored interrupt or macro service request.

Each interrupt mask flag in MK0 and MK1L is the same flag as the interrupt mask flag in the interrupt control register. MK0 and MK1L are provided for blanket control of interrupt masking.

RESET input sets MK0 to FFFFH and MK1L to FFH, and all maskable interrupts are disabled.

★

Figure 16-2. Format of Interrupt Mask Registers (MK0, MK1L)

<Byte access>

Address: 0FFACH, 0FFADH, 0FFAEH      After reset: FFH      R/W

Symbol	7	6	5	4	3	2	1	0
MK0L	CSIMK0	KSMK	TMMK01	TMMK00	PMK2	PMK1	PMK0	WDTMK
MK0H	SRMK0	SERMK0	CSIMK2	REMMK	ADMK	TMMK51	TMMK50	CSIMK1
MK1L	1	1	1	1	1	WTMK	WTIMK	STMK0

××MKn	Interrupt request enable/disable
0	Interrupt servicing enable
1	Interrupt servicing disable

<Word access>

Address: 0FFACH      After reset: 0FFFFH      R/W

Symbol	15	14	13	12	11	10	9	8
MK0	SRMK0	SERMK0	CSIMK2	REMMK	ADMK	TMMK51	TMMK50	CSIMK1
	7	6	5	4	3	2	1	0
	CSIMK0	KSMK	TMMK01	TMMK00	PMK2	PMK1	PMK0	WDTMK

××MKn	Interrupt request enable/disable
0	Interrupt servicing enable
1	Interrupt servicing disable

**16.3.3 In-service priority register (ISPR)**

The ISPR shows the priority level of the maskable interrupt currently being serviced and the non-maskable interrupt being serviced. When a maskable interrupt request is acknowledged, the bit corresponding to the priority of that interrupt request is set to 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 to 1, and remains set until the service program ends.

When an RETI instruction or RETCS instruction is executed, the bit, among those set to 1 in the ISPR, that corresponds to the highest-priority interrupt request is automatically cleared to 0 by hardware.

The contents of the ISPR are not changed by execution of an RETB or RETCSB instruction.  $\overline{\text{RESET}}$  input sets the ISPR to 00H.

**Figure 16-3. Format of In-Service Priority Register (ISPR)**

Address: 0FFA8H		After reset: 00H		R				
Symbol	7	6	5	4	3	2	1	0
ISPR	0	WDTS	0	0	ISPR3	ISPR2	ISPR1	ISPR0

WDTS	Watchdog timer interrupt servicing status
0	Watchdog timer interrupt (non-maskable interrupt: INTWDT) is not acknowledged.
1	Watchdog timer interrupt (non-maskable interrupt: INTWDT) is acknowledged.

ISPRn	Priority level (n = 0 to 3)
0	Interrupt of priority level n is not acknowledged.
1	Interrupt of priority level n is acknowledged.

**Caution** The in-service priority register (ISPR) is a read-only register. The microcontroller may malfunction if this register is written.



**16.3.4 Interrupt mode control register (IMC)**

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 IMC is manipulated, the interrupt disabled state (DI state) should be set first to prevent malfunction.

IMC can be read or written to using a 1-bit or 8-bit manipulation instruction.

$\overline{\text{RESET}}$  input sets IMC to 80H.

**Figure 16-4. Format of Interrupt Mode Control Register (IMC)**

Address: 0FFAAH		After reset: 80H		R/W					
Symbol	7	6	5	4	3	2	1	0	
IMC	PRSL	0	0	0	0	0	0	0	0

PRSL	Nesting control of maskable interrupt (lowest level)
0	Interrupts with level 3 (lowest level) can be nested.
1	Nesting of interrupts with level 3 (lowest level) is disabled.

**16.3.5 Watchdog timer mode register (WDM)**

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 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 WDM, #byte is written), system initialization should be performed by the program.

Other write instructions (MOV WDM, A; AND WDM, #byte; and SET1 WDM.7) 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.

WDM can be read at any time using a data transfer instruction.

RESET input sets WDM to 00H.

**Figure 16-5. Format of Watchdog Timer Mode Register (WDM)**

Address: 0FFC2H	After reset: 00H				R/W			
Symbol	7	6	5	4	3	2	1	0
WDM	RUN	0	0	0	0	WDT2	WDT1	0
	RUN		Specifies operation of watchdog timer (refer to <b>Figure 9-2</b> ).					
	WDT2	WDT1	Specifies count clock of watchdog timer (refer to <b>Figure 9-2</b> ).					

**Caution** The watchdog timer mode register (WDM) can be written only by using a dedicated instruction (MOV WDM, #byte).

**16.3.6 Interrupt select control register (SNMI)**

SNMI selects whether to use an interrupt request signal from the watchdog timer as a maskable or non-maskable interrupts signal.

Since the bit of this register can be set (1) only once after reset, the bit should be cleared (0) by reset.

SNMI is set using a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$  input sets SNMI to 00H.

**Figure 16-6. Format of Interrupt Select Control Register (SNMI)**

Address: 0FFA9H	After reset: 00H	R/W						
Symbol	7	6	5	4	3	2	1	0
SNMI	0	0	0	0	0	0	SWDT	0

SWDT	Watchdog timer interrupt selection
0	Use as non-maskable interrupt (INTWDT). Interrupt servicing cannot be disabled with interrupt mask register.
1	Use as maskable interrupt (INTWDTM). Vectored interrupts and macro service can be used. Interrupt servicing can be disabled with interrupt mask register.

**16.3.7 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 lower 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, the PSWL is saved to the stack and the IE flag is cleared to 0. The 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 to 0. The PSWL is restored from the fixed area in the register bank by an RETCSI or RETCSB instruction.

RESET input sets PSWL to 00H.

**Figure 16-7. Format of Program Status Word (PSWL)**

After reset: 00H

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

S	Used for normal instruction execution
Z	
RSS	
AC	

IE	Enable or disable accepting interrupt
0	Disable
1	Enable

P/V	Used for normal instruction execution
CY	

### 16.4 Software Interrupt Acknowledgment Operations

A software interrupt is acknowledged in response to execution of a BRK or BRKCS instruction. Software interrupts cannot be disabled.

#### 16.4.1 BRK instruction software interrupt acknowledgment operation

When a BRK instruction is executed, the program status word (PSW) and program counter (PC) are saved in that order to the stack, the IE flag is cleared to 0, the vector table (003EH/003FH) contents are loaded into the lower 16 bits of the PC, and 0000B into the higher 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. Use the RETB instruction.

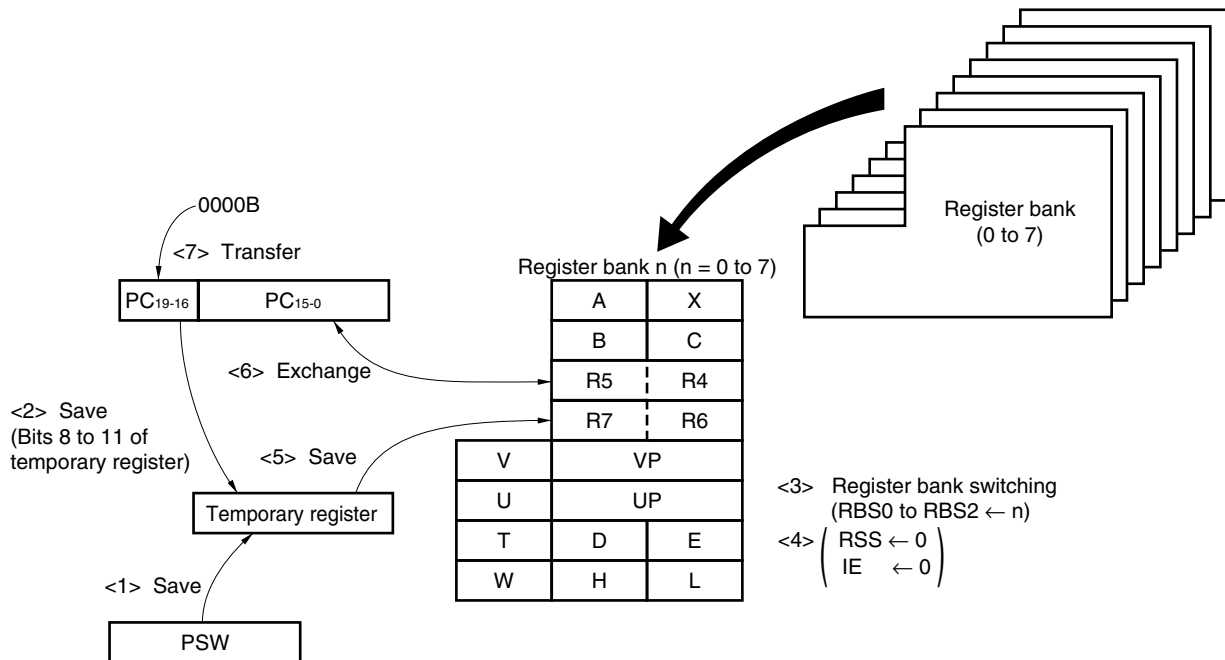
#### 16.4.2 BRKCS instruction software interrupt (software context switching) acknowledgment operation

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.

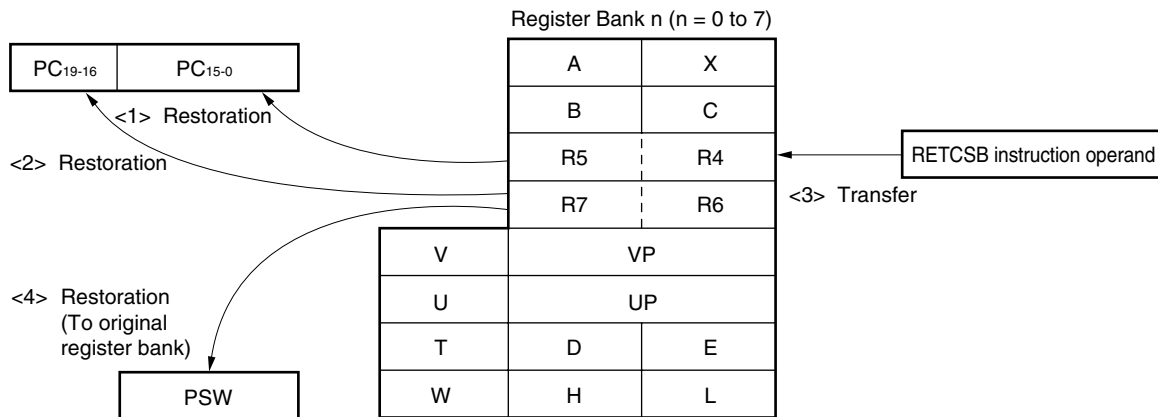
**Figure 16-8. Context Switching Operation by Execution of BRKCS Instruction**



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. Use the RETCSB instruction.

Figure 16-9. Return from BRKCS Instruction Software Interrupt (RETCSB Instruction Operation)



### 16.5 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 to 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 **16.12 Restoring Interrupt Function to Initial State**.

### 16.6 Non-Maskable Interrupt Acknowledgment Operation

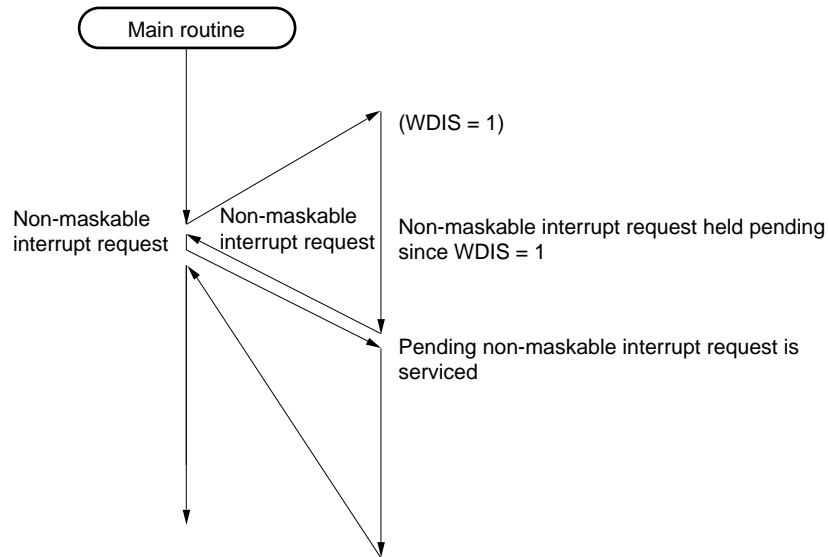
Non-maskable interrupts are acknowledged even in the interrupt disabled state.

Except in the cases described in **16.9 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 of the PSW is cleared to 0, the in-service priority register (ISPR) bit corresponding to the acknowledged non-maskable interrupt is set to 1, the vector table contents are loaded into the PC, and a branch is performed. The ISPR bit that is set to 1 is the WDTS bit.

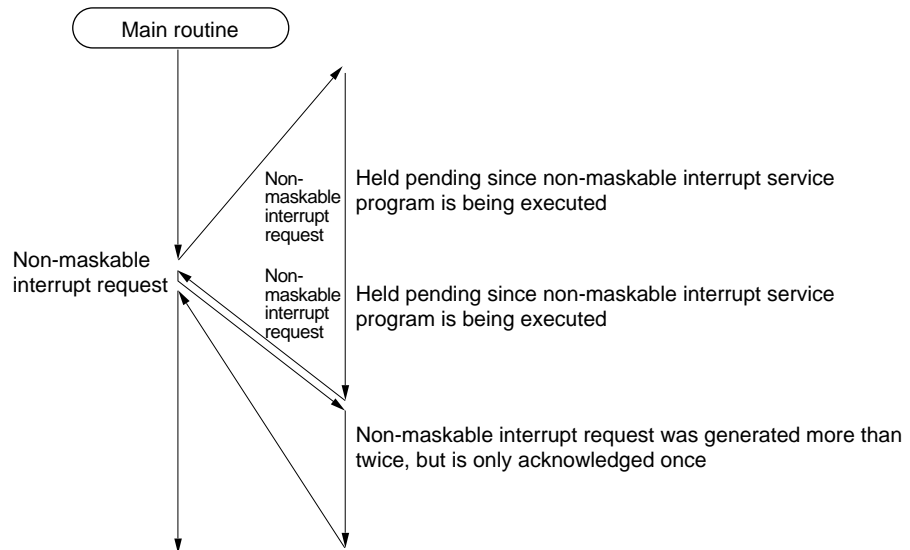
Even if the same non-maskable interrupt request is generated more than once during execution of the non-maskable interrupt service program, only one non-maskable interrupt is acknowledged after completion of the non-maskable interrupt service program.

Figure 16-10. Non-Maskable Interrupt Request Acknowledgment Operations

(a) When a new non-maskable interrupt request is generated during non-maskable interrupt service program execution



(b) When a non-maskable interrupt request is generated twice during non-maskable interrupt service program execution



- Cautions**
1. Macro service requests are acknowledged and serviced 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. If you restart a program from the initial state after a non-maskable interrupt acknowledgment, refer to 16.12 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 16.9. 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) (refer to Table 3-6 in 3.8 Special Function Registers (SFRs)), 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 a software inadvertently loops. Therefore, the program after RESET release must be as shown below.

```
CSEG AT 0
DW   STRT
CSEG BASE

STRT:
LOCATION 0FH; or LOCATION 0H
MOVG SP, #imm24
```



## 16.7 Maskable Interrupt Acknowledgment Operation

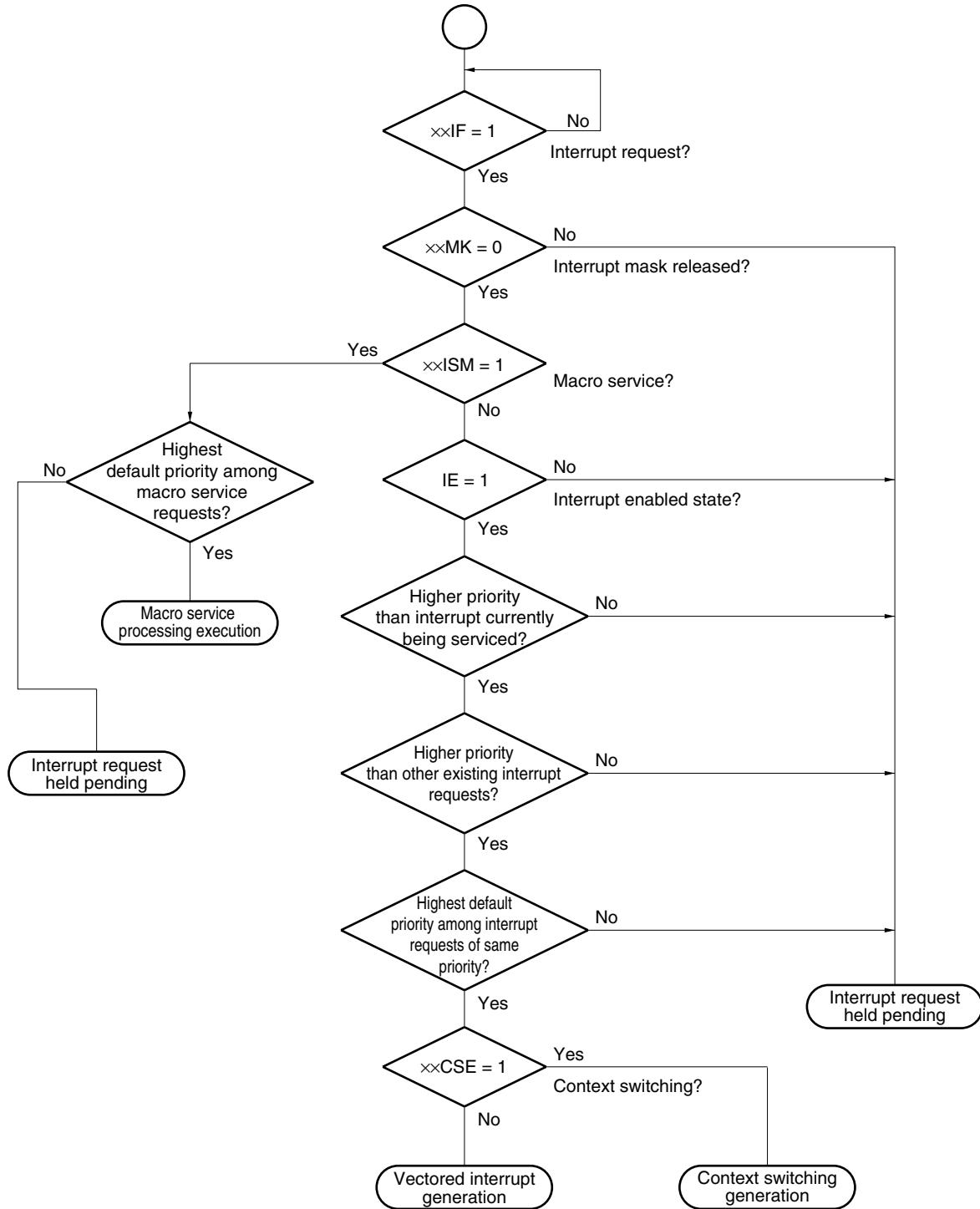
A maskable interrupt can be acknowledged when the interrupt request flag is set to 1 and the mask flag for that interrupt is cleared to 0. When servicing is performed by macro service, the interrupt is acknowledged and serviced by macro service immediately. In the case of vectored interrupt and context switching, an interrupt is acknowledged in the interrupt enabled state (when the IE flag is set to 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 16-11.

Figure 16-11. Interrupt Request Acknowledgment Processing Algorithm



**16.7.1 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 to 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 to 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.

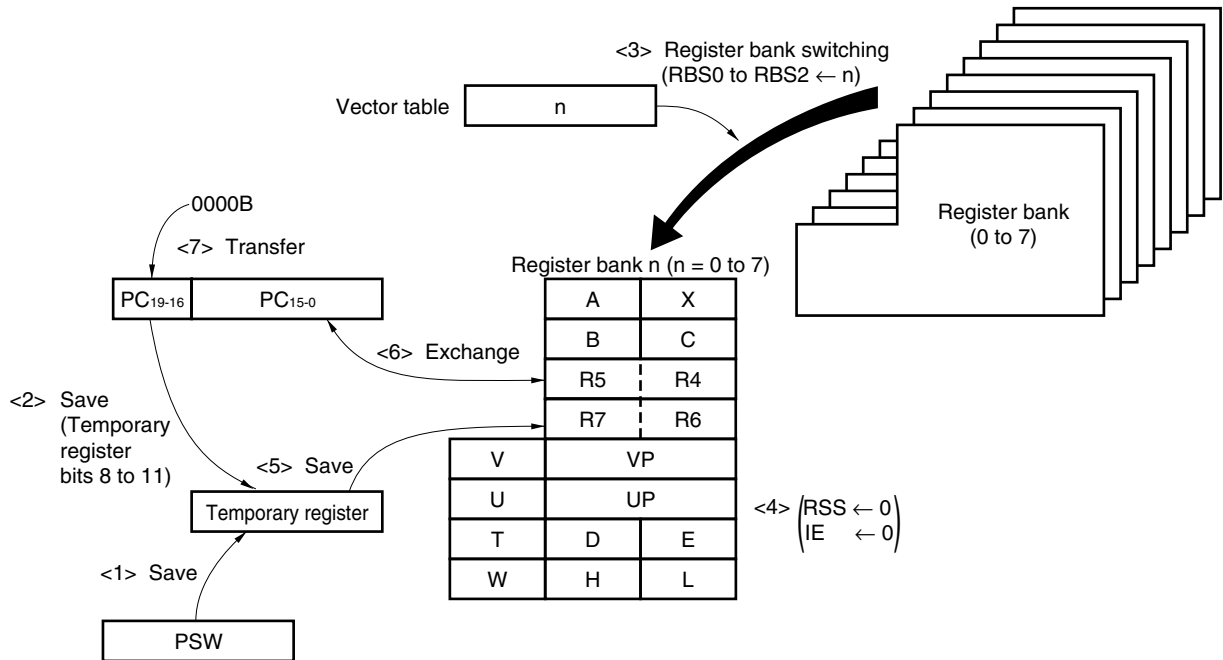
**16.7.2 Context switching**

Initiation of the context switching function is enabled by setting the context switching enable flag of the interrupt control register to 1.

When an interrupt request for which the context switching function is enabled is acknowledged, the register bank specified by 3 bits of the lower 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 branching is performed to the interrupt service program.

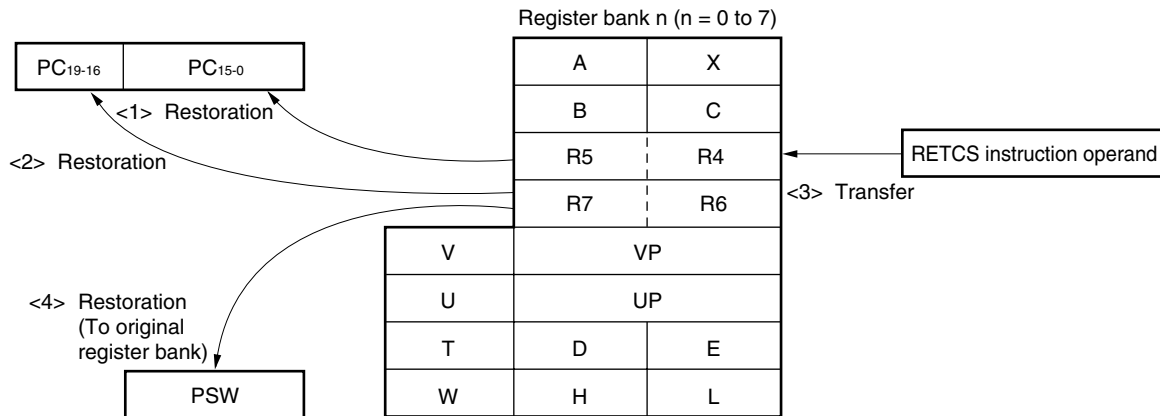
**Figure 16-12. Context Switching Operation by Generation of Interrupt Request**



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 serviced by context switching. Subsequent interrupt acknowledgment will not be performed normally if a different instruction is used.

Figure 16-13. Return from Interrupt that Uses Context Switching by Means of RETCS Instruction



**16.7.3 Maskable interrupt priority levels**

The  $\mu$ PD784975A performs multiple interrupt servicing in which an interrupt is acknowledged during servicing of another interrupt. Multiple 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 service is performed in accordance with the priority preassigned to each interrupt request (default priority) (refer to **Table 16-2**). In programmable priority control, interrupt requests are divided into four levels according to the setting of the priority specification flag. Interrupt requests for which multiple interrupt is permitted are shown in Table 16-5.

Since the IE flag is cleared to 0 automatically when an interrupt is acknowledged, when multiple interrupt is used, the IE flag should be set to 1 to enable interrupts by executing an IE instruction in the interrupt service program, etc.

**Table 16-5. Multiple Interrupt Processing**

Priority of Interrupt Currently Being Acknowledged	ISPR Value	IE Flag in PSW	PRSL in IMC Flag	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, or 2
2	0000×100	0	×	• All macro service only
		1	×	• All macro service • Maskable interrupts specified as priority 0 or 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	0100××××	×	×	• All macro service only

Figure 16-14. Examples of Servicing When Another Interrupt Request Is Generated During Interrupt Service (1/3)

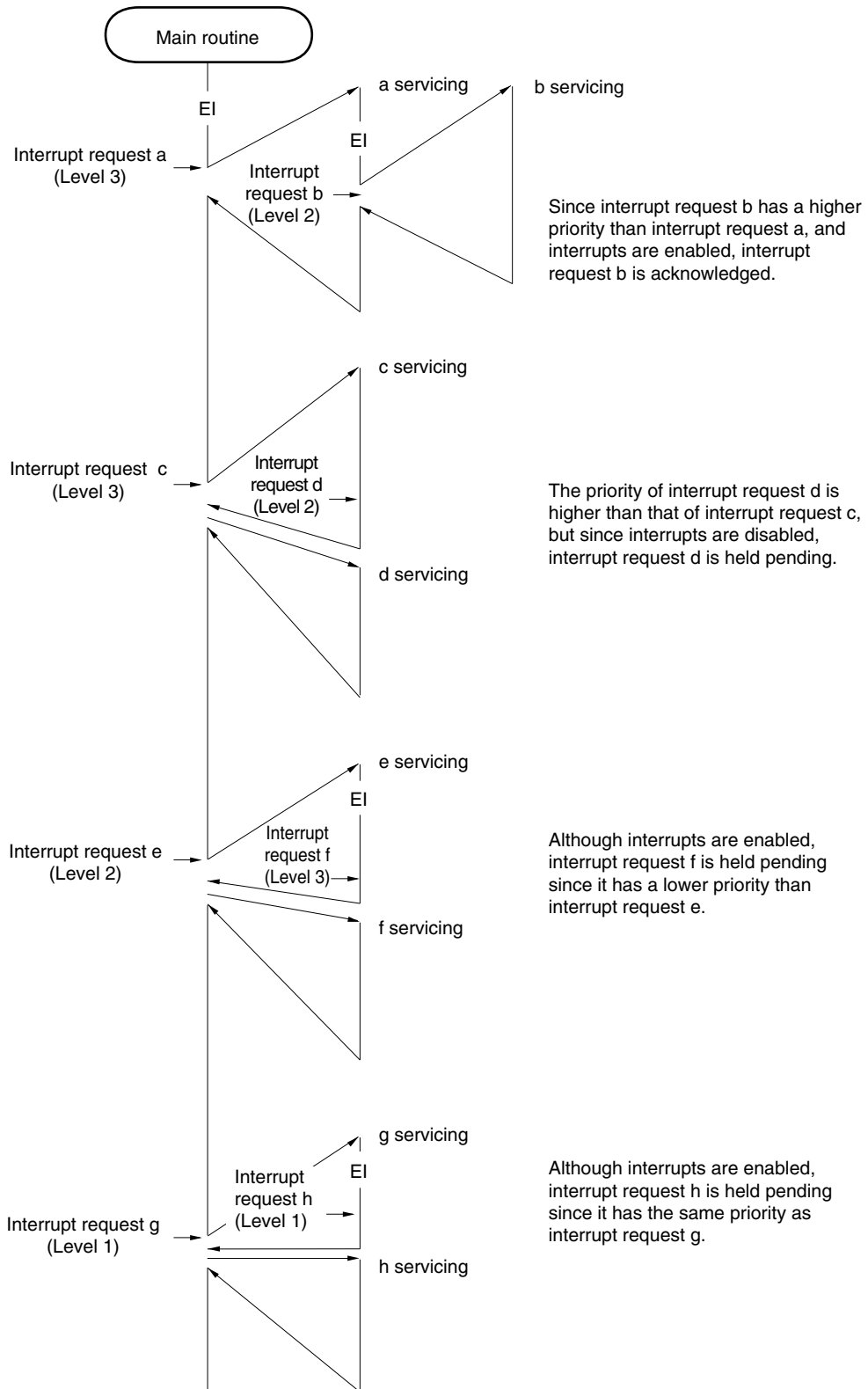
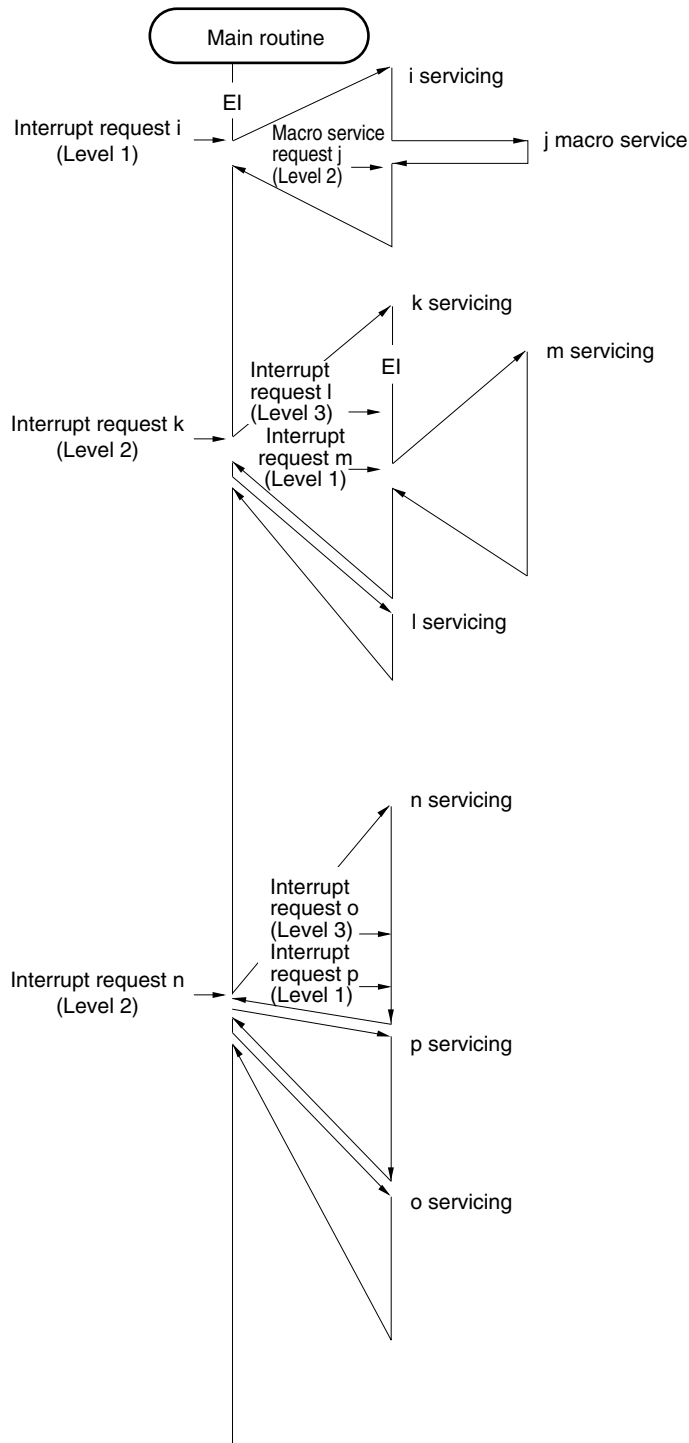


Figure 16-14. Examples of Servicing When Another Interrupt Request Is Generated During Interrupt Service (2/3)

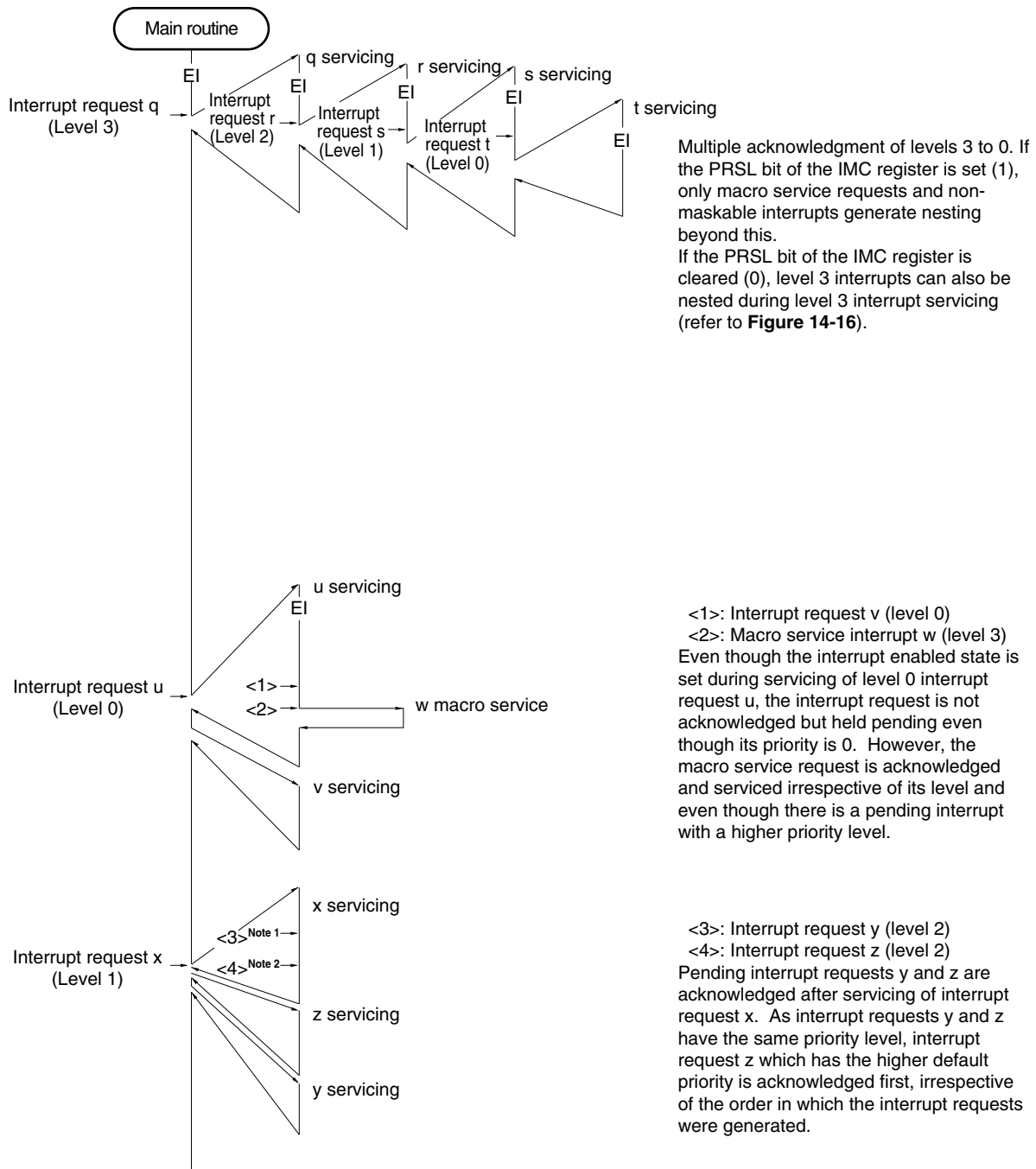


The macro service request is serviced irrespective of interrupt enabling/disabling and priority.

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 servicing of interrupt request n performed in the interrupt disabled state, interrupt requests o and p are held pending. After interrupt request n servicing, 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 16-14. Examples of Servicing When Another Interrupt Request Is Generated During Interrupt Service (3/3)**

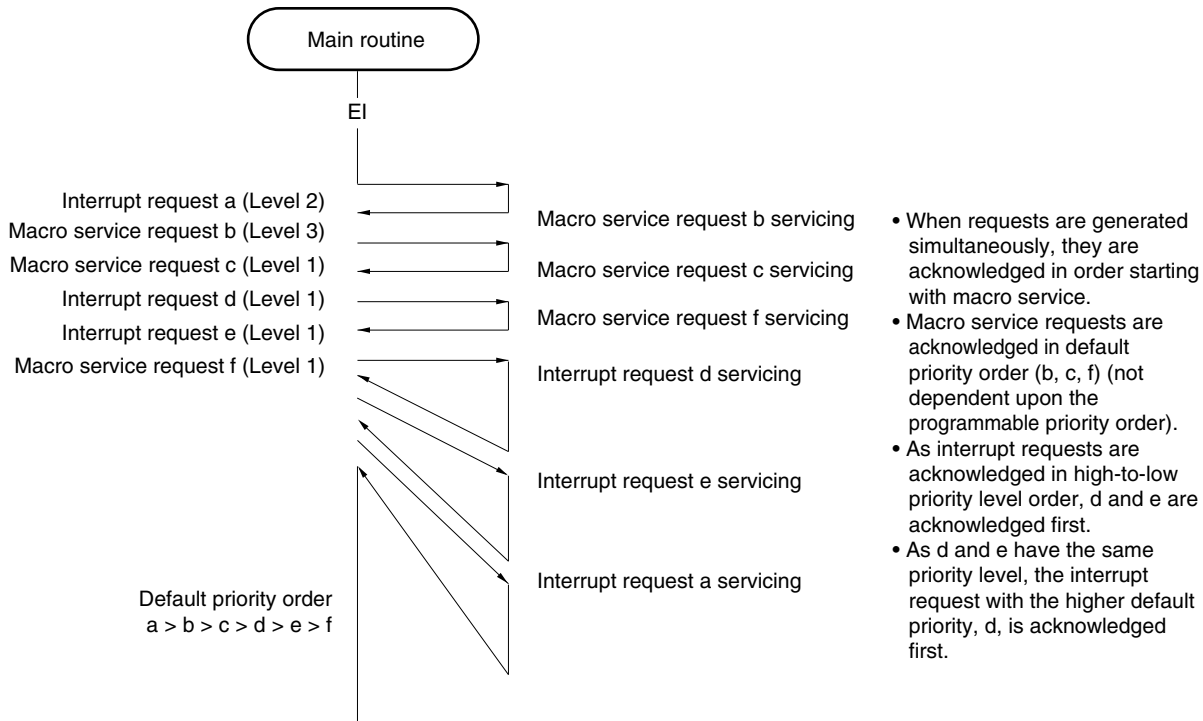


- Notes 1.** Low default priority  
**2.** High default priority

- Remarks 1.** “a” to “z” in the figure above 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.

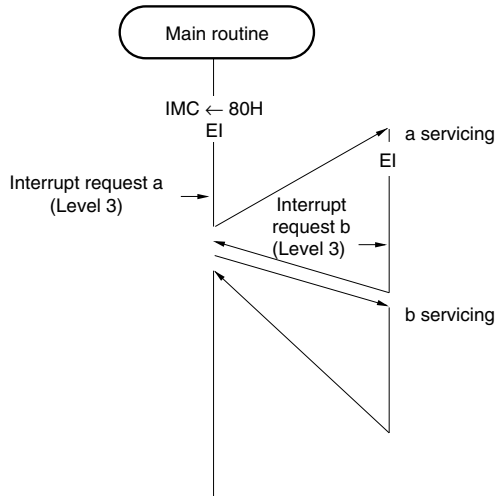


Figure 16-15. Examples of Servicing of Simultaneously Generated Interrupts



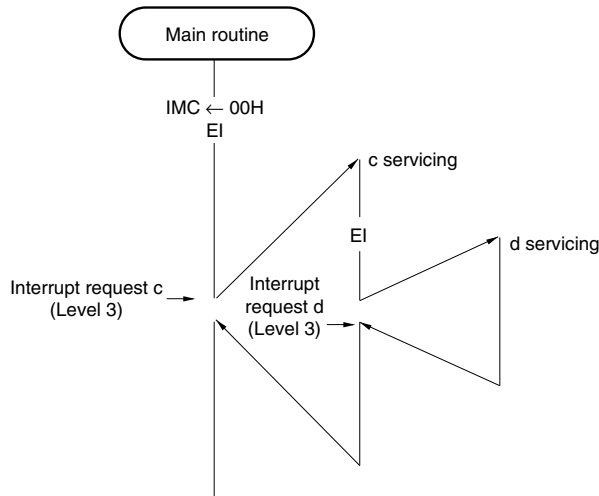
**Remark** “a” to “f” in the figure above are arbitrary names used to differentiate between the interrupt requests and macro service requests.

Figure 16-16. Differences in Level 3 Interrupt Acknowledgment According to IMC Register Setting



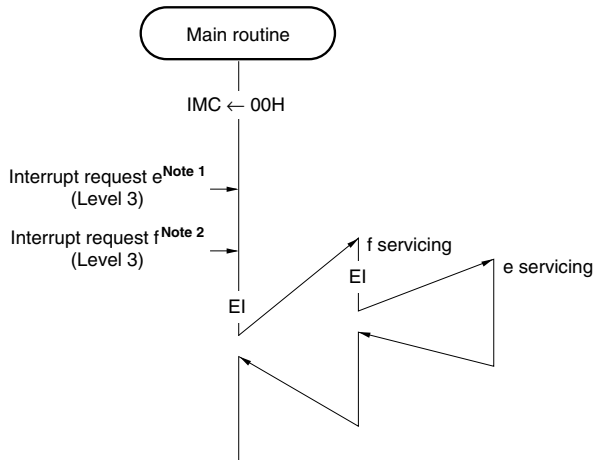
The PRSL bit of the IMC is set to 1, and nesting between level 3 interrupts is disabled.

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 servicing (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 e 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 servicing of interrupt request f, pending interrupt request e is acknowledged since PRSL = 0.

- Notes 1. Low default priority
- 2. High default priority

- Remarks 1. "a" to "f" in the figure above are arbitrary names used to differentiate between the interrupt requests.
- 2. High/low default priorities in the figure indicate the relative priority levels of the two interrupt requests.

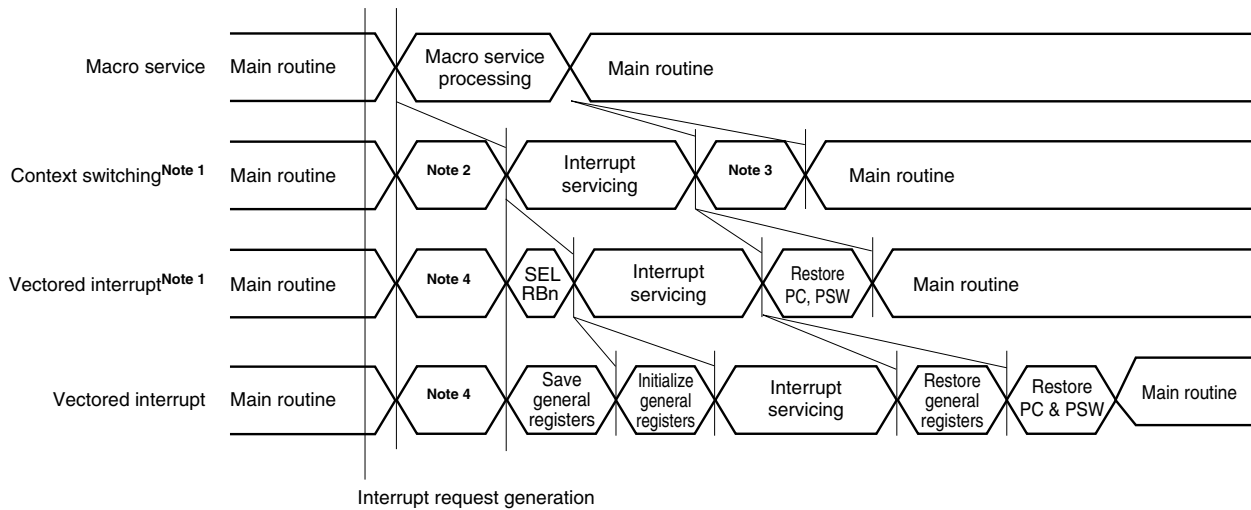
## 16.8 Macro Service Function

### 16.8.1 Outline of macro service function

Macro service is one method of servicing 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 16-17. Differences Between Vectored Interrupt and Macro Service Processing**



- Notes**
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

### 16.8.2 Types of macro service

Macro service can be used with the 19 kinds of interrupts shown in Table 16-6. There are three kinds of operation, which can be used to suit the application.

**Table 16-6. Interrupts for Which Macro Service Can be Used**

Default Priority	Interrupt Request Generation Source	Generating Unit	Macro Service Control Word Address
0	INTWDTM (watchdog timer overflow (when interval time is selected))	Watchdog timer	0FE06H
1	INTP0 (pin input edge detection)	Edge detection	0FE08H
2	INTP1 (pin input edge detection)		0FE0AH
3	INTP2 (pin input edge detection)		0FE0CH
4	INTTM00 (occurrence of signal indicating a match between the 16-bit timer counter (TM0) and capture compare register (CR00))	16-bit timer/event counter 0	0FE0EH
5	INTTM01 (occurrence of signal indicating a match between the 16-bit timer counter (TM0) and capture compare register (CR01))		0FE10H
6	INTKS (timing of key scanning from VFD controller/driver)	VFD controller/driver	0FE12H
7	INTCSI0 (end of 3-wire transfer of CSI0)	Serial interface	0FE14H
8	INTCSI1 (end of 3-wire transfer of CSI1)		0FE16H
9	INTTM50 (match between the 8-bit timer counter (TM50) and 8-bit compare register (CR50))	8-bit PWM timer 50 (TM50)	0FE18H
10	INTTM51 (match between the 8-bit timer counter (TM51) and 8-bit compare register (CR51))	8-bit PWM timer 51 (TM51)	0FE1AH
11	INTAD (end of A/D conversion)	A/D converter	0FE1CH
12	INTREM (generation of remote control receive interrupt by 16-bit timer/event counter 0)	16-bit timer/event counter 0	0FE1EH
13	INTCSI2 (end of CSI2 3-wire transfer)	Serial interface (SIO2)	0FE20H
14	INTSER0 (occurrence of UART receive error)	Asynchronous serial interface (UART)	0FE22H
15	INTSR0 (end of reception by UART)		0FE24H
16	INTST0 (end of transmission by UART)		0FE26H
17	INTWT1 (reference interval time signal from watch timer)	Watch timer	0FE28H
18	INTWT (watch timer overflow)		0FE2AH

- Remarks 1.** The default priority is a fixed number. This indicates the order of priority when two or more macro service requests specified as having the same priority are generated simultaneously.
- 2.** CSI: Clocked synchronous serial interface  
 UART: Asynchronous serial interface

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 0FE06H to 0FE1DH when the LOCATION 0H instruction is executed, and addresses 0FFE06H to 0FFE1DH 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 1 MB memory space can be used.

**(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 generator.

When MSC is 0, a vector interrupt can be generated.

To restart the macro service, MSC must be set again.

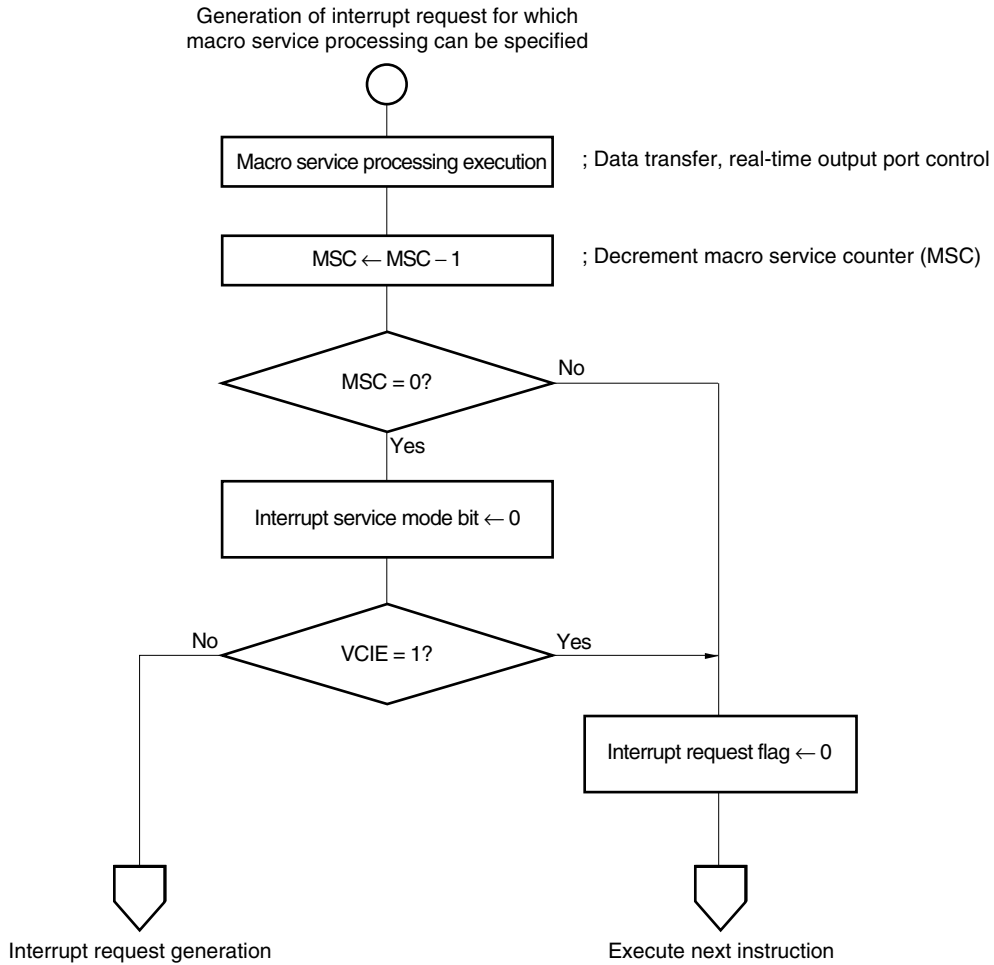
MSC is fixed to 16 bits and cannot be used as an 8-bit counter.

### 16.8.3 Basic macro service operation

Interrupt requests for which the macro service processing generated by the algorithm shown in Figure 16-10 can be specified are basically serviced in the sequence shown in Figure 16-18.

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 an interrupt mask flag in the interrupt mask register (MK0, MK1L) to 1. Macro service processing can be executed in the interrupt disabled state and during execution of an interrupt service program.

**Figure 16-18. 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 FE06H to FE1DH when the LOCATION 0H instruction is executed, or FFE06H to FFE1DH when the LOCATION 0FH instruction is executed.

**16.8.4 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.

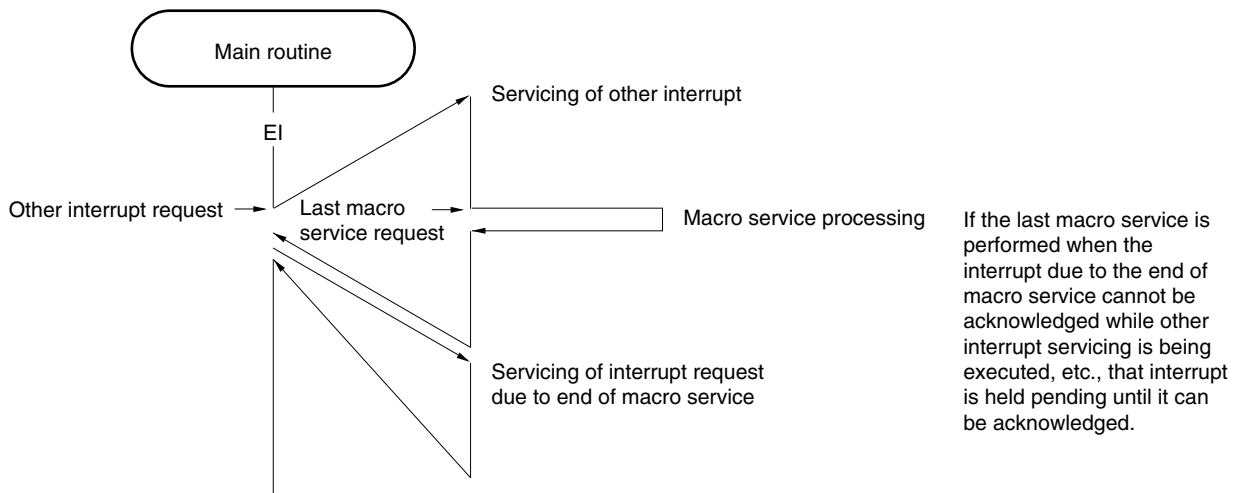
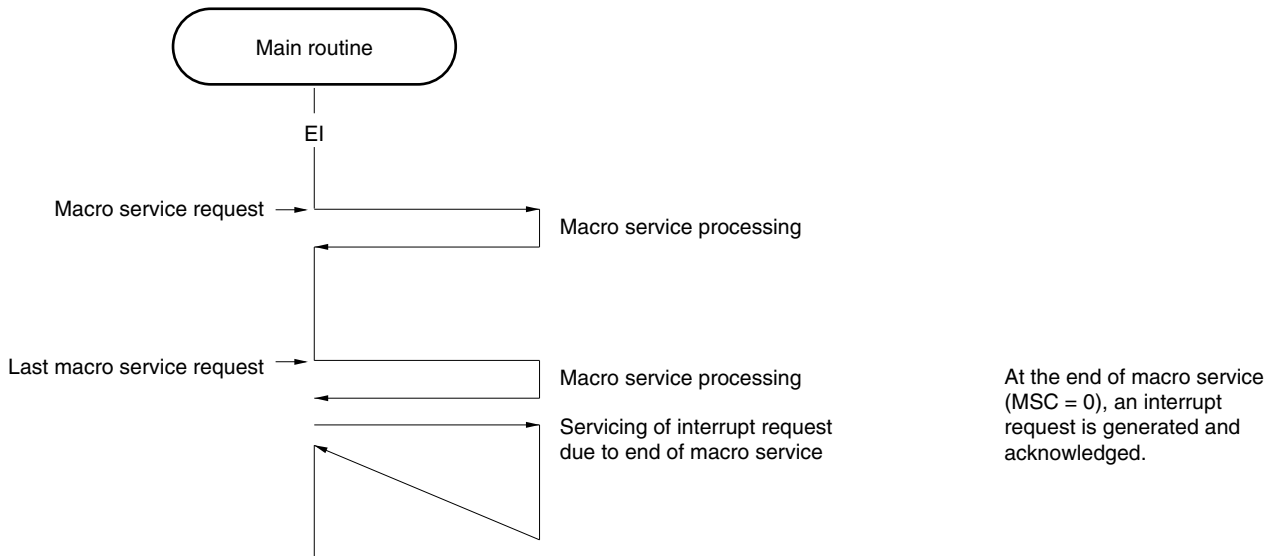
**(1) When VCIE bit is 0**

In this mode, an interrupt is generated as soon as the macro service ends. Figure 16-19 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:

- A/D conversion result fetch (INTAD)
- Compare register update as the result of a match between a timer counter and the compare register (INTTM00, INTTM01, INTTM50, INTTM51)
- Timer counter capture register read due to edge input to the INTPn pin (INTP0, INTP1, INTP2)

Figure 16-19. Operation at End of Macro Service When VCIE = 0





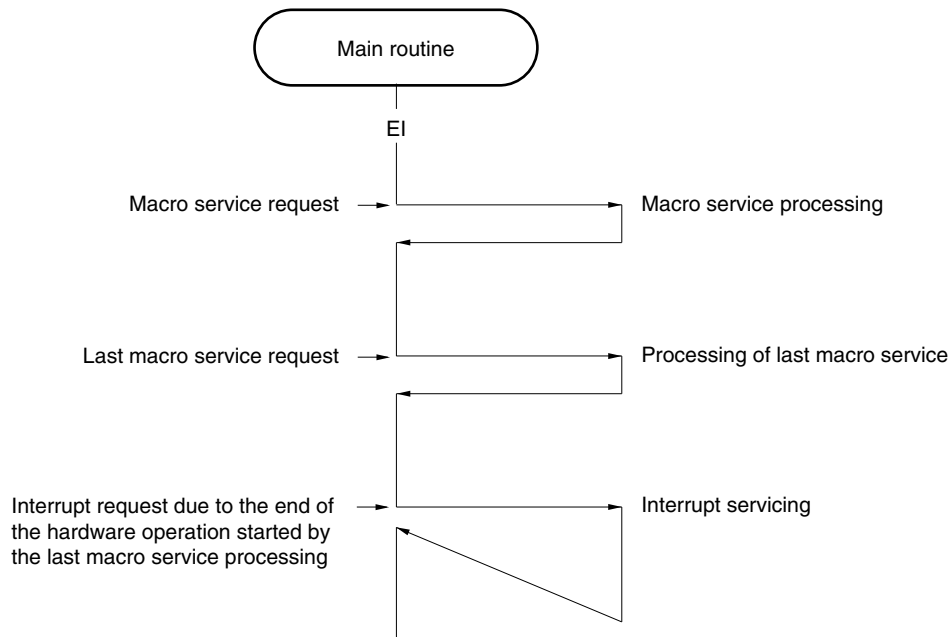
**(2) When VCIE bit is 1**

In this mode, an interrupt is not generated after macro service ends. Figure 16-20 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:

- Clock synchronous serial interface receive data transfers (INTCSI0, INTCSI1)

**Figure 16-20. Operation at End of Macro Service When VCIE = 1**



16.8.5 Macro service control registers

(1) Macro service control word

The  $\mu$ PD784975A 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 16-21 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 16-21. Format of Macro Service Control Word

Reserved word	Address		Source
WTCHP	0FE2BH	Channel pointer	INTWT
WTMMD	0FE2AH	Mode register	
WTICHP	0FE29H	Channel pointer	INTWT1
WTIMMD	0FE28H	Mode register	
STCHP0	0FE27H	Channel pointer	INTST0
STMMD0	0FE26H	Mode register	
SRCHP0	0FE25H	Channel pointer	INTSR0
SRMMD0	0FE24H	Mode register	
SERCHP0	0FE23H	Channel pointer	INTSER0
SERMMD0	0FE22H	Mode register	
CSICHP2	0FE21H	Channel pointer	INTCSI2
CSIMMD2	0FE20H	Mode register	
REMCHP0	0FE1FH	Channel pointer	INTREM
REMMMD0	0FE1EH	Mode register	
ADCHP	0FE1DH	Channel pointer	INTAD
ADMMD	0FE1CH	Mode register	
TMCHP51	0FE1BH	Channel pointer	INTTM51
TMMMD51	0FE1AH	Mode register	
TMCHP50	0FE19H	Channel pointer	INTTM50
TMMMD50	0FE18H	Mode register	
CSICHP1	0FE17H	Channel pointer	INTCSI1
CSIMMD1	0FE16H	Mode register	
CSICHP0	0FE15H	Channel pointer	INTCSI0
CSIMMD0	0FE14H	Mode register	
KSCHP	0FE13H	Channel pointer	INTKS
KSMMD	0FE12H	Mode register	
TMCHP01	0FE11H	Channel pointer	INTTM01
TMMMD01	0FE10H	Mode register	
TMCHP00	0FE0FH	Channel pointer	INTTM00
TMMMD00	0FE0EH	Mode register	
PCHP2	0FE0DH	Channel pointer	INTP2
PMMD2	0FE0CH	Mode register	
PCHP1	0FE0BH	Channel pointer	INTP1
PMMD1	0FE0AH	Mode register	
PCHP0	0FE09H	Channel pointer	INTP0
PMMD0	0FE08H	Mode register	
WDTCHP	0FE07H	Channel pointer	INTWDTM
WDTMMD	0FE06H	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 (refer to **Figure 16-21**).

The format of the macro service mode register is shown in Figure 16-22.

Figure 16-22. Format of Macro Service Mode Register (1/2)

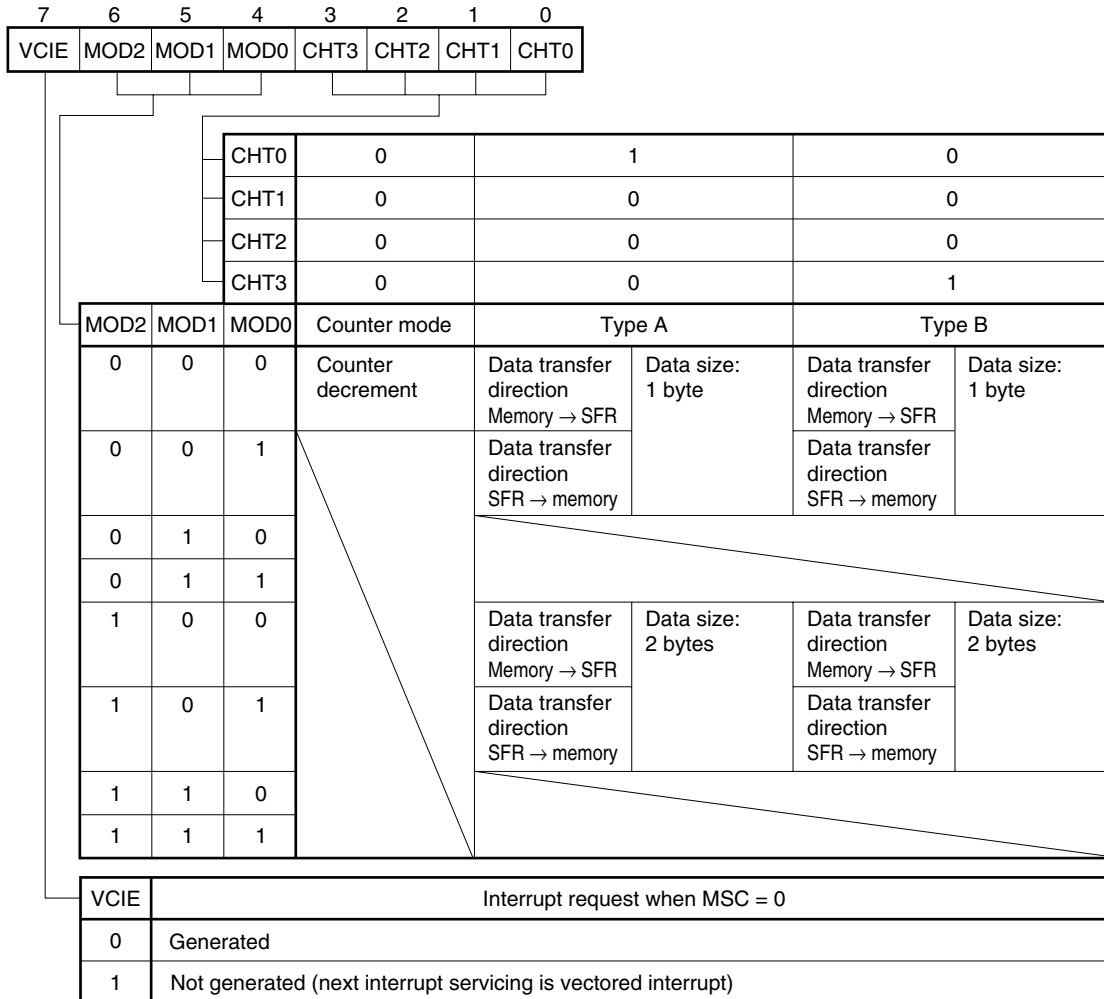
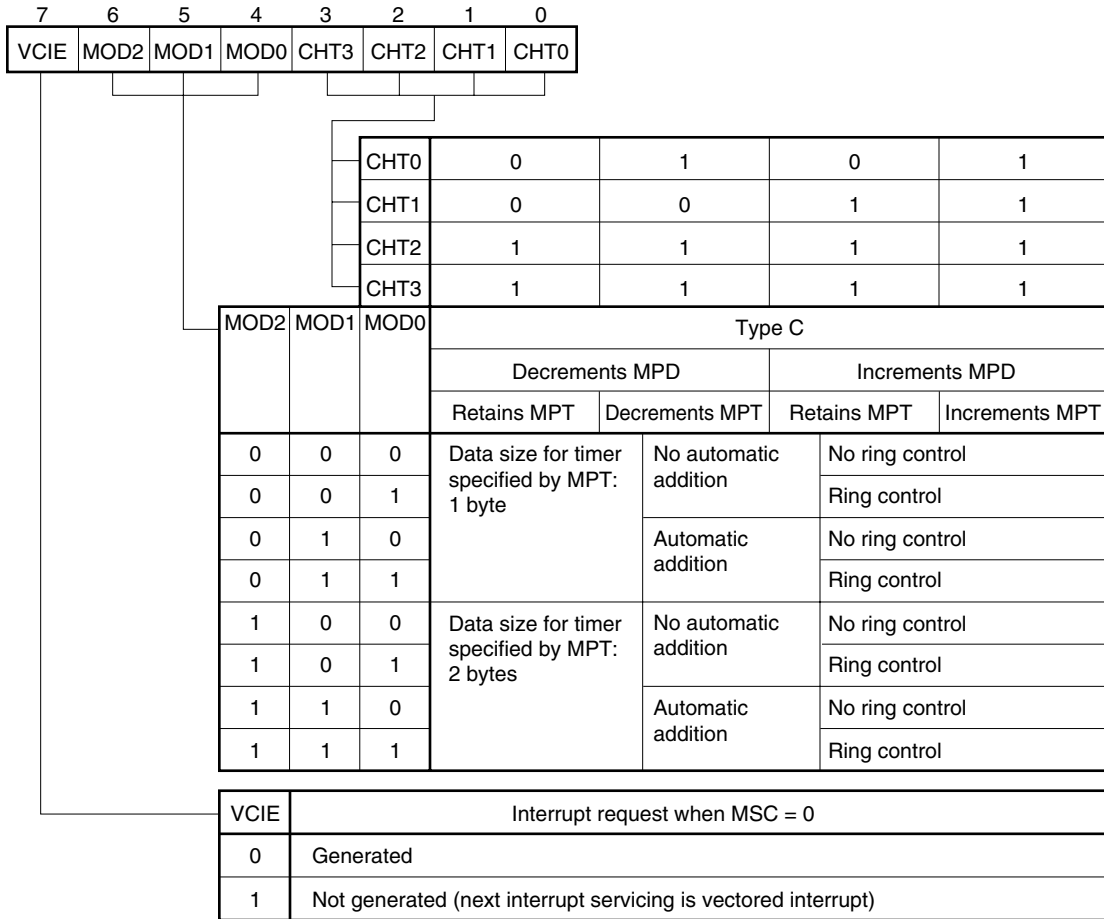


Figure 16-22. Format of Macro Service Mode Register (2/2)



**(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 FE06H to FE1DH when the LOCATION 0H instruction is executed, or FFE06H to FFE1DH when the LOCATION 0FH instruction is executed, and the higher 16 bits of the address are fixed. Therefore, the lower 8 bits of the data stored to the highest address of the macro service channel are set in the macro service channel pointer.

**16.8.6 Macro service type A**

**(1) Operation**

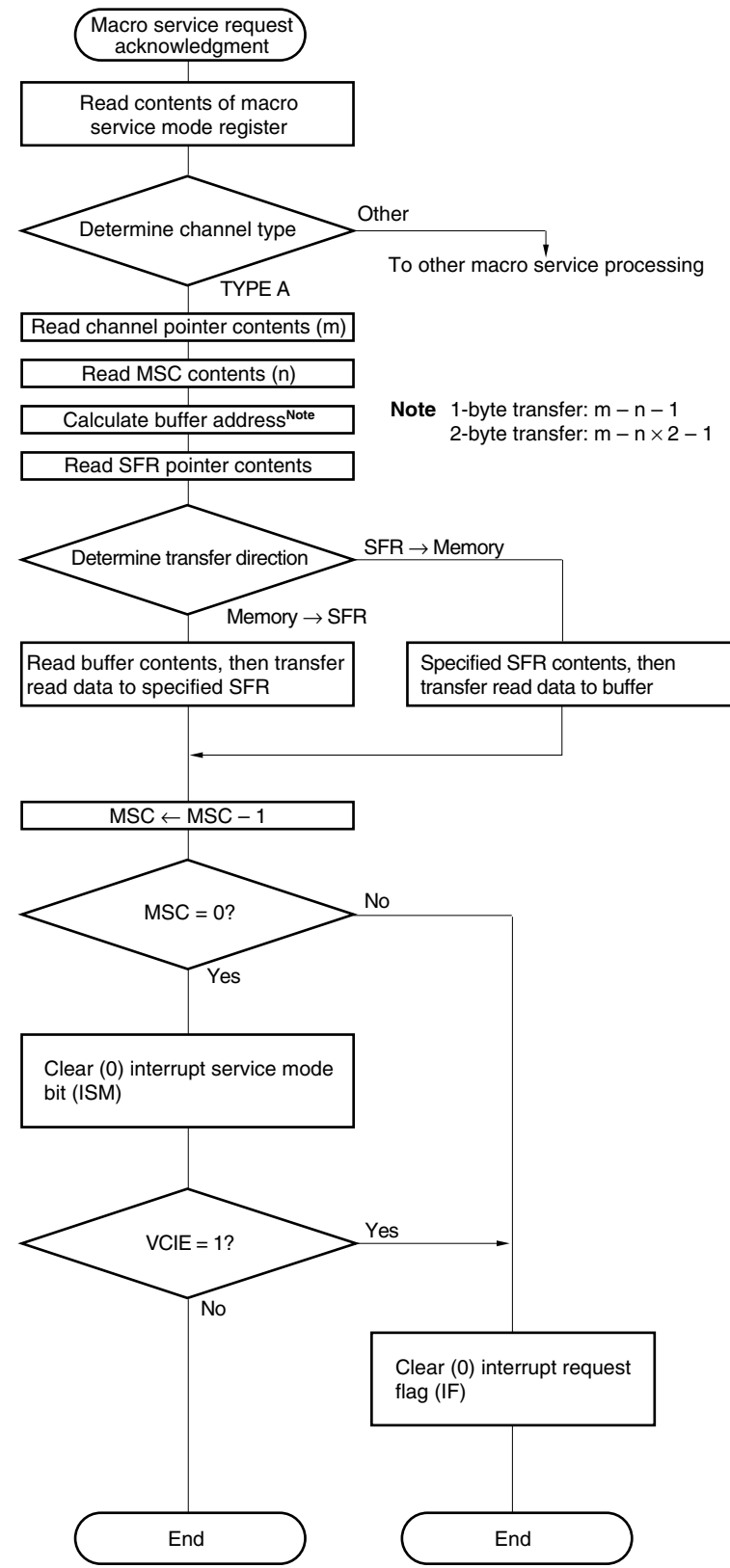
Data transfers are performed between buffer memory in the macro service channel and an SFR specified in the macro service channel.

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 16-23. Macro Service Data Transfer Processing Flow (Type A)



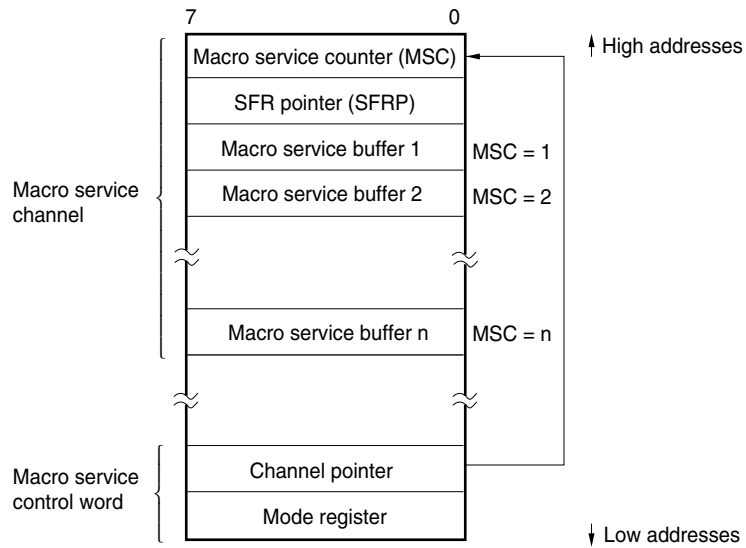
(Vectored interrupt request generation)

**(2) Macro service channel configuration**

The channel pointer and 8-bit macro service counter (MSC) indicate the buffer address in internal RAM (FE06H to FE1DH when the LOCATION 0H instruction is executed, or FFE06H to FFE1DH when the LOCATION 0FH instruction is executed) which is the transfer source or transfer destination (refer to **Figure 16-24**). In the channel pointer, the lower 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 lower 8 bits of the SFR address are written to the SFRP.

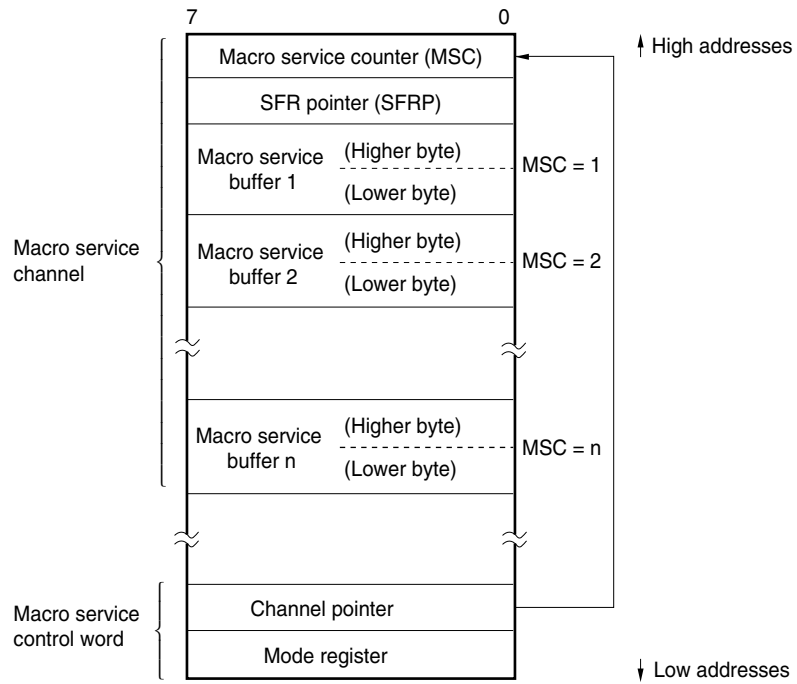
**Figure 16-24. Type A Macro Service Channel**

**(a) 1-byte transfers**



$$\text{Macro service buffer address} = (\text{channel pointer}) - (\text{macro service counter}) - 1$$

(b) 2-byte transfers



$$\text{Macro service buffer address} = (\text{channel pointer}) - (\text{macro service counter}) \times 2 - 1$$

16.8.7 Macro service type B

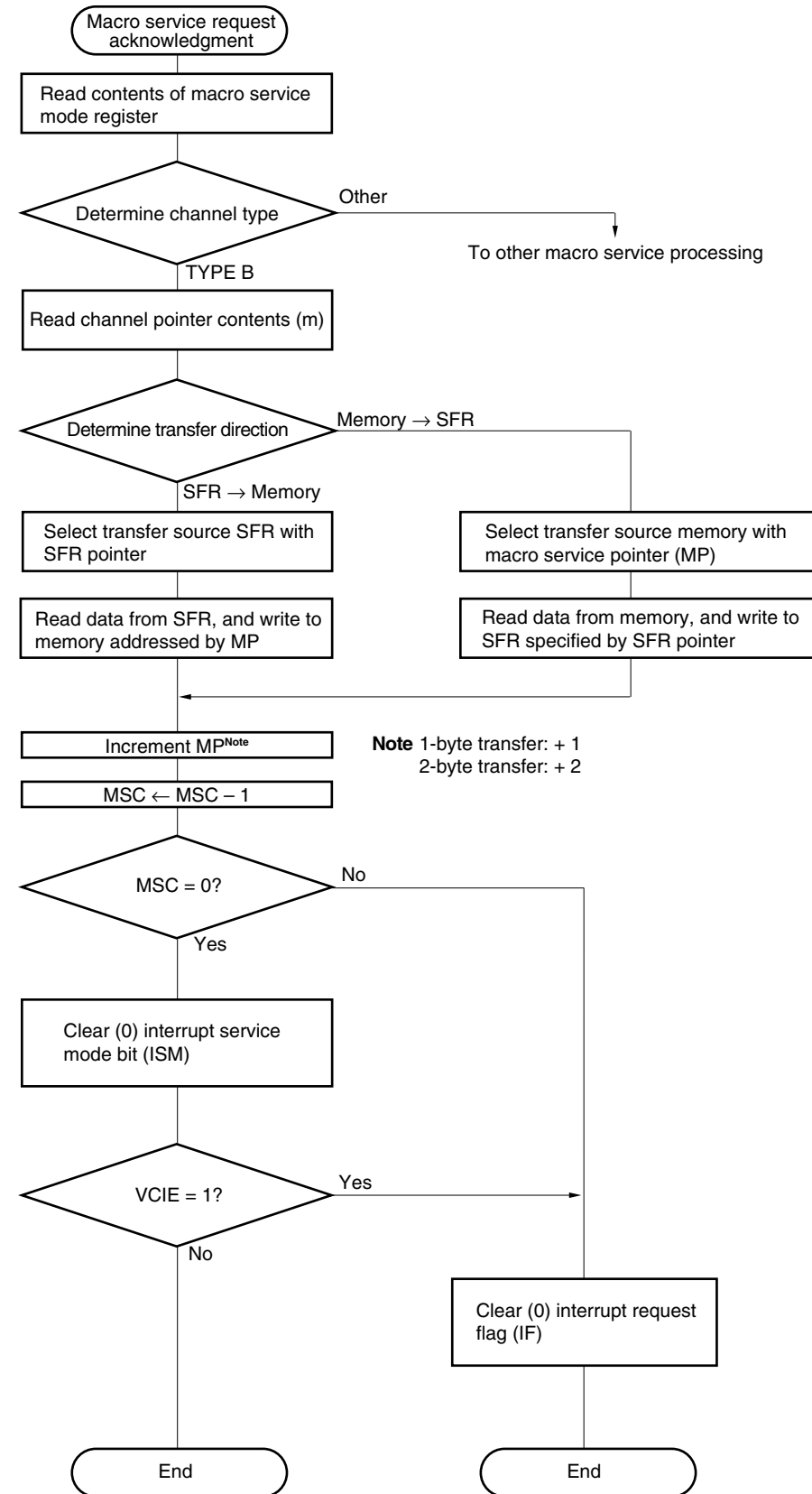
(1) Operation

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 KB of data buffer area when 8-bit data is transferred or 128 KB of data buffer area when 16-bit data is transferred can be set in 1 MB of any address space.

Figure 16-25. 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 MB memory space that is the transfer destination or transfer source.

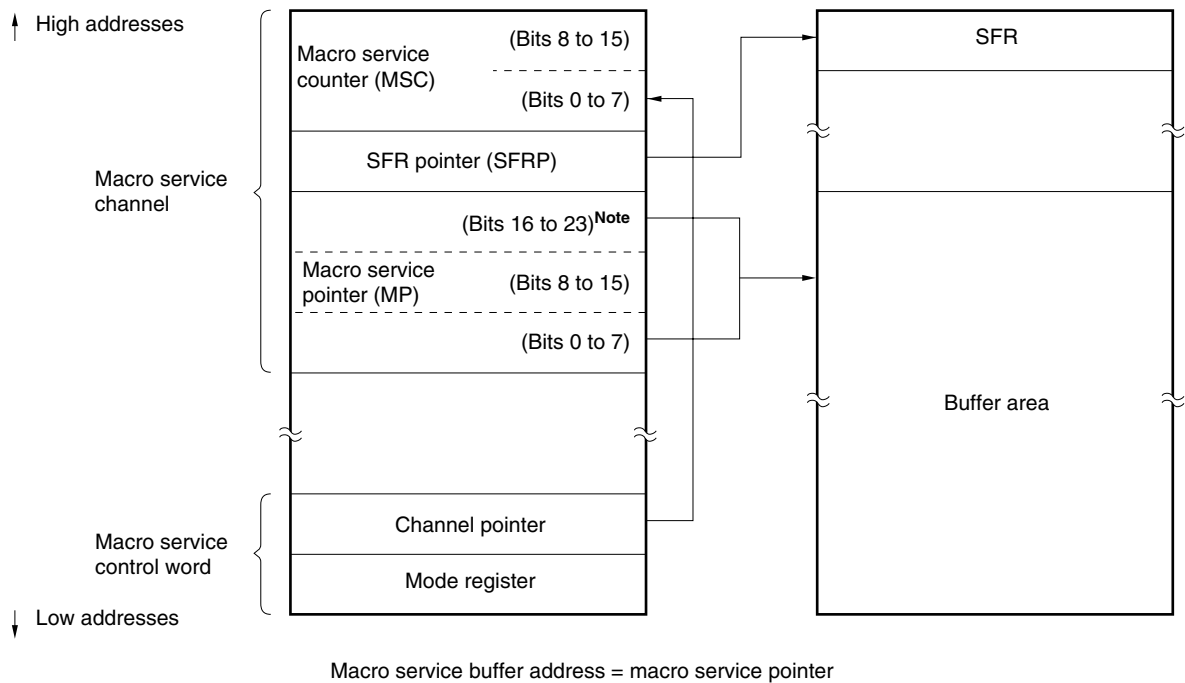
The lower 8 bits of the SFR that is the transfer destination or transfer source is written to the SFR pointer (SFRP).

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 0FE06H to 0FE1DH when the LOCATION 0H instruction is executed, or 0FFE06H to 0FFE1DH when the LOCATION 0FH instruction is executed.

The macro service channel is indicated by the channel pointer as shown in Figure 16-26. In the channel pointer, the lower 8 bits of the address are written to the macro service counter in the macro service channel.

**Figure 16-26. Type B Macro Service Channel**

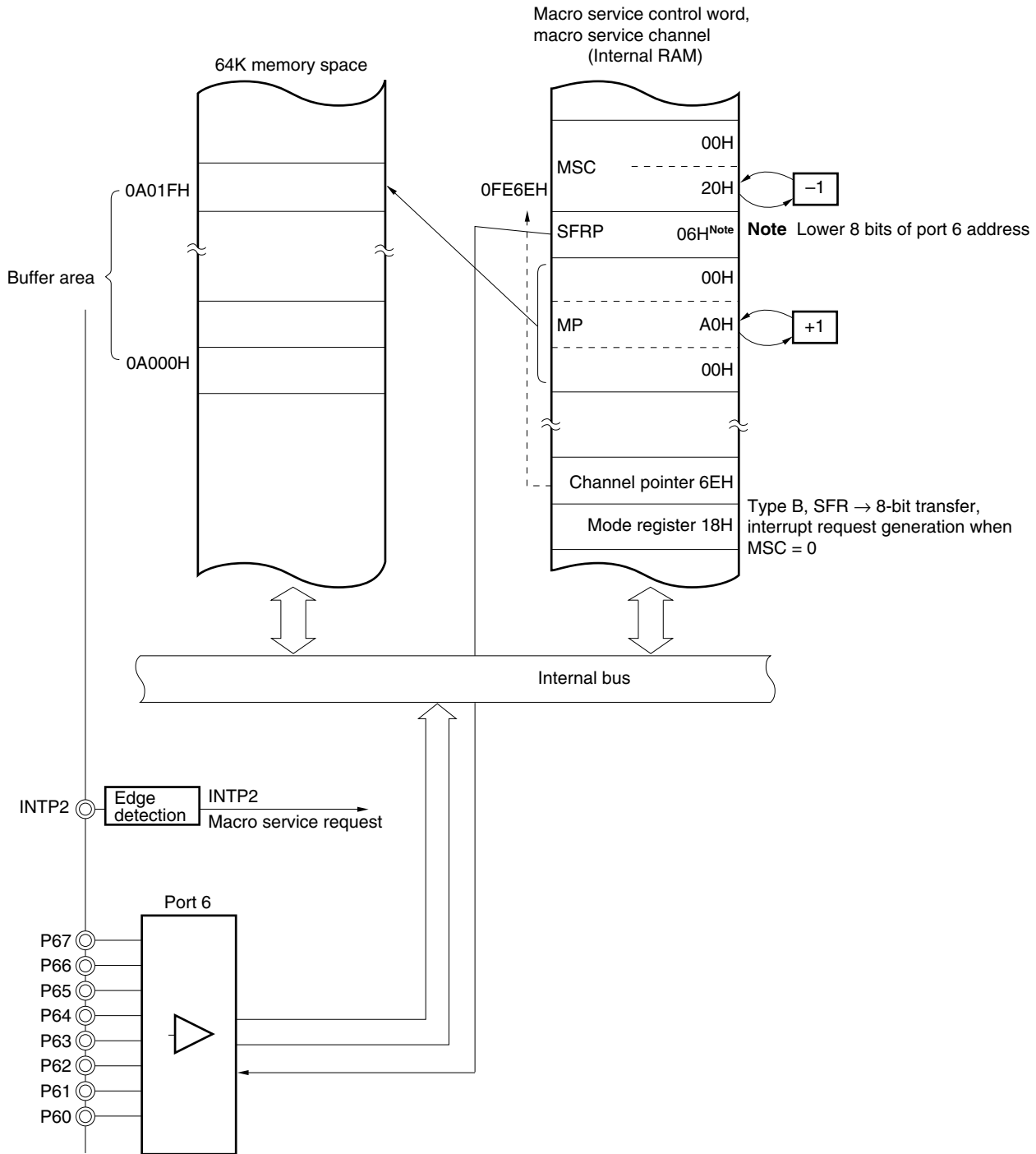


**Note** Be sure to set bits 20 to 23 to 0.

(3) Example of use of type B

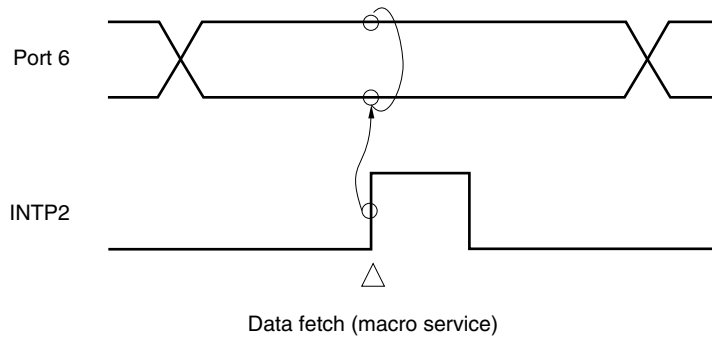
An example is shown below in which parallel data is input from port 6 in synchronization with an external signal. The INTP2 external interrupt pin is used for synchronization with the external signal.

Figure 16-27. Parallel Data Input Synchronized with External Interrupts



**Remark** Macro service channel addresses in the figure are the values when the LOCATION 0H instruction is executed. When the LOCATION 0FH instruction is executed, 0F0000H should be added to the values in the figure.

Figure 16-28. Timing of Parallel Data Input



### 16.8.8 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, 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 data macro service pointer (MPD).

#### (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 16-29. Macro Service Data Transfer Processing Flow (Type C) (1/2)

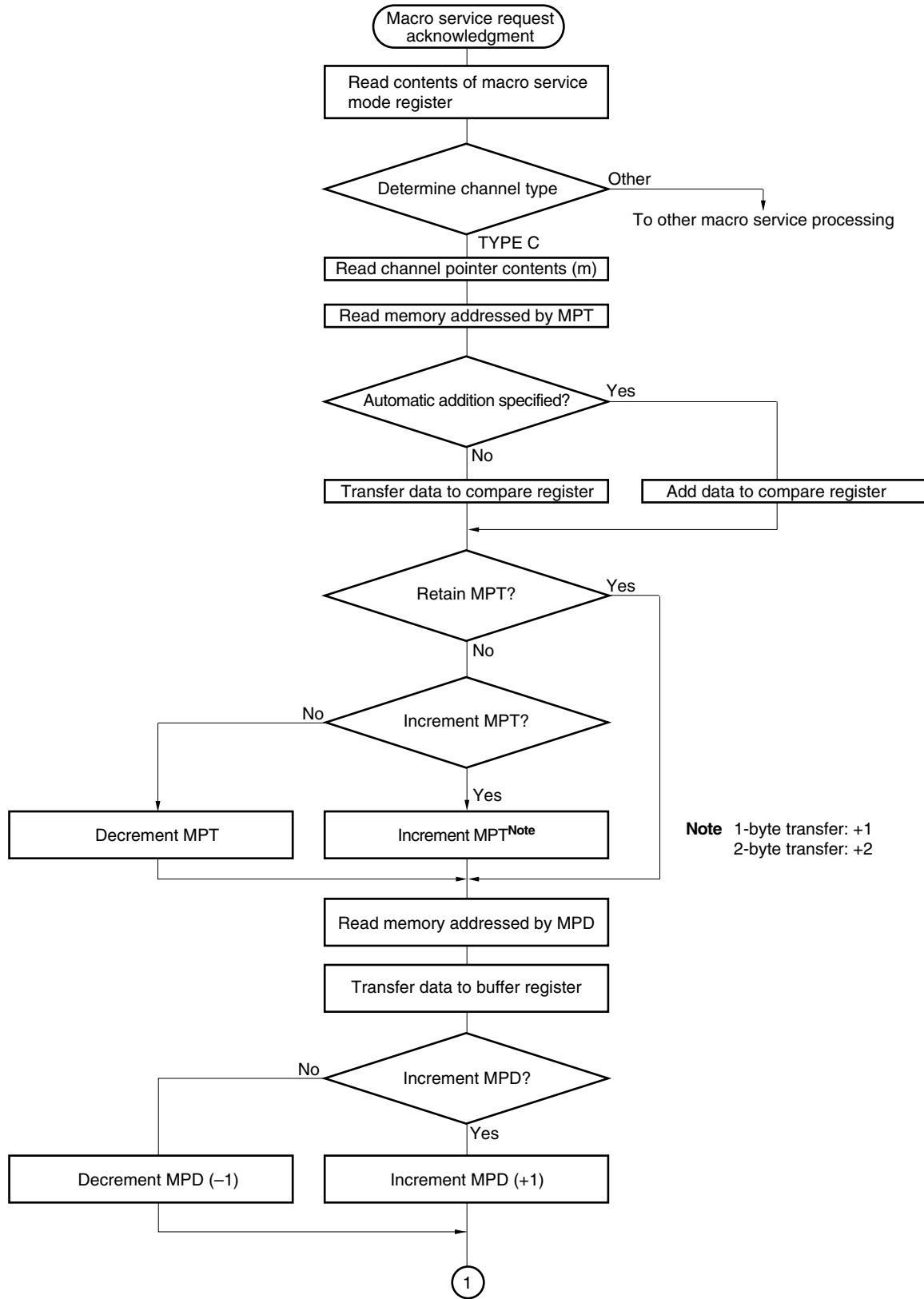
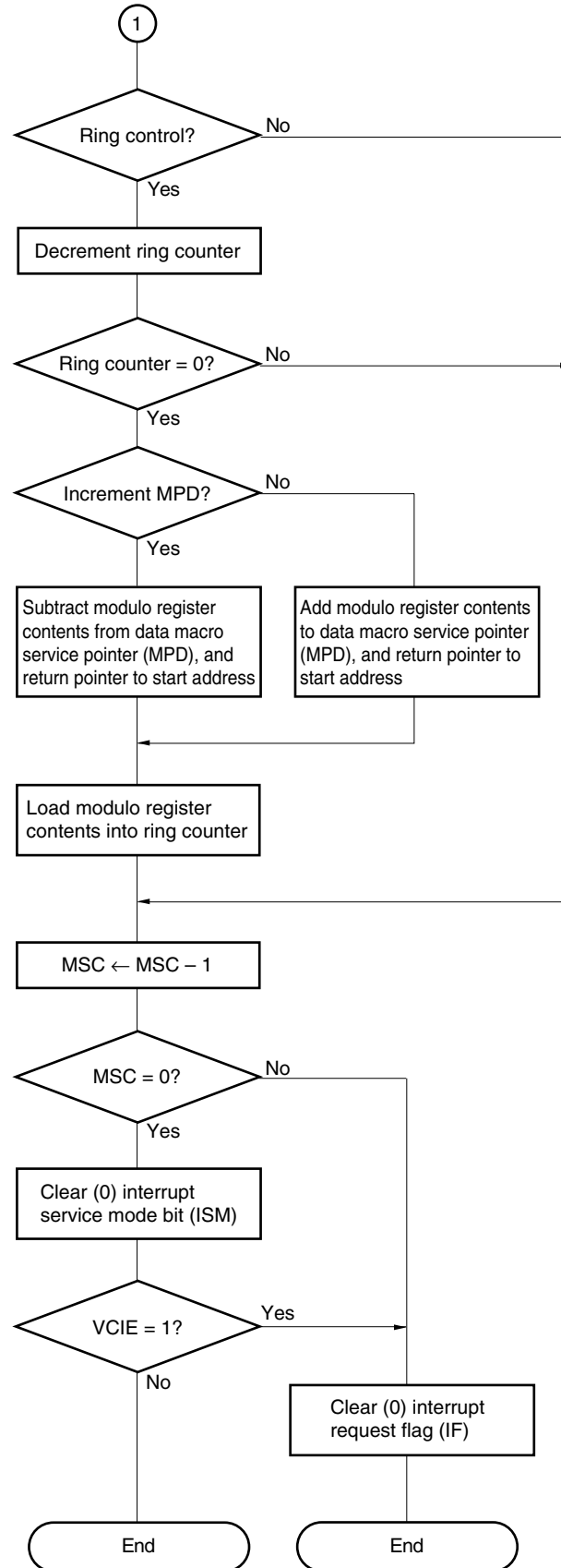


Figure 16-29. Macro Service Data Transfer Processing Flow (Type C) (2/2)



(Vectored interrupt request generation)

**(2) Macro service channel configuration**

There are two kinds of type C macro service channel, as shown in Figure 16-30.

The timer macro service pointer (MPT) mainly indicates the data buffer area in the 1 MB memory space to be transferred or added to the timer counter compare register.

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.

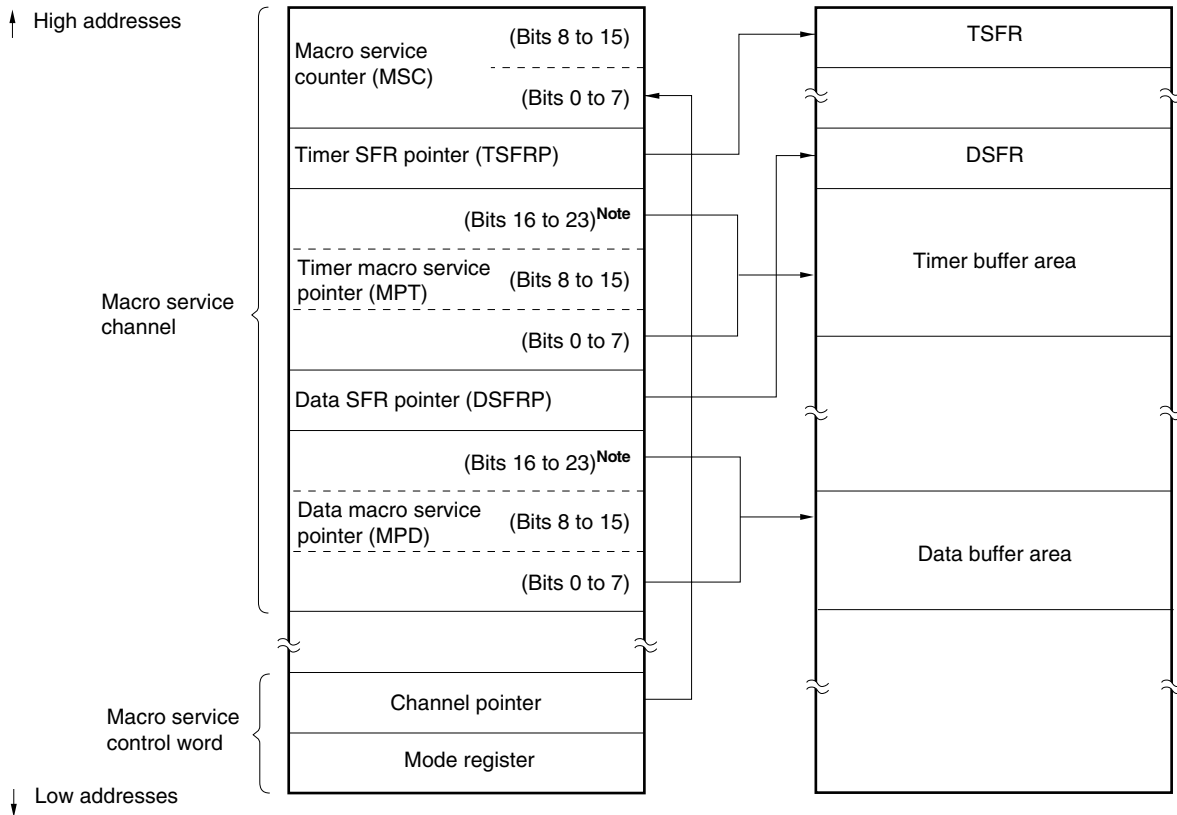
The macro service counter (MSC) is a 16-bit counter that specifies the number of data transfers.

The lower 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 0FE06H to 0FE1DH when the LOCATION 0H instruction is executed, or 0FFE06H to 0FFE1DH when the LOCATION 0FH instruction is executed. The macro service channel is indicated by the channel pointer as shown in Figure 16-30. In the channel pointer, the lower 8 bits of the address are written to the macro service counter in the macro service channel.

**Figure 16-30. Type C Macro Service Channel (1/2)**

**(a) No ring control**

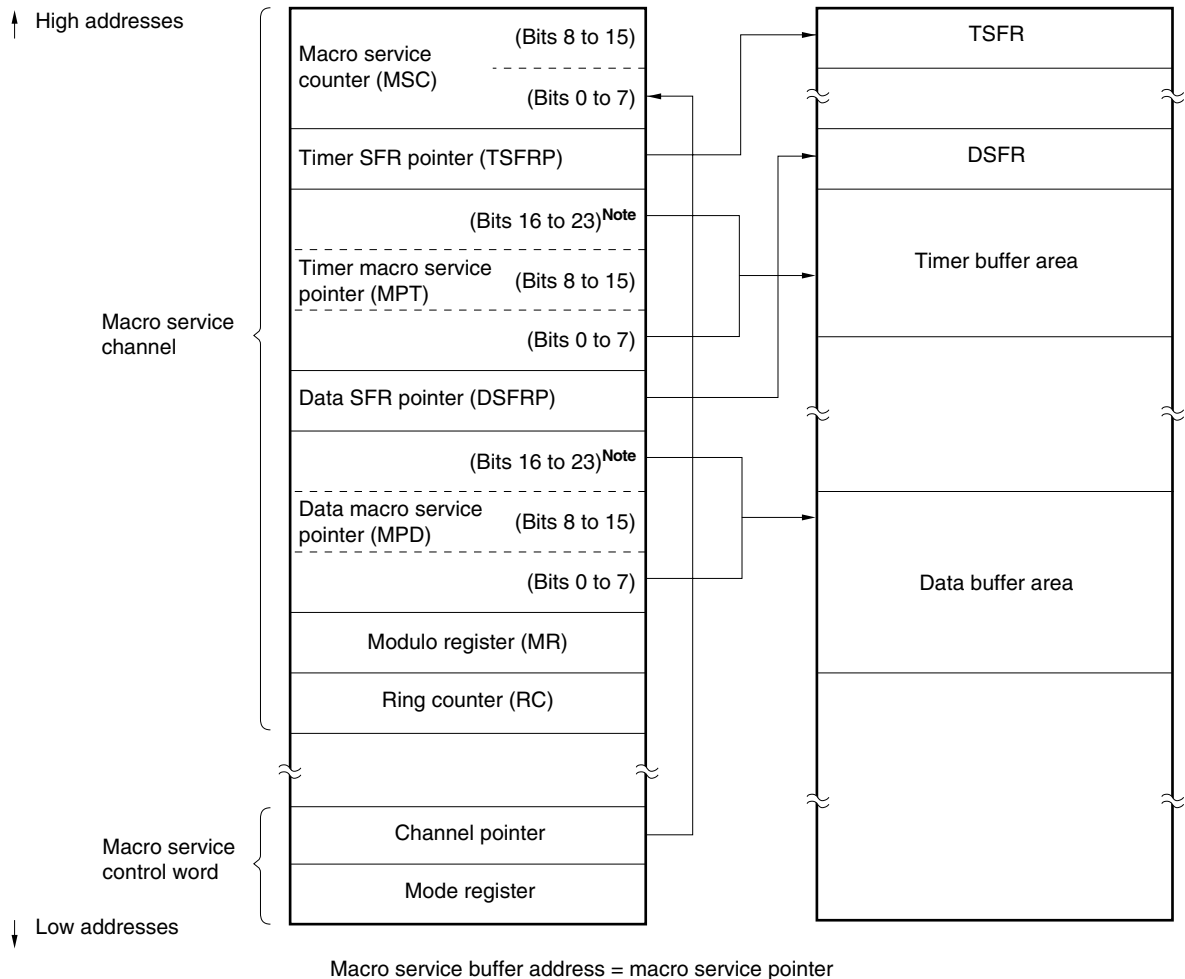


Macro service buffer address = macro service pointer

**Note** Be sure to set bits 20 to 23 to 0.

Figure 16-30. Type C Macro Service Channel (2/2)

## (b) With ring control



**Note** Be sure to set bits 20 to 23 to 0.

## (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  $MSC = 0$ .

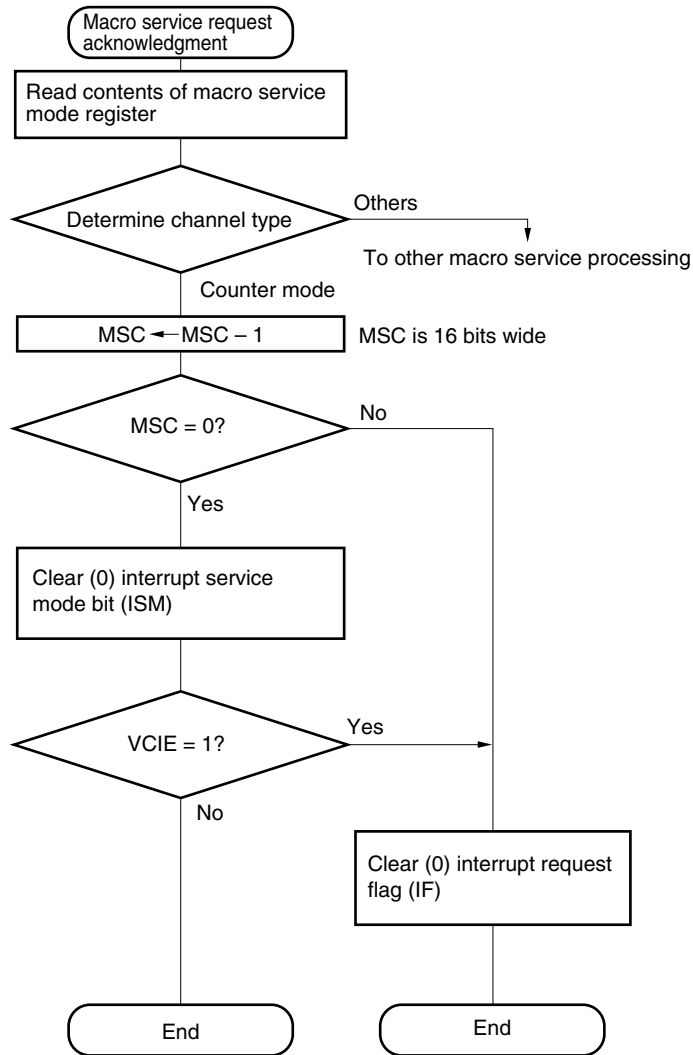
## 16.8.9 Counter mode

## (1) Operation

MSC is decremented the number of times preset 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 16-31. Macro Service Data Transfer Processing Flow (Counter Mode)



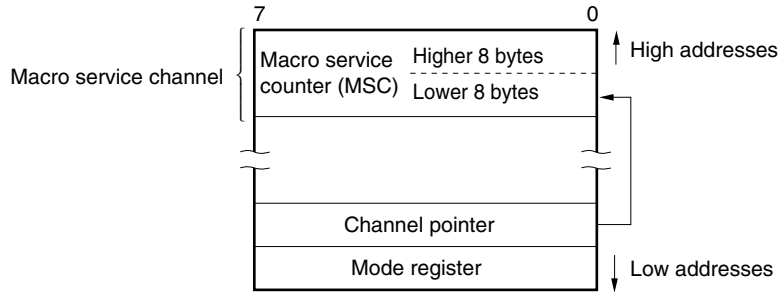
(Vectored interrupt request generation)



**(2) Configuration of macro service channel**

The macro service channel consists of only a 16-bit macro service counter (MSC). The lower 8 bits of the address of the MSC are written to the channel pointer.

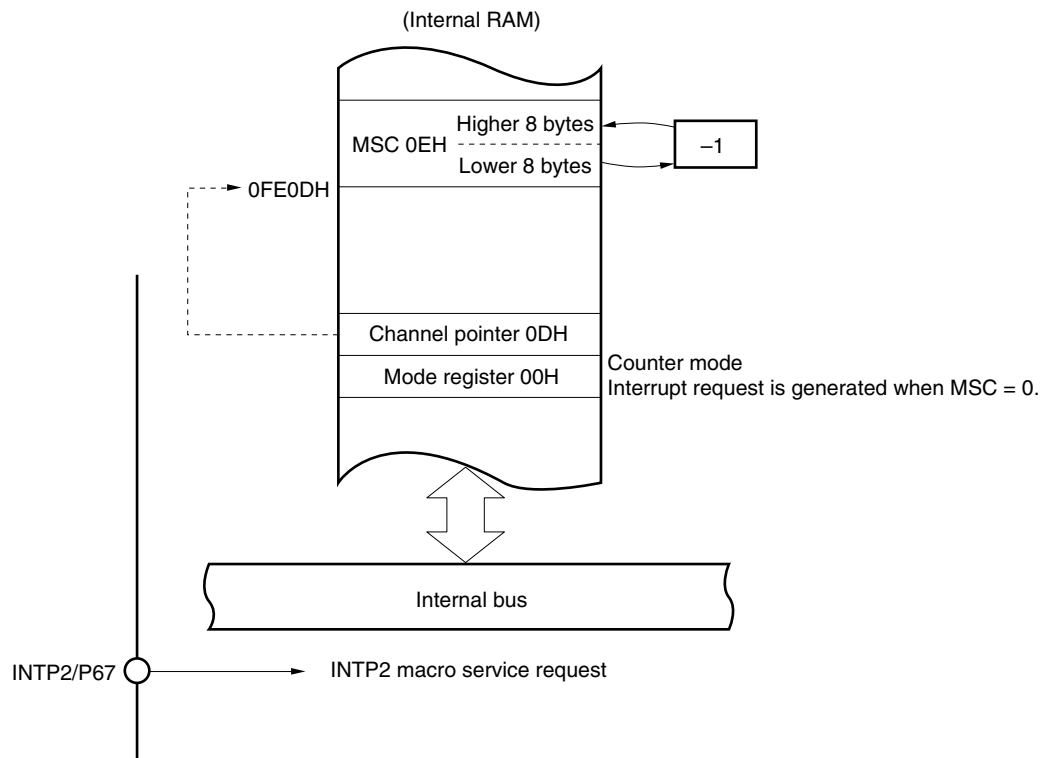
**Figure 16-32. Counter Mode**



**(3) Example of using counter mode**

Here is an example of counting the number of edges input to external interrupt pin INTP2.

**Figure 16-33. Counting Number of Edges**



**Remark** The internal RAM address in the figure above is the value when the LOCATION 0H instruction is executed.  
 When the LOCATION 0FH instruction is executed, add 0F0000H to this value.

## 16.9 When Interrupt Requests and Macro Service Are Temporarily Held Pending

When the following instructions are executed, interrupt acknowledgment and macro service processing is pending 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  
POPU post  
MOV PSWL, A  
MOV PSWL, #byte  
MOVG SP, #imm24

Write instruction and bit manipulation instruction (excluding BT and BF) to interrupt control registers<sup>Note</sup>, MK0, MK1L, IMC, and ISPR.

PSW bit manipulation instruction

(Excluding the BT PSWL.bit, \$addr20 instruction, BF PSWL.bit, \$addr20 instruction, BT PSWH.bit, \$addr20 instruction, BF PSWH.bit, \$addr20 instruction, SET1 CY instruction, NOT1 CY instruction, and CLR1 CY instruction)

**Note** Interrupt control registers: WDTIC, PIC0, PIC1, PIC2, TMIC00, TMIC01, KSIC, CSIIC0, CSIIC1, TMIC50, TMIC51, ADIC, REMIC, CSIIC2, SERIC0, SRIC0, STIC0, WTIC, WTIC

**Caution** If problems are caused by a long pending period for interrupts and macro service when the instructions to be applied are used in succession, insert an instruction such as NOP to create a timing that can receive interrupts and macro service requests without leaving them pending.

## 16.10 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:

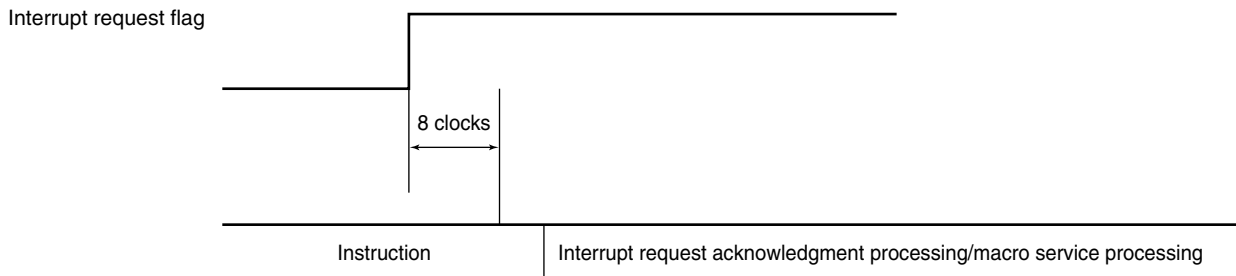
MOVM, XCHM, MOVBK, XCHBK  
CMPME, CMPMNE, CMPMC, CMPMNC  
CMPBKE, CMPBKNE, CMPBKC, CMPBKNC  
SACW

### 16.11 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 (refer to **16.9 When Interrupt Requests and Macro Service Are Temporarily Held Pending** for deferred instructions).

**Figure 16-34. Interrupt Request Generation and Acknowledgment (Unit: Clock =  $1/f_{\text{CLK}}$ )**



**16.11.1 Interrupt acknowledge processing time**

The time shown in Table 16-7 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 16-7. Interrupt Acknowledge Processing Time**

(Unit: Clock = 1/f<sub>CLK</sub>)

Vector Table	IROM						EMEM					
	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 0H 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
2. 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).
  3. If the vector table is EMEM, and if wait states are inserted in reading the vector table, add 2m 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.
  4. 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.
  5. 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.
  6. The number of wait states is the sum of the number of address wait states and the number of access wait states.

## 16.11.2 Processing time of macro service

Macro service processing time differs depending on the type of the macro service, as shown in Table 16-8.

**Table 16-8. Macro Service Processing Time**

(Units: Clock =  $1/f_{CLK}$ )

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.

## 16.12 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, etc., the entire system must be restored to its initial state. In the  $\mu$ PD784975A, 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  $\overline{\text{RESET}}$  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, #RETVAl  ; Forcibly change SP location
                RETI   ; Forcibly terminate running interrupt service program, return
                        address = IRESL

RETVAl:
                DW     LOWW (IRESL) ; Stack data to return to IRESL with RETI instruction
                DB     0
                DB     HIGHW (IRESL) ; LOWW and HIGHW are assembler operators for calculating
                        lower 16 bits and higher 16 bits, respectively

NEXT:

```

- After this, on-chip 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).

### 16.13 Cautions

- (1) The in-service priority register (ISPR) is read-only. Writing to this register may result in malfunction.
- (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. Use the RETB instruction.
- (4) The RETCS instruction must not be used to return from a software interrupt caused by a BRKCS instruction. Use the RETCSB instruction.
- (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 serviced 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. If you restart a program from the initial state after a no-maskable interrupt acknowledgement, refer to **16.12 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 priority 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 **16.9**. 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) (refer to **Table 3-6** in **3.8 Special Function Registers (SFRs)**), and the CPU becomes deadlocked, or an unexpected signal output from a pin, or 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 a software inadvertently loops.

Therefore, the program following  $\overline{\text{RESET}}$  release must be as follows.

```

CSEG  AT 0
      DW   STRT
CSEG  BASE
STRT:
      LOCATION 0FH; or LOCATION 0H
      MOVG SP, #imm24

```

- (10) If problems are caused by a long pending period for interrupts and macro service when the instructions to be applied are used in succession, insert an instruction such as NOP to create a timing that can receive interrupts and macro service requests without leaving them pending.



## CHAPTER 17 STANDBY FUNCTION

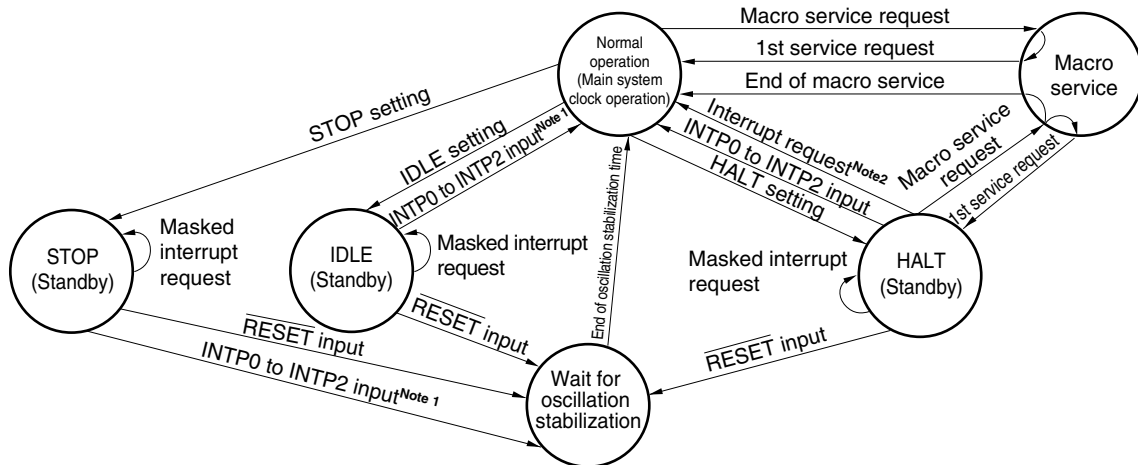
### 17.1 Configuration and Function

The  $\mu$ PD784975A 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 17-1.

Figure 17-1. Standby Mode Transition Diagram



**Notes** 1. When INTPO to INTP2 are not masked

2. Unmasked interrupt request only

**Remark** The watchdog timer must not be used to release the standby mode (STOP, HALT, or IDLE mode).

## 17.2 Control Registers

### 17.2.1 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, STBC 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.

STBC can be read at any time using a data transfer instruction.

RESET input sets STBC to 30H.

Figure 17-2 shows the format of STBC.

Figure 17-2. Format of Standby Control Register (STBC)

Address: 0FFC0H After reset: 30H R/W

Symbol	7	6	5	4	3	2	1	0
STBC	0	0	CK1	CK0	0	0	STP	HLT

CK1	CK0	CPU clock selection <sup>Note</sup> (in through-rate clock mode or oscillation division mode)
0	0	$f_{xx}$ ( $f_x$ , $f_x/2$ )
0	1	$f_{xx}/2$ ( $f_x/2$ , $f_x/2^2$ )
1	0	$f_{xx}/2^2$ ( $f_x/2^2$ , $f_x/2^3$ )
1	1	$f_{xx}/2^3$ ( $f_x/2^3$ , $f_x/2^4$ )

STP	HLT	Operation specification flag
0	0	Normal operation mode
0	1	HALT mode (cleared automatically when HALT mode is released)
1	0	STOP mode (cleared automatically when STOP mode is released)
1	1	IDLE mode (cleared automatically when IDLE mode is released)

**Note** A CPU clock can also be selected using the oscillation mode select register (CC).

- 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 (to 1) before setting the STOP mode. If the STOP mode is used with the EXTC bit of the OSTS cleared (to 0) when using external clock input, the  $\mu$ PD784975A may suffer damage or its reliability may be degraded. 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.
  2. Execute an NOP instruction three times after the standby instruction (after the standby mode has been released). Otherwise, the standby instruction cannot be executed if execution of the standby instruction and an interrupt request contend, and the interrupt is acknowledged after two or more instructions following the standby instruction have been executed. The instruction that is executed before acknowledging the interrupt is the one that is executed within up to 6 clocks after the standby instruction has been executed.

**Example**

```
MOV STBC, #byte
NOP
NOP
NOP
```

**Remark**  $f_{xx}$ : Main system clock frequency ( $f_x$  or  $f_x/2$ )  
 $f_x$ : Main system clock oscillation frequency

### 17.2.2 Oscillation stabilization time specification register (OSTS)

OSTS specifies the oscillator operation and the oscillation stabilization time when STOP mode is released. The EXTC bit of 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 (to 1).

Bits OSTS0 to OSTS2 of OSTS select the oscillation stabilization time when STOP mode is released. In general, an oscillation stabilization time of at least 40 ms 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.

OSTS is set using a 1-bit or 8-bit transfer instruction.

$\overline{\text{RESET}}$  input sets OSTS to 00H.

Figure 17-3 shows the format of OSTS.

Figure 17-3. Format of Oscillation Stabilization Time Specification Register (OSTS)

Address: 0FFCFH After reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
OSTS	EXTC	0	0	0	0	OSTS2	OSTS1	OSTS0

EXTC	External clock selection
0	When crystal/ceramic oscillation is used
1	When external clock is used

EXTC	OSTS2	OSTS1	OSTS0	Oscillation stabilization time selection
0	0	0	0	$2^{19}/f_{xx}$ (41.9 ms)
0	0	0	1	$2^{18}/f_{xx}$ (21.0 ms)
0	0	1	0	$2^{17}/f_{xx}$ (10.5 ms)
0	0	1	1	$2^{16}/f_{xx}$ (5.2 ms)
0	1	0	0	$2^{15}/f_{xx}$ (2.6 ms)
0	1	0	1	$2^{14}/f_{xx}$ (1.3 ms)
0	1	1	0	$2^{13}/f_{xx}$ (655 $\mu$ s)
0	1	1	1	$2^{12}/f_{xx}$ (328 $\mu$ s)
1	×	×	×	$512/f_{XT}$ (41.0 $\mu$ s)

- Cautions**
1. When crystal/ceramic oscillation is used, the EXTC bit must be cleared (to 0) before use. If the EXTC bit is set (to 1), oscillation will stop.
  2. If the STOP mode is used when using external clock input, the EXTC bit must be set (to 1) before setting the STOP mode. If the STOP mode is used with the EXTC bit cleared (to 0), the  $\mu$ PD784975A may suffer damage or its reliability may be degraded.
  3. 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. When the EXTC bit is set (to 1), the  $\mu$ PD784975A only operates with the clock input to the X2 pin.

- Remarks**
1. The values in parentheses are valid for operation when  $f_{xx}$  is 12.5 MHz.
  2. ×: don't care

## 17.3 HALT Mode

### 17.3.1 HALT mode setting and operating states

The HALT mode is selected by setting (to 1) the HLT bit of the standby control register (STBC).

The only writes that can be performed on 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.

If interrupts are enabled (when the IE bit of the program status word (PSW) is set to 1, write a NOP instruction three times after the instruction that sets the HALT mode (after releasing the HALT mode). Otherwise, two or more instructions may be executed before an interrupt is acknowledged (after releasing the HALT mode). As a result, the execution sequence of the interrupt processing and instructions may be changed. To prevent troubles due to changes in the execution sequence, the above processing is necessary.

**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, etc. before entering HALT mode.

**Table 17-1. Operating States in HALT Mode**

Clock oscillator	Operating
Internal system clock	Operating
CPU	Operation stopped <sup>Note</sup>
I/O lines	Retain state prior to HALT mode setting
Peripheral functions	Continue operating
Internal RAM	Retained

**Note** Macro service processing is executed.

### 17.3.2 HALT mode release

HALT mode can be released by the following two sources.

- 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 17-2. Figure 17-4 shows operations after HALT mode release.

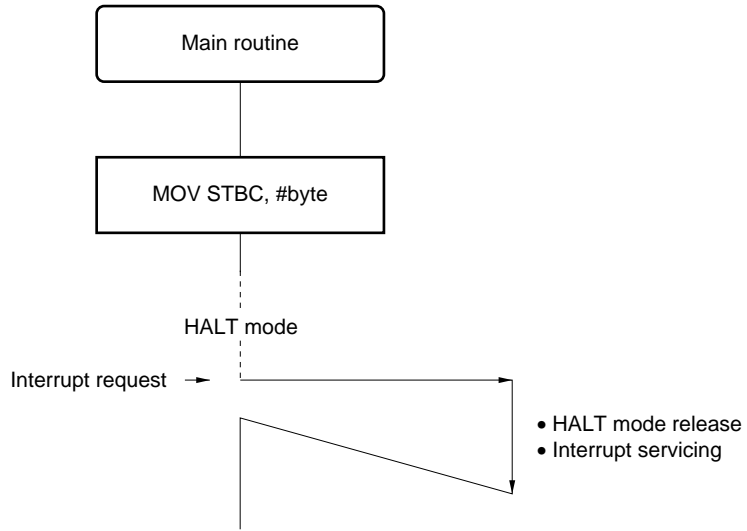
Table 17-2. HALT Mode Release and Operations After Release

Release Source	MK <sup>Note 1</sup>	IE <sup>Note 2</sup>	State on Release	Operation After Release
RESET input	×	×	—	Normal reset operation
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>Note 4</sup> cleared (to 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>Note 4</sup> is cleared (to 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>Note 3</sup> )
	0	0	—	
	1	×	—	HALT mode maintained
Macro service request	0	×	—	Macro service processing execution End condition not established → HALT mode again End condition established → If VCIE <sup>Note 5</sup> = 1: HALT mode again If VCIE <sup>Note 5</sup> = 0: Same as release by maskable interrupt request
			1	×

- Notes**
1. Interrupt mask bit in individual interrupt request source
  2. Interrupt enable flag in the program status word (PSW)
  3. Pending interrupt requests are acknowledged when acknowledgment becomes possible.
  4. Bit in the interrupt mode control register (IMC)
  5. Bit in macro service mode register of macro service control word in individual macro service request source

Figure 17-4. Operation After HALT Mode Release (1/4)

(1) When interrupt generates after HALT mode has been set



(2) Reset after HALT mode has been set

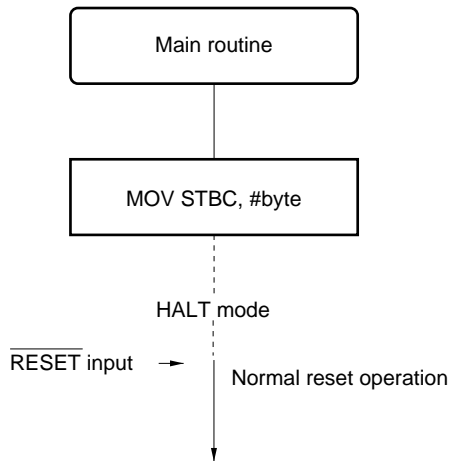
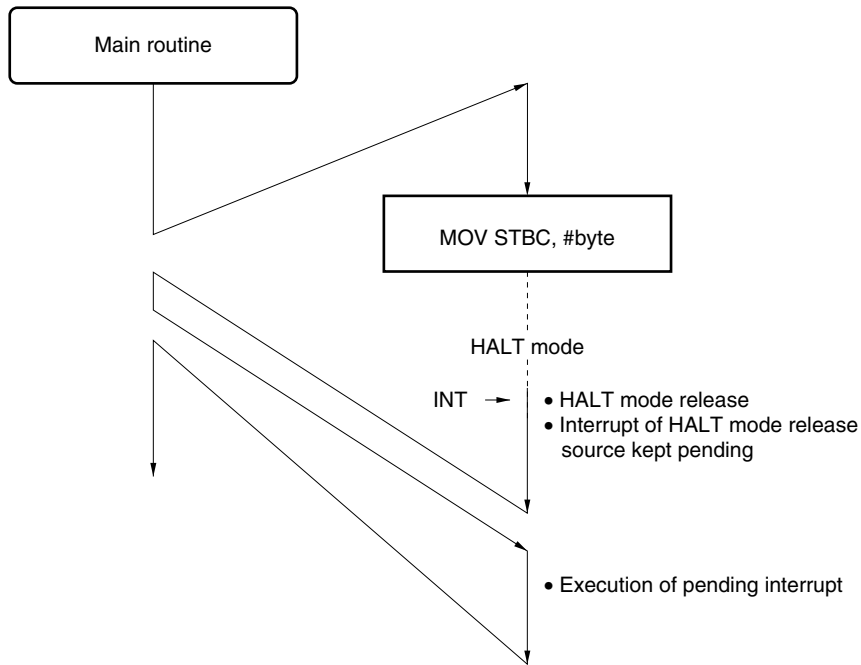




Figure 17-4. Operation After HALT Mode Release (2/4)

(3) When HALT mode is set while interrupt routine with priority higher than or same as that of interrupt of release source



(4) When HALT mode is set while interrupt routine with priority lower than that of interrupt of release source

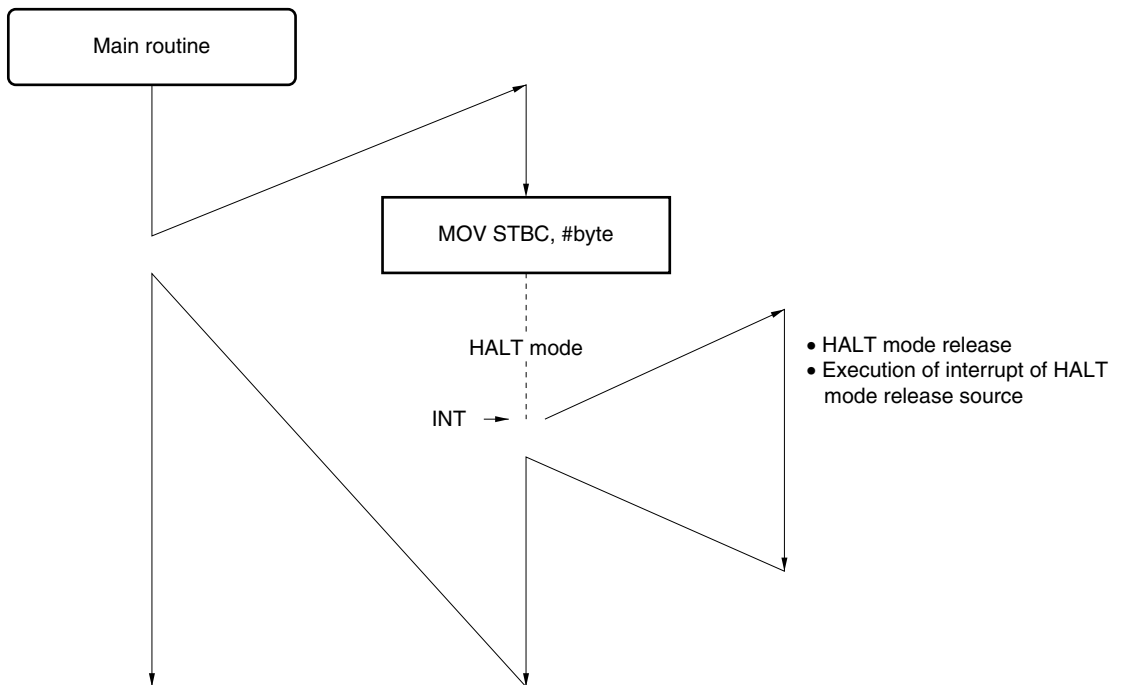
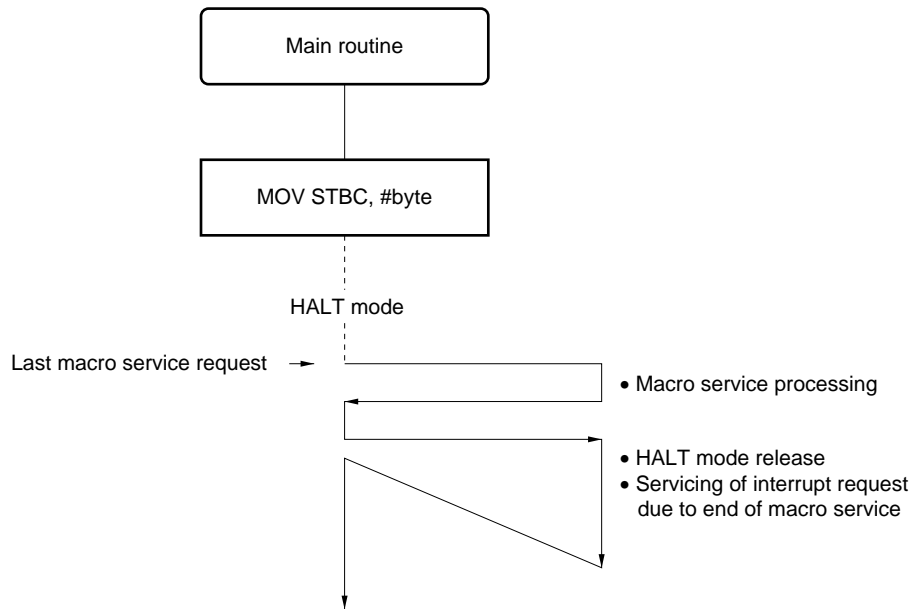


Figure 17-4. Operation After HALT Mode Release (3/4)

(5) When macro service request is generated in HALT mode

(a) When end condition of macro service is satisfied and interrupt request is generated immediately (VCIE = 0)



(b) When end condition of macro service is not satisfied, or if end condition of macro service is satisfied but interrupt request is not generated immediately (VCIE = 1)

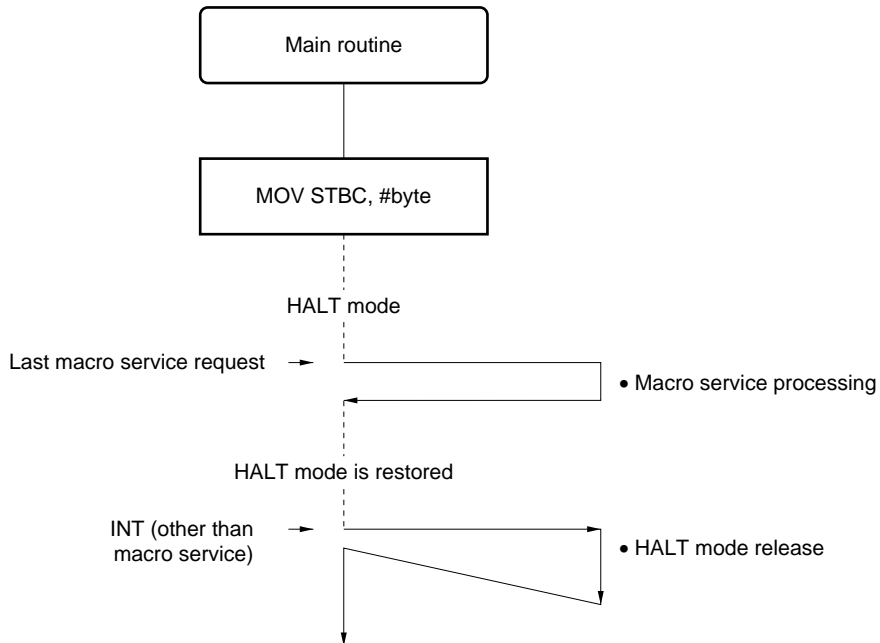
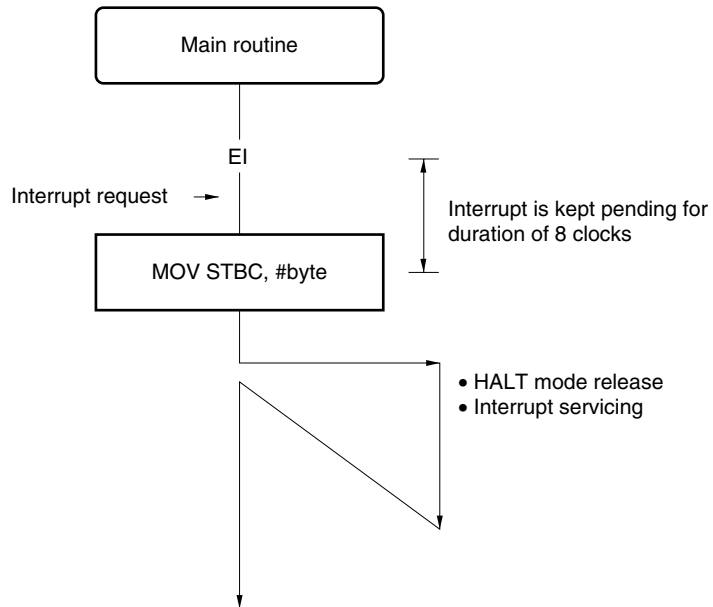
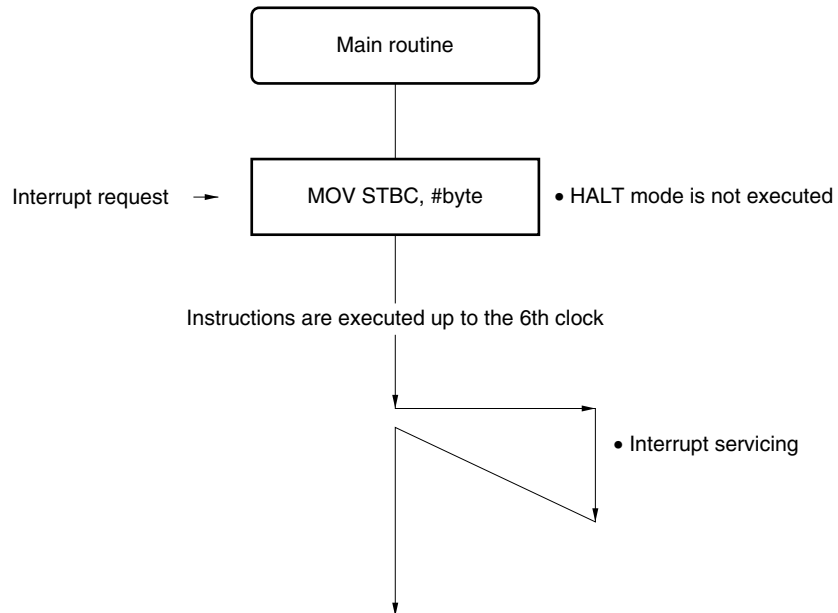


Figure 17-4. Operation After HALT Mode Release (4/4)

(6) When interrupt generates during execution of instruction that temporarily keeps interrupt pending, and if HALT mode is set while that interrupt is kept pending



(7) When HALT instruction and interrupt contend



**(1) Release by maskable interrupt request**

The HALT mode release by a maskable interrupt request can only be performed by an interrupt for which the interrupt mask flag is 0.

When the HALT mode is released, if an interrupt can be acknowledged when the interrupt request enable flag (IE) is set (to 1), a branch is made to the interrupt service program. If the interrupt cannot be acknowledged and if the IE flag is cleared (to 0), execution is resumed from the instruction following the instruction that set the HALT mode. Refer to **16.7 Maskable Interrupt Acknowledgment Operation** for details of interrupt acknowledgment.

With macro service, the HALT mode is released temporarily, service is performed once, then the HALT mode is restored. When macro service has been performed the specified number of times, the HALT mode is released if the VCIE bit in the macro service mode register of the macro service control word is cleared (to 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 (to 1), the HALT mode is entered again and is released by the next interrupt request.

Table 17-3. HALT Mode Release by Maskable Interrupt Request

Release Source	MK <sup>Note 1</sup>	IE <sup>Note 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>Note 4</sup> cleared (to 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>Note 4</sup> is cleared (to 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>Note 3</sup> )
	0	0	—	
	1	×	—	HALT mode maintained
Macro service request	0	×	—	Macro service processing execution End condition not established → HALT mode again End condition established → If VCIE <sup>Note 5</sup> = 1: HALT mode again If VCIE <sup>Note 5</sup> = 0: Same as release by maskable interrupt request
			1	×

- Notes**
1. Interrupt mask bit in individual interrupt request source
  2. Interrupt enable flag in the program status word (PSW)
  3. Pending interrupt requests are acknowledged when acknowledgment becomes possible.
  4. Bit in the interrupt mode control register (IMC)
  5. Bit in macro service mode register of macro service control word in individual macro service request source

## (2) 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.

## 17.4 STOP Mode

### 17.4.1 STOP mode setting and operating states

The STOP mode is selected by setting (to 1) the STP bit of the standby control register (STBC).

The only writes that can be performed on STBC 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.

If interrupts are enabled (when the IE bit of the program status word (PSW) is set to 1), write a NOP instruction three times after the instruction that sets the STOP mode (after releasing the STOP mode). Otherwise, two or more instructions may be executed before an interrupt is acknowledged. As a result, the execution sequence of the interrupt processing and instructions may be changed. To prevent troubles due to changes in the execution sequence, the above processing is necessary.

**Caution** If the STOP mode is set when the condition to release the HALT mode is satisfied (refer to 17.3.2 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 17-4. 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
16-bit timer/event counter	Operation stopped
8-bit PWM timer	Operable only when an external input clock (TIO50, TIO51) is selected as the count clock
Watchdog timer	Stopped (timer is initialized)
A/D converter	Operation stopped <sup>Note 1</sup>
3-wire serial interface	Operation stopped <sup>Note 2</sup>
Asynchronous serial interface	Operation stopped <sup>Note 3</sup>
Watch timer	Operation stopped
External interrupt (INTP0 to INTP2)	Operable
Internal RAM	Retained

- Notes**
1. A/D converter operation is stopped, but if the ADCS bit of the A/D converter mode register (ADM) is set (to 1), the current consumption does not decrease.
  2. The serial input pin supports a 3 V interface (i.e., it is a low-threshold pin). To prevent current being input to the Schmitt input buffer in STOP and IDLE modes, therefore, the Schmitt input buffer is turned off (the buffer output is “L”). Disable the serial interface (SIO0, SIO1, SIO2, and UART) before setting STOP or IDLE mode, and re-set the interface after STOP or IDLE mode has been released.

- Cautions**
- 1.** When the STOP mode is used in a system that uses an external clock, the EXTC bit of OSTS must be set (to 1). If STOP mode setting is performed in a system to which an external clock is input when the EXTC bit of OSTS is cleared (to 0), the current consumption increases. 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 5.4 Main System Clock Oscillator).
  - 2.** The ADCS bit of the A/D converter mode register (ADM) should be cleared (to 0).

**17.4.2 STOP mode release**

The STOP mode is released by INTP0 to INTP2 input, and  $\overline{\text{RESET}}$  input.

Release sources and an outline of operations after release are shown in Table 17-5. Figure 17-5 shows operations after STOP mode release.

**Table 17-5. STOP Mode Release and Operations After Release**

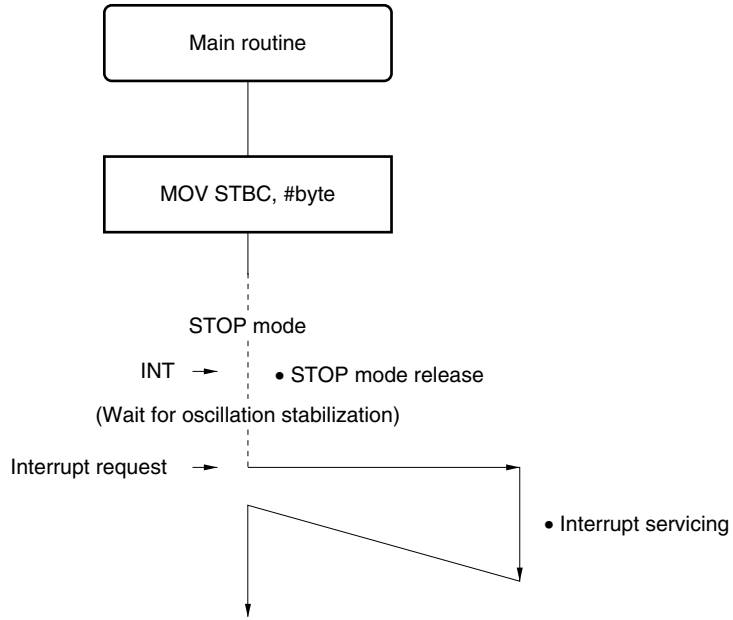
Release Source	MK <sup>Note 1</sup>	ISM <sup>Note 2</sup>	IE <sup>Note 3</sup>	State on Release	Operation After Release
$\overline{\text{RESET}}$ input	×	×	×	—	Normal reset operation
INTP0 to INTP2 pin input	0	0	1	<ul style="list-style-type: none"> <li>• Interrupt service program not being executed</li> <li>• Low-priority maskable interrupt service program being executed</li> <li>• PRSL bit<sup>Note 5</sup> cleared (to 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>Note 5</sup> is cleared (to 0), excluding execution of priority level 3 interrupt service program)</li> <li>• High-priority interrupt service program being executed</li> </ul>	
	0	0	0	—	STOP mode maintained
	1	0	×	—	
×	1	×	—		

- Notes**
1. Interrupt mask bit in individual interrupt request source
  2. Macro service enable flag in individual interrupt request source
  3. Interrupt enable flag in the program status word (PSW)
  4. Pending interrupt requests are acknowledged when acknowledgment becomes possible.
  5. Bit in the interrupt mode control register (IMC)



Figure 17-5. Operation After STOP Mode Release (1/3)

(1) When interrupt generates after STOP mode has been set



(2) Reset after STOP mode has been set

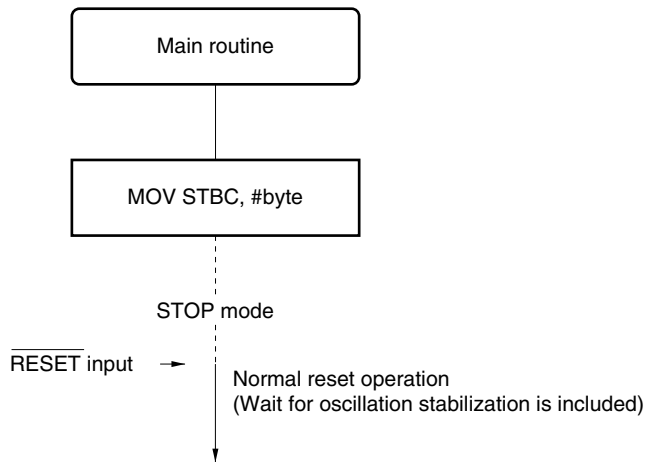
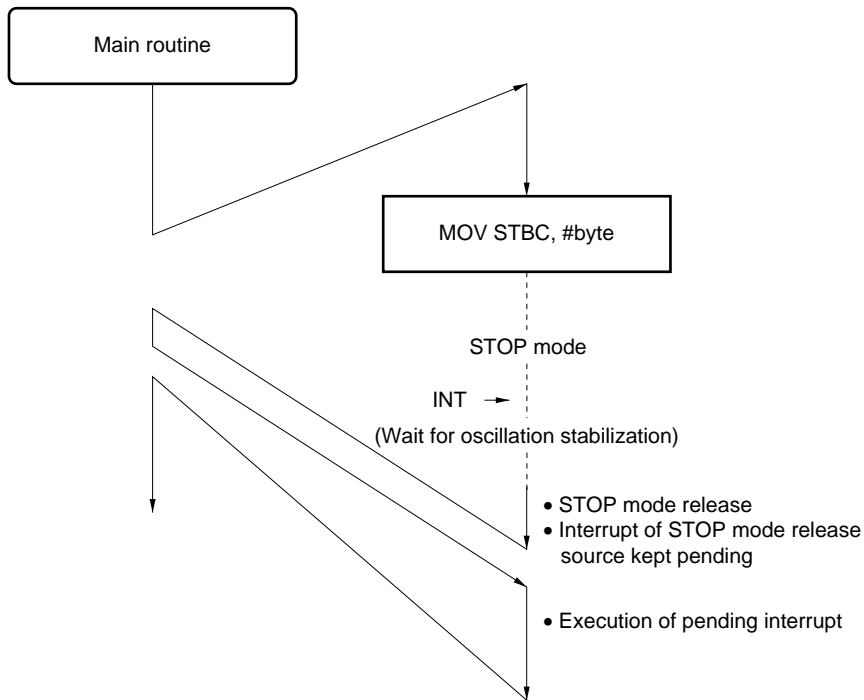
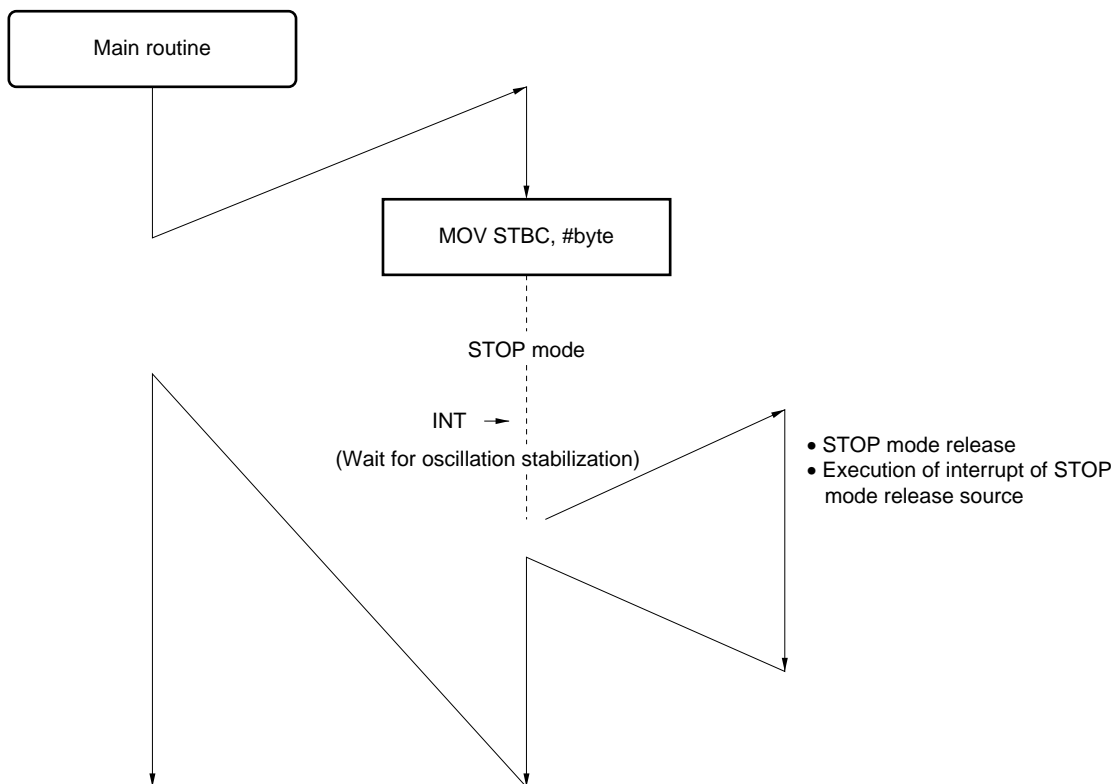


Figure 17-5. Operation After STOP Mode Release (2/3)

(3) When STOP mode is set while interrupt routine with priority higher than or same as that of interrupt of release source



(4) When STOP mode is set while interrupt routine with priority lower than that of interrupt of release source



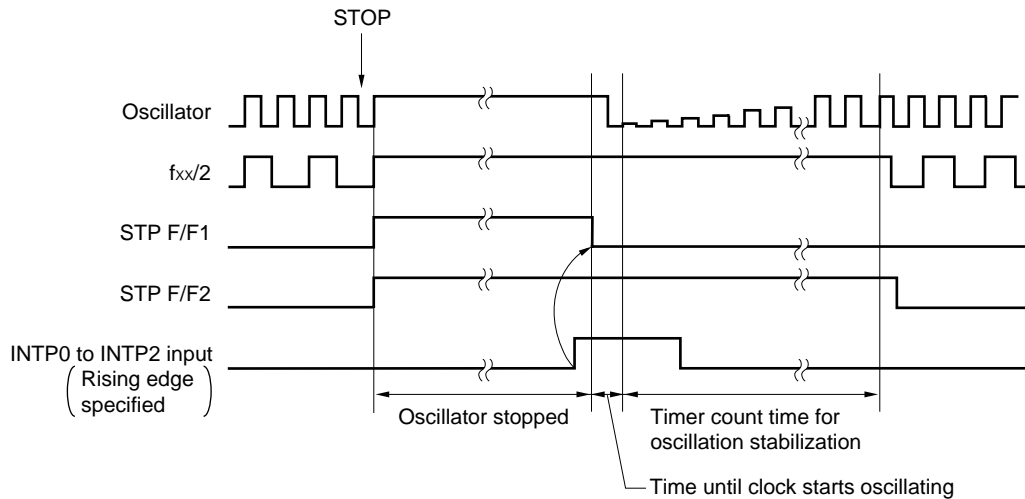


**(1) STOP mode release by INTP0 to INTP2 input**

When masking of interrupts by INTP0 to INTP2 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 INTP0 to INTP2 input. Following this, the STOP mode is released after the oscillation stabilization time specified by the oscillation stabilization time specification register (OSTS) elapses.

When the  $\mu$ PD784975A is released from the STOP mode, if an interrupt can be acknowledged when the interrupt enable flag (IE) is set (to 1), a branch is made to the interrupt service program. If the interrupt cannot be acknowledged and if the IE flag is cleared (to 0), execution is resumed from the instruction following the instruction that set the STOP mode. Refer to **16.7 Maskable Interrupt Acknowledgment Operation** for details of interrupt acknowledgment.

**Figure 17-6. STOP Mode Release by INTP0 to INTP2 Input**



★

**(2) STOP mode release by RESET input**

When the  $\overline{\text{RESET}}$  input goes from high to low, the reset state is established. The clock starts oscillating at the rising edge of  $\overline{\text{RESET}}$ . When the oscillation stabilization timer expires, the normal operation starts. At this point, the internal data memory retains the contents prior to the STOP mode setting.

## 17.5 IDLE Mode

### 17.5.1 IDLE mode setting and operating states

The IDLE mode is selected by setting (to 1) both the STP bit and the HLT bit of the standby control register (STBC).

The only writes that can be performed on 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.

If interrupts are enabled (when the IE bit of the program status word (PSW) is set to 1), write a NOP instruction three times after the instruction that sets the IDLE mode (after releasing the IDLE mode). Otherwise, two or more instructions may be executed before an interrupt is acknowledged. As a result, the execution sequence of the interrupt processing and instructions may be changed. To prevent troubles due to changes in the execution sequence, the above processing is necessary.

**Caution** If the IDLE mode is set when the condition to release the HALT mode is satisfied (refer to 17.3.2 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 17-6. 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
16-bit timer/event counter	Operation stopped
8-bit PWM timer	Operable only when an external input clock (TIO50, TIO51) is selected as the count clock
Watchdog timer	Stopped (timer is initialized)
A/D converter	Operation stopped <sup>Note 1</sup>
3-wire serial interface	Operation stopped <sup>Note 2</sup>
Asynchronous serial interface	Operation stopped <sup>Note 2</sup>
Watch timer	Operable
External interrupt (INTP0 to INTP2)	Operable
Internal RAM	Retained

- Notes**
1. A/D converter operation is stopped, but if the ADCS bit of the A/D converter mode register (ADM) is set, the current consumption does not decrease.
  2. The serial input pin supports a 3 V interface (i.e., it is a low-threshold pin). To prevent current being input to the Schmitt input buffer in STOP and IDLE modes, therefore, the Schmitt input buffer is turned off (the buffer output is "L"). Disable the serial interface (SIO0, SIO1, SIO2, and UART) before setting STOP or IDLE mode, and re-set the interface after STOP or IDLE mode has been released.

**Caution** The ADCS bit of the A/D converter mode register (ADM) should be reset.

**17.5.2 IDLE mode release**

The IDLE mode is released by INTP0 to INTP2 input, or  $\overline{\text{RESET}}$  input.

Release source and an outline of operations after release are shown in Table 17-7. Figure 17-7 shows operations after IDLE mode release.

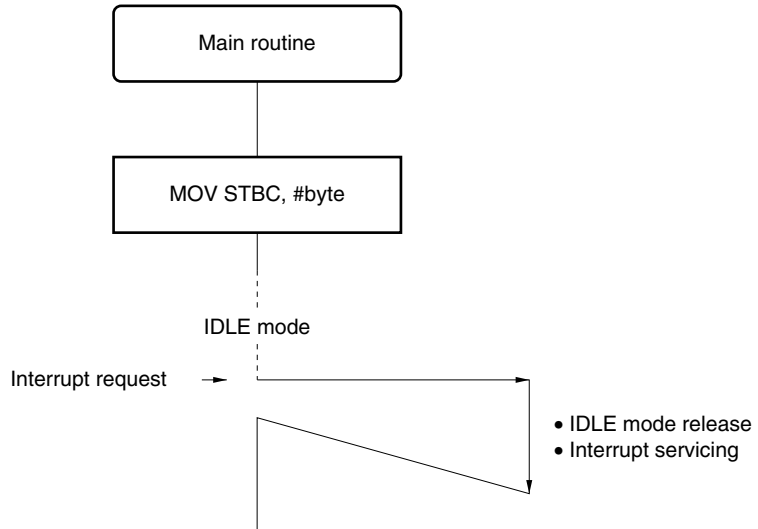
**Table 17-7. IDLE Mode Release and Operations After Release**

Release Source	MK <sup>Note 1</sup>	ISM <sup>Note 2</sup>	IE <sup>Note 3</sup>	State on Release	Operation After Release
$\overline{\text{RESET}}$ input	×	×	×	—	Normal reset operation
INTP0 to INTP2 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>Note 5</sup> cleared (to 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>Note 5</sup> is cleared (to 0), excluding execution of priority level 3 interrupt service program)</li> <li>High-priority interrupt service program being executed</li> </ul>	
	0	0	0	—	IDLE mode maintained
	1	0	×	—	
×	1	×	×		

- Notes**
1. Interrupt mask bit in individual interrupt request source
  2. Macro service enable flag in individual interrupt request source
  3. Interrupt enable flag in the program status word (PSW)
  4. Pending interrupt requests are acknowledged when acknowledgment becomes possible.
  5. Bit in the interrupt mode control register (IMC)

Figure 17-7. Operation After IDLE Mode Release (1/3)

(1) When interrupt generates after IDLE mode has been set



(2) Reset after IDLE mode has been set

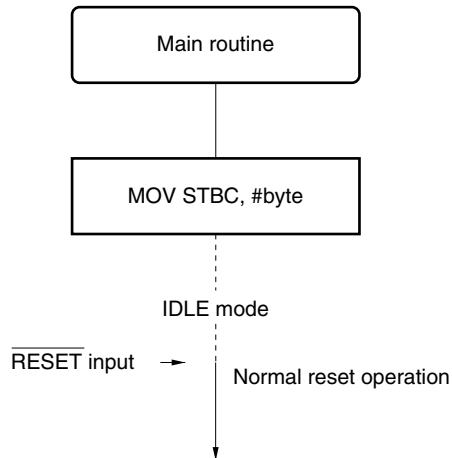
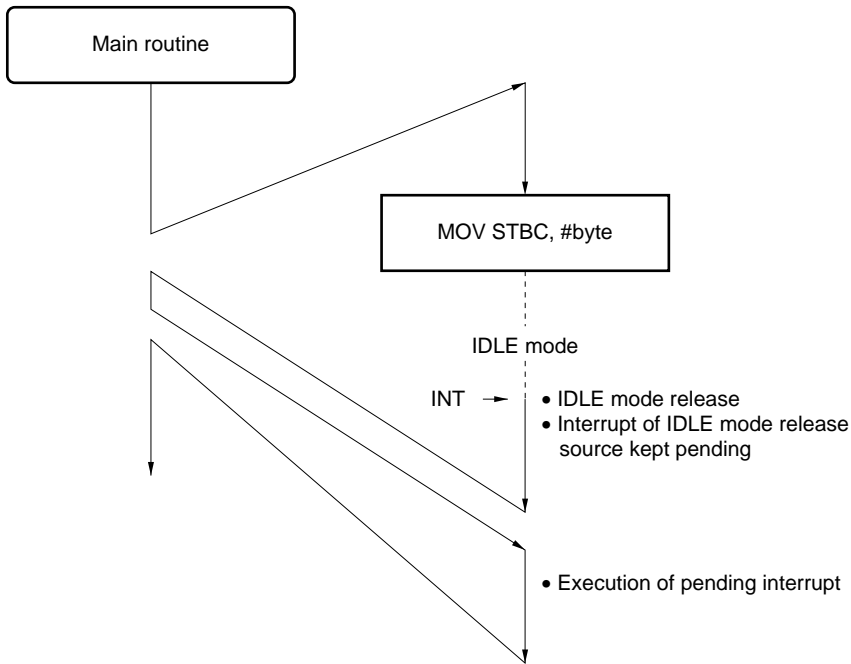


Figure 17-7. Operation After IDLE Mode Release (2/3)

(3) When IDLE mode is set while interrupt routine with priority higher than or same as that of interrupt of release source



(4) When IDLE mode is set while interrupt routine with priority lower than that of interrupt of release source

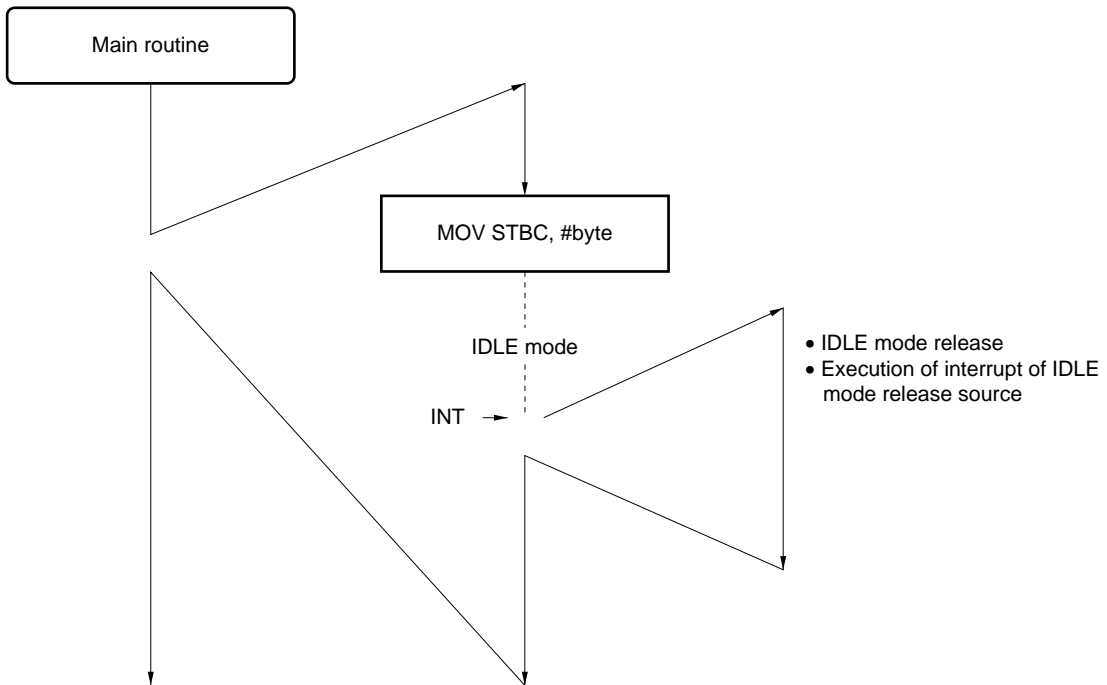
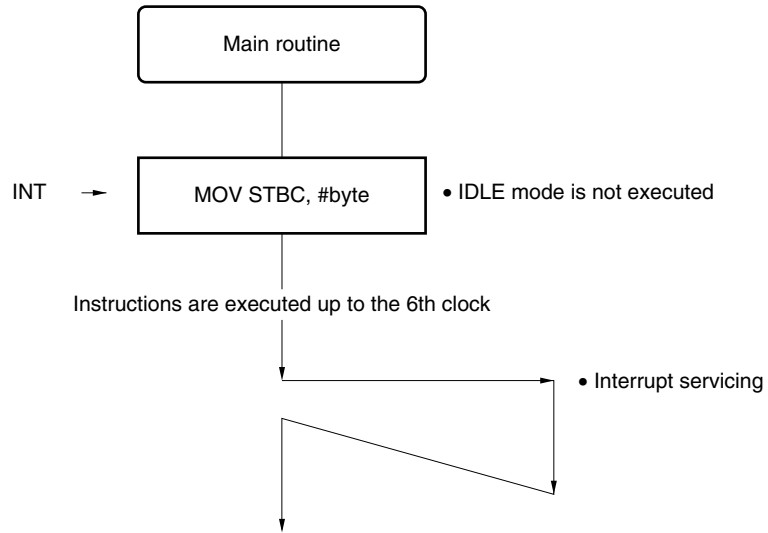




Figure 17-7. Operation After IDLE Mode Release (3/3)

## (5) When IDLE instruction and interrupt content



**(1) IDLE mode release by INTP0 to INTP2 input**

When masking of interrupts by INTP0 to INTP2 input is released and macro service is disabled, the IDLE mode is released when the valid edge specified by external interrupt mode register 1 (INTM1) is input to the INTP0 to INTP2 input.

When the  $\mu$ PD784975A is released from the IDLE mode, if an interrupt can be acknowledged when the interrupt enable flag (IE) is set (to 1), a branch is made to the interrupt service program. If the interrupt cannot be acknowledged and if the IE flag is cleared (to 0), execution is resumed from the instruction following the instruction that set the IDLE mode.

Refer to **16.7 Maskable Interrupt Acknowledgment Operation** for details of interrupt acknowledgment.

**(2) IDLE mode release by  $\overline{\text{RESET}}$  input**

When the  $\overline{\text{RESET}}$  input goes from high to low, the reset state is established. The clock starts oscillating at the rising edge of  $\overline{\text{RESET}}$ . When the oscillation stabilization timer expires, the normal operation starts. At this point, the internal data memory retains the contents prior to the IDLE mode setting.

## 17.6 Check Items When STOP Mode/IDLE Mode Is Used

Check items required to reduce the current consumption when STOP mode or 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 on-chip hardware, and therefore the on-chip 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, this 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  $\mu$ PD784975A may be adversely affected.

Also ensure that an intermediate potential is not applied.

### (3) Are on-chip 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 on-chip pull-up resistor is not used.

### ★ (4) A/D converter

The current flowing to the  $AV_{DD}$  pin can be reduced by clearing the ADCS 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  $AV_{DD}$  pin with external circuitry.

When  $ADCS = 1$ , the  $AV_{DD}$  pin can be used with the same potential as  $V_{SS1}$ .

## 17.7 Cautions

- (1) If the HALT/STOP/IDLE mode (standby mode hereafter) setting is performed when a condition that release the HALT mode (refer to **17.3.2 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 the standby mode.
- (2) When crystal/ceramic oscillation is used, the EXTC bit must be cleared (to 0) before use. If the EXTC bit is set (to 1), oscillation will stop.
- (3) When the STOP mode is used in a system that uses an external clock, the EXTC bit of OSTS must be set (to 1). If STOP mode setting is performed in a system to which an external clock is input when the EXTC bit of OSTS is cleared (to 0), the current consumption increases.  
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 **5.4 Main System Clock Oscillator**).
- (4) In the STOP and IDLE modes, the ADCS bit of the A/D converter mode register (ADM) should be cleared (to 0).
- (5) Execute an NOP instruction three times after the standby instruction (after the standby mode has been released). Otherwise, the standby instruction cannot be executed if execution of the standby instruction and an interrupt request contend, and the interrupt is acknowledged after two or more instructions following the standby instruction have been executed. The instruction that is executed before acknowledging the interrupt is the one that is executed within up to 6 clocks after the standby instruction has been executed.

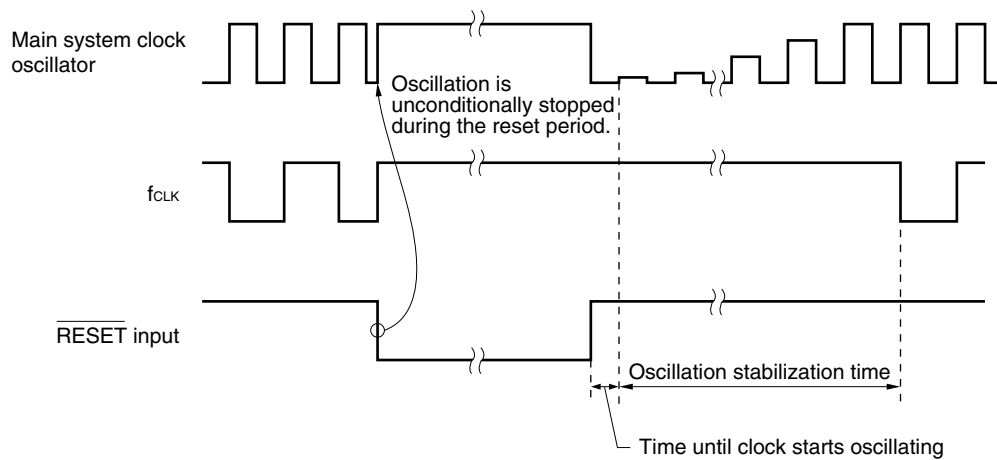
**Example**   MOV STBC, #byte  
          NOP  
          NOP  
          NOP  
          :

## CHAPTER 18 RESET FUNCTION

When a low level is input to the  $\overline{\text{RESET}}$  input pin, system reset is performed. The hardware enters the states listed in Figure 18-1. Since the oscillation of the main system clock unconditionally stops during the reset period, the current consumption of the entire system can be reduced.

When  $\overline{\text{RESET}}$  input goes from low to high, the reset state is released. After the count time of the timer for oscillation stabilization (41.9 ms@ 12.5 MHz operation), the content of the reset vector table is set in the program counter (PC). Execution branches to the address set in the PC, and program execution starts from the branch destination address. Therefore, the reset can start from any address.

**Figure 18-1. Oscillation of Main System Clock in Reset Period**



★

To prevent error operation caused by noise, a noise eliminator based on an analog delay is installed at the  $\overline{\text{RESET}}$  input pin.

Figure 18-2. Accepting Reset Signal

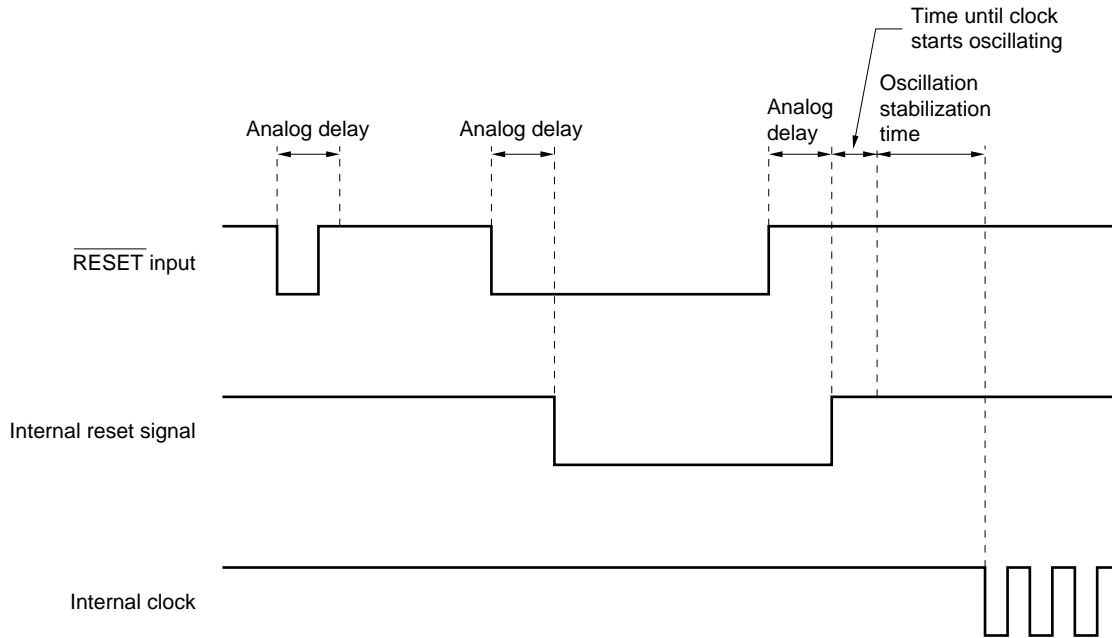


Table 18-1. State During/After Reset for All Hardware Resets

Hardware	State During Reset ( $\overline{\text{RESET}} = \text{L}$ )	State After Reset ( $\overline{\text{RESET}} = \text{H}$ )
Main system clock oscillator	Oscillation stops	Oscillation starts
Program counter (PC)	Undefined	Set a value in the reset vectored table.
Stack pointer (SP)	Undefined	
Program status word (PSW)	Initialize to 0000H.	
Internal RAM	This is undefined. However, when the standby state is released by a reset, the value is saved before setting standby.	
I/O lines	The input and output buffers turn off.	High impedance
Other hardware	Initialize to the fixed state <sup>Note</sup> .	

**Note** Refer to the After Reset column of **Table 3-6 Special Function Register (SFR) List**.

## CHAPTER 19 $\mu$ PD78F4976A PROGRAMMING

The flash memory can be written when installed in the target system (on board). The dedicated flash programmer (Flashpro III (part number FL-PR3, PG-FP3)) is connected to the host machine and target system.

**Remark** FL-PR3 is a product of Naito Densai Machida Mfg. Co., Ltd.

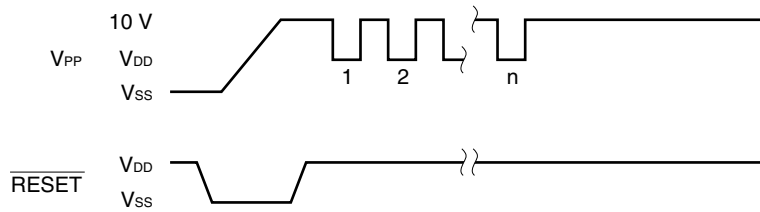
### 19.1 Selecting Communication Protocol

Flashpro III writes to flash memory by serial communication. The communication protocol is selected from Table 19-1 then writing is performed. The selection of the communication protocol has the format shown in Figure 19-1. Each communication protocol is selected by the number of  $V_{PP}$  pulses shown in Table 19-1.

**Table 19-1. Communication Protocols**

Communication Protocol	No. of Channels	Pins Used	No. of $V_{PP}$ Pulses
3-wire serial I/O	2	$\overline{SCK0}/P27$ SO0/P26 SI0/P25	0
		$\overline{SCK1}/P62$ SO1/P61 SI1/P60	1
Handshake (HS)	1	$\overline{SCK0}/27$ SO0/P26 SI0/P25 P20	3

**Caution** Select the communication protocol by using number of  $V_{PP}$  pulses given in Table 19-1.

**Figure 19-1. Format of Communication Protocol Selection**

## 19.2 Flash Memory Programming Functions

By transmitting and receiving various commands and data by the selected communication protocol, operations such as writing to the flash memory are performed. Table 19-2 shows the major functions.

**Table 19-2. Major Functions in Flash Memory Programming**

Function	Description
Area erase	Erase the contents of the specified memory area where one memory area is 16 KB.
Area blank check	Checks the erase state of the specified area.
Data write	Writes to the flash memory based on the start write address and the number of data written (number of bytes).
Area verify	Compares the data input to the contents of the specified memory area.

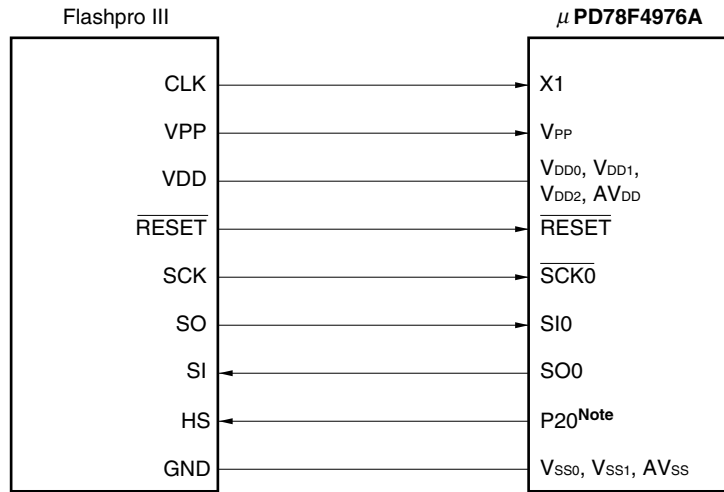
Flash memory verification is performed by supplying data from the outside via the serial interface, collating the supplied data with the contents of a specific area or the entire memory, and then indicating to the outside whether there is any conflicting data. The flash memory is not provided with a read function. This verification method keeps the flash memory contents from being accessed by unauthorized persons.



### 19.3 Connecting Flashpro III

The connection between the Flashpro III and the  $\mu$ PD78F4976A differs depending on the communication protocol (3-wire serial I/O). Figure 19-2 are the connection diagram.

**Figure 19-2. Connecting Flashpro III in 3-Wire Serial I/O Mode (When Using 3-Wire Serial I/O0)**



**Note** Used only when handshake communication is selected.

## CHAPTER 20 INSTRUCTION OPERATION

### 20.1 Examples

#### (1) Operand expression format and description (1/2)

Expression Format	Description
r, r <sup>Note 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>Note 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>Note 2</sup>	AX(RP0), BC(RP1), RP2, RP3, VP(RP4), UP(RP5), DE(RP6), HL(RP7)
rp1 <sup>Note 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 (refer to <b>Table 3-6 Special Function Register (SFR) List.</b> )
sfrp	Special function register symbol (16-bit manipulation register: refer to <b>Table 3-6 Special Function Register (SFR) List.</b> )
post <sup>Note 2</sup>	AX(RP0), BC(RP1), RP2, RP3, VP(RP4), UP(RP5)/PSW, DE(RP6), HL(RP7) Multiple descriptions are possible. However, UP is restricted to the PUSH/POP instruction, and PSW is restricted to the PUSHU/POPU instruction.
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	Everything under mem except [WHL+] and [WHL-]
mem2	[TDE], [WHL]
mem3	[AX], [BC], [RP2], [RP3], [VVP], [UUP], [TDE], [WHL]

- Notes**
1. By setting the RSS bit to 1, R4 to R7 can be used as X, A, C, and B. Use this function only when 78K/III Series programs are also used.
  2. By setting the RSS bit to 1, RP2 and RP3 can be used as AX and BC. Use this function only when 78K/III Series programs are also used.

(1) Operand expression format and description (2/2)

Expression Format	Description
<b>Note</b>	
saddr, saddr'	FD20H to FF1FH Immediate data or label
saddr1	FE00H to FEFFH Immediate data or label
saddr2	FD20H to FDFFH, FF00H - FF1FH Immediate data or label
saddrp	FD20H to FF1EH Immediate data or label (when manipulating 16 bits)
saddrp1	FE00H to FEFFH Immediate data or label (when manipulating 16 bits)
saddrp2	FD20H to FDFFH, FF00H - FF1EH Immediate data or label (when manipulating 16 bits)
saddrg	FD20H to FEFDH Immediate data or label (when manipulating 24 bits)
saddrg1	FE00H to FEFDH Immediate data or label (when manipulating 24 bits)
saddrg2	FD20H to FDFFH Immediate data or label (when manipulating 24 bits)
addr24	0H to FFFFFFFH Immediate data or label
addr20	0H to FFFFFH Immediate data or label
addr16	0H to FFFFH Immediate data or label
addr11	800H to FFFH Immediate data or label
addr8	0FE00H to 0FEFFH <sup>Note</sup> Immediate data or label
addr5	40H to 7EH Immediate data or label
imm24	24-bit immediate data or label
word	16-bit immediate data or label
byte	8-bit immediate data or label
bit	3-bit immediate data or label
n	3-bit immediate data
locaddr	0H or 0FH

**Note** When 0H is set by the LOCATION instruction, these addresses become the addresses shown here. When 0FH is set by the LOCATION instruction, the values of the addresses shown here added to F0000H become the addresses.

**(2) Symbols in “Operand” column**

Symbol	Description
+	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 reverse
[ ]	Indirect addressing
[%]	24-bit indirect addressing

**(3) Symbols in “Flags” column**

Symbol	Description
(Blank)	Not changed
0	Clear to 0
1	Set to 1
×	Set or clear based on the result
P	Operate with the P/V flag as the parity flag
V	Operate with the P/V flag as the overflow flag
R	Restore the previously saved value

**(4) Symbols in “Operation” column**

Symbol	Description
jdisp8	Two’s complement data (8 bits) of the relative address distance between the head address of the next instruction and the branch address
jdisp16	Two’s complement data (16 bits) of the relative address distance between the head address of the next instruction and the 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 operand**

mem Mode	Register Indirect Addressing		Based Addressing	Indexed Addressing	Based Indexed Addressing
No. of bytes	1	2 <sup>Note</sup>	3	5	2

**Note** This becomes a 1-byte instruction only when [TDE], [WHL], [TDE+], [TDE-], [WHL+], or [WHL-] is described in mem in the MOV instruction.

**(6) Number of bytes of instruction that includes saddr, saddrp, r, or rp in operand**

The number of bytes in an instruction that has saddr, saddrp, r, or rp in the operand is described in two parts divided by a slash (/). The following table shows the number of bytes in each one.

Description	No. of Bytes on Left Side	No. of Bytes on Right Side
saddr	saddr2	saddr1
saddrp	saddrp2	saddrp1
r	r1	r2
rp	rp1	rp2

**(7) Descriptions of instructions that include mem in operand and string instructions**

The TDE, WHL, VVP, and UUP (24-bit registers) operands can be described by DE, HL, VP, and UP. However, when DE, HL, VP, and UP are described, they are handled as TDE, WHL, VVP, and UUP (24-bit registers).

20.2 List of Operations

(1) 8-bit data transfer instruction: MOV

Mnemonic	Operand	Bytes	Operation	Flags					
				S	Z	AC	P/V	CY	
MOV	r, #byte	2/3	r ← byte						
	saddr, #byte	3/4	(saddr) ← byte						
	sfr, #byte	3	sfr ← byte						
	!addr16, #byte	5	(saddr16) ← byte						
	!!addr24, #byte	6	(addr24) ← byte						
	r, r'	2/3	r ← r'						
	A, r	1/2	A ← r						
	A, saddr2	2	A ← (saddr2)						
	r, saddr	3	r ← (saddr)						
	saddr2, A	2	(saddr2) ← A						
	saddr, r	3	(saddr) ← r						
	A, sfr	2	A ← sfr						
	r, sfr	3	r ← sfr						
	sfr, A	2	sfr ← A						
	sfr, r	3	sfr ← r						
	saddr, saddr'	4	(saddr) ← (saddr')						
	r, !addr16	4	r ← (addr16)						
	!addr16, r	4	(addr16) ← r						
	r, !!addr24	5	r ← (addr24)						
	!!addr24, r	5	(addr24) ← r						
	A, [saddrp]	2/3	A ← ((saddrp))						
	A, [%saddrg]	3/4	A ← ((saddrg))						
	A, mem	1-5	A ← (mem)						
	[saddrp], A	2/3	((saddrp)) ← A						
	[%saddrg], A	3/4	((saddrg)) ← A						
	mem, A	1-5	(mem) ← A						
	PSWL #byte	3	PSWL ← byte			x	x	x	x
	PSWH #byte	3	PSWH ← byte						
	PSWL, A	2	PSWL ← A			x	x	x	x
	PSWH, A	2	PSWH ← A						
	A, PSWL	2	A ← PSWL						
	A, PSWH	2	A ← PSWH						
r3, #byte	3	r3 ← byte							
A, r3	2	A ← r3							
r3, A	2	r3 ← A							

(2) 16-bit data transfer instruction: MOVW

Mnemonic	Operand	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	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MOVG	rg, #imm24	5	rg ← imm24					
	rg, rg'	2	rg ← rg'					
	rg, !addr24	5	rg ← (addr24)					
	!addr24, rg	5	(addr24) ← rg					
	rg, saddrg	3	rg ← (saddrg)					
	saddrg, rg	3	(saddrg) ← rg					
	WHL, [%saddrg]	3/4	WHL ← ((saddrg))					
	[%saddrg], WHL	3/4	((saddrg)) ← WHL					
	WHL, mem1	2-5	WHL ← (mem1)					
	mem1, WHL	2-5	(mem1) ← WHL					

(4) 8-bit data exchange instruction: **XCH**

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
XCH	r, r'	2/3	r ↔ r'					
	A, r	1/2	A ↔ r					
	A, saddr2	2	A ↔ (saddr2)					
	r, saddr	3	r ↔ (saddr)					
	r, sfr	3	r ↔ sfr					
	saddr, saddr'	4	(saddr) ↔ (saddr')					
	r, !addr16	4	r ↔ (addr16)					
	r, !addr24	5	r ↔ (addr24)					
	A, [saddrp]	2/3	A ↔ ((saddrp))					
	A, [%saddrg]	3/4	A ↔ ((saddrg))					
	A, mem	2-5	A ↔ (mem)					



(5) 16-bit data exchange instruction: XCHW

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
XCHW	rp, rp'	2	rp ↔ rp'					
	AX, saddrp2	2	AX ↔ (saddrp2)					
	rp, saddrp	3	rp ↔ (saddrp)					
	rp, sfrp	3	rp ↔ sfrp					
	AX, [saddrp]	3/4	AX ↔ ((saddrp))					
	AX, [%saddrg]	3/4	AX ↔ ((saddrg))					
	AX, !addr16	4	AX ↔ (addr16)					
	AX, !!addr24	5	AX ↔ (addr24)					
	saddrp, saddrp'	4	(saddrp) ↔ (saddrp')					
	AX, mem	2-5	AX ↔ (mem)					

(6) 8-bit arithmetic instructions: ADD, ADDC, SUB, SUBC, CMP, AND, OR, XOR

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ADD	A, #byte	2	A, CY ← A + byte	×	×	×	V	×
	r, #byte	3	r, CY ← r + byte	×	×	×	V	×
	saddr, #byte	3/4	(saddr), CY ← (saddr) + byte	×	×	×	V	×
	sfr, #byte	4	sfr, CY ← sfr + byte	×	×	×	V	×
	r, r'	2/3	r, CY ← r + r'	×	×	×	V	×
	A, saddr2	2	A, CY ← A + (saddr2)	×	×	×	V	×
	r, saddr	3	r, CY ← r + (saddr)	×	×	×	V	×
	saddr, r	3	(saddr), CY ← (saddr) + r	×	×	×	V	×
	r, sfr	3	r, CY ← r + sfr	×	×	×	V	×
	sfr, r	3	sfr, CY ← sfr + r	×	×	×	V	×
	saddr, saddr'	4	(saddr), CY ← (saddr) + (saddr')	×	×	×	V	×
	A, [saddrp]	3/4	A, CY ← A + ((saddrp))	×	×	×	V	×
	A, [%saddrg]	3/4	A, CY ← A + ((saddrg))	×	×	×	V	×
	[saddrp], A	3/4	((saddrp)), CY ← ((saddrp)) + A	×	×	×	V	×
	[%saddrg], A	3/4	((saddrg)), CY ← ((saddrg)) + A	×	×	×	V	×
	A, !addr16	4	A, CY ← A + (addr16)	×	×	×	V	×
	A, !!addr24	5	A, CY ← A + (addr24)	×	×	×	V	×
	!addr16, A	4	(addr16), CY ← (addr16) + A	×	×	×	V	×
	!!addr24, A	5	(addr24), CY ← (addr24) + A	×	×	×	V	×
	A, mem	2-5	A, CY ← A + (mem)	×	×	×	V	×
	mem, A	2-5	(mem), CY ← (mem) + A	×	×	×	V	×

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ADDC	A, #byte	2	$A, CY \leftarrow A + \text{byte} + CY$	x	x	x	V	x
	r, #byte	3	$r, CY \leftarrow r + \text{byte} + CY$	x	x	x	V	x
	saddr, #byte	3/4	$(\text{saddr}), CY \leftarrow (\text{saddr}) + \text{byte} + CY$	x	x	x	V	x
	sfr, #byte	4	$\text{sfr}, CY \leftarrow \text{sfr} + \text{byte} + CY$	x	x	x	V	x
	r, r'	2/3	$r, CY \leftarrow r + r' + CY$	x	x	x	V	x
	A, saddr2	2	$A, CY \leftarrow A + (\text{saddr}2) + CY$	x	x	x	V	x
	r, saddr	3	$r, CY \leftarrow r + (\text{saddr}) + CY$	x	x	x	V	x
	saddr, r	3	$(\text{saddr}), CY \leftarrow (\text{saddr}) + r + CY$	x	x	x	V	x
	r, sfr	3	$r, CY \leftarrow r + \text{sfr} + CY$	x	x	x	V	x
	sfr, r	3	$\text{sfr}, CY \leftarrow \text{sfr} + r + CY$	x	x	x	V	x
	saddr, saddr'	4	$(\text{saddr}), CY \leftarrow (\text{saddr}) + (\text{saddr}') + CY$	x	x	x	V	x
	A, [saddrp]	3/4	$A, CY \leftarrow A + ((\text{saddrp})) + CY$	x	x	x	V	x
	A, [%saddrg]	3/4	$A, CY \leftarrow A + ((\text{saddrg})) + CY$	x	x	x	V	x
	[saddrp], A	3/4	$((\text{saddrp})), CY \leftarrow ((\text{saddrp})) + A + CY$	x	x	x	V	x
	[%saddrg], A	3/4	$((\text{saddrg})), CY \leftarrow ((\text{saddrg})) + A + CY$	x	x	x	V	x
	A, !addr16	4	$A, CY \leftarrow A + (\text{addr}16) + CY$	x	x	x	V	x
	A, !!addr24	5	$A, CY \leftarrow A + (\text{addr}24) + CY$	x	x	x	V	x
	!addr16, A	4	$(\text{addr}16), CY \leftarrow (\text{addr}16) + A + CY$	x	x	x	V	x
	!!addr24, A	5	$(\text{addr}24), CY \leftarrow (\text{addr}24) + A + CY$	x	x	x	V	x
	A, mem	2-5	$A, CY \leftarrow A + (\text{mem}) + CY$	x	x	x	V	x
mem, A	2-5	$(\text{mem}), CY \leftarrow (\text{mem}) + A + CY$	x	x	x	V	x	

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
SUB	A, #byte	2	A, CY ← A – byte	×	×	×	V	×
	r, #byte	3	r, CY ← r – byte	×	×	×	V	×
	saddr, #byte	3/4	(saddr), CY ← (saddr) – byte	×	×	×	V	×
	sfr, #byte	4	sfr, CY ← sfr – byte	×	×	×	V	×
	r, r'	2/3	r, CY ← r – r'	×	×	×	V	×
	A, saddr2	2	A, CY ← A – (saddr2)	×	×	×	V	×
	r, saddr	3	r, CY ← r – (saddr)	×	×	×	V	×
	saddr, r	3	(saddr), CY ← (saddr) – r	×	×	×	V	×
	r, sfr	3	r, CY ← r – sfr	×	×	×	V	×
	sfr, r	3	sfr, CY ← sfr – r	×	×	×	V	×
	saddr, saddr'	4	(saddr), CY ← (saddr) – (saddr')	×	×	×	V	×
	A, [saddrp]	3/4	A, CY ← A – ((saddrp))	×	×	×	V	×
	A, [%saddrg]	3/4	A, CY ← A – ((saddrg))	×	×	×	V	×
	[saddrp], A	3/4	((saddrp)), CY ← ((saddrp)) – A	×	×	×	V	×
	[%saddrg], A	3/4	((saddrg)), CY ← ((saddrg)) – A	×	×	×	V	×
	A, !addr16	4	A, CY ← A – (addr16)	×	×	×	V	×
	A, !!addr24	5	A, CY ← A – (addr24)	×	×	×	V	×
	!addr16, A	4	(addr16), CY ← (addr16) – A	×	×	×	V	×
	!!addr24, A	5	(addr24), CY ← (addr24) – A	×	×	×	V	×
	A, mem	2-5	A, CY ← A – (mem)	×	×	×	V	×
mem, A	2-5	(mem), CY ← (mem) – A	×	×	×	V	×	

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
SUBC	A, #byte	2	A, CY ← A – byte – CY	×	×	×	V	×
	r, #byte	3	r, CY ← r – byte – CY	×	×	×	V	×
	saddr, #byte	3/4	(saddr), CY ← (saddr) – byte – CY	×	×	×	V	×
	sfr, #byte	4	sfr, CY ← sfr – byte – CY	×	×	×	V	×
	r, r'	2/3	r, CY ← r – r' – CY	×	×	×	V	×
	A, saddr2	2	A, CY ← A – (saddr2) – CY	×	×	×	V	×
	r, saddr	3	r, CY ← r – (saddr) – CY	×	×	×	V	×
	saddr, r	3	(saddr), CY ← (saddr) – r – CY	×	×	×	V	×
	r, sfr	3	r, CY ← r – sfr – CY	×	×	×	V	×
	sfr, r	3	sfr, CY ← sfr – r – CY	×	×	×	V	×
	saddr, saddr'	4	(saddr), CY ← (saddr) – (saddr') – CY	×	×	×	V	×
	A, [saddrp]	3/4	A, CY ← A – ((saddrp)) – CY	×	×	×	V	×
	A, [%saddrg]	3/4	A, CY ← A – ((saddrg)) – CY	×	×	×	V	×
	[saddrp], A	3/4	((saddrp)), CY ← ((saddrp)) – A – CY	×	×	×	V	×
	[%saddrg], A	3/4	((saddrg)), CY ← ((saddrg)) – A – CY	×	×	×	V	×
	A, !addr16	4	A, CY ← A – (addr16) – CY	×	×	×	V	×
	A, !!addr24	5	A, CY ← A – (addr24) – CY	×	×	×	V	×
	!addr16, A	4	(addr16), CY ← (addr16) – A – CY	×	×	×	V	×
	!!addr24, A	5	(addr24), CY ← (addr24) – A – CY	×	×	×	V	×
	A, mem	2-5	A, CY ← A – (mem) – CY	×	×	×	V	×
mem, A	2-5	(mem), CY ← (mem) – A – CY	×	×	×	V	×	

Mnemonic	Operand	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	Operand	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	$A \leftarrow A \wedge (\text{saddr}2)$	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	$A \leftarrow A \wedge (\text{addr}16)$	x	x			P
	A, !!addr24	5	$A \leftarrow A \wedge (\text{addr}24)$	x	x			P
	!addr16, A	4	$(\text{addr}16) \leftarrow (\text{addr}16) \wedge A$	x	x			P
	!!addr24, A	5	$(\text{addr}24) \leftarrow (\text{addr}24) \wedge A$	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	Operand	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	$A \leftarrow A \vee (\text{saddr}2)$	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	$A \leftarrow A \vee (\text{addr}16)$	x	x			P
	A, !!addr24	5	$A \leftarrow A \vee (\text{saddr}24)$	x	x			P
	!addr16, A	4	$(\text{addr}16) \leftarrow (\text{addr}16) \vee A$	x	x			P
	!!addr24, A	5	$(\text{addr}24) \leftarrow (\text{aaddr}24) \vee A$	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	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
XOR	A, #byte	2	$A \leftarrow A \nabla \text{byte}$	x	x			P
	r, #byte	3	$r \leftarrow r \nabla \text{byte}$	x	x			P
	saddr, #byte	3/4	$(\text{saddr}) \leftarrow (\text{saddr}) \nabla \text{byte}$	x	x			P
	sfr, #byte	4	$\text{sfr} \leftarrow \text{sfr} \nabla \text{byte}$	x	x			P
	r, r'	2/3	$r \leftarrow r \nabla r'$	x	x			P
	A, saddr2	2	$A \leftarrow A \nabla (\text{saddr}2)$	x	x			P
	r, saddr	3	$r \leftarrow r \nabla (\text{saddr})$	x	x			P
	saddr, r	3	$(\text{saddr}) \leftarrow (\text{saddr}) \nabla r$	x	x			P
	r, sfr	3	$r \leftarrow r \nabla \text{sfr}$	x	x			P
	sfr, r	3	$\text{sfr} \leftarrow \text{sfr} \nabla r$	x	x			P
	saddr, saddr'	4	$(\text{saddr}) \leftarrow (\text{saddr}) \nabla (\text{saddr}')$	x	x			P
	A, [saddrp]	3/4	$A \leftarrow A \nabla ((\text{saddrp}))$	x	x			P
	A, [%saddrg]	3/4	$A \leftarrow A \nabla ((\text{saddrg}))$	x	x			P
	[saddrp], A	3/4	$((\text{saddrp})) \leftarrow ((\text{saddrp})) \nabla A$	x	x			P
	[%saddrg], A	3/4	$((\text{saddrg})) \leftarrow ((\text{saddrg})) \nabla A$	x	x			P
	A, !addr16	4	$A \leftarrow A \nabla (\text{addr}16)$	x	x			P
	A, !!addr24	5	$A \leftarrow A \nabla (\text{addr}24)$	x	x			P
	!addr16, A	4	$(\text{addr}16) \leftarrow (\text{addr}16) \nabla A$	x	x			P
	!!addr24, A	5	$(\text{addr}24) \leftarrow (\text{addr}24) \nabla A$	x	x			P
	A, mem	2-5	$A \leftarrow A \nabla (\text{mem})$	x	x			P
mem, A	2-5	$(\text{mem}) \leftarrow (\text{mem}) \nabla A$	x	x			P	



(7) 16-bit arithmetic instructions: ADDW, SUBW, CMPW

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ADDW	AX, #word	3	AX, CY ← AX + word	×	×	×	V	×
	rp, #word	4	rp, CY ← rp + word	×	×	×	V	×
	rp, rp'	2	rp, CY ← rp + rp'	×	×	×	V	×
	AX, saddrp2	2	AX, CY ← AX + (saddrp2)	×	×	×	V	×
	rp, saddrp	3	rp, CY ← rp + (saddrp)	×	×	×	V	×
	saddrp, rp	3	(saddrp), CY ← (saddrp) + rp	×	×	×	V	×
	rp, sfrp	3	rp, CY ← rp + sfrp	×	×	×	V	×
	sfrp, rp	3	sfrp, CY ← sfrp + rp	×	×	×	V	×
	saddrp, #word	4/5	(saddrp), CY ← (saddrp) + word	×	×	×	V	×
	sfrp, #word	5	sfrp, CY ← sfrp + word	×	×	×	V	×
	saddrp, saddrp'	4	(saddrp), CY ← (saddrp) + (saddrp')	×	×	×	V	×
SUBW	AX, #word	3	AX, CY ← AX – word	×	×	×	V	×
	rp, #word	4	rp, CY ← rp – word	×	×	×	V	×
	rp, rp'	2	rp, CY ← rp – rp'	×	×	×	V	×
	AX, saddrp2	2	AX, CY ← AX – (saddrp2)	×	×	×	V	×
	rp, saddrp	3	rp, CY ← rp – (saddrp)	×	×	×	V	×
	saddrp, rp	3	(saddrp), CY ← (saddrp) – rp	×	×	×	V	×
	rp, sfrp	3	rp, CY ← rp – sfrp	×	×	×	V	×
	sfrp, rp	3	sfrp, CY ← sfrp – rp	×	×	×	V	×
	saddrp, #word	4/5	(saddrp), CY ← (saddrp) – word	×	×	×	V	×
	sfrp, #word	5	sfrp, CY ← sfrp – word	×	×	×	V	×
	saddrp, saddrp'	4	(saddrp), CY ← (saddrp) – (saddrp')	×	×	×	V	×
CMPW	AX, #word	3	AX – word	×	×	×	V	×
	rp, #word	4	rp – word	×	×	×	V	×
	rp, rp'	2	rp – rp'	×	×	×	V	×
	AX, saddrp2	2	AX – (saddrp2)	×	×	×	V	×
	rp, saddrp	3	rp – (saddrp)	×	×	×	V	×
	saddrp, rp	3	(saddrp) – rp	×	×	×	V	×
	rp, sfrp	3	rp – sfrp	×	×	×	V	×
	sfrp, rp	3	sfrp – rp	×	×	×	V	×
	saddrp, #word	4/5	(saddrp) – word	×	×	×	V	×
	sfrp, #word	5	sfrp – word	×	×	×	V	×
	saddrp, saddrp'	4	(saddrp) – (saddrp')	×	×	×	V	×

(8) 24-bit arithmetic instructions: ADDG, SUBG

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ADDG	rg, rg'	2	rg, CY ← rg + rg'	x	x	x	V	x
	rg, #imm24	5	rg, CY ← rg + imm24	x	x	x	V	x
	WHL, saddrg	3	WHL, CY ← WHL + (saddrg)	x	x	x	V	x
SUBG	rg, rg'	2	rg, CY ← rg - rg'	x	x	x	V	x
	rg, #imm24	5	rg, CY ← rg - imm24	x	x	x	V	x
	WHL, saddrg	3	WHL, CY ← WHL - (saddrg)	x	x	x	V	x

(9) Multiplicative instructions: MULU, MULUW, MULW, DIVUW, DIVUX

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MULU	r	2/3	AX ← A × r					
MULUW	rp	2	AX (higher), rp (lower) ← AX × rp					
MULW	rp	2	AX (higher), rp (lower) ← AX × rp					
DIVUW	r	2/3	AX (quotient), r (remainder) ← AX ÷ r <sup>Note 1</sup>					
DIVUX	rp	2	AXDE (quotient), rp (remainder) ← AXDE ÷ rp <sup>Note 2</sup>					

Notes 1. When r = 0, r ← X, AX ← FFFFH

2. When rp = 0, rp ← DE, AXDE ← FFFFFFFFH

(10) Special arithmetic instructions: MACW, MACSW, SACW

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MACW	byte	3	AXDE ← (B) × (C) + AXDE, B ← B + 2, C ← C + 2, byte ← byte - 1 End if (byte = 0 or P/V = 1)	x	x	x	V	x
MACSW	byte	3	AXDE ← (B) × (C) + AXDE, B ← B + 2, C ← C + 2, byte ← byte - 1 if byte = 0 then End if P/V = 1 then if overflow AXDE ← 7FFFFFFFH, End if underflow AXDE ← 80000000H, End	x	x	x	V	x
SACW	[TDE+], [WHL+]	4	AX ←  (TDE) - (WHL)  + AX, TDE ← TED + 2, WHL ← WHL + 2 C ← C - 1 End if (C = 0 or CY = 1)	x	x	x	V	x

**(11) Increment and decrement instructions: INC, DEC, INCW, DECW, INCG, DECG**

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
INC	r	1/2	$r \leftarrow r + 1$	x	x	x	V	
	saddr	2/3	$(saddr) \leftarrow (saddr) + 1$	x	x	x	V	
DEC	r	1/2	$r \leftarrow r - 1$	x	x	x	V	
	saddr	2/3	$(saddr) \leftarrow (saddr) - 1$	x	x	x	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) Decimal adjust instructions: ADJBA, ADJBS, CVTBW**

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ADJBA		2	Decimal adjust accumulator after addition	x	x	x	P	x
ADJBS		2	Decimal adjust accumulator after subtract	x	x	x	P	x
CVTBW		1	$X \leftarrow A, A \leftarrow 00H$ if $A_7 = 0$ $X \leftarrow A, A \leftarrow FFH$ if $A_7 = 1$					

(13) Shift and rotate instructions: ROR, ROL, RORC, ROLC, SHR, SHL, SHRW, SHLW, ROR4, ROL4

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ROR	r, n	2/3	$(CY, r_7 \leftarrow r_0, r_{m-1} \leftarrow r_m) \times n$ n = 0 to 7				P	×
ROL	r, n	2/3	$(CY, r_0 \leftarrow r_7, r_{m+1} \leftarrow r_m) \times n$ n = 0 to 7				P	×
RORC	r, n	2/3	$(CY \leftarrow r_0, r_7 \leftarrow CY, r_{m-1} \leftarrow r_m) \times n$ n = 0 to 7				P	×
ROLC	r, n	2/3	$(CY \leftarrow r_7, r_0 \leftarrow CY, r_{m+1} \leftarrow r_m) \times n$ n = 0 to 7				P	×
SHR	r, n	2/3	$(CY \leftarrow r_0, r_7 \leftarrow 0, r_{m-1} \leftarrow r_m) \times n$ n = 0 to 7	×	×	0	P	×
SHL	r, n	2/3	$(CY \leftarrow r_7, r_0 \leftarrow 0, r_{m+1} \leftarrow r_m) \times n$ n = 0 to 7	×	×	0	P	×
SHRW	rp, n	2	$(CY \leftarrow rp_0, rp_{15} \leftarrow 0, rp_{m-1} \leftarrow rp_m) \times n$ n = 0 to 7	×	×	0	P	×
SHLW	rp, n	2	$(CY \leftarrow rp_{15}, rp_0 \leftarrow 0, rp_{m+1} \leftarrow rp_m) \times n$ n = 0 to 7	×	×	0	P	×
ROR4	mem3	2	$A_{3-0} \leftarrow (mem3)_{3-0}, (mem3)_{7-4} \leftarrow A_{3-0},$ $(mem3)_{3-0} \leftarrow (mem3)_{7-4}$					
ROL4	mem3	2	$A_{3-0} \leftarrow (mem3)_{7-4}, (mem3)_{3-0} \leftarrow A_{3-0},$ $(mem3)_{7-4} \leftarrow (mem3)_{3-0}$					

(14) Bit manipulation instructions: MOV1, AND1, OR1, XOR1, NOT1, SET1, CLR1

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MOV1	CY, saddr.bit	3/4	$CY \leftarrow (saddr.bit)$					×
	CY, sfr.bit	3	$CY \leftarrow sfr.bit$					×
	CY, X.bit	2	$CY \leftarrow X.bit$					×
	CY, A.bit	2	$CY \leftarrow A.bit$					×
	CY, PSWL.bit	2	$CY \leftarrow PSWL.bit$					×
	CY, PSWH.bit	2	$CY \leftarrow PSWH.bit$					×
	CY, !addr16.bit	5	$CY \leftarrow !addr16.bit$					×
	CY, !!addr24.bit	2	$CY \leftarrow !!addr24.bit$					×
	CY, mem2.bit	2	$CY \leftarrow mem2.bit$					×
	saddr.bit, CY	3/4	$(saddr.bit) \leftarrow CY$					
	sfr.bit, CY	3	$sfr.bit \leftarrow CY$					
	X.bit, CY	2	$X.bit \leftarrow CY$					
	A.bit, CY	2	$A.bit \leftarrow CY$					
	PSWL.bit, CY	2	$PSWL.bit \leftarrow CY$	×	×	×	×	×
	PSWH.bit, CY	2	$PSWH.bit \leftarrow CY$					
	!addr16.bit, CY	5	$!addr16.bit \leftarrow CY$					
	!!addr24.bit, CY	6	$!!addr24.bit \leftarrow CY$					
	mem2.bit, CY	2	$mem2.bit \leftarrow CY$					

Mnemonic	Operand	Bytes	Operation	Flags					
				S	Z	AC	P/V	CY	
AND1	CY, saddr.bit	3/4	$CY \leftarrow CY \wedge (\text{saddr.bit})$						x
	CY, /saddr.bit	3/4	$CY \leftarrow CY \wedge \overline{(\text{saddr.bit})}$						x
	CY, sfr.bit	3	$CY \leftarrow CY \wedge \text{sfr.bit}$						x
	CY, /sfr.bit	3	$CY \leftarrow CY \wedge \overline{\text{sfr.bit}}$						x
	CY, X.bit	2	$CY \leftarrow CY \wedge X.bit$						x
	CY, /X.bit	2	$CY \leftarrow CY \wedge \overline{X.bit}$						x
	CY, A.bit	2	$CY \leftarrow CY \wedge A.bit$						x
	CY, /A.bit	2	$CY \leftarrow CY \wedge \overline{A.bit}$						x
	CY, PSWL.bit	2	$CY \leftarrow CY \wedge \text{PSWL.bit}$						x
	CY, /PSWL.bit	2	$CY \leftarrow CY \wedge \overline{\text{PSWL.bit}}$						x
	CY, PSWH.bit	2	$CY \leftarrow CY \wedge \text{PSWH.bit}$						x
	CY, /PSWH.bit	2	$CY \leftarrow CY \wedge \overline{\text{PSWH.bit}}$						x
	CY, laddr16.bit	5	$CY \leftarrow CY \wedge \text{laddr16.bit}$						x
	CY, /laddr16.bit	5	$CY \leftarrow CY \wedge \overline{\text{laddr16.bit}}$						x
	CY, !!addr24.bit	2	$CY \leftarrow CY \wedge \text{!!addr24.bit}$						x
	CY, /!addr24.bit	6	$CY \leftarrow CY \wedge \overline{\text{!addr24.bit}}$						x
	CY, mem2.bit	2	$CY \leftarrow CY \wedge \text{mem2.bit}$						x
	CY, /mem2.bit	2	$CY \leftarrow CY \wedge \overline{\text{mem2.bit}}$						x
OR1	CY, saddr.bit	3/4	$CY \leftarrow CY \vee (\text{saddr.bit})$						x
	CY, /saddr.bit	3/4	$CY \leftarrow CY \vee \overline{(\text{saddr.bit})}$						x
	CY, sfr.bit	3	$CY \leftarrow CY \vee \text{sfr.bit}$						x
	CY, /sfr.bit	3	$CY \leftarrow CY \vee \overline{\text{sfr.bit}}$						x
	CY, X.bit	2	$CY \leftarrow CY \vee X.bit$						x
	CY, /X.bit	2	$CY \leftarrow CY \vee \overline{X.bit}$						x
	CY, A.bit	2	$CY \leftarrow CY \vee A.bit$						x
	CY, /A.bit	2	$CY \leftarrow CY \vee \overline{A.bit}$						x
	CY, PSWL.bit	2	$CY \leftarrow CY \vee \text{PSWL.bit}$						x
	CY, /PSWL.bit	2	$CY \leftarrow CY \vee \overline{\text{PSWL.bit}}$						x
	CY, PSWH.bit	2	$CY \leftarrow CY \vee \text{PSWH.bit}$						x
	CY, /PSWH.bit	2	$CY \leftarrow CY \vee \overline{\text{PSWH.bit}}$						x
	CY, laddr16.bit	5	$CY \leftarrow CY \vee \text{laddr16.bit}$						x
	CY, /laddr16.bit	5	$CY \leftarrow CY \vee \overline{\text{laddr16.bit}}$						x
	CY, !!addr24.bit	2	$CY \leftarrow CY \vee \text{!!addr24.bit}$						x
	CY, /!addr24.bit	6	$CY \leftarrow CY \vee \overline{\text{!addr24.bit}}$						x
	CY, mem2.bit	2	$CY \leftarrow CY \vee \text{mem2.bit}$						x
	CY, /mem2.bit	2	$CY \leftarrow CY \vee \overline{\text{mem2.bit}}$						x

Mnemonic	Operand	Bytes	Operation	Flags					
				S	Z	AC	P/V	CY	
XOR1	CY, saddr.bit	3/4	$CY \leftarrow CY \nabla (\text{saddr.bit})$						x
	CY, sfr.bit	3	$CY \leftarrow CY \nabla \text{sfr.bit}$						x
	CY, X.bit	2	$CY \leftarrow CY \nabla X.\text{bit}$						x
	CY, A.bit	2	$CY \leftarrow CY \nabla A.\text{bit}$						x
	CY, PSWL.bit	2	$CY \leftarrow CY \nabla \text{PSWL.bit}$						x
	CY, PSWH.bit	2	$CY \leftarrow CY \nabla \text{PSWH.bit}$						x
	CY, !addr16.bit	5	$CY \leftarrow CY \nabla \text{!addr16.bit}$						x
	CY, !!addr24.bit	2	$CY \leftarrow CY \nabla \text{!!addr24.bit}$						x
	CY, mem2.bit	2	$CY \leftarrow CY \nabla \text{mem2.bit}$						x
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	$\text{PSWL.bit} \leftarrow \overline{\text{PSWL.bit}}$	x	x	x	x	x	
	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	$CY \leftarrow \overline{CY}$						x
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	$\text{PSWL.bit} \leftarrow 1$	x	x	x	x	x	
	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	$\text{PSWL.bit} \leftarrow 0$	x	x	x	x	x	
	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	Operand	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	$\{(SP - 2) \leftarrow post, SP \leftarrow SP - 2\} \times m^{\text{Note}}$					
	rg	2	$(SP - 3) \leftarrow rg, SP \leftarrow SP - 3$					
PUSHU	post	2	$\{(UUP - 2) \leftarrow post, UUP \leftarrow UUP - 2\} \times m^{\text{Note}}$					
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	$\{post \leftarrow (SP), SP \leftarrow SP + 2\} \times m^{\text{Note}}$					
	rg	2	$rg \leftarrow (SP), SP \leftarrow SP + 3$					
POPU	post	2	$\{post \leftarrow (UUP), UUP \leftarrow UUP + 2\} \times m^{\text{Note}}$					
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	$SP \leftarrow SP - 1$					

**Note** m is the number of registers specified by post.

(16) Call return instructions: CALL, CALLF, CALLT, BRK, BRKCS, RET, RETI, RETB, RETCS, RETCSB

Mnemonic	Operand	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	$(SP - 3) \leftarrow (PC + 2)$ , $SP \leftarrow SP - 3$ , $PC_{HW} \leftarrow 0$ , $PC_{LW} \leftarrow \text{rp}$					
	rg	2	$(SP - 3) \leftarrow (PC + 2)$ , $SP \leftarrow SP - 3$ , $PC \leftarrow \text{rg}$					
	[rp]	2	$(SP - 3) \leftarrow (PC + 2)$ , $SP \leftarrow SP - 3$ , $PC_{HW} \leftarrow 0$ , $PC_{LW} \leftarrow (\text{rp})$					
	[rg]	2	$(SP - 3) \leftarrow (PC + 2)$ , $SP \leftarrow SP - 3$ , $PC \leftarrow (\text{rg})$					
	!\$!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	$(SP - 3) \leftarrow (PC + 1)$ , $SP \leftarrow SP - 3$ $PC_{HW} \leftarrow 0$ , $PC_{CW} \leftarrow (\text{addr5})$					
BRK		1	$(SP - 2) \leftarrow \text{PSW}$ , $(SP - 1)_{0-3} \leftarrow (PC + 1)_{HW}$ , $(SP - 4) \leftarrow (PC + 1)_{LW}$ , $SP \leftarrow SP - 4$ $PC_{HW} \leftarrow 0$ , $PC_{LW} \leftarrow (003EH)$					
BRKCS	RBn	2	$PC_{LW} \leftarrow \text{RP2}$ , $\text{RP3} \leftarrow \text{PSW}$ , $\text{RBS2} - 0 \leftarrow n$ , $\text{RSS} \leftarrow 0$ , $\text{IE} \leftarrow 0$ , $\text{RP3}_{8-11} \leftarrow \text{PC}_{HW}$ , $\text{PC}_{HW} \leftarrow 0$					
RET		1	$PC \leftarrow (SP)$ , $SP \leftarrow SP + 3$					
RETI		1	$PC_{LW} \leftarrow (SP)$ , $PC_{HW} \leftarrow (SP + 3)_{0-3}$ , $\text{PSW} \leftarrow (SP + 2)$ , $SP \leftarrow SP + 4$ The flag with the highest priority that is set to one in the ISPR is cleared to 0.	R	R	R	R	R
RETB		1	$PC_{LW} \leftarrow (SP)$ , $PC_{HW} \leftarrow (SP + 3)_{0-3}$ , $\text{PSW} \leftarrow (SP + 2)$ , $SP \leftarrow SP + 4$	R	R	R	R	R
RETCS	!addr16	3	$\text{PSW} \leftarrow \text{RP3}$ , $PC_{LW} \leftarrow \text{RP2}$ , $\text{RP2} \leftarrow \text{addr16}$ , $PC_{HW} \leftarrow \text{RP3}_{8-11}$ The flag with the highest priority that is set to one in the ISPR is cleared to 0.	R	R	R	R	R
RETCSB	!addr16	4	$\text{PSW} \leftarrow \text{RP3}$ , $PC_{LW} \leftarrow \text{RP2}$ , $\text{RP2} \leftarrow \text{addr16}$ , $PC_{HW} \leftarrow \text{RP3}_{8-11}$	R	R	R	R	R



**(17) Unconditional branch instruction: BR**

Mnemonic	Operand	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, BNV, BPO, BV, BPE, BP, BN, BLT, BGE, BLE, BGT, BNH, BH, BF, BT, BTCLR, BFSET, DBNZ

Mnemonic	Operand	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	$PC \leftarrow PC + 4^{\text{Note}} + \text{jdisp8}$ 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						

**Note** This is used when the number of bytes is four. When five, it becomes  $PC \leftarrow PC + 5 + \text{jdisp8}$ .

Mnemonic	Operand	Bytes	Operation	Flags					
				S	Z	AC	P/V	CY	
BT	saddr.bit, \$addr20	3/4	$PC \leftarrow PC + 3^{\text{Note 1}} + \text{jdisp8}$ 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	{ $PC \leftarrow PC + 4^{\text{Note 2}} + \text{jdisp8}$ , (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	x	x	x	x	x	
	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						

- Notes** 1. This is used when the number of bytes is three. When four, it becomes  $PC \leftarrow PC + 4 + \text{jdisp8}$ .  
 2. This is used when the number of bytes is four. When five, it becomes  $PC \leftarrow PC + 5 + \text{jdisp8}$ .

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
BFSET	saddr.bit, \$addr20	4/5	{PC ← PC + 4 <sup>Note 2</sup> + 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	x	x	x	x	x
	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					
	saddr, \$addr20	3/4	(saddr) ← (saddr) - 1, PC ← PC + 3 <sup>Note 1</sup> + jdisp8 if (saddr) ≠ 0					

- Notes** 1. This is used when the number of bytes is three. When four, it becomes PC ← PC + 4 + jdisp8.  
 2. This is used when the number of bytes is four. When five, it becomes PC ← PC + 5 + jdisp8.

**(19) CPU control instructions: MOV, LOCATION, SEL, SWRS, NOP, EI, DI**

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MOV	STBC, #byte	4	STBC ← byte					
	WDM, #byte	4	WDM ← byte					
LOCATION	locaddr	4	Specification of the higher word of the location address of the SFR and internal data area					
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 operation					
EI		1	IE ← 1 (Enable interrupt)					
DI		1	IE ← 0 (Disable interrupt)					

**(20) String instructions: MOVTLBW, MOVML, XCHML, MOVBLK, XCHBK, CMPME, CMPMNE, CMPMC, CMPMNC, CMPBKE, CMPBKNE, CMPBKC, CMPBKNC**

Mnemonic	Operand	Bytes	Operation	Flags					
				S	Z	AC	P/V	CY	
MOVTLBW	!addr8, byte	4	(addr8 + 2) ← (addr8), byte ← byte – 1, addr8 ← addr8 – 2 End if byte = 0						
MOVML	[TDE+], A	2	(TDE) ← A, TDE ← TDE + 1, C ← C – 1 End if C = 0						
	[TDE–], A	2	(TDE) ← A, TDE ← TDE – 1, C ← C – 1 End if C = 0						
XCHML	[TDE+], A	2	(TDE) ↔ A, TDE ← TDE + 1, C ← C – 1 End if C = 0						
	[TDE–], A	2	(TDE) ↔ A, TDE ← TDE – 1, C ← C – 1 End if C = 0						
MOVBLK	[TDE+], [WHL+]	2	(TDE) ← (WHL), TDE ← TDE + 1, WHL ← WHL + 1, C ← C – 1 End if C = 0						
	[TDE–], [WHL–]	2	(TDE) ← (WHL), TDE ← TDE – 1, WHL ← WHL – 1, C ← C – 1 End if C = 0						
XCHBK	[TDE+], [WHL+]	2	(TDE) ↔ (WHL), TDE ← TDE + 1, WHL ← WHL + 1, C ← C – 1 End if C = 0						
	[TDE–], [WHL–]	2	(TDE) ↔ (WHL), TDE ← TDE – 1, WHL ← WHL – 1, C ← C – 1 End if C = 0						
CMPME	[TDE+], A	2	(TDE) – A, TDE ← TDE + 1, C ← C – 1 End if C = 0 or Z = 0	x	x	x	V	x	
	[TDE–], A	2	(TDE) – A, TDE ← TDE – 1, C ← C – 1 End if C = 0 or Z = 0	x	x	x	V	x	
CMPMNE	[TDE+], A	2	(TDE) – A, TDE ← TDE + 1, C ← C – 1 End if C = 0 or Z = 1	x	x	x	V	x	
	[TDE–], A	2	(TDE) – A, TDE ← TDE – 1, C ← C – 1 End if C = 0 or Z = 1	x	x	x	V	x	
CMPMC	[TDE+], A	2	(TDE) – A, TDE ← TDE + 1, C ← C – 1 End if C = 0 or CY = 0	x	x	x	V	x	
	[TDE–], A	2	(TDE) – A, TDE ← TDE – 1, C ← C – 1 End if C = 0 or CY = 0	x	x	x	V	x	
CMPMNC	[TDE+], A	2	(TDE) – A, TDE ← TDE + 1, C ← C – 1 End if C = 0 or CY = 1	x	x	x	V	x	
	[TDE–], A	2	(TDE) – A, TDE ← TDE – 1, C ← C – 1 End if C = 0 or CY = 1	x	x	x	V	x	
CMPBKE	[TDE+], [WHL+]	2	(TDE) – (WHL), TDE ← TDE + 1, WHL ← WHL + 1, C ← C – 1 End if C = 0 or Z = 0	x	x	x	V	x	
	[TDE–], [WHL–]	2	(TDE) – (WHL), TDE ← TDE – 1, WHL ← WHL – 1, C ← C – 1 End if C = 0 or Z = 0	x	x	x	V	x	

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
CMPBKNE	[TDE+], [WHL+]	2	(TDE) – (WHL), TDE ← TDE + 1, WHL ← WHL + 1, C ← C –1 End if C = 0 or Z = 1	x	x	x	V	x
	[TDE–], [WHL–]	2	(TDE) – (WHL), TDE ← TDE – 1, WHL ← WHL – 1, C ← C –1 End if C = 0 or Z = 1	x	x	x	V	x
CMPBKC	[TDE+], [WHL+]	2	(TDE) – (WHL), TDE ← TDE + 1, WHL ← WHL + 1, C ← C –1 End if C = 0 or CY = 0	x	x	x	V	x
	[TDE–], [WHL–]	2	(TDE) – (WHL), TDE ← TDE – 1, WHL ← WHL – 1, C ← C –1 End if C = 0 or CY = 0	x	x	x	V	x
CMPBKNC	[TDE+], [WHL+]	2	(TDE) – (WHL), TDE ← TDE + 1, WHL ← WHL + 1, C ← C –1 End if C = 0 or CY = 1	x	x	x	V	x
	[TDE–], [WHL–]	2	(TDE) – (WHL), TDE ← TDE – 1, WHL ← WHL – 1, C ← C –1 End if C = 0 or CY = 1	x	x	x	V	x

20.3 Lists of Addressing Instructions

(1) 8-bit instructions (values in parentheses are combined to express the A description as r.)

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, MOVW, XCHW, CMPME, CMPMNE, CMPMNC, CMPMC, MOVBK, XCHBK, CMPBKE, CMPBKNE, CMPBKNC, CMPBKC

Table 20-1. 8-Bit Addressing Instructions

Second operand / First operand	#byte	A	r r'	saddr saddr'	sfr	!addr16 !!addr24	mem [saddrp] [%saddrg]	r3 PSWL PSWH	[WHL+] [WHL-]	n	None <sup>Note 2</sup>
A	(MOV) ADD <sup>Note 1</sup>	(MOV) (XCH) (ADD) <sup>Note 1</sup>	MOV XCH (ADD) <sup>Note 1</sup>	(MOV) <sup>Note 6</sup> (XCH) <sup>Note 6</sup> (ADD) <sup>Notes 1, 6</sup>	MOV (XCH) (ADD) <sup>Note 1</sup>	(MOV) (XCH) ADD <sup>Note 1</sup>	MOV XCH ADD <sup>Note 1</sup>	MOV	(MOV) (XCH) (ADD) <sup>Note 1</sup>		
r	MOV ADD <sup>Note 1</sup>	(MOV) (XCH) (ADD) <sup>Note 1</sup>	MOV XCH ADD <sup>Note 1</sup>	MOV XCH ADD <sup>Note 1</sup>	MOV XCH ADD <sup>Note 1</sup>	MOV XCH				ROR <sup>Note 3</sup>	MULU DIVUW INC DEC
saddr	MOV ADD <sup>Note 1</sup>	(MOV) <sup>Note 6</sup> (ADD) <sup>Note 1</sup>	MOV ADD <sup>Note 1</sup>	MOV XCH ADD <sup>Note 1</sup>							INC DEC DBNZ
sfr	MOV ADD <sup>Note 1</sup>	MOV (ADD) <sup>Note 1</sup>	MOV ADD <sup>Note 1</sup>								PUSH POP
!addr16 !!addr24	MOV	MOV ADD <sup>Note 1</sup>	MOV								
mem [saddrp] [%saddrg]		MOV ADD <sup>Note 1</sup>									
mem3											ROR4 ROL4
r3 PSWL PSWH	MOV	MOV									
B, C											DBNZ
STBC, WDM	MOV										
[TDE+] [TDE-]		(MOV) (ADD) <sup>Note 1</sup> MOVW <sup>Note 4</sup>							MOVBK <sup>Note 5</sup>		

- Notes**
- ADDC, SUB, SUBC, AND, OR, XOR, and CMP are identical to ADD.
  - There is no second operand, or the second operand is not an operand address.
  - ROL, RORC, ROLC, SHR, and SHL are identical to ROR.
  - XCHW, CMPME, CMPMNE, CMPMNC, and CMPMC are identical to MOVW.
  - XCHBK, CMPBKE, CMPBKNE, CMPBKNC, and CMPBKC are identical to MOVBK.
  - When saddr is saddr2 in this combination, the instruction has a short code length.

(2) 16-bit instructions (values in parentheses are combined to express AX description as rp.)

MOVM, XCHW, ADDW, SUBW, CMPW, MULUW, MULW, DIVUX, INCW, DECW, SHRW, SHLW, PUSH, POP, ADDWG, SUBWG, PUSHU, POPU, MOVTBLW, MACW, MACSW, SACW

Table 20-2. 16-bit Addressing Instructions

Second operand / First operand	#word	AX	rp rp'	saddrp saddrp'	sfrp	!addr16 !!addr24	mem [saddrp] [%saddrg]	[WHL+]	byte	n	None <sup>Note 2</sup>
AX	(MOVW) ADDW <sup>Note 1</sup>	(MOVW) (XCHW) (ADD) <sup>Note 1</sup>	(MOVW) (XCHW) (ADDW) <sup>Note 1</sup>	(MOVW) <sup>Note 3</sup> (XCHW) <sup>Note 3</sup> (ADDW) <sup>Notes 1,3</sup>	MOVW (XCHW) (ADDW) <sup>Note 1</sup>	(MOVW) XCHW	MOVW XCHW	(MOVW) (XCHW)			
rp	MOVW ADDW <sup>Note 1</sup>	(MOVW) (XCHW) (ADDW) <sup>Note 1</sup>	MOVW XCHW ADDW <sup>Note 1</sup>	MOVW XCHW ADDW <sup>Note 1</sup>	MOVW XCHW ADDW <sup>Note 1</sup>	MOVW				SHRW SHLW	MULW <sup>Note 4</sup> INCW DECW
saddrp	MOVW ADDW <sup>Note 1</sup>	(MOVW) <sup>Note 3</sup> (ADDW) <sup>Note 1</sup>	MOVW ADDW <sup>Note 1</sup>	MOVW XCHW ADDW <sup>Note 1</sup>							INCW DECW
sfrp	MOVW ADDW <sup>Note 1</sup>	MOVW (ADDW) <sup>Note 1</sup>	MOVW (ADDW) <sup>Note 1</sup>								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

- Notes**
1. SUBW and CMPW are identical to ADDW.
  2. There is no second operand, or the second operand is not an operand address.
  3. When saddrp is saddrp2 in this combination, this is a short code length instruction.
  4. MULUW and DIVUX are identical to MULW.



**(3) 24-bit instructions (values in parentheses are combined to express WHL description as rg.)**  
 MOVG, ADDG, SUBG, INCG, DECG, PUSH, POP

**Table 20-3. 24-bit Addressing Instructions**

Second operand \ First operand	#imm24	WHL	rg rg'	saddrg	!!addr24	mem1	[%saddrg]	SP	None <sup>Note</sup>
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
saddrg		(MOVG)	MOVG						
!!addr24		(MOVG)	MOVG						
mem1		MOVG							
[%saddrg]		MOVG							
SP	MOVG	MOVG							INCG DECG

**Note** There is no second operand, or the second operand is not an operand address.

**(4) Bit manipulation instructions**

MOV1, AND1, OR1, XOR1, SET1, CLR1, NOT1, BT, BF, BTCLR, BFSET

**Table 20-4. Bit Manipulation Instruction Addressing Instructions**

Second operand \ First operand	CY	saddr.bit A.bit PSWL.bit mem2.bit !addr16.bit !!addr24.bit	sfr.bit X.bit PSWH.bit	/saddr.bit /A.bit /PSWL.bit /mem2.bit /!addr16.bit /!!addr24.bit	None <sup>Note</sup>
CY		MOV1 AND1 OR1 XOR1		AND1 OR1	NOT1 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

**Note** There is no second operand, or the second operand is not an operand address.

**(5) Call return instructions and branch instructions**

CALL, CALLF, CALLT, BRK, RET, RETI, RETB, RETCS, RETCSB, BRKCS, BR, BNZ, BNE, BZ, BE, BNC, BNL, BC, BL, BNV, BPO, BV, BPE, BP, BN, BLT, BGE, BLE, BGT, BNH, BH, BF, BT, BTCLR, BFSET, DBNZ

**Table 20-5. Call Return Instructions and Branch Instruction Addressing Instructions**

Instruction Address Operand	\$addr20	!addr20	!addr16	!!addr20	rp	rg	[rp]	[rg]	!addr11	[addr5]	RBn	None
Basic instructions	BC <sup>Note</sup> BR	CALL BR	CALL BR RETCS RETCSB	CALL BR	CALL BR	CALL BR	CALL BR	CALL BR	CALLF CALLT		BRKCS	BRK RET RETI RETB
Composite instructions	BF BT BTCLR BFSET DBNZ											

**Note** BNZ, BNE, BZ, BE, BNC, BNL, BL, BNV, BPO, BV, BPE, BP, BN, BLT, BGE, BLE, BGT, BNH, and BH are identical to BC.

**(6) Other instructions**

ADJBA, ADJBS, CVTBW, LOCATION, SEL, NOT, EI, DI, SWRS

Absolute Maximum Ratings ( $T_A = 25^\circ\text{C}$ )

Parameter	Symbol	Conditions	Ratings	Unit	
Supply voltage	$V_{DD}$		-0.3 to +6.5	V	
	$V_{PP}$	$\mu\text{PD78F4976A}$ only	-0.3 to +10.5	V	
	$AV_{LOAD}$		$V_{DD} - 40$ to $V_{DD} + 0.3$	V	
	$AV_{DD}$		-0.3 to $V_{DD} + 0.3$	V	
	$AV_{SS}$		-0.3 to +0.3	V	
Input voltage	$V_{I1}$	Ports 0, 1 (other than analog input pin), ports 2, 4, 6, X1, X2, RESET	-0.3 to $V_{DD} + 0.3$	V	
	$V_{I2}$	Port 5	N-ch open drain	-0.3 to +13	V
			N-ch open drain, with mask option ( $\mu\text{PD784975A}$ only)	-0.3 to $V_{DD} + 0.3$	V
	$V_{I3}$	Ports 7, 8, 9	P-ch open drain	$V_{DD} - 40$ to $V_{DD} + 0.3$	V
Analog input voltage	$V_{AN}$	ANI0 to ANI11 (when used as analog input pin)	$AV_{SS} - 0.3$ to $AV_{DD} + 0.3$	V	
Output voltage	$V_{O1}$	Total of all pins of ports 2, 4 to 6	-0.3 to $V_{DD} + 0.3$	V	
	$V_{OD}$	Total of all pins of ports 7 to 10, FIP0 to FIP15	P-ch open drain $V_{DD} - 40$ to $V_{DD} + 0.3$	V	
Output current, low	$I_{OL}$	Per pin (ports 2, 6)	10	mA	
		Per pin (ports 4, 5)	20	mA	
		Total of all pins of ports 2, 4, 5, 6	200	mA	
Output current, high	$I_{OH}$	Per pin (ports 2, 4, 6)	-10	mA	
		Total of all pins of ports 2, 4, 6	-30	mA	
		Per pin (FIP0 to FIP15)	-15	mA	
		Per pin (FIP16 to FIP47)	-5	mA	
		Total of all pins of FIP0 to FIP47	-225	mA	
Total power dissipation	$P_T$	$T_A = -40$ to $+60^\circ\text{C}$	800	mW	
		$T_A = +85^\circ\text{C}$	600	mW	
Operating ambient temperature	$T_A$		-40 to +85	$^\circ\text{C}$	
Storage temperature	$T_{stg}$	$\mu\text{PD784975A}$	-65 to +150	$^\circ\text{C}$	
		$\mu\text{PD78F4976A}$	-65 to +125	$^\circ\text{C}$	

**Caution** Product quality may suffer if the absolute maximum rating is exceeded even momentarily for any parameter. That is, the absolute maximum ratings are rated values at which the product is on the verge of suffering physical damage, and therefore the product must be used under conditions that ensure that the absolute maximum ratings are not exceeded.

- Remarks**
1. Unless specified otherwise, the characteristics of alternate-function pins are the same as those of port pins.
  2. Refer to 14.7 **Calculation of Total Power Dissipation** for details of how to calculate the total power dissipation.

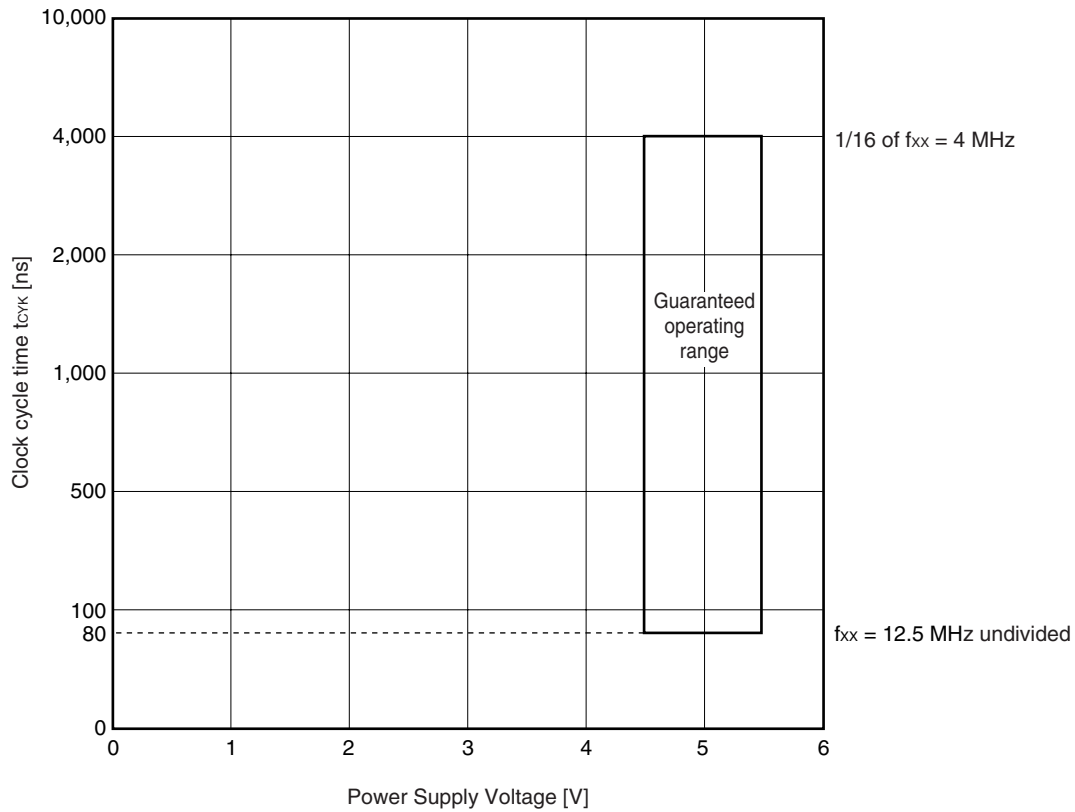
**Operating Conditions**

- Clock frequency

Clock Frequency	Supply Voltage
4 MHz ≤ f <sub>xx</sub> ≤ 12.5 MHz	4.5 V ≤ V <sub>DD</sub> ≤ 5.5 V

- Operating ambient temperature (T<sub>A</sub>): -40 to +85°C
- Power supply voltage and clock cycle time: Refer to **Figure 21-1**
- f<sub>xx</sub>: Main system clock frequency

**Figure 21-1. Power Supply Voltage and Clock Cycle Time**



**Capacitance (T<sub>A</sub> = 25°C, V<sub>DD</sub> = V<sub>SS</sub> = 0 V)**

Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit	
Input capacitance	C <sub>IN</sub>	f <sub>xx</sub> = 1 MHz Unmeasured pins returned to 0 V.			15	pF	
Output capacitance	C <sub>OUT</sub>		Ports 0, 1			35	pF
I/O capacitance	C <sub>IO</sub>		Port 10, FIP0 to FIP15			15	pF
			Ports 2, 4, 6			20	pF
			Port 5			35	pF
		Ports 7, 8, 9					

Main System Clock Oscillator Characteristics (T<sub>A</sub> = -40 to +85°C)

Resonator	Recommended Circuit	Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Ceramic resonator or crystal resonator		Oscillation frequency	f <sub>x</sub>		4		12.5	MHz
		Oscillation stabilization time <sup>Note</sup>	f <sub>sx</sub>	When reset is released		2 <sup>19</sup> /f <sub>xx</sub>		ns
				When STOP mode is released		<b>Note</b>		ns
External clock		Oscillation frequency	f <sub>x</sub>	ENMP = 0	8		25	MHz
				ENMP = 1	4		12.5	MHz
		Oscillation stabilization time <sup>Note</sup>	f <sub>sx</sub>	When reset is released		2 <sup>19</sup> /f <sub>xx</sub>		ns
				When STOP mode is released		<b>Note</b>		ns
		Input high-/low-level width	t <sub>wxH</sub> , t <sub>wxL</sub>		18		125	ns
Input rising/falling time	t <sub>xR</sub> , t <sub>xF</sub>		0		5	ns		

**Note** The oscillation stabilization time is the time required for oscillation to stabilize after power (V<sub>DD</sub>) application.

**Caution** When using the main system clock oscillator, wire as follows in the area enclosed by the broken lines in the above figures to avoid an adverse effect from wiring capacitance.

- Keep the wiring length as short as possible.
- Do not cross the wiring with other signal lines.
- Do not route the wiring near a signal line through which a high fluctuating current flows.
- Always make the ground point of the oscillator capacitor the same potential as V<sub>SS1</sub>.
- Do not ground the capacitor to a ground pattern through which a high current flows.
- Do not fetch signals from the oscillator.

- Remarks**
1. f<sub>x</sub>: Main system clock oscillation frequency  
f<sub>xx</sub>: Main system clock frequency
  2. For the resonator selection and oscillator constant, customers are requested to either evaluate the oscillation themselves or apply to the resonator manufacturer for evaluation.

DC Characteristics (T<sub>A</sub> = –40 to +85°C, V<sub>DD</sub> = AV<sub>DD</sub> = 4.5 to 5.5 V, V<sub>SS</sub> = AV<sub>SS</sub> = 0 V) (1/3)

Parameter	Symbol	Conditions		MIN.	TYP.	MAX.	Unit
Input voltage, low	V <sub>IL1</sub>	P00 to P03, P10 to P17, P26, P40 to P47, P61		0		0.3V <sub>DD</sub>	V
	V <sub>IL2</sub>	P20, P63 to P67, X1, X2, $\overline{\text{RESET}}$		0		0.2V <sub>DD</sub>	V
	V <sub>IL3</sub>	P25, P27, P55 <sup>Note 1</sup> , P57 <sup>Note 1</sup> , P60, P62		0		0.1V <sub>DD</sub> + 0.4	V
	V <sub>IL4</sub>	P50 to P57 (N-ch open drain)		0		0.3V <sub>DD</sub>	V
	V <sub>IL5</sub>	P70 to P77, P80 to P87, P90 to P97 (P-ch open drain)		V <sub>DD</sub> – 35		0.3V <sub>DD</sub>	V
Input voltage, high	V <sub>IH1</sub>	P00 to P03, P10 to P17, P26, P40 to P47, P61		0.7V <sub>DD</sub>		V <sub>DD</sub>	V
	V <sub>IH2</sub>	P20, P63 to P67, X1, X2, $\overline{\text{RESET}}$		0.8V <sub>DD</sub>		V <sub>DD</sub>	V
	V <sub>IH3</sub>	P25, P27, P55 <sup>Note 1</sup> , P57 <sup>Note 1</sup> , P60, P62		0.3V <sub>DD</sub> + 0.7		V <sub>DD</sub>	V
	V <sub>IH4</sub>	P50 to P57 (N-ch open drain)		0.7V <sub>DD</sub>		12	V
	V <sub>IH5</sub>	P70 to P77, P80 to P87, P90 to P97 (P-ch open drain)		0.7V <sub>DD</sub>		V <sub>DD</sub>	V
Output voltage, low	V <sub>OL1</sub>	I <sub>OL</sub> = 1.6 mA	P20, P25 to P27, P60 to P67			0.4	V
	V <sub>OL2</sub>	I <sub>OL</sub> = 10 mA	P40 to P47			1.5	V
	V <sub>OL3</sub>	I <sub>OL</sub> = 15 mA	P50 to P57			2.0	V
Output voltage, high	V <sub>OH1</sub>	I <sub>OH</sub> = –1 mA		V <sub>DD</sub> – 1.0			V
	V <sub>OH2</sub>	I <sub>OH</sub> = –100 μA		V <sub>DD</sub> – 0.5			V
Input leakage current, low	I <sub>LIL1</sub>	V <sub>I</sub> = 0 V	For pins other than P50 to P57, P70 to P77, P80 to P87, P90 to P97, X1, and X2			–10	μA
	I <sub>LIL2</sub>					–20	μA
	I <sub>LIL3</sub>					–10 <sup>Note 2</sup>	μA
	I <sub>LIL4</sub>	V <sub>I</sub> = –35 V	P70 to P77, P80 to P87, P90 to P97			–10	μA
Input leakage current, high	I <sub>LIH1</sub>	V <sub>I</sub> = V <sub>DD</sub>	For pins other than P50 to P57, P70 to P77, P80 to P87, P90 to P97, X1, and X2			10	μA
	I <sub>LIH2</sub>					20	μA
	I <sub>LIH3</sub>	V <sub>I</sub> = 12 V	P50 to P57			10	μA
	I <sub>LIH4</sub>	V <sub>I</sub> = V <sub>DD</sub>	P70 to P77, P80 to P87, P90 to P97			10	μA
Output leakage current, low <sup>Note 3</sup>	I <sub>LOL1</sub>	V <sub>O</sub> = 0 V				–10	μA
	I <sub>LOL2</sub>	V <sub>O</sub> = V <sub>LOAD</sub> = –35 V	P70 to P77, P80 to P87, P90 to P97, P100 to P107, FIP0 to FIP15			–10	μA
Output leakage current, high <sup>Note 3</sup>	I <sub>LOH1</sub>	V <sub>O</sub> = V <sub>DD</sub>				10	μA
	I <sub>LOH2</sub>	V <sub>O</sub> = 12 V				10	μA

- Notes**
- High-level and low-level input voltages for P55 and P57 apply to V<sub>IH3</sub> and V<sub>IL3</sub> only when SIO2 is used. They are V<sub>IH4</sub> and V<sub>IL4</sub> when the port is used.
  - When pull-up resistors are not connected to P50 to P57 (specified by a mask option), a low-level input leakage current of –200 μA (MAX.) flows for only 2 clocks after a read instruction has been executed to port 5 (P50 to P57). At times other than this 2-clock interval, a –10 μA (MAX.) current flows.
  - The current flowing to on-chip pull-up and pull-down resistors is not included.

**Remark** Unless otherwise specified, the characteristics of alternate-function pins are the same as those of port pins.

**DC Characteristics (T<sub>A</sub> = -40 to +85°C, V<sub>DD</sub> = AV<sub>DD</sub> = 4.5 to 5.5 V, V<sub>SS</sub> = AV<sub>SS</sub> = 0 V) (2/3)**

Parameter	Symbol	Conditions		MIN.	TYP.	MAX.	Unit
VFD output current	I <sub>OD</sub>	V <sub>OD</sub> = V <sub>DD</sub> - 2 V	FIP0 to FIP15			-10	mA
			FIP16 to FIP47			-3	mA
V <sub>DD</sub> power supply current <sup>Note</sup>	I <sub>DD1</sub>	Operating mode	μDP784975A		15	35	mA
			μDP78F4976A		20	40	mA
	I <sub>DD2</sub>	HALT mode	μDP784975A		7	18	mA
			μDP78F4976A		8	20	mA
	I <sub>DD3</sub>	IDLE mode	When watch timer operation stops		1	2.5	mA
IDLE mode		When watch timer operates		1.5	3.5	mA	
Data retention voltage	V <sub>DDDR</sub>	STOP mode		2.5		5.5	V
Data retention power supply current <sup>Note</sup>	I <sub>DDDR</sub>	STOP mode	V <sub>DD</sub> = 2.5 V		2	10	μA
			V <sub>DD</sub> = 4.5 to 5.5 V		10	50	μA
Software pull-up resistor	R <sub>1</sub>	V <sub>I</sub> = 0 V	P20, P25 to P27, P40 to P47, P60 to P67	10	30	100	kΩ
On-chip mask option pull-up resistor (μPD784975A only)	R <sub>2</sub>	P50 to P57		20	40	90	kΩ

**Note** The current flowing to ports, the VFD output pin, software pull-up resistors and on-chip pull-down resistors (specified by a mask option) is not included.

**Remark** Unless otherwise specified, the characteristics of alternate-function pins are the same as those of port pins.

**DC Characteristics ( $T_A = -40$  to  $+85^\circ\text{C}$ ,  $V_{DD} = AV_{DD} = 4.5$  to  $5.5$  V,  $V_{SS} = AV_{SS} = 0$  V) (3/3)**

Parameter	Symbol	Conditions		MIN.	TYP.	MAX.	Unit
On-chip pull-down resistor <sup>Note 1</sup>	R <sub>30</sub>	$V_{OD} - V_{LOAD} = 35$ V	FIP0 to FIP15	30	90	230	k $\Omega$
		$T_A = -40$ to $+85^\circ\text{C}$		50	90	165	k $\Omega$
	$V_{OD} - V_{LOAD} = 35$ V, $T_A = 20$ to $40^\circ\text{C}$	25		50	130	k $\Omega$	
	R <sub>31</sub> <sup>Note 2</sup>	$V_{OD} - V_{LOAD} = 35$ V					
On-chip mask-option pull-up resistor ( $\mu\text{PD784975A}$ only)	R <sub>40</sub>	$V_{OD} - V_{LOAD} = 35$ V	FIP16 to FIP47	30	90	230	k $\Omega$
		$T_A = -40$ to $+85^\circ\text{C}$		50	90	165	k $\Omega$
	$V_{OD} - V_{LOAD} = 35$ V, $T_A = 20$ to $+40^\circ\text{C}$	25		50	130	k $\Omega$	
	R <sub>41</sub>	$V_{OD} - V_{LOAD} = 35$ V					

**Notes 1.** The values for on-chip pull-down resistors (R<sub>30</sub> and R<sub>31</sub>) and on-chip mask-option pull-down resistors (R<sub>40</sub> and R<sub>41</sub>) can be selected according to the following conditions.

Part Number	Conditions		On-Chip Pull-Down Resistor <sup>Note 2</sup>	On-Chip Mask-Option Pull-Down Resistor <sup>Note 3</sup>
	With/Without Option Resistor	Resistor Value <sup>Note 1</sup>		
$\mu\text{PD78F4976A}$	–	–	R <sub>30</sub>	No
$\mu\text{PD784975A}$	Without	90 k $\Omega$ (TYP.)	R <sub>30</sub>	No
		50 k $\Omega$ (TYP.)	R <sub>31</sub>	
	With	90 k $\Omega$ (TYP.)	R <sub>30</sub>	R <sub>40</sub>
		50 k $\Omega$ (TYP.)	R <sub>31</sub>	R <sub>41</sub>

- Notes 1.** The mixed use of resistor values is not possible for both on-chip pull-down resistors and on-chip mask-option pull-down resistors.
- 2.** The resistor value selected for all of pins FIP0 to FIP15 is connected as an on-chip pull-down resistor regardless of the use of the option resistor.
- 3.** The use of the option resistor of the on-chip mask-option pull-down resistor can be specified in 1-bit units. The selected resistor value is connected to bits for which the use of the option resistor is specified.
- 2.** The  $\mu\text{PD784975A}$  only

**Remark** Unless otherwise specified, the characteristics of alternate-function pins are the same as those of port pins.



**AC Characteristics (T<sub>A</sub> = -40 to +85°C, V<sub>DD</sub> = AV<sub>DD</sub> = 4.5 to 5.5 V, V<sub>SS</sub> = AV<sub>SS</sub> = 0 V)**

**(1) Basic operation**

Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
System clock cycle time	t <sub>cyk</sub>		80			ns

**(2) External interrupt/reset timing**

Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
INTP <sub>n</sub> low-level width	t <sub>WITL</sub>		10			μs
INTP <sub>n</sub> high-level width	t <sub>WITH</sub>		10			μs
RESET low-level width	t <sub>WRSL</sub>		10			μs
RESET high-level width	t <sub>WRSH</sub>		10			μs

**Remark** n = 0 to 2

(3) Serial operation ( $T_A = -40$  to  $+85^\circ\text{C}$ ,  $V_{DD} = AV_{DD} = 4.5$  to  $5.5$  V,  $V_{SS} = AV_{SS} = 0$  V)

(a) 3-wire serial I/O mode ( $\overline{\text{SCKn}}$ : Internal clock output)

Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
$\overline{\text{SCKn}}$ cycle time	$t_{\text{KCY1}}$	Fastest setting by CSIMn: $f_{\text{xx}}/8$ ( $f_{\text{xx}} = 12.5$ MHz)	640			ns
$\overline{\text{SCKn}}$ low-level width	$t_{\text{KL1}}$	$t_{\text{KCY1}}/2 - 50$	270			ns
$\overline{\text{SCKn}}$ high-level width	$t_{\text{KH1}}$	$t_{\text{KCY1}}/2 - 50$	270			ns
SIn setup time (to $\overline{\text{SCKn}} \uparrow$ )	$t_{\text{SIK1}}$		70			ns
SIn hols time (from $\overline{\text{SCKn}} \uparrow$ )	$t_{\text{SI1}}$		80			ns
Delay time from $\overline{\text{SCKn}} \downarrow$ to SOn output	$t_{\text{KS01}}$				80	ns

**Remark**  $n = 0$  or  $1$

(b) 3-wire serial I/O mode ( $\overline{\text{SCKn}}$ : External clock input)

Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
$\overline{\text{SCKn}}$ cycle time	$t_{\text{KCY2}}$		640			ns
$\overline{\text{SCKn}}$ low-level width	$t_{\text{KL2}}$	$t_{\text{KCY2}}/2 - 50$	270			ns
$\overline{\text{SCKn}}$ high-level width	$t_{\text{KH2}}$	$t_{\text{KCY2}}/2 - 50$	270			ns
SIn setup time (to $\overline{\text{SCKn}} \uparrow$ )	$t_{\text{SIK2}}$		70			ns
SIn hols time (from $\overline{\text{SCKn}} \uparrow$ )	$t_{\text{SI2}}$		80			ns
Delay time from $\overline{\text{SCKn}} \downarrow$ to SOn output	$t_{\text{KS02}}$				80	ns

**Remark**  $n = 0$  or  $1$

**Caution** The  $\overline{\text{SCK2}}$  pin of serial interface 2 (SIO2) is an N-ch open drain pin. Therefore, if internal clock output is selected, the clock output from the pin is not a waveform with 50% duty. The values set to bit 1 (SCL21) and bit 0 (SCL20) of serial operation mode register 2 (CSIM2) are the possible clocks assuming that a 10 k $\Omega$  pull-up resistor is connected with operation at  $f_{\text{xx}} = 4.194$  MHz. Even if the clock set by the SCL21 and SCL20 bits of the CSIM2 register is selected, it may not operate correctly in conditions other than above, and depending on the board wiring capacitance, etc. Be sure to evaluate the operation before use.

(c) UART mode

Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
ASCK0 cycle time	$t_{\text{KCY3}}$		417			ns
ASCK0 low-level width	$t_{\text{KL3}}$		208			ns
ASCK0 high-level width	$t_{\text{KH3}}$		208			ns

**A/D Converter Characteristics (T<sub>A</sub> = -40 to +85°C, V<sub>DD</sub> = AV<sub>DD</sub> = 4.5 to 5.5 V, V<sub>SS</sub> = AV<sub>SS</sub> = 0 V)**

Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit	
Resolution			8	8	8	bit	
Overall error <sup>Notes 1, 2</sup>					±10	%FSR	
Conversion time <sup>Note 3</sup>	t <sub>CONV</sub>		14			μs	
Analog input voltage	V <sub>IAN</sub>		AV <sub>SS</sub>		AV <sub>DD</sub>	μs	
Resistance between AV <sub>DD</sub> and AV <sub>SS</sub> <sup>Note 4</sup>	R <sub>REF</sub>	When A/D converter is not operating		21.4		kΩ	
AV <sub>DD</sub> power supply current	AI <sub>DD1</sub>	Operation mode		1	3	mA	
	AI <sub>DD2</sub>	HALT mode		1	3	mA	
	AI <sub>DD3</sub>	IDLE mode <sup>Note 5</sup>		10	50	μA	
A/D converter data retention power supply current	AI <sub>DDDR</sub>	STOP mode <sup>Note 5</sup>	AV <sub>DDDR</sub> = 2.5 V		2	10	μA
			AV <sub>DDDR</sub> = 4.5 to 5.5 V		10	50	μA

- Notes**
1. Quantization error (±1/2 LSB) is not included.
  2. Overall error is indicated as a ratio to the full-scale value.
  3. Set the value so that the A/D conversion time is 14 μs or longer.
  4. This is the resistor value for the series resistor string only.
  5. Stop the A/D conversion operation (by setting bit 7 (ADCS) of the A/D converter mode register (ADM) to 0) before setting IDLE or STOP mode; otherwise the above specifications are not guaranteed.

**Flash Memory Programming Characteristics ( $\mu$ PD78F4976A only)****( $T_A = 10$  to  $40^\circ\text{C}$ ,  $V_{DD} = AV_{DD} = 4.5$  to  $5.5$  V,  $V_{SS} = AV_{SS} = 0$  V,  $V_{PP} = 9.7$  to  $10.3$  V) (1/2)****(1) Basic characteristics**

Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Operating frequency	$f_{XX}$		4		12.5	MHz
Oscillation frequency <sup>Note</sup>	$f_X$	Other than handshake mode	8		25	MHz
		Handshake mode	4		12.5	MHz
Power supply voltage	$V_{DD}$		4.5		5.5	V
	$V_{PPL}$	When detecting $V_{PP}$ low level	0		$0.2V_{DD}$	V
	$V_{PP}$	When detecting $V_{PP}$ high level	$0.9V_{DD}$		$1.1V_{DD}$	V
	$V_{PPH}$	When detecting $V_{PP}$ high voltage	9.7	10	10.3	V
Operating temperature	$T_A$		-40		85	$^\circ\text{C}$
Storage temperature	$T_{stg}$		-65		125	$^\circ\text{C}$
Programming temperature	$T_{PRG}$		10		40	$^\circ\text{C}$

**Note** Use the ceramic or crystal resonator at  $f_X = 4$  to 12.5 MHz.

- Remarks**
1. After executing the program command, execute the verify command to confirm that the write operation has been completed normally.
  2. Handshake mode is the CSI write mode that uses P20. Handshake mode can be used with the PG-FR3 and FL-PR3.

**Flash Memory Programming Characteristics ( $\mu$ PD78F4976A only)**

( $T_A = 10$  to  $40^\circ\text{C}$ ,  $V_{DD} = AV_{DD} = 4.5$  to  $5.5$  V,  $V_{SS} = AV_{SS} = 0$  V,  $V_{PP} = 9.7$  to  $10.3$  V) (2/2)

**(2) Write erase characteristics**

Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
$V_{PP}$ power supply voltage	$V_{PP2}$	During flash memory programming	9.7	10.0	10.3	V
$V_{DD}$ power supply current	$I_{DD}$	When $V_{PP} = V_{PP2}$ , $f_{XX} = 12.5$ MHz			40	mA
$V_{PP}$ power supply current	$I_{PP}$	When $V_{PP} = V_{PP2}$			100	mA
Step erase time	$T_{er}$	<b>Note 1</b>		0.2		s
Overall erase time per area	$T_{era}$	When step erase time = 0.2 s <sup>Note 2</sup>			20	s/area
Write-back time	$T_{wb}$	<b>Note 3</b>		50		ms
Number of write-backs per write-back command	$C_{wb}$	When write-back time = 50 ms <sup>Note 4</sup>			60	times/write-back command
Number of erase/write-backs	$C_{erwb}$				16	times
Step write time	$T_{wr}$	<b>Note 5</b>		50		$\mu$ s
Overall write time per word	$T_{wrw}$	When step write time = 50 $\mu$ s (1 word = 1 byte) <sup>Note 6</sup>	50		500	$\mu$ s/word
Number of rewrites per area	$C_{erwr}$	1 erase + 1 write after erase = 1 rewrite <sup>Note 7</sup>	20 <sup>Note 8</sup>			times/area

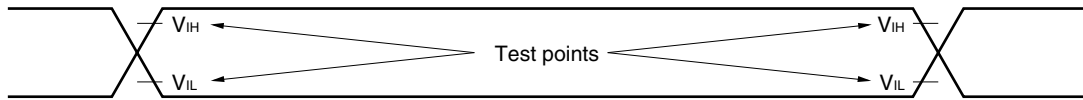
- Notes**
1. The recommend setting value for the step erase time is 0.2 s.
  2. The rewrite time before erasure and the erase verify time (write-back time) is not included.
  3. The recommended setting value for the write-back time is 50 ms.
  4. Write-back is executed once by the issuance of the write-back command. Therefore, the retry times must be the maximum value minus the number of commands issued.
  5. Recommended value of the step write time is 50  $\mu$ s.
  6. The actual write time per word is 100  $\mu$ s longer. The internal verify time during or after a write is not included.
  7. When a product is first written after shipment, "erase  $\rightarrow$  write" and "write only" are both taken as one rewrite.  
Example P: Write, E: Erase  
Shipped product  $\rightarrow$  P  $\rightarrow$  E  $\rightarrow$  P  $\rightarrow$  E  $\rightarrow$  P: 3 rewrites  
Shipped product  $\rightarrow$  E  $\rightarrow$  P  $\rightarrow$  E  $\rightarrow$  P  $\rightarrow$  E  $\rightarrow$  P: 3 rewrites
  8. The operation when rewriting is performed more than 20 times cannot be guaranteed.

- Remarks**
1. The range of the operating clock during flash memory programming is the same as the range during normal operation.
  2. When using the PG-FP3 or FL-PR3, the time parameters that need to be downloaded from the parameter files for write/erase are automatically set. Unless otherwise directed, do not change the set values.

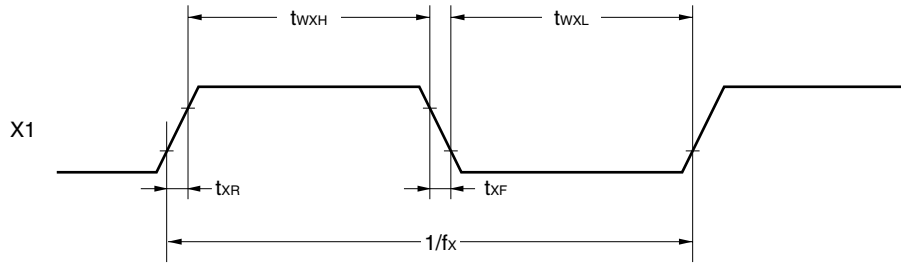
**Data Retention Characteristics ( $T_A = -40$  to  $+85^\circ\text{C}$ ,  $V_{DD} = AV_{DD} = 4.5$  to  $5.5$  V,  $V_{SS} = AV_{SS} = 0$  V)**

Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Data retention voltage	$V_{DDDR}$		2.5		5.5	V
Data retention power supply current	$I_{DDDR}$	$V_{DD} = 2.5$ V		2	10	$\mu\text{A}$
		$4.5 \text{ V} \leq V_{DD} \leq 5.5 \text{ V}$		10	50	$\mu\text{A}$
$V_{DD}$ rise time	$t_{RVD}$		200			$\mu\text{s}$
$V_{DD}$ fall time	$t_{FVD}$		200			$\mu\text{s}$
$V_{DD}$ hold power supply voltage (from STOP mode setting)	$t_{HVD}$		0			ms
STOP release signal input time	$t_{DREL}$		0			ms
Oscillation stabilization wait time	$t_{WAIT}$	Crystal resonator	30			ms
		Ceramic resonator	5			ms
Data retention low-level input voltage	$V_{ILDR}$	All input ports	0		$0.1V_{DDDR}$	V
Data retention high-level input voltage	$V_{IHDR}$	All input ports	$0.9V_{DDDR}$		$V_{DDDR}$	V

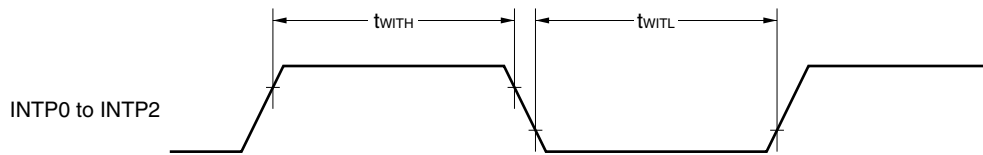
**AC Timing Test Points**



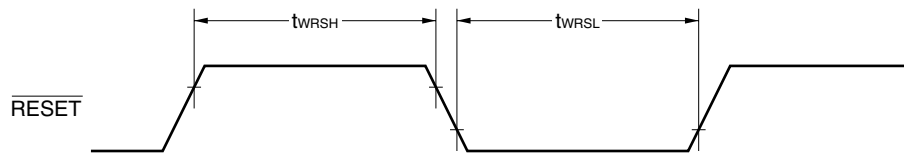
**Clock Timing**



**Interrupt Input Timing**

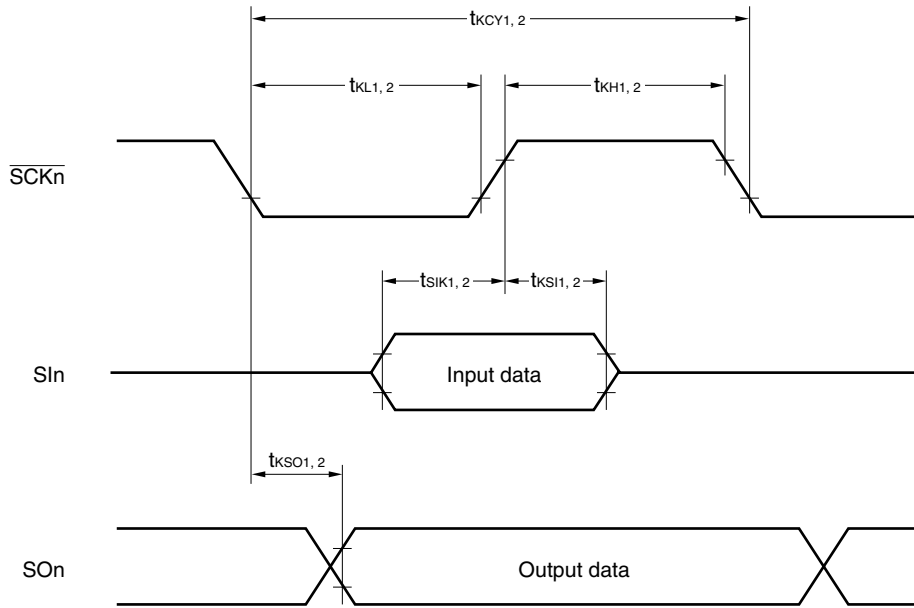


**Reset Input Timing**



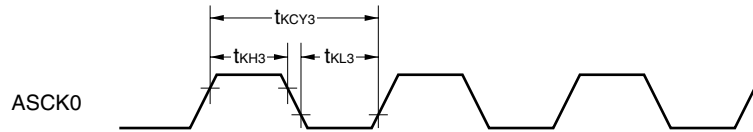
Serial Operation

(1) 3-wire serial I/O mode



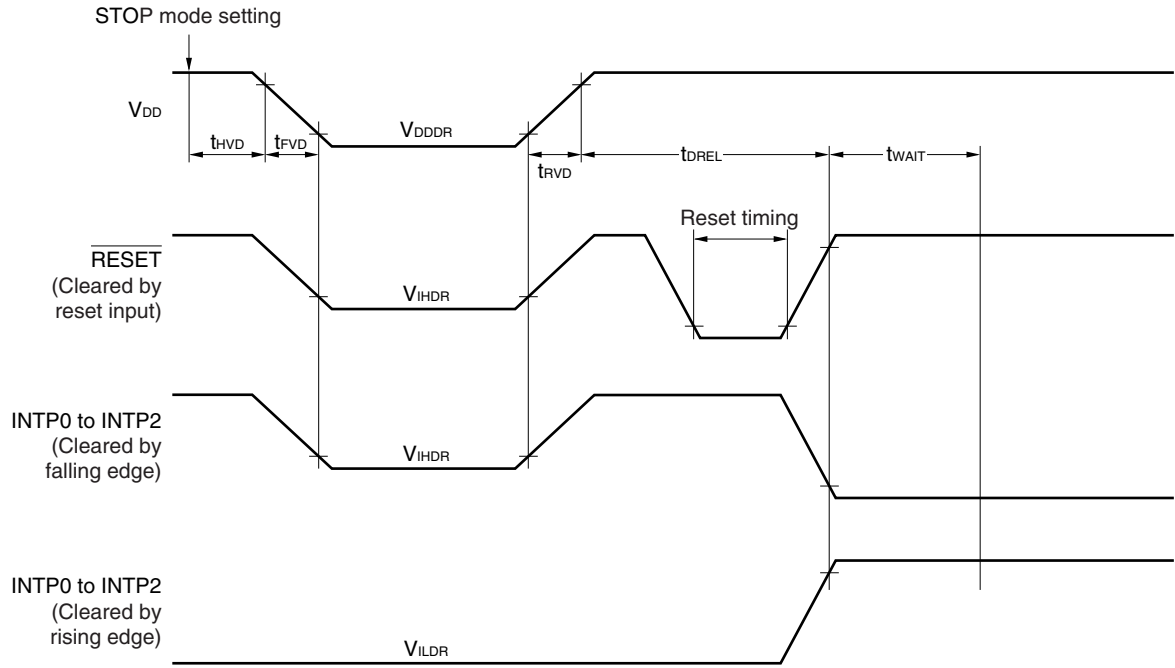
Remark  $n = 0$  or  $1$

(2) UART mode

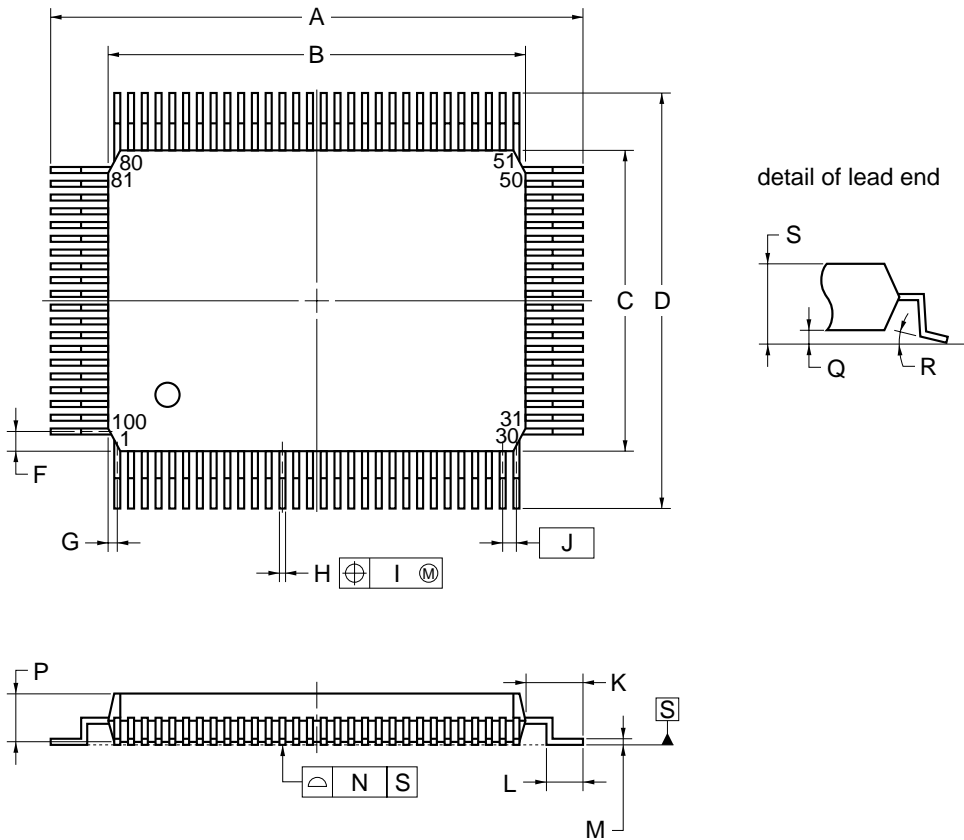




Data Retention Characteristics



## 100-PIN PLASTIC QFP (14x20)

**NOTE**

Each lead centerline is located within 0.15 mm of its true position (T.P.) at maximum material condition.

ITEM	MILLIMETERS
A	23.6±0.4
B	20.0±0.2
C	14.0±0.2
D	17.6±0.4
F	0.8
G	0.6
H	0.30±0.10
I	0.15
J	0.65 (T.P.)
K	1.8±0.2
L	0.8±0.2
M	0.15 <sup>+0.10</sup> <sub>-0.05</sub>
N	0.10
P	2.7±0.1
Q	0.1±0.1
R	5°±5°
S	3.0 MAX.

**P100GF-65-3BA1-4**

**Remark** The external dimensions and material of the ES version are the same as those of the mass-produced version.

## CHAPTER 23 RECOMMENDED SOLDERING CONDITIONS

The  $\mu$ PD784975A should be soldered and mounted under the following recommended conditions.

For details of the recommended soldering conditions, refer to the document **Semiconductor Device Mounting Technology Manual (C10535E)**.

For soldering methods and conditions other than those recommended below, contact an NEC sales representative.

**Remark** The recommended soldering conditions for the  $\mu$ PD78F4976A are undetermined.

**Table 23-1. Surface Mounting Type Soldering Conditions**

**$\mu$ PD784975AGF-xxx-3BA: 100-pin plastic QFP (14 × 20)**

Soldering Method	Soldering Conditions	Recommended Condition Symbol
Infrared reflow	Package peak temperature: 235°C, Time: 30 seconds max. (at 210°C or higher), Count: Two times or less	IR35-00-2
VPS	Package peak temperature: 215°C, Time: 40 seconds max. (at 200°C or higher), Count: Two times or less	VP15-00-2
Wave soldering	Solder bath temperature: 260°C max., Time: 10 seconds max., Count: Once, Preheating temperature: 120°C max. (package surface temperature)	WS60-00-1
Partial heating	Pin temperature: 300°C max., Time: 3 seconds max. (per pin row)	—

**Caution** Do not use different soldering methods together (except for partial heating).

## APPENDIX A DEVELOPMENT TOOLS

The configurations of the development tools necessary for developing the systems that use the  $\mu$ PD784976A Subseries products are shown in the following pages.

- **Regarding the PC98-NX series**

Unless otherwise specified, products supported by IBM PC/AT™ compatibles can also be used in the PC98-NX series. When using the PC98-NX series, refer to the explanation of IBM PC/AT compatibles.

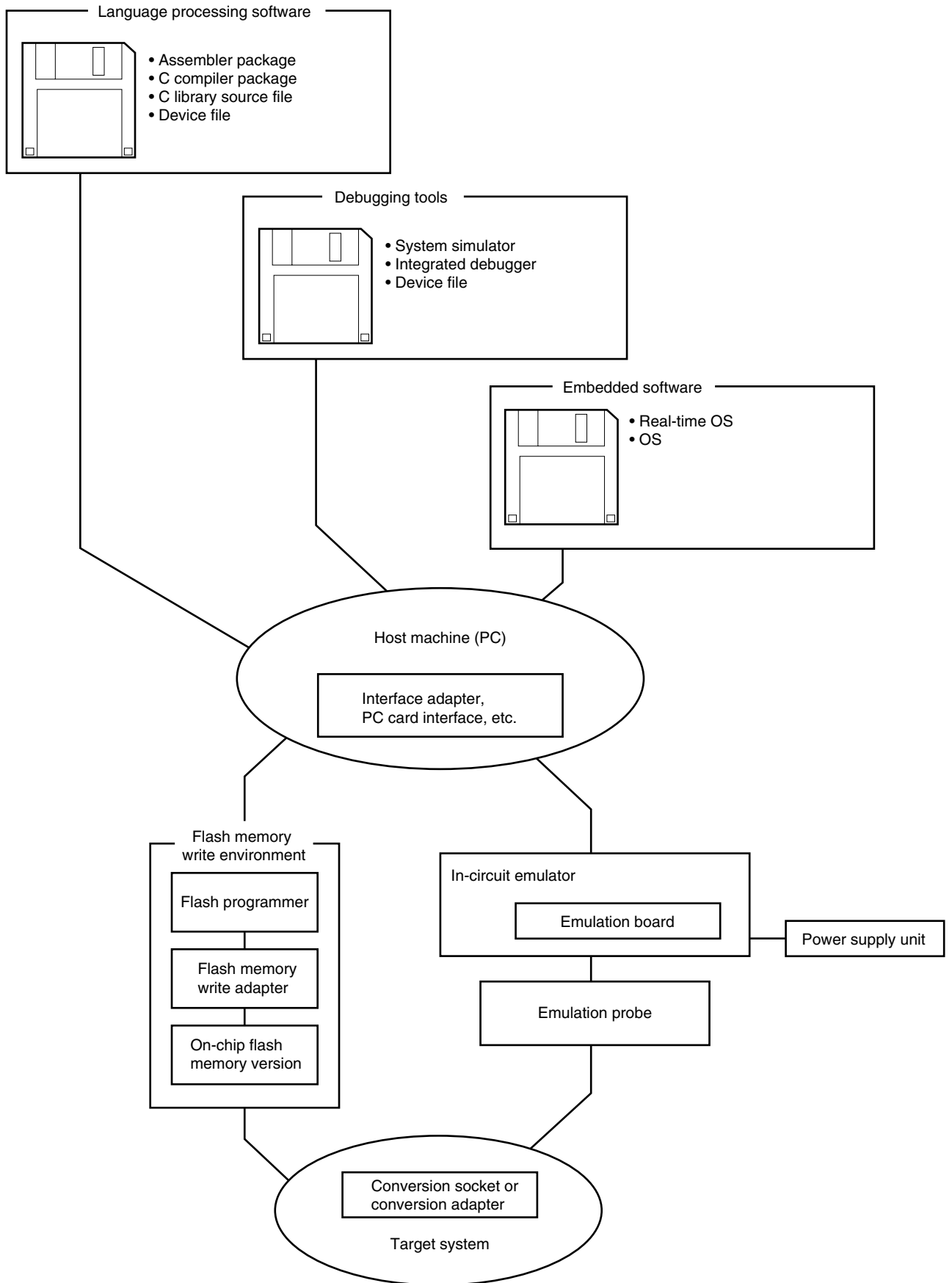
- **Regarding Windows**

Unless otherwise specified, “Windows” indicates the following OSs.

- Windows 95, 98, 2000
- Windows NT™ Ver. 4.0

★

Figure A-1. Development Tool Configuration



**A.1 Language Processing Software**

★

<p>SP78K4 78K/IV Series software package</p>	<p>This is a software package that includes the development tools common to the 78K/IV Series.</p> <p>Part number: <math>\mu</math>SxxxxSP78K4</p>
<p>RA78K4 Assembler package</p>	<p>This assembler converts programs written in mnemonics into object codes executable with a microcontroller.</p> <p>Further, this assembler is provided with functions capable of automatically creating symbol tables and branch instruction optimization.</p> <p>This assembler should be used in combination with an optional device file (DF784976).</p> <p><b>&lt;Caution when using RA78K4 in PC environment&gt;</b> This assembler package is a DOS-based application. It can also be used in Windows, however, by using the Project Manager (included in assembler package) in Windows.</p> <p>Part number: <math>\mu</math>SxxxxRA78K4</p>
<p>CC78K4 C compiler package</p>	<p>This compiler converts programs written in C language into object codes executable with a microcontroller.</p> <p>This compiler should be used in combination with an optional assembler package and device file.</p> <p><b>&lt;Caution when using CC78K4 in PC environment&gt;</b> This C compiler package is a DOS-based application. It can also be used in Windows, however, by using the Project Manager (included in assembler package) in Windows.</p> <p>Part number: <math>\mu</math>SxxxxCC78K4</p>
<p>DF784976<sup>Note</sup> Device file</p>	<p>This file contains information peculiar to the device.</p> <p>This device file should be used in combination with an optional tool (RA78K4, CC78K4, SM78K4, and ID78K4-NS).</p> <p>Corresponding OSs and host machines differ depending on the tool to be used.</p> <p>Part number: <math>\mu</math>SxxxxDF784976</p>
<p>CC78K4-L C library source file</p>	<p>This is a source file of the functions that configure the object library included in the C compiler package.</p> <p>This file is required in order to match the object library included in C compiler package to the customer's specifications.</p> <p>The operating environment does not depend on the OS because this is a source file.</p> <p>Part number: <math>\mu</math>SxxxxCC78K4-L</p>

**Note** The DF784976 can be used in common with the RA78K4, CC78K4, SM78K4, and ID78K4-NS.

★ **Remark** xxxx in the part number differs depending on the host machine and OS used.

μSxxxxSP78K4

xxxx	Host Machine	OS	Supply Medium
AB17	PC-9800 series,	Windows (Japanese version)	CD-ROM
BB17	IBM PC/AT or compatibles	Windows (English version)	

μSxxxxRA78K4

μSxxxxCC78K4

xxxx	Host Machine	OS	Supply Medium
AB13	PC-9800 series, IBM PC/AT or compatibles	Windows (Japanese version)	3.5-inch 2HD FD
BB13		Windows (English version)	
AB17		Windows (Japanese version)	CD-ROM
BB17		Windows (English version)	
3P17	HP9000 series 700™	HP-UX™ (Rel. 10.10)	
3K17	SPARCstation™	SunOS™ (Rel. 4.1.4), Solaris™ (Rel. 2.5.1)	

μSxxxxDF784976

μSxxxxCC78K4-L

xxxx	Host Machine	OS	Supply Medium
AB13	PC-9800 series, IBM PC/AT or compatibles	Windows (Japanese version)	3.5-inch 2HD FD
BB13		Windows (English version)	
3P16	HP9000 series 700	HP-UX (Rel. 10.10)	DAT
3K13	SPARCstation	SunOS (Rel. 4.1.4), Solaris (Rel. 2.5.1)	3.5-inch 2HD FD
3K15			1/4-inch CGMT

## A.2 Flash Memory Writing Tools

Flashpro III (part number: FL-PR3, PG-FP3) Flash Programmer	Flash programmer dedicated to microcontrollers with on-chip flash memory.
FA-100GF Flash Memory Writing Adapter	Flash memory writing adapter used connected to the Flashpro III. • FA-100GF: 100-pin plastic QFP (GF-3BA type)

**Remark** FL-PR3 and FA-100GF are products made by Naito Densai Machida Mfg. Co., Ltd.  
Phone: +8-45-475-4191 Naito Densai Machida Mfg. Co., Ltd.

### A.3 Debugging Tools

#### ★ A.3.1 Hardware

IE-78K4-NS In-circuit emulator	The in-circuit emulator serves to debug hardware and software when developing application systems using a 78K/IV Series product. It supports integrated debugger (ID78K4-NS). This emulator should be used in combination with power supply unit, emulation probe, and interface adapter which is required to connect this emulator to the host machine.
IE-70000-MC-PS-B Power supply unit	This adapter is used for supplying power from a receptacle of 100 to 240 V AC.
IE-70000-98-IF-C Interface adapter	This adapter is required when using the PC-9800 series computer (except notebook type) as the IE-78K4-NS host machine (supporting C bus).
IE-70000-CD-IF-A PC card interface	These PC card and interface cable are required when using a notebook-type PC as the IE-78K4-NS host machine (supporting PCMCIA socket).
IE-70000-PC-IF-C Interface adapter	This adapter is required when using the IBM PC/AT compatible computers as the IE-78K4-NS host machine (supporting ISA bus).
IE-70000-PCI-IF-A Interface adapter	This adapter is required when using a personal computer provided with a PCI bus as the IE-78K4-NS host machine.
IE-784976-NS-EM1 Emulation board	This board emulates the operations of the peripheral hardware peculiar to a device. It should be used in combination with an in-circuit emulator.
NP-100GF Emulation probe	This probe is used to connect the in-circuit emulator to the target system and is designed for 100-pin plastic QFP (GF-3BA type).
EV-9200GF-100 Conversion socket (Refer to <b>Figures A-5 and A-6</b> )	This conversion socket connects the NP-100GF to the target system board designed to mount a 100-pin plastic QFP (GF-3BA type).
NP-100GF-TQ Emulation probe	This probe is used to connect the in-circuit emulator to the target system and is designed for 100-pin plastic QFP (GF-3BA type).
TGF-100RBP Conversion connector	This conversion connector connects the NP-100GF-TQ to the target system board designed to mount a 100-pin plastic QFP (GF-3BA type).

- Remarks**
1. NP-100GF and NP-100GF-TQ are products made by Naito Densei Machida Mfg. Co., Ltd.  
Phone: +8-45-475-4191 Naito Densei Machida Mfg. Co., Ltd.
  2. The TGF-100RBP is a product made by TOKYO ELETECH CORPORATION.  
For further information, contact Daimaru Kogyo, Ltd.  
Tokyo Electronic Division (TEL: +81-3-3820-7112)  
Osaka Electronic Division (TEL: +81-6-6244-6672)
  3. EV-9200GF-100 is sold in five-unit sets.



A.3.2 Software (1/2)

SM78K4 System simulator	This system simulator is used to perform debugging at C source level or assembler level while simulating the operation of the target system on a host machine. This simulator runs on Windows. Use of the SM78K4 allows the execution of application logical testing and performance testing on an independent basis from hardware development without having to use an in-circuit emulator, thereby providing higher development efficiency and software quality. The SM78K4 should be used in combination with an optional device file (DF784976). Part number: $\mu$ SxxxxSM78K4
----------------------------	---

**Remark** xxxx in the part number differs depending on the host machine and OS used.

$\mu$ SxxxxSM78K4

★

xxxx	Host Machine	OS	Supply Medium
AB13	IBM PC/AT compatibles	Windows (Japanese version)	3.5-inch 2HC FD
BB13		Windows (English version)	
AB17		Windows (Japanese version)	CD-ROM
BB17		Windows (English version)	

A.3.2 Software (2/2)

ID78K4-NS Integrated debugger (supporting in-circuit emulator IE-78K4-NS)	This debugger is a control program to debug 78K/IV Series microcontrollers. It adopts a graphical user interface, which is equivalent visually and operationally to Windows or OSF/Motif™. It also has an enhanced debugging function for C language programs, and thus trace results can be displayed on screen in C-language level by using the window integration function which links a trace result with its source program, disassembled display, and memory display. In addition, by incorporating function modules such as task debugger and system performance analyzer, the efficiency of debugging programs, which run on real-time OSs can be improved. It should be used in combination with the optional device file (DF784976).
	Part number: $\mu$ SxxxxID78K4-NS

**Remark** xxxx in the part number differs depending on the host machine and OS used.

$\mu$ SxxxxID78K4-NS

★

xxxx	Host Machine	OS	Supply Medium
AB13	IBM PC/AT compatibles	Windows (Japanese version)	3.5-inch 2HC FD
BB13		Windows (English version)	
AB17		Windows (Japanese version)	CD-ROM
BB17		Windows (English version)	

★ **A.4 Notes on Target System Design**

The following shows a diagram of the connection conditions between the emulation probe, conversion socket, and conversion connector. Design your system making allowances for conditions such as the form of parts mounted on the target system as shown below.

**Figure A-2. Distance Between In-Circuit Emulator and Conversion Socket**

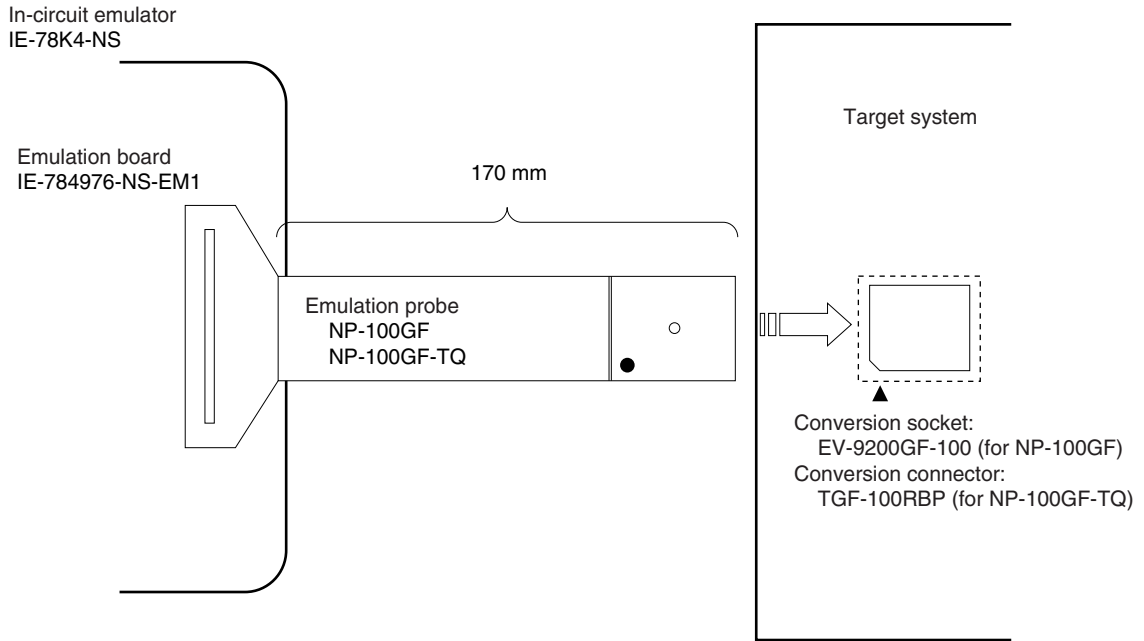
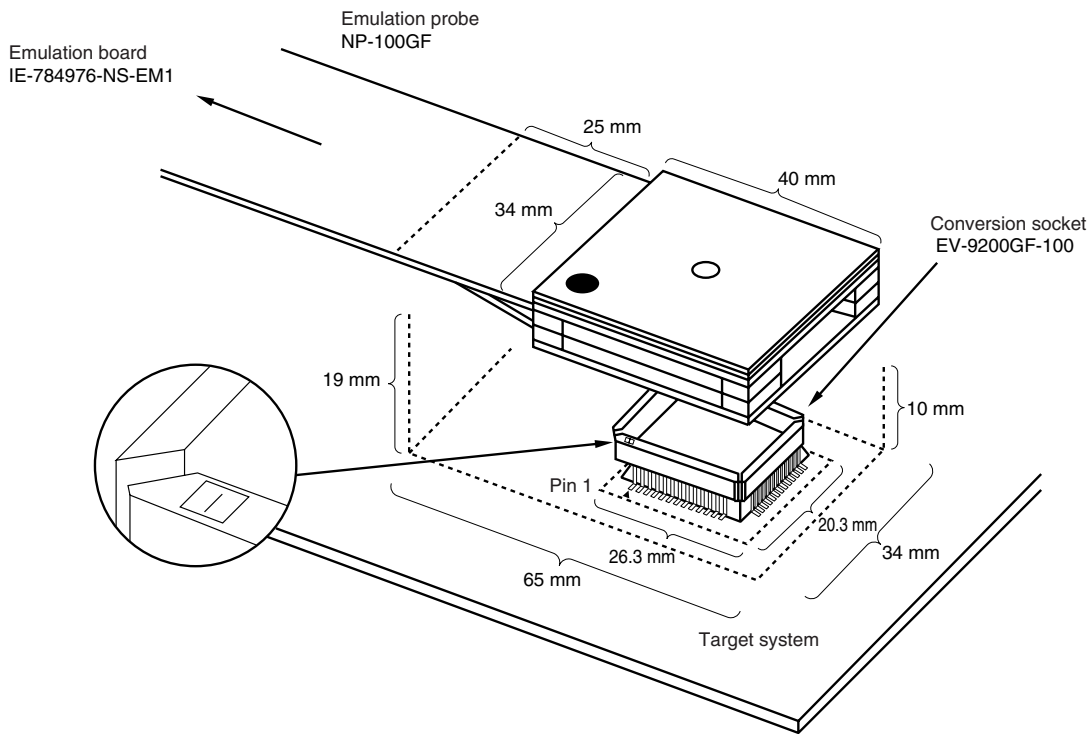
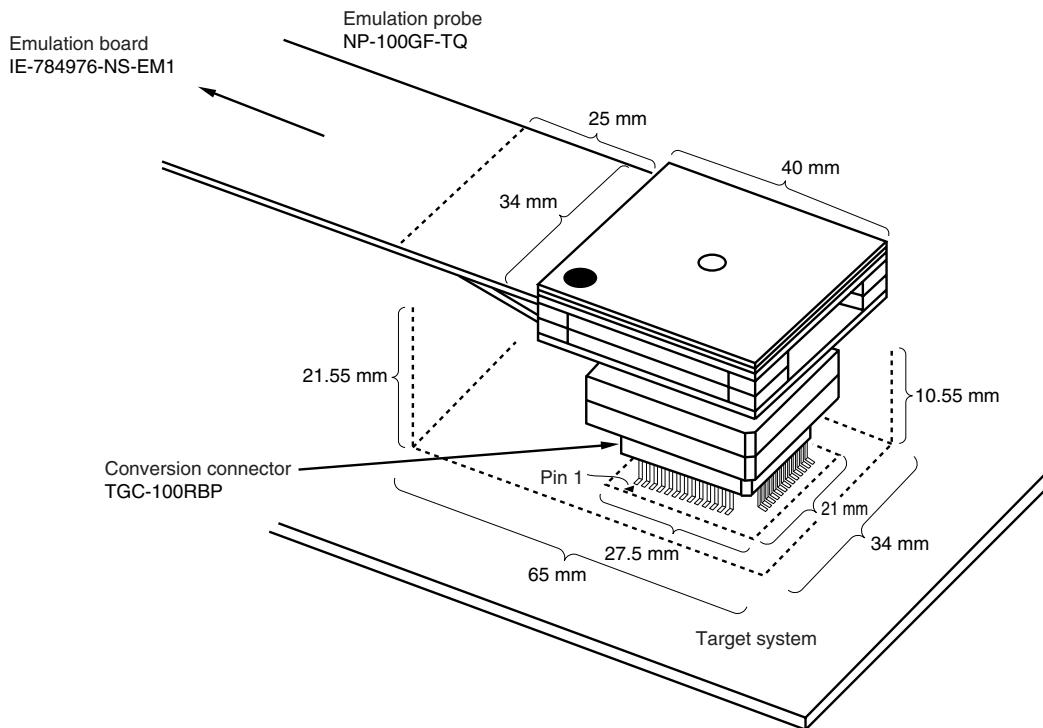


Figure A-3. Conditions for Target System Connection (1)



**Remark** The NP-100GF is a product of Naito Densai Machida Mfg. Co., Ltd.

Figure A-4. Conditions for Target System Connection (2)

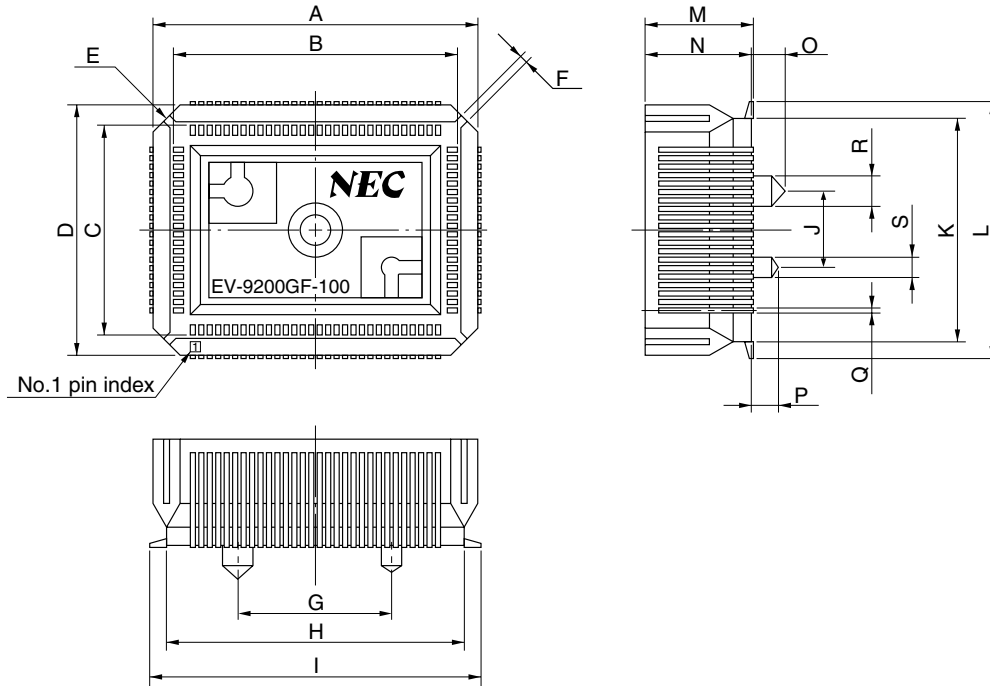


**Remark** The NP-100GF-TQ is a product of Naito Densai Machida Mfg. Co., Ltd.  
The TGC-100RBP is a product of Tokyo Eletech Corporation.

**A.5 Conversion Socket (EV-9200GF-100)**

- (1) The package drawing of the conversion socket (EV-9200GF-100) and recommended board installation pattern  
 This is combined with the NP-100GF or EP-78064GF-R and mounted on the board.

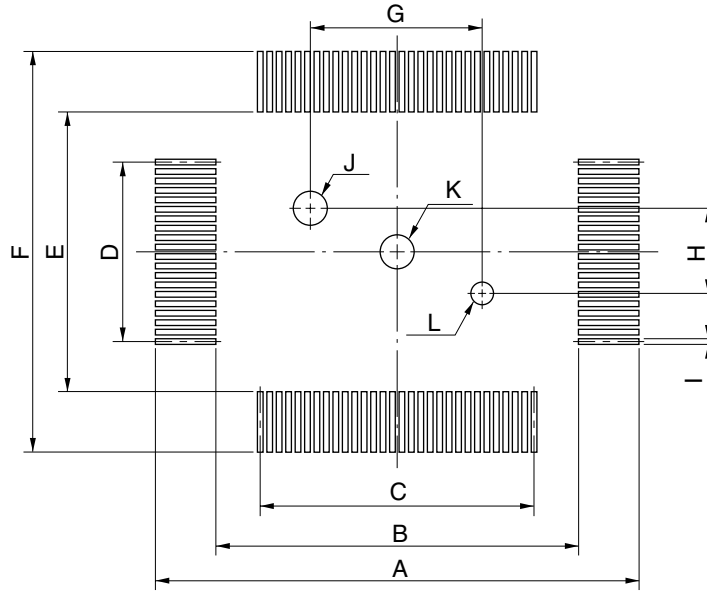
**Figure A-5. Package Drawing of EV-9200GF-100 (Reference) (Units: mm)**



EV-9200GF-100-G0E

ITEM	MILLIMETERS	INCHES
A	24.6	0.969
B	21	0.827
C	15	0.591
D	18.6	0.732
E	4-C 2	4-C 0.079
F	0.8	0.031
G	12.0	0.472
H	22.6	0.89
I	25.3	0.996
J	6.0	0.236
K	16.6	0.654
L	19.3	0.76
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	φ 2.3	φ 0.091
S	φ 1.5	φ 0.059

Figure A-6. Recommended Board Installation Pattern of EV-9200GF-100 (Reference) (Units: mm)



EV-9200GF-100-P1E

ITEM	MILLIMETERS	INCHES
A	26.3	1.035
B	21.6	0.85
C	$0.65 \pm 0.02 \times 29 = 18.85 \pm 0.05$	$0.026^{+0.001}_{-0.002} \times 1.142 = 0.742^{+0.002}_{-0.002}$
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	15.6	0.614
F	20.3	0.799
G	$12 \pm 0.05$	$0.472^{+0.003}_{-0.002}$
H	$6 \pm 0.05$	$0.236^{+0.003}_{-0.002}$
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).

## APPENDIX B SOFTWARE FOR EMBEDDED USE

The following software for embedded use is available to help users develop and maintain programs for the  $\mu$ PD784976A Subseries microcomputers.

### Real-time OS

RX78K/4 Real-time OS	This real-time OS complies with the $\mu$ ITRON specification. It consists of the RX78K/4 nucleus and a tool (configurator) for creating information tables. It is used in combination with an optional assembler package (RA78K4) and device file (DF84976). <b>&lt;Caution when using RX78/IV in PC environment&gt;</b> This real-time OS is a DOS-based application. Execute it from the Windows DOS prompt. <hr/> Part number: $\mu$ SxxxxRX78K4- $\Delta\Delta\Delta\Delta$
-------------------------	---

**Caution** Before purchasing RX78K/4, submit a purchase application form and conclude a license agreement.

**Remark** Codes “xxxx” and “ $\Delta\Delta\Delta\Delta$ ” in the part number used on the order form vary depending on the host machine and OS on which RX78K/IV is used.

$\mu$ SxxxxSM78K4- $\Delta\Delta\Delta\Delta$

$\Delta\Delta\Delta\Delta$	Product Overview	Upper Limit on Quantity Usable for Mass Production
001	Object for evaluation	Not to be used for mass production
100K	Objects for mass production	100,000 units
001M		1,000,000 units
010M		10,000,000 units
S01	Source program	Source program of object for mass production

xxxx	Host Machine	OS	Distribution Medium
AA13	PC-9800 series	Windows (Japanese version) <sup>Note</sup>	3.5-inch 2HD FD
AB13	IBM PC/AT compatible	Windows (Japanese version) <sup>Note</sup>	3.5-inch 2HC FD
BB13		Windows (English version) <sup>Note</sup>	
3P16	HP9000 series 700	HP-UX (Rel. 10.10)	DAT (DOS)
3K13	SPARCstation	SunOS (Rel. 4.1.4), Solaris (Rel. 2.5.1)	3.5-inch 2HD FD
3K15			1/4-inch CGMT

**Note** Operable also in a DOS environment

## APPENDIX C REGISTER INDEX

### C.1 Register Name Index (Alphabetic Order)

16-bit capture/compare register 00 (CR00) .....	116
16-bit capture/compare register 01 (CR01) .....	117
16-bit timer counter 0 (TM0) .....	116
16-bit timer mode control register 0 (TMC0) .....	118
8-bit compare register 50 (CR50) .....	150
8-bit compare register 51 (CR51) .....	150
8-bit timer counter 50 (TM50) .....	150
8-bit timer counter 51 (TM51) .....	150
8-bit timer mode control register 50 (TMC50) .....	152
8-bit timer mode control register 51 (TMC51) .....	152
<b>[A]</b>	
A/D conversion result register (ADCR) .....	175
A/D converter input select register (ADIS) .....	178
A/D converter mode register (ADM) .....	177
Asynchronous serial interface mode register 0 (ASIM0) .....	203
Asynchronous serial interface status register 0 (ASIS0) .....	205
<b>[B]</b>	
Baud rate generator control register 0 (BRGC0) .....	206
<b>[C]</b>	
Capture/compare control register 0 (CRC0) .....	119
<b>[D]</b>	
Display mode register 0 (DSPM0) .....	222
Display mode register 1 (DSPM1) .....	223
Display mode register 2 (DSPM2) .....	224
<b>[E]</b>	
External interrupt falling edge enable register (EGN0) .....	234
External interrupt rising edge enable register (EGP0) .....	234
<b>[I]</b>	
In-service priority register (ISPR) .....	248
Internal memory size switching register (IMS) .....	55
Interrupt control register .....	243
Interrupt mask register 0H (MK0H) .....	246
Interrupt mask register 0L (MK0L) .....	246
Interrupt mask register 1L (MK1L) .....	246
Interrupt mode control register (IMC) .....	249
Interrupt select control register (SNMI) .....	251



<b>[M]</b>		
	Memory expansion mode register (MM) .....	44
<b>[O]</b>		
	Oscillation mode select register (CC) .....	104
	Oscillation stabilization time specification register (OSTS) .....	105, 300
<b>[P]</b>		
	Port 0 (P0) .....	78
	Port 1 (P1) .....	79
	Port 10 (P10) .....	94
	Port 2 (P2) .....	80
	Port 2 mode register (PM2) .....	95
	Port 4 (P4) .....	83
	Port 4 mode register (PM4) .....	95
	Port 5 (P5) .....	84
	Port 5 mode register (PM5) .....	95
	Port 6 (P6) .....	88
	Port 6 mode register (PM6) .....	95
	Port 7 (P7) .....	91
	Port 8 (P8) .....	92
	Port 9 (P9) .....	93
	Port read 7 (PLR7) .....	91
	Port read 8 (PLR8) .....	92
	Port read 9 (PLR9) .....	93
	Prescaler mode register 0 (PRM0) .....	120
	Program status word (PSW) .....	56, 252
	Pull-up resistor option register (PUO) .....	97
	Pull-up resistor option register 2 (PU2) .....	96
<b>[R]</b>		
	Receive buffer register 0 (RXB0) .....	202
	Remote controller receive mode register (REMM) .....	121
<b>[S]</b>		
	Serial I/O shift register 0 (SIO0) .....	190
	Serial I/O shift register 1 (SIO1) .....	190
	Serial I/O shift register 2 (SIO2) .....	190
	Serial operation mode register 0 (CSIM0) .....	191
	Serial operation mode register 1 (CSIM1) .....	191
	Serial operation mode register 2 (CSIM2) .....	191
	Standby control register (STBC) .....	102, 298
<b>[T]</b>		
	Timer clock select register 50 (TCL50) .....	151
	Timer clock select register 51 (TCL51) .....	151
	Transmit shift register 0 (TXS0) .....	202

**[W]**

Watch timer clock select register (WTCL) .....	173
Watch timer mode control register (WTM) .....	171
Watchdog timer mode register (WDM) .....	166, 250

## C.2 Register Symbol Index

### [A]

ADCR:	A/D conversion result register .....	175
ADIC:	Interrupt control register .....	245
ADIS:	A/D converter input select register .....	178
ADM:	A/D converter mode register .....	177
ASIM0:	Asynchronous serial interface mode register 0 .....	203
ASIS0:	Asynchronous serial interface status register 0 .....	205

### [B]

BRGC0:	Baud rate generator control register 0 .....	206
--------	--	-----

### [C]

CC:	Oscillation mode select register .....	104
CR00:	16-bit capture/compare register 00 .....	116
CR01:	16-bit capture/compare register 01 .....	117
CR50:	8-bit compare register 50 .....	150
CR51:	8-bit compare register 51 .....	150
CRC0:	Capture/compare control register 0 .....	119
CSIIC0:	Interrupt control register .....	244
CSIIC1:	Interrupt control register .....	244
CSIIC2:	Interrupt control register .....	245
CSIM0:	Serial operation mode register 0 .....	191
CSIM1:	Serial operation mode register 1 .....	191
CSIM2:	Serial operation mode register 2 .....	191

### [D]

DSPM0:	Display mode register 0 .....	222
DSPM1:	Display mode register 1 .....	223
DSPM2:	Display mode register 2 .....	224

### [E]

EGN0:	External interrupt falling edge enable register 0 .....	234
EGP0:	External interrupt rising edge enable register 0 .....	234

### [I]

IMC:	Interrupt mode control register .....	249
IMS:	Internal memory size switching register .....	55
ISPR:	In-service priority register .....	248

### [K]

KSIC:	Interrupt control register .....	244
-------	----------------------------------	-----

### [M]

MK0H:	Interrupt mask register 0H .....	246
MK0L:	Interrupt mask register 0L .....	246
MK1L:	Interrupt mask register 1L .....	246
MM:	Memory expansion mode register .....	44

**[O]**

OSTS: Oscillation stabilization time specification register ..... 105, 300

**[P]**

P0: Port 0 ..... 78  
 P1: Port 1 ..... 79  
 P2: Port 2 ..... 80  
 P4: Port 4 ..... 83  
 P5: Port 5 ..... 84  
 P6: Port 6 ..... 88  
 P7: Port 7 ..... 91  
 P8: Port 8 ..... 92  
 P9: Port 9 ..... 93  
 P10: Port 10 ..... 94  
 PIC0: Interrupt control register ..... 244  
 PIC1: Interrupt control register ..... 244  
 PIC2: Interrupt control register ..... 244  
 PLR7: Port read 7 ..... 91  
 PLR8: Port read 8 ..... 92  
 PLR9: Port read 9 ..... 93  
 PM2: Port 2 mode register ..... 95  
 PM4: Port 4 mode register ..... 95  
 PM5: Port 5 mode register ..... 95  
 PM6: Port 6 mode register ..... 95  
 PRM0: Prescaler mode register 0 ..... 120  
 PSW: Program status word ..... 56, 252  
 PU2: Pull-up resistor option register 2 ..... 96  
 PUO: Pull-up resistor option register ..... 97

**[R]**

REMIC: Interrupt control register ..... 245  
 REMM: Remote controller receive mode register ..... 121  
 RXB0: Receive buffer register 0 ..... 202

**[S]**

SERIC0: Interrupt control register ..... 245  
 SIO0: Serial I/O shift register 0 ..... 190  
 SIO1: Serial I/O shift register 1 ..... 190  
 SIO2: Serial I/O shift register 2 ..... 190  
 SNMI: Interrupt select control register ..... 251  
 SRIC0: Interrupt control register ..... 245  
 STBC: Standby control register ..... 102, 298  
 STIC0: Interrupt control register ..... 245

[T]

TCL50:	Timer clock select register 50 .....	151
TCL51:	Timer clock select register 51 .....	151
TM0:	16-bit timer counter 0 .....	116
TM50:	8-bit timer counter 50 .....	150
TM51:	8-bit timer counter 51 .....	150
TMC0:	16-bit timer mode control register 0 .....	118
TMC50:	8-bit timer mode control register 50 .....	152
TMC51:	8-bit timer mode control register 51 .....	152
TMIC00:	Interrupt control register .....	244
TMIC01:	Interrupt control register .....	244
TMIC50:	Interrupt control register .....	245
TMIC51:	Interrupt control register .....	245
TXS0:	Transmit shift register 0 .....	202

[W]

WDTIC:	Interrupt control register .....	244
WDM:	Watchdog timer mode register .....	166, 250
WTCL:	Watch timer clock select register .....	173
WTIC:	Interrupt control register .....	249
WTIIC:	Interrupt control register .....	249
WTM:	Watch timer mode control register .....	171

## APPENDIX D REVISION HISTORY

A history of the revisions up to this edition is shown below. “Applied to:” indicates the chapters to which the revision was applied.

(1/2)

Edition	Description	Applied to:
2nd	<ul style="list-style-type: none"> <li>• Modification of <b>78K/IV Series Lineup</b></li> <li>• Modification of <b>Caution</b> and <b>Remark</b> in <b>1.4 Pin Configuration (Top View)</b></li> </ul>	<b>CHAPTER 1 GENERAL</b>
	<ul style="list-style-type: none"> <li>• Modification of description in <b>2.2.13 AV<sub>DD</sub></b></li> <li>• Modification of <b>Table 2-1 Types of Pin I/O Circuits and Recommended Connection of Unused Pins</b></li> </ul>	<b>CHAPTER 2 PIN FUNCTIONS</b>
	<ul style="list-style-type: none"> <li>• Addition of interrupt mask register 1L to <b>Table 3-6 Special Function Register (SFR) List</b></li> </ul>	<b>CHAPTER 3 CPU ARCHITECTURE</b>
	<ul style="list-style-type: none"> <li>• Correction of <b>Table 4-3 Port Mode Register and Output Latch Setting When Alternate Function Is Used</b></li> <li>• Modification of description in <b>4.5 Selecting Mask Option</b></li> </ul>	<b>CHAPTER 4 PORT FUNCTIONS</b>
	<ul style="list-style-type: none"> <li>• Modification of description in <b>(8) AV<sub>DD</sub> pin</b> in <b>11.2 Configuration of A/D Converter</b></li> <li>• Modification of <b>Figure 11-9 Timing of A/D Conversion End Interrupt Request Generation</b></li> <li>• <b>11.5 Notes on A/D Converter</b> Addition of <b>(10) Timing that makes A/D conversion result undefined</b> and <b>(11) Cautions on board design</b> and modification of description in <b>(12) Reading A/D conversion result register (ADCR)</b></li> </ul>	<b>CHAPTER 11 A/D CONVERTER</b>
	<ul style="list-style-type: none"> <li>• Modification of <b>Remark</b> in <b>Figure 12-4 Format of Serial Operation Mode Register 2</b></li> <li>• Addition of <b>Table 12-2 Serial Interface Operation Mode Settings</b></li> <li>• Modification of <b>Remark</b> in <b>12.4.2 3-wire serial I/O mode (b) Format of serial operation mode register 2</b></li> </ul>	<b>CHAPTER 12 SERIAL INTERFACE</b>
	<ul style="list-style-type: none"> <li>• Addition of <b>Table 13-3 Serial Interface Operation Mode Settings</b></li> </ul>	<b>CHAPTER 13 ASYNCHRONOUS SERIAL INTERFACE</b>
	<ul style="list-style-type: none"> <li>• Modification of <b>Table 16-3 Control Registers</b></li> <li>• Modification of description in <b>16.3.2 Interrupt mask registers (MK0, MK1L)</b></li> <li>• Modification of <b>Figure 16-2 Format of Interrupt Mask Registers (MK0, MK1L)</b></li> </ul>	<b>CHAPTER 16 INTERRUPT FUNCTION</b>
	<ul style="list-style-type: none"> <li>• Modification of <b>Figure 17-6 STOP Mode Release by INTP0 to INTP2 Input</b></li> <li>• Modification of description in <b>(4) A/D converter</b> in <b>17.6 Check Items When STOP Mode/IDLE Mode Is Used</b></li> </ul>	<b>CHAPTER 17 STANDBY FUNCTION</b>
	<ul style="list-style-type: none"> <li>• Modification of <b>Figure 18-1 Oscillation of Main System Clock in Reset Period</b></li> </ul>	<b>CHAPTER 18 RESET FUNCTION</b>
<ul style="list-style-type: none"> <li>• Addition of chapter</li> </ul>	<b>CHAPTER 21 ELECTRICAL SPECIFICATIONS</b>	
<ul style="list-style-type: none"> <li>• Addition of chapter</li> </ul>	<b>CHAPTER 22 PACKAGE DRAWINGS</b>	

Edition	Description	Applied to:
2nd	<ul style="list-style-type: none"> <li>• Addition of chapter</li> </ul>	<p><b>CHAPTER 23 RECOMMENDED SOLDERING CONDITIONS</b></p>
	<ul style="list-style-type: none"> <li>• Modification of <b>Figure A-1 Development Tool Configuration</b></li> <li>• Addition of SP78K4 to <b>A.1 Language Processing Software</b></li> <li>• Modification of <b>Remark</b></li> <li>• Modification of <b>A.3.1 Hardware</b></li> <li>• Modification of <b>Remark</b> in <b>A.3.2 Software</b></li> <li>• Addition of A.4 <b>Notes on Target System Design</b></li> </ul>	<p><b>APPENDIX A DEVELOPMENT TOOLS</b></p>

## Facsimile Message

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.

From:

Name

Company

Tel.

FAX

Address

*Thank you for your kind support.*

<b>North America</b> NEC Electronics Inc. Corporate Communications Dept. Fax: +1-800-729-9288 +1-408-588-6130	<b>Hong Kong, Philippines, Oceania</b> NEC Electronics Hong Kong Ltd. Fax: +852-2886-9022/9044	<b>Taiwan</b> NEC Electronics Taiwan Ltd. Fax: +886-2-2719-5951
<b>Europe</b> NEC Electronics (Europe) GmbH Market Communication Dept. Fax: +49-211-6503-274	<b>Korea</b> NEC Electronics Hong Kong Ltd. Seoul Branch Fax: +82-2-528-4411	<b>Asian Nations except Philippines</b> NEC Electronics Singapore Pte. Ltd. Fax: +65-250-3583
<b>South America</b> NEC do Brasil S.A. Fax: +55-11-6462-6829	<b>P.R. China</b> NEC Electronics Shanghai, Ltd. Fax: +86-21-6841-1137	<b>Japan</b> NEC Semiconductor Technical Hotline Fax: +81- 44-435-9608

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"/>