To our customers,

Old Company Name in Catalogs and Other Documents

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

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

April 1st, 2010
Renesas Electronics Corporation

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

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

Notice

1. All information included in this document is current as of the date this document is issued. Such information, however, is subject to change without any prior notice. Before purchasing or using any Renesas Electronics products listed herein, please confirm the latest product information with a Renesas Electronics sales office. Also, please pay regular and careful attention to additional and different information to be disclosed by Renesas Electronics such as that disclosed through our website.
2. Renesas Electronics does not assume any liability for infringement of patents, copyrights, or other intellectual property rights of third parties by or arising from the use of Renesas Electronics products or technical information described in this document. No license, express, implied or otherwise, is granted hereby under any patents, copyrights or other intellectual property rights of Renesas Electronics or others.
3. You should not alter, modify, copy, or otherwise misappropriate any Renesas Electronics product, whether in whole or in part.
4. Descriptions of circuits, software and other related information in this document are provided only to illustrate the operation of semiconductor products and application examples. You are fully responsible for the incorporation of these circuits, software, and information in the design of your equipment. Renesas Electronics assumes no responsibility for any losses incurred by you or third parties arising from the use of these circuits, software, or information.
5. When exporting the products or technology described in this document, you should comply with the applicable export control laws and regulations and follow the procedures required by such laws and regulations. You should not use Renesas Electronics products or the technology described in this document for any purpose relating to military applications or use by the military, including but not limited to the development of weapons of mass destruction. Renesas Electronics products and technology may not be used for or incorporated into any products or systems whose manufacture, use, or sale is prohibited under any applicable domestic or foreign laws or regulations.
6. Renesas Electronics has used reasonable care in preparing the information included in this document, but Renesas Electronics does not warrant that such information is error free. Renesas Electronics assumes no liability whatsoever for any damages incurred by you resulting from errors in or omissions from the information included herein.
7. Renesas Electronics products are classified according to the following three quality grades: “Standard”, “High Quality”, and “Specific”. The recommended applications for each Renesas Electronics product depends on the product’s quality grade, as indicated below. You must check the quality grade of each Renesas Electronics product before using it in a particular application. You may not use any Renesas Electronics product for any application categorized as “Specific” without the prior written consent of Renesas Electronics. Further, you may not use any Renesas Electronics product for any application for which it is not intended without the prior written consent of Renesas Electronics. Renesas Electronics shall not be in any way liable for any damages or losses incurred by you or third parties arising from the use of any Renesas Electronics product for an application categorized as “Specific” or for which the product is not intended where you have failed to obtain the prior written consent of Renesas Electronics. The quality grade of each Renesas Electronics product is “Standard” unless otherwise expressly specified in a Renesas Electronics data sheets or data books, etc.

   “Standard”: Computers; office equipment; communications equipment; test and measurement equipment; audio and visual equipment; home electronic appliances; machine tools; personal electronic equipment; and industrial robots.

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

   “Specific”: Aircraft; aerospace equipment; submersible repeaters; nuclear reactor control systems; medical equipment or systems for life support (e.g. artificial life support devices or systems), surgical implantations, or healthcare intervention (e.g. excision, etc.), and any other applications or purposes that pose a direct threat to human life.
8. You should use the Renesas Electronics products described in this document within the range specified by Renesas Electronics, especially with respect to the maximum rating, operating supply voltage range, movement power voltage range, heat radiation characteristics, installation and other product characteristics. Renesas Electronics shall have no liability for malfunctions or damages arising out of the use of Renesas Electronics products beyond such specified ranges.
9. Although Renesas Electronics endeavors to improve the quality and reliability of its products, semiconductor products have specific characteristics such as the occurrence of failure at a certain rate and malfunctions under certain use conditions. Further, Renesas Electronics products are not subject to radiation resistance design. Please be sure to implement safety measures to guard them against the possibility of physical injury, and injury or damage caused by fire in the event of the failure of a Renesas Electronics product, such as safety design for hardware and software including but not limited to redundancy, fire control and malfunction prevention, appropriate treatment for aging degradation or any other appropriate measures. Because the evaluation of microcomputer software alone is very difficult, please evaluate the safety of the final products or system manufactured by you.
10. Please contact a Renesas Electronics sales office for details as to environmental matters such as the environmental compatibility of each Renesas Electronics product. Please use Renesas Electronics products in compliance with all applicable laws and regulations that regulate the inclusion or use of controlled substances, including without limitation, the EU RoHS Directive. Renesas Electronics assumes no liability for damages or losses occurring as a result of your noncompliance with applicable laws and regulations.
11. This document may not be reproduced or duplicated, in any form, in whole or in part, without prior written consent of Renesas Electronics.
12. Please contact a Renesas Electronics sales office if you have any questions regarding the information contained in this document or Renesas Electronics products, or if you have any other inquiries.

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

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

RENESAS

User's Manual

μPD784054

16-Bit Single-Chip Microcontrollers

Hardware

μPD784054
μPD784054(A)
μPD784054(A1)
μPD784054(A2)

Document No. U11719EJ3V1UD00 (3rd edition)
Date Published August 2005 N CP(K)

Printed in Japan

[MEMO]

NOTES FOR CMOS DEVICES

① VOLTAGE APPLICATION WAVEFORM AT INPUT PIN

Waveform distortion due to input noise or a reflected wave may cause malfunction. If the input of the CMOS device stays in the area between V_{IL} (MAX) and V_{IH} (MIN) due to noise, etc., the device may malfunction. Take care to prevent chattering noise from entering the device when the input level is fixed, and also in the transition period when the input level passes through the area between V_{IL} (MAX) and V_{IH} (MIN).

② HANDLING OF UNUSED INPUT PINS

Unconnected CMOS device inputs can be cause of malfunction. If an input pin is unconnected, it is possible that an internal input level may be generated due to noise, etc., causing malfunction. CMOS devices behave differently than Bipolar or NMOS devices. Input levels of CMOS devices must be fixed high or low by using pull-up or pull-down circuitry. Each unused pin should be connected to V_{DD} or GND via a resistor if there is a possibility that it will be an output pin. All handling related to unused pins must be judged separately for each device and according to related specifications governing the device.

③ PRECAUTION AGAINST ESD

A 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 when it has occurred. Environmental control must be adequate. When it is dry, a humidifier should be used. It is recommended to avoid using insulators that easily build up 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 benches and floors should be grounded. The operator should be grounded using a wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions need to be taken for PW boards with mounted semiconductor devices.

④ STATUS BEFORE INITIALIZATION

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

⑤ POWER ON/OFF SEQUENCE

In the case of a device that uses different power supplies for the internal operation and external interface, as a rule, switch on the external power supply after switching on the internal power supply. When switching the power supply off, as a rule, switch off the external power supply and then the internal power supply. Use of the reverse power on/off sequences may result in the application of an overvoltage to the internal elements of the device, causing malfunction and degradation of internal elements due to the passage of an abnormal current.

The correct power on/off sequence must be judged separately for each device and according to related specifications governing the device.

⑥ INPUT OF SIGNAL DURING POWER OFF STATE

Do not input signals or an I/O pull-up power supply while the device is not powered. The current injection that results from input of such a signal or I/O pull-up power supply may cause malfunction and the abnormal current that passes in the device at this time may cause degradation of internal elements.

Input of signals during the power off state must be judged separately for each device and according to related specifications governing the device.

FIP and IEBus are trademarks of NEC Electronics 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 International Business Machines Corporation.
SPARCstation is a trademark of SPARC International, Inc.
Solaris and SunOS are trademarks of Sun Microsystems, Inc.
HP9000 series 700 and HP-UX are trademarks of Hewlett-Packard Company.
Ethernet is a trademark of Zerox Corporation.
TRON is an abbreviation of The Realtime Operating system Nucleus.
ITRON is an abbreviation of Industrial TRON.

These commodities, technology or software, must be exported in accordance with the export administration regulations of the exporting country. Diversion contrary to the law of that country is prohibited.

- **The information in this document is current as of August, 2005. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC Electronics data sheets or data books, etc., for the most up-to-date specifications of NEC Electronics products. Not all products and/or types are available in every country. Please check with an NEC Electronics 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 the prior written consent of NEC Electronics. NEC Electronics assumes no responsibility for any errors that may appear in this document.
- NEC Electronics does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from the use of NEC Electronics 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 Electronics 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 a customer's equipment shall be done under the full responsibility of the customer. NEC Electronics assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.
- While NEC Electronics endeavors to enhance the quality, reliability and safety of NEC Electronics 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 Electronics products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features.
- NEC Electronics products are classified into the following three quality grades: "Standard", "Special" and "Specific".

  The "Specific" quality grade applies only to NEC Electronics products developed based on a customer-designated "quality assurance program" for a specific application. The recommended applications of an NEC Electronics product depend on its quality grade, as indicated below. Customers must check the quality grade of each NEC Electronics 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 Electronics products is "Standard" unless otherwise expressly specified in NEC Electronics data sheets or data books, etc. If customers wish to use NEC Electronics products in applications not intended by NEC Electronics, they must contact an NEC Electronics sales representative in advance to determine NEC Electronics' willingness to support a given application.

(Note)

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

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

M8E 02. 11-1

Regional Information

Some information contained in this document may vary from country to country. Before using any NEC Electronics product in your application, please contact the NEC Electronics 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.

[GLOBAL SUPPORT]
http://www.necel.com/en/support/support.html

NEC Electronics America, Inc. (U.S.)
Santa Clara, California
Tel: 408-588-6000
800-366-9782

NEC Electronics (Europe) GmbH
Duesseldorf, Germany
Tel: 0211-65030

- **Sucursal en España**
  Madrid, Spain
  Tel: 091-504 27 87
- **Succursale Française**
  Vélizy-Villacoublay, France
  Tel: 01-30-67 58 00
- **Filiale Italiana**
  Milano, Italy
  Tel: 02-66 75 41
- **Branch The Netherlands**
  Eindhoven, The Netherlands
  Tel: 040-265 40 10
- **Tyskland Filial**
  Taeby, Sweden
  Tel: 08-63 87 200
- **United Kingdom Branch**
  Milton Keynes, UK
  Tel: 01908-691-133

NEC Electronics Hong Kong Ltd.
Hong Kong
Tel: 2886-9318

NEC Electronics Hong Kong Ltd.
Seoul Branch
Seoul, Korea
Tel: 02-558-3737

NEC Electronics Shanghai Ltd.
Shanghai, P.R. China
Tel: 021-5888-5400

NEC Electronics Taiwan Ltd.
Taipei, Taiwan
Tel: 02-2719-2377

NEC Electronics Singapore Pte. Ltd.
Novena Square, Singapore
Tel: 6253-8311

J05.6

Major Revisions in This Edition

Page	Contents
U11719EJ2V0UD00 → U11719EJ3V0UD00	
p. 29	**CHAPTER 1 GENERAL** • Completion of development of the following product μPD78F4046 • Update of **78K/IV Series Product Lineup**
p. 47	**CHAPTER 2 PIN FUNCTIONS** Addition of description on BWD pin in **Table 2-4 I/O Circuit Type of Each Pin and Recommended Processing of Unused Pins**
p. 268	**CHAPTER 12 ASYNCHRONOUS SERIAL INTERFACE/3-WIRE SERIAL I/O** Addition of cautions on start bit during UART transmission to **12.5 Cautions**
p. 365 p. 378	**CHAPTER 16 STANDBY FUNCTION** • Modification of **Figure 16-1 Diagram of Standby Mode Transition** • Modification of description in **16.6 (5) A/D converter**
p. 385 p. 387	**CHAPTER 18 μPD78F4046** • Addition of description on Flashpro III • Addition of cautions in **18.3 Cautions**
p. 423	Addition of **CHAPTER 20 ELECTRICAL SPECIFICATIONS (μPD784054)**
p. 429	Addition of **CHAPTER 21 ELECTRICAL SPECIFICATIONS (μPD784054(A))**
p. 435	Addition of **CHAPTER 22 ELECTRICAL SPECIFICATIONS (μPD784054(A1))**
p. 441	Addition of **CHAPTER 23 ELECTRICAL SPECIFICATIONS (μPD784054(A2))**
p. 447	Addition of **CHAPTER 24 TIMING CHARTS**
p. 452	Addition of **CHAPTER 25 PACKAGE DRAWING**
p. 453	Addition of **CHAPTER 26 RECOMMENDED SOLDERING CONDITIONS**
p. 455 pp. 458, 459 p. 459 pp. 460, 461 p. 462 p. 463	**APPENDIX A DEVELOPMENT TOOLS** • Addition of description on host machines and OSs • Addition of SP78K4 to **A.1 Language Processing Software,** modification of description in **Remark** • Addition of description on Flashpro III in **Remark** in **A.2 Flash Memory Writing Tools** • Addition and modification of description in **A.3.1 Hardware** • Modification of description in **Remark** in **A.3.2 Software** • Addition of **A.4 Cautions on Designing Target System**
p. 467	Modification of description in **APPENDIX B EMBEDDED SOFTWARE**
U11719EJ3V0UD00 → U11719EJ3V1UD00	
p. 30	Modification of **1.2 Ordering Information**
p. 30	Modification of **1.3 Quality Grades**
p. 453	Addition of **Table 26-1. Surface Mounting Type Soldering Conditions (2)**

The mark ★ shows major revised points.

INTRODUCTION

Intended Reader

This manual is intended for user engineers who understand the functions of the μPD784054 and wish to design application systems using this subseries.

The relevant products are as follows:

- Standard products: μPD784054
- Special products : μPD784054(A), (A1), (A2)

Purpose

The purpose of this manual is to give users an understanding of the various hardware functions of the μPD784054.

Organization

The μPD784054 manual is divided into two volumes – the hardware volume (this manual) and the instruction volume.

Hardware volume	Instruction volume
Pin functions	CPU functions
Internal block functions	Addressing
Interrupts	Instruction set
Other on-chip peripheral functions	
Electrical specifications	

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

How to Read This Manual

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

♦ To readers using this manual as a μPD784054(A), 784054(A1), 784054(A2) manual:

→ The μPD784054 is treated as the representative model. Therefore, when using this manual for the μPD784054(A), 784054(A1), 784054(A2) manual, μPD784054 should be read as each product name as appropriate. For the differences between products, refer to **1.8 Differences between μPD784054 and μPD784046 Subseries**, **1.9 Differences between μPD784054 and μPD784054(A)**, and **1.10 Differences between μPD784054(A), 784054(A1), and 784054(A2)**.

The application examples presented in this manual are for the "standard" quality models in general-purpose electronic systems. If you wish to use the applications presented in this manual for electronic systems that require "special" quality models, thoroughly study the parts and circuits to be actually used, and their quality grade.

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

→ Use **APPENDIX C REGISTER INDEX**.

♦ If the device operates strangely after debugging:

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

♦ For a general understanding of the functions

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

♦ For the details of the instruction functions:

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

♦ To find out about electrical specifications

→ Refer to the each chapter of electrical specifications.

♦ To find out about application examples of each function,

→ Refer to the **Application Note** separately available.

Legend

Significance in data notation	:	High-order digit on left, low-order digit on right
Active-low notation	:	$\overline{\times\times\times}$ (Line above pin or signal name)
Note	:	Explanation of item marked with **Note** in the text
Caution	:	Item to be especially noted
Remark	:	Supplementary information
Numeric notations	:	Binary ..... ××××B or××××
		Decimal ..... ××××
		Hexadecimal..... ××××H

Register Notation

	7	6	5	4	③	2	1	0
EDC	B	1	0	×	A	1	0	×

Where the bit number is marked with a circle, the bit name is reserved for NEC Electronics assembler and is defined as a sfr variable by the #pragma sfr directive for C compiler.

Write Operation	Read Operation
0 or 1 is written. The operation is not affected by either value.	0 or 1 is read.
0 must be written	0 is read.
1 must be written	1 is read.
A value is written according to the function to be used.	A value is read according to the operating status.

Code combinations marked "Setting prohibited" in the register notations in the text must not be written.

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

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

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

Documents Related to Devices

Document Name	Document No.
μPD784054 Subseries User's Manual - Hardware	This manual
78K/IV Series Application Note - Software Fundamentals	U10095E
78K/IV Series User's Manual - Instructions	U10905E

Documents Related to Development 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
SM78K Series Ver. 2.30 or Later System Simulator	Operation (Windows™ Based)	U15373E
	External Part User Open Interface Specification	U15802E
ID78K Series Integrated Debugger Ver. 2.30 or Later	Operation (Windows Based)	U15185E
RX78K4 Real-time OS	Fundamentals	U10603E
	Installation	U10604E
Project Manager Ver 3.12 or Later (Windows Based)		U14610E

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

Document Name	Document No.
IE-78K4-NS In-Circuit Emulator	U13356E
IE-784046-NS-EM1 Emulation Board	U13744E
IE-784000-R In-Circuit Emulator	U12903E
IE-784046-R-EM1 Emulation Board	U11677E

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 Flash Memory Writing (User's Manuals)

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

Other Related Documents

Document Name	Document No.
SEMICONDUCTOR SELECTION GUIDE - Products and Packages -	X13769X
Semiconductor Device Mount Manual	**Note**
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

Note See the "Semiconductor Device Mount Manual" website (http://www.necel.com/pkg/en/mount/index.html).

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

LIST OF FIGURES (1/6)

LIST OF FIGURES (2/6)

LIST OF FIGURES (3/6)

LIST OF FIGURES (4/6)

LIST OF FIGURES (5/6)

LIST OF FIGURES (6/6)

LIST OF TABLES (1/3)

LIST OF TABLES (2/3)

LIST OF TABLES (3/3)

CHAPTER 1 GENERAL

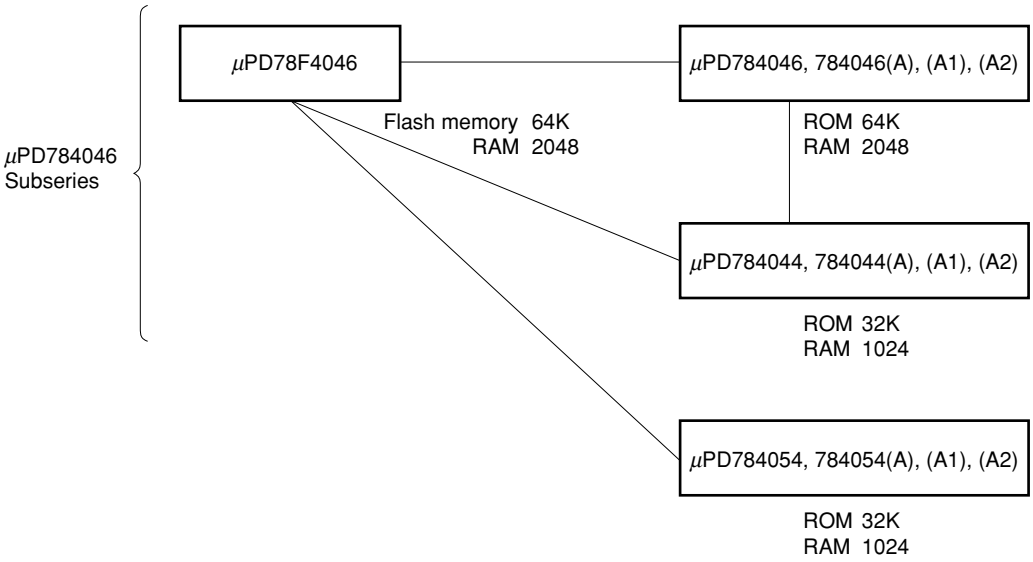
The μPD784054 is a product in the 78K/IV series and is provided with a 10-bit A/D converter. The 78K/IV series is a collection of 16-bit single-chip microcontrollers with a high-performance CPU that has a function to access a 1M-byte memory space.

The μPD784054 is based on the μPD784046 Subseries in the 78K/IV series but does not have the real-time output function and two timer/counter units of the μPD784046 Subseries. It is provided with a standby function invalid mode.

The μPD784054 has a 32K-byte mask ROM and a 1024-byte RAM. In addition, it also has a high-performance timer, 10-bit A/D converter, and two independent serial interface channels.

★ The μPD78F4046 is available as a flash memory model that can operate at the same supply voltage as the mask ROM model. The μPD78F4046 is a model of the μPD784046 Subseries and its functions are different from the μPD784054. For the differences, refer to **1.8 Differences between μPD784054 and μPD784046 Subseries**.

The μPD784054(A), 784054(A1), and 784054(A2) are the "special" quality revisions of the μPD784054.



These products can be used for the following applications.

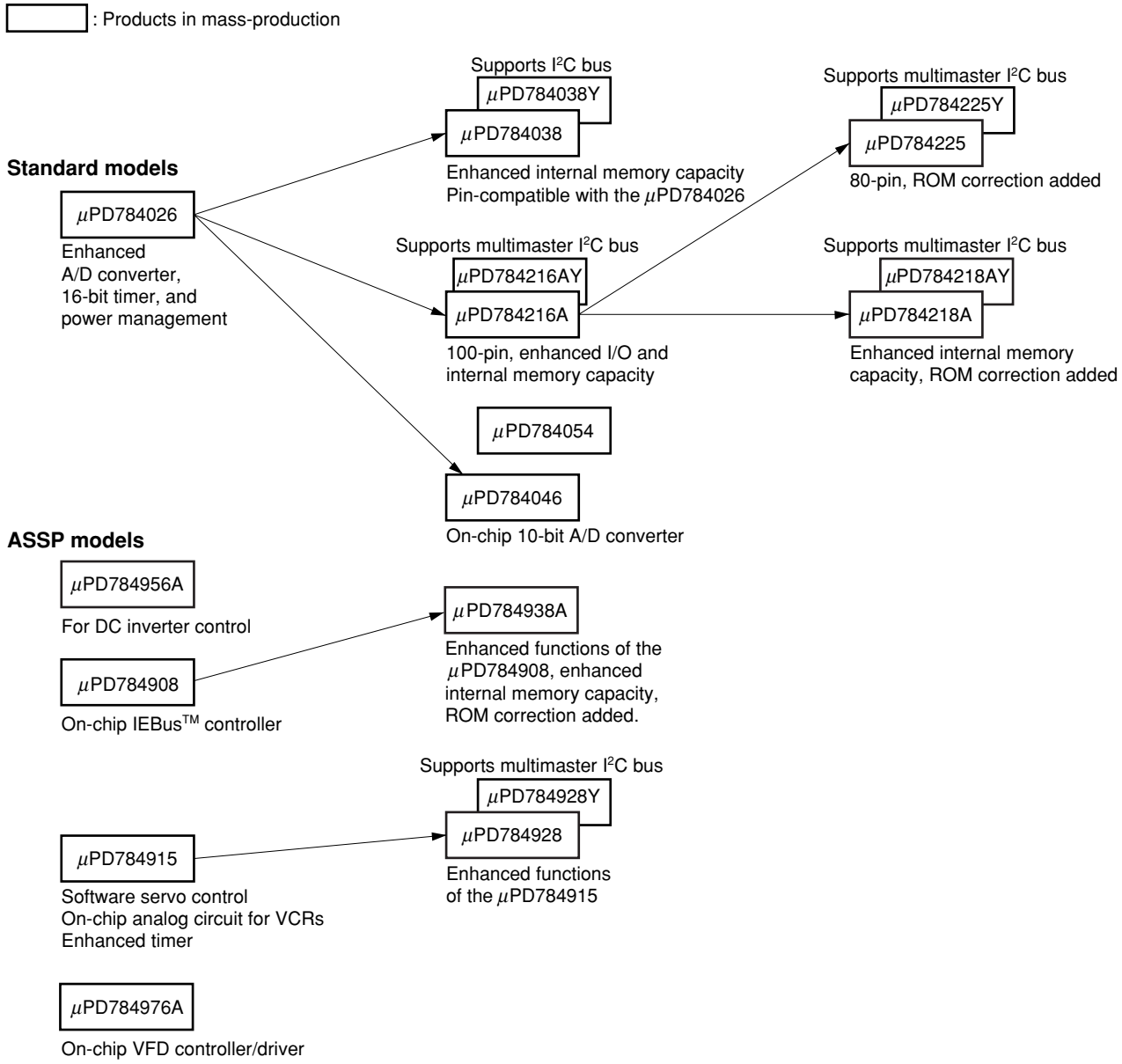
[Standard models]

- Office machines such as PPCs and printers
- Factory machines such as robots and automatic machine tools

[Special models]

- Control units of automotive appliances

★ **78K/IV Series Product Lineup**



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

1.1 Features

- 78K/IV series
- Minimum instruction execution time: 125 ns (at internal 16-MHz)
- Internal memory
  - ROM
    Mask ROM : 32K bytes
  - RAM : 1024 bytes
- I/O port: 64 pins
- Timer : 16-bit timer × 3 units
- Watchdog timer : 1 channel
- A/D converter : 10-bit resolution × 16 channels
- Serial interface
  UART/IOE (3-wire serial I/O): 2 channels (with baud rate generator)
- Interrupt controller (4 priority levels)
  Vectored interrupt/macro service/context switching
- Standby function
  HALT/STOP/IDLE/standby function invalid mode
- Supply voltage : V_{DD} = 4.5 to 5.5 V

★ 1.2 Ordering Information

Part Number	Package
μPD784054GC-×××-3B9	80-pin plastic QFP (14 x 14)
μPD784054GC(A)-×××-3B9	80-pin plastic QFP (14 x 14)
μPD784054GC(A1)-×××-3B9	80-pin plastic QFP (14 x 14)
μPD784054GC(A2)-×××-3B9	80-pin plastic QFP (14 x 14)
μPD784054GC-×××-3B9-A	80-pin plastic QFP (14 x 14)

Remarks 1. ××× indicates ROM code suffix.

2. Products that have the part numbers suffixed by “-A” are lead-free products.

★ 1.3 Quality Grades

Part Number	Package	Quality Grade
μPD784054GC-×××-3B9	80-pin plastic QFP (14 x 14)	Standard
μPD784054GC(A)-×××-3B9	80-pin plastic QFP (14 x 14)	Special
μPD784054GC(A1)-×××-3B9	80-pin plastic QFP (14 x 14)	Special
μPD784054GC(A2)-×××-3B9	80-pin plastic QFP (14 x 14)	Special
μPD784054GC-×××-3B9-A	80-pin plastic QFP (14 x 14)	Standard

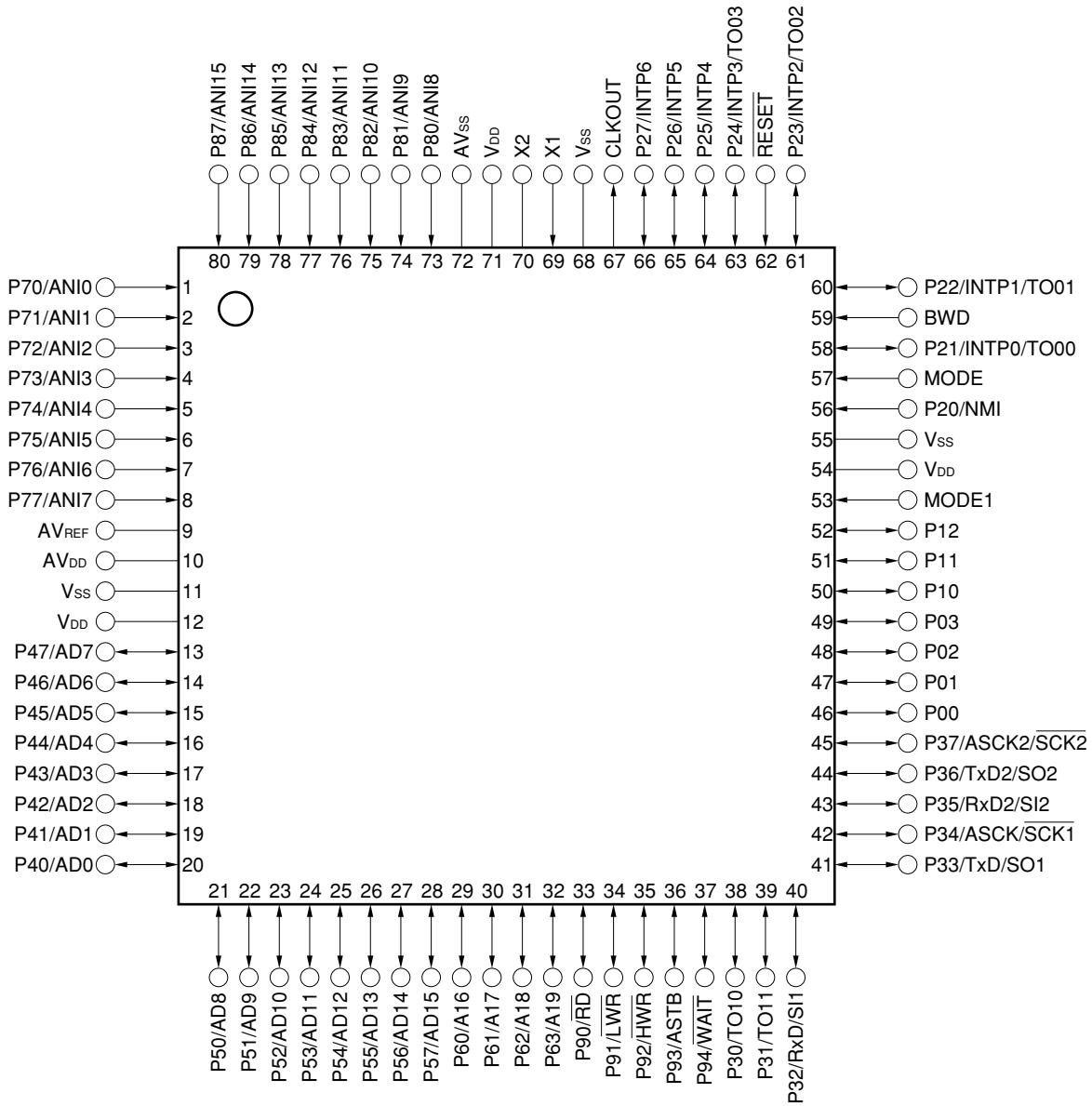
Remarks 1. ××× indicates ROM code suffix.

2. Products that have the part numbers suffixed by “-A” are lead-free products.

Please refer to "Quality Grades on NEC Semiconductor Devices" (Document No. C11531E) published by NEC Electronics Corporation to know the specification of quality grade on the devices and its recommended applications.

1.4 Pin Configuration (Top View)

- **80-pin plastic QFP (14 x 14)**

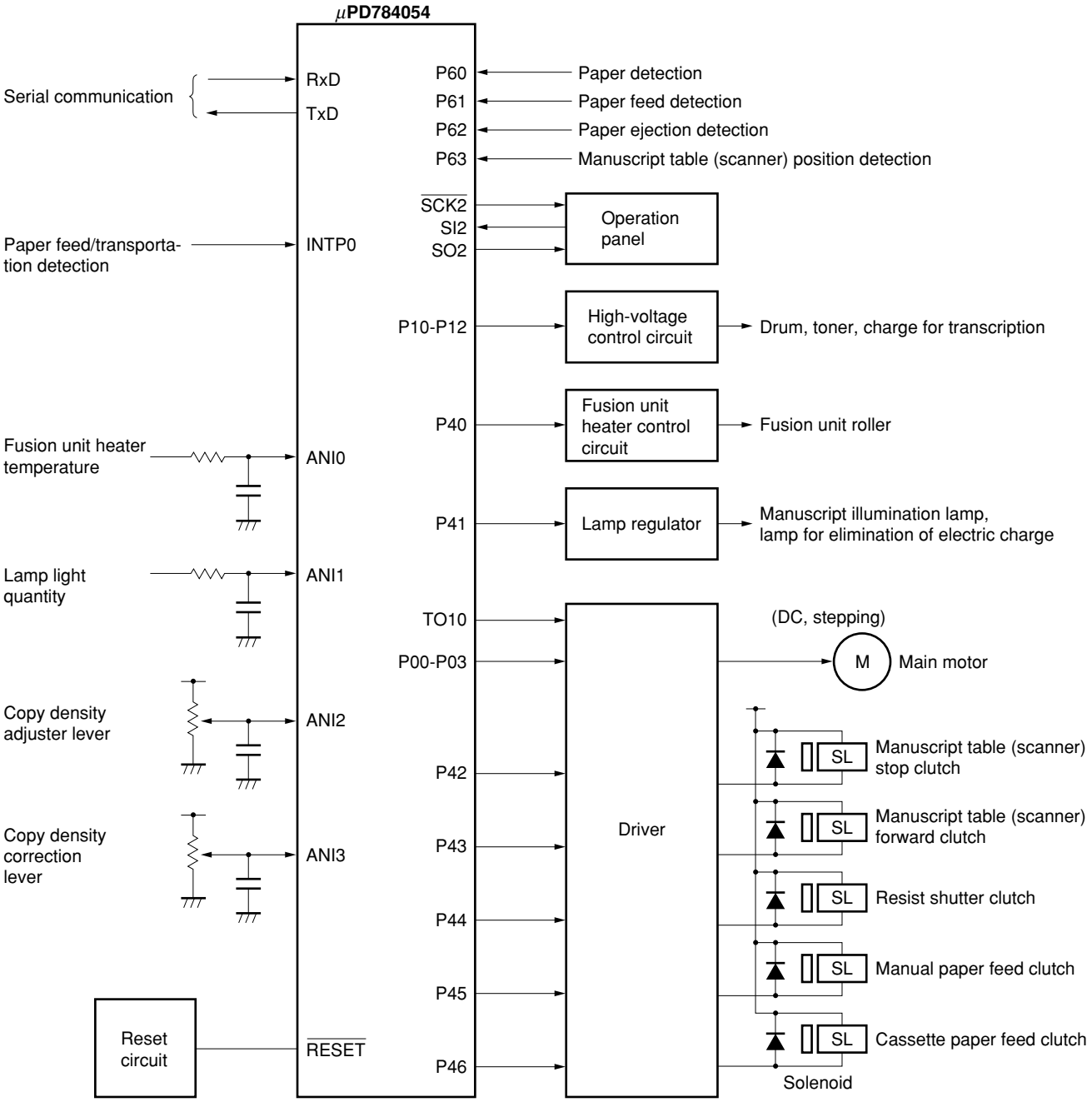


Cautions 1. Do not directly connect the MODE pin to V_{SS}.

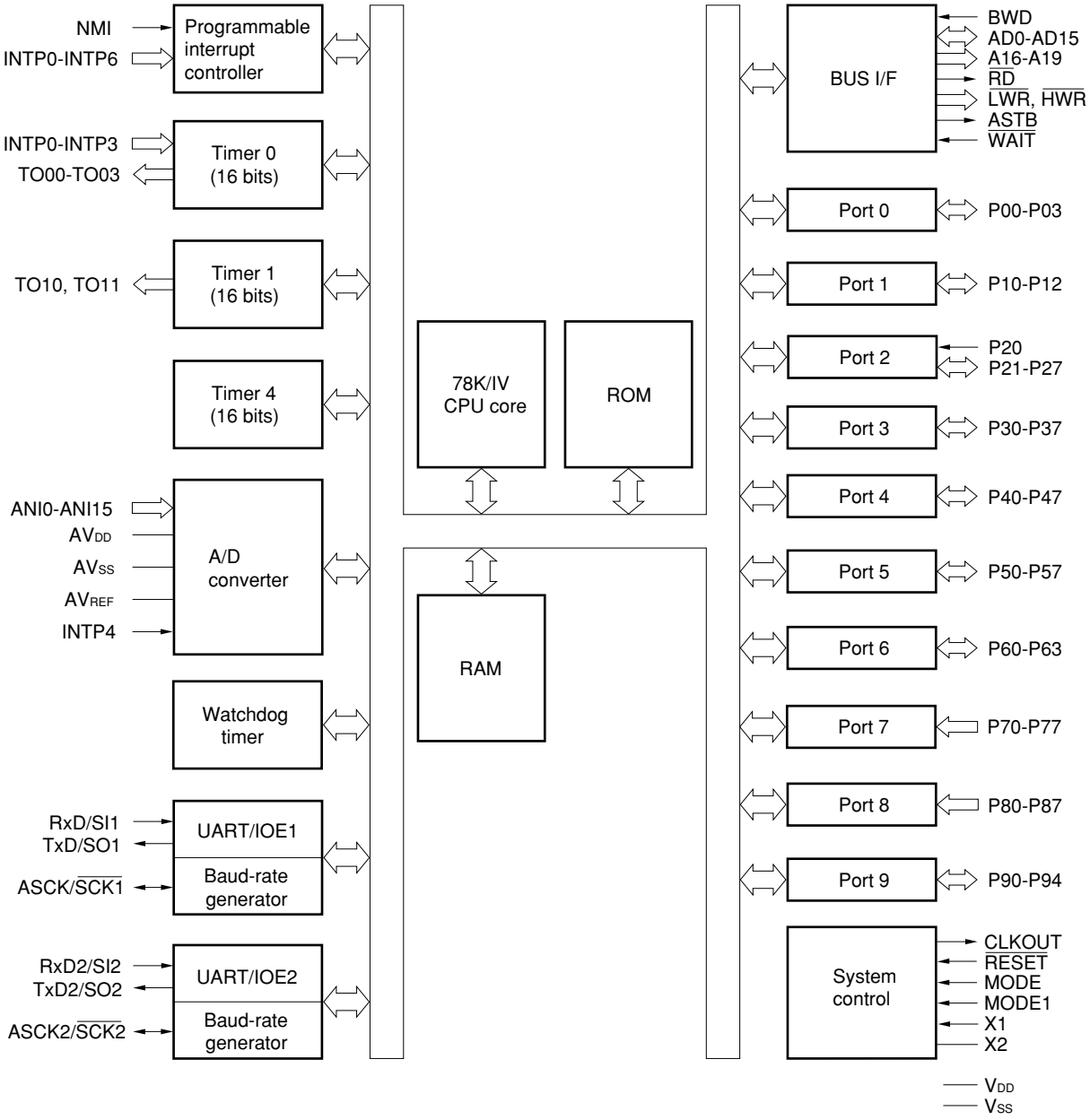
2. Normally, directly connect the MODE1 pin to V_{SS}.

A16-A19 : Address Bus
AD0-AD15 : Address/Data Bus
ANI0-ANI15 : Analog Input
ASCK, ASCK2 : Asynchronous Serial Clock
ASTB : Address Strobe
AV_{DD} : Analog Power Supply
AV_{REF} : Analog Reference Voltage
AV_{SS} : Analog Ground
BWD : Bus Width Definition
CLKOUT : Clock Out
$\overline{HWR}$: High Address Write Strobe
INTP0-INTP6 : Interrupt from Peripherals
$\overline{LWR}$: Low Address Write Strobe
MODE, MODE1 : Mode
NMI : Non-maskable Interrupt
P00-P03 : Port0
P10-P12 : Port1
P20-P27 : Port2
P30-P37 : Port3
P40-P47 : Port 4
P50-P57 : Port 5
P60-P63 : Port 6
P70-P77 : Port 7
P80-P87 : Port 8
P90-P94 : Port 9
$\overline{RD}$: Read Strobe
$\overline{RESET}$: Reset
RxD, RxD2 : Receive Data
$\overline{SCK1}$, $\overline{SCK2}$: Serial Clock
SI1, SI2 : Serial Input
SO1, SO2 : Serial Output
TO00-TO03, TO10, TO11: Timer Output
TxD, TxD2 : Transmit Data
V_{DD} : Power Supply
V_{SS} : Ground
$\overline{WAIT}$: Wait
X1, X2 : Crystal

1.5 System Configuration Example (PPC)



1.6 Block Diagram



1.7 List of Functions

Item		Function
Number of basic instructions (mnemonics)		113
General-purpose register		8 bits × 16 registers × 8 banks, or 16 bits × 8 registers × 8 banks (memory mapping)
Minimum instruction execution time		125 ns (at internal 16 MHz operation)
Internal memory	ROM	32 KB (mask ROM)
	RAM	1024 B
Memory space		1 MB with program and data memories combined
I/O port	Total	64 lines
	Input	17 lines
	I/O	47 lines
Pins with ancillary functions[Note]	Pin with pull-up resistor	29 pins
Timer		Timer 0 (16 bits): Timer register × 1, Capture/Compare register × 4; Pulse output: • Toggle output • Set/Reset output
		Timer 1 (16 bits): Timer register × 1, Compare register × 2; Pulse output: • Toggle output • Set/Reset output
		Timer 4 (16 bits): Timer register × 1, Compare register × 2
A/D converter		10-bit resolution × 16 channels
Serial interface		UART/IOE (3-wire serial I/O): 2 channels (with baud rate generator)
Watchdog timer		1 channel
Interrupt	Hardware source	23 (internal: 19, external: 8 (shared with internal: 4)
	Software source	BRK instruction, BRKCS instruction, operand error
	Non-maskable	Internal: 1, external : 1
	Maskable	Internal: 18, external: 7 (shared with internal: 4)
		• 4 priority levels • Three processing formats: vectored interrupt/macro service/context switching
Bus sizing		8 bits/16 bits external data bus width selectable
Standby		HALT/STOP/IDLE/standby function invalid mode
Supply voltage		V_{DD} = 4.5 to 5.5 V
Package		80-pin plastic QFP (14 x 14)

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

1.8 Differences between μPD784054 and μPD784046 Subseries

The differences between μPD784054 and μPD784046 Subseries are shown in Table 1-1.

Table 1-1. Differences between μPD784054 and μPD784046 Subseries

Item \ Part Number	μPD784054	μPD784046 Subseries		
		μPD784044	μPD784046	μPD78F4046
Internal ROM	32 KB (mask ROM)		64 KB (mask ROM)	64 KB (flash memory)
Internal RAM	1024 B		2048 B	
Port 1	P10-P12	P10-P13		
Real-time output port	Not provided	4 bits × 1		
Timer/counter	16 bits timer × 3 units	16 bits timer/counter × 2 units 16 bits timer × 3 units		
Standby function	HALT/STOP/IDLE/ standby function invalid mode	HALT/STOP/IDLE mode		
MODE1 pin	Provided	Not provided		
Function of pin 57	Mode			Mode/V_{PP}
Interrupt hardware source	23	27		

1.9 Differences between μPD784054 and μPD784054(A)

Table 1-2. Differences between μPD784054 and μPD784054(A)

Item \ Part Number	μPD784054	μPD784054(A)
Quality grade	Standard	Special
Operating ambient temperature (T_A)	−10 to +70°C	−40 to +85°C
Operating frequency	8 to 32 MHz	8 to 25 MHz
Minimum instruction execution time	125 ns (operates at 16 MHz internally)	160 ns (operates at 12.5 MHz internally)
DC characteristics	V_{DD} supply current differs.	
AC characteristics	Bus timing and serial operation differ.	
A/D converter characteristics	Conversion time and sampling time differ.	

1.10 Differences between μPD784054(A), 784054(A1), and 784054(A2)

Table 1-3. Differences between μPD784054(A), 784054(A1), and 784054(A2)

Item \ Part Number	μPD784054(A)	μPD784054(A1)	μPD784054(A2)
Operating ambient temperature (T_A)	−40 to +85°C	−40 to +110°C	−40 to +125°C
Operating frequency	8 to 25 MHz	8 to 20 MHz	
Minimum instruction execution time	160 ns (operates at 12.5 MHz internally)	200 ns (operates at 10 MHz internally)	
DC characteristics	Analog pin input leakage current, V_{DD} supply current, and data retention current differ.		
AC characteristics	Bus timing and serial operation differ.		
A/D converter characteristics	AV_{REF} current, A/D converter data retention current differ.		

CHAPTER 2 PIN FUNCTIONS

2.1 List of Pin Functions

(1) Port (1/2)

Pin Name	I/O	Dual-Function Pins	Function	
P00-P03	I/O	–	Port 0 (P0): • 4-bit I/O port • Can be set in input/output mode bit-wise. • Pins in input mode can all be connected to pull-up resistors at once via software by software settings.	
P10-P12	I/O	–	Port 1 (P1): • 3-bit I/O port • Can be set in input/output mode bit-wise.	
P20	Input	NMI	Port 2 (P2): • 8-bit I/O port	Input only
P21	I/O	INTP0/TO00		Can be set in input/output mode bit-wise.
P22		INTP1/TO01		
P23		INTP2/TO02		
P24		INTP3/TO03		
P25		INTP4		
P26		INTP5		
P27		INTP6		
P30	I/O	TO10	Port 3 (P3): • 8-bit I/O port • Can be set in input/output mode bit-wise.	
P31		TO11		
P32		RxD/SI1		
P33		TxD/SO1		
P34		ASCK/$\overline{\text{SCK1}}$		
P35		RxD2/SI2		
P36		TxD2/SO2		
P37		ASCK2/$\overline{\text{SCK2}}$		
P40-P47	I/O	AD0-AD7	Port 4 (P4): • 8-bit I/O port • Can be set in input/output mode bit-wise. • Pins in input mode can all be connected to pull-up resistors at once via software by software settings.	
P50-P57	I/O	AD8-AD15	Port 5 (P5): • 8-bit I/O port • Can be set in input/output mode bit-wise. • Pins in input mode can all be connected to pull-up resistors at once via software by software settings.	
P60-P63	I/O	A16-A19	Port 6 (P6): • 4-bit I/O port • Can be set in input/output mode bit-wise. • Pins in input mode can all be connected to pull-up resistors at once via software by software settings.	

(1) Port (2/2)

Pin Name	I/O	Dual-Function Pins	Function
P70-P77	Input	ANI0-ANI7	Port 7 (P7): • 8-bit input port
P80-P87	Input	ANI8-ANI15	Port 8 (P8): • 8-bit input port
P90	I/O	$\overline{RD}$	Port 9 (P9): • 5-bit I/O port • Can be set in input/output mode bit-wise. • Pins in input mode can all be connected to pull-up resistors at once via software by software settings.
P91		$\overline{LWR}$	
P92		$\overline{HWR}$	
P93		ASTB	
P94		$\overline{WAIT}$	

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

Pin Name	I/O	Dual-Function Pins	Function	
NMI	Input	P20	Non-maskable interrupt request input	
INTP0		P21/TO00	External interrupt request input	Capture trigger signal of CC00
INTP1		P22/TO01		Capture trigger signal of CC01
INTP2		P23/TO02		Capture trigger signal of CC02
INTP3		P24/TO03		Capture trigger signal of CC03
INTP4		P25		Conversion start trigger input of A/D converter
INTP5		P26		–
INTP6		P27		
TO00	Output	P21/INTP0	Timer output	
TO01		P22/INTP1		
TO02		P23/INTP2		
TO03		P24/INTP3		
TO10		P30		
TO11		P31		
RxD	Input	P32/SI1	Serial data input (UART0)	
RxD2		P35/SI2	Serial data input (UART2)	
TxD	Output	P33/SO1	Serial data output (UART0)	
TxD2		P36/SO2	Serial data output (UART2)	
ASCK	Input	P34/$\overline{SCK1}$	Baud rate clock input (UART0)	
ASCK2		P37/$\overline{SCK2}$	Baud rate clock input (UART2)	
SI1	Input	P32/RxD	Serial data input (3-wire serial I/O1)	
SI2		P35/RxD2	Serial data input (3-wire serial I/O2)	
SO1	Output	P33/TxD	Serial data output (3-wire serial I/O1)	
SO2		P36/TxD2	Serial data output (3-wire serial I/O2)	
$\overline{SCK1}$	I/O	P34/ASCK	Serial clock input/output (3-wire serial I/O1)	
$\overline{SCK2}$		P37/ASCK2	Serial clock input/output (3-wire serial I/O2)	
AD0-AD7	I/O	P40 to P47	Lower multiplexed address/data bus when external memory is connected	

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

Pin Name	I/O	Dual-Function Pins	Function
AD8-AD15[Note]	I/O	P50 to P57	• When 8-bit bus is specified Higher address bus when external memory is connected • When external 16-bit bus is specified Higher multiplexed address/data bus when external memory is connected
A16-A19[Note]	Output	P60 to P63	Higher address bus when external memory is connected
$\overline{RD}$	Output	P90	Read strobe to external memory
$\overline{LWR}$	Output	P91	• When external 8-bit bus is specified Write strobe to external memory • When external 16-bit bus is specified Write strobe to external memory located at lower position
$\overline{HWR}$		P92	Write strobe to external memory located at higher position when external 16-bit bus is specified
ASTB	Output	P93	Timing signal output to externally latch address information output from AD0 to AD15 pins to access external memory
$\overline{WAIT}$	Input	P94	Inserts wait.
BWD	Input	–	Sets bus width.
MODE	Input	–	Directly connect this pin to V_{SS} (this pin specifies test mode of IC).
MODE1	Input	–	Specifies standby function invalid mode. Connect this pin to V_{SS} when this mode is not used.
CLKOUT	Output	–	Clock output. Low level is output in the IDLE mode or STOP mode, otherwise f_{XX} (oscillation frequency) is always output.
X1	Input	–	Connect crystal for system clock oscillation (clock can also be input to X1).
X2	–	–	
$\overline{RESET}$	Input	–	Chip reset
ANI0-ANI7	Input	P70 to P77	Analog voltage input for A/D converter
ANI8-ANI15		P80 to P87	
AV_{REF}	–	–	Reference voltage for A/D converter
AV_{DD}		–	Positive power for A/D converter
AV_{SS}		–	GND for A/D converter
V_{DD}		–	Positive power
V_{SS}	–	–	GND

Note The number of pins used as address bus pins differs depending on the external address space (refer to **CHAPTER 15 LOCAL BUS INTERFACE FUNCTION**).

2.2 Description of Pin Functions

(1) P00 to P03 (Port 0) ... 3-state I/O

Port 0 is a 4-bit I/O port with an output latch. This port can be set in the input or output mode in 1-bit units by using port 0 mode register (PM0). Each pin of this port is provided with a software programmable pull-up resistor.

When $\overline{\text{RESET}}$ is input, this port is set in the input mode (output high-impedance status), and the contents of the output latch are undefined.

(2) P10 to P12 (Port 1) ... 3-state I/O

Port 1 is a 3-bit I/O port with an output latch. This port can be set in the input or output mode in 1-bit units by using port 1 mode register (PM1).

When $\overline{\text{RESET}}$ is input, this port is set in the input mode (output high-impedance status), and the contents of the output latch are undefined.

(3) P20 to P27 (Port 2) ... 3-state I/O

Port 2 is an 8-bit I/O port with an output latch. This port can be set in the input or output mode in 1-bit units by using port 2 mode register (PM2) (however, P20 is input-only).

In addition to the input/output port function, port 2 also functions as external interrupt signals, and to output the timer signal of timer 0 (refer to **Table 2-1**). P21 to P24 serve as the timer output pins of timer 0 if so specified by port 2 mode control register (PMC2). The level of each pin of this port can always be read or tested regardless of the multiplexed function. All the eight pins are Schmitt-trigger input pins to prevent malfunctioning due to noise.

When $\overline{\text{RESET}}$ is input, this port is set in the input mode (output high-impedance status), and the contents of the output latch are undefined.

Table 2-1. Operation Mode of Port 2

(n = 0 to 7)

Mode	Port Mode		Control Signal Output Mode
Set condition	PMC2n = 0		PMC2n = 1
	PM2n = 0	PM2n = 1	PM2n = ×
P20	–	Input port/NMI input[Note]	–
P21	Output port	Input port/INTP0 input	TO00 output
P22		Input port/INTP1 input	TO01 output
P23		Input port/INTP2 input	TO02 output
P24		Input port/INTP3 input	TO03 output
P25		Input port/INTP4 input	–
P26		Input port/INTP5 input	
P27		Input port/INTP6 input	

Note The NMI input pin accepts an interrupt request regardless of whether interrupts are enabled or disabled.

Remark ×: don't care

(a) Port mode

(i) Function as port pin

Each port pin set in the port mode by the port 2 mode control register (PMC2) can be set in the input or output mode in 1-bit units by the port 2 mode register (PM2) (however, P20 is fixed in input only).

(ii) Function as control signal input pins

If PMC2n (n = 0 to 7) bit of PMC2 is "0" and if PM2n (n = 0 to 7) bit of PM2 is "1", the pins of port 2 can be used as the following control signal input pins.

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

  This pin inputs an external non-maskable interrupt request. Whether the interrupt request is detected at the rising or falling edge can be specified by using external interrupt mode register 0 (INTM0).

- **INTP0 to INTP6 (Interrupt from Peripherals)**

  These pins input external interrupt requests. When the valid edge specified by external interrupt mode registers (INTM0 and INTM1) is detected on the INTP0 to INTP6 pins, an interrupt occurs (refer to **CHAPTER 13 EDGE DETECTION FUNCTION**).

  The INTP0 to INTP4 pins can also be used as external trigger input pins of each function, as follows:

- INTP0 ... Capture trigger input pin of capture/compare register 00 (CC00) of timer 0
- INTP1 ... Capture trigger input pin of capture/compare register 01 (CC01) of timer 0
- INTP2 ... Capture trigger input pin of capture/compare register 02 (CC02) of timer 0
- INTP3 ... Capture trigger input pin of capture/compare register 03 (CC03) of timer 0
- INTP4 ... External trigger input pin of A/D converter

(b) Control signal output mode

The P21 to P24 pins can be used as the timer output pins (TO00 to TO03) of timer 0 in 1-bit units if so specified by the port 2 mode control register (PMC2).

(4) P30 to P37 (Port 3) ... 3-state I/O

Port 3 is an 8-bit I/O port with an output latch. This port can be set in the input or output mode in 1-bit units by using port 3 mode register (PM3).

In addition to the input/output port function, port 3 also has a function to input or output control signals. The operation mode of each pin can be specified by using port 3 mode control register (PMC3), as shown in Table 2-2. The level of each pin of this port can always be read or tested regardless of the multiplexed function.

When $\overline{\text{RESET}}$ is input, this port is set in the input mode (output high-impedance status), and the contents of the output latch are undefined.

Table 2-2. Operation Mode of Port 3

(n = 0 to 7)

Mode	Port Mode	Control Signal I/O Mode
Setting condition	PMC3n = 0	PMC3n = 1
P30	I/O port	TO10 output
P31		TO11 output
P32		RxD/SI1 input
P33		TxD/SO1 output
P34		ASCK input/$\overline{\text{SCK1}}$ I/O
P35		RxD2/SI2 input
P36		TxD2/SO2 output
P37		ASCK2 input/$\overline{\text{SCK2}}$ I/O

(a) Port mode

Each port pin set in the port mode by the port 3 mode control register (PMC3) can be set in the input or output mode by the port 3 mode register (PM3).

(b) Control signal I/O mode

Each pin of port 3 can be set in the control mode in 1-bit units by using the port 3 mode control register (PMC3).

(i) TO10, TO11 (Timer Output)

These are timer output pins of timer 1.

(ii) RxD, RxD2 (Receive Data)

These are serial data input pins of the asynchronous serial interface.

(iii) TxD, TxD2 (Transmit data)

These are serial data output pins of the asynchronous serial interface.

(iv) SI1, SI2 (Serial Input)

These are serial data input pins of the 3-wire serial I/O.

(v) SO1, SO2 (Serial Output)

These are serial data output pins of the 3-wire serial I/O.

(vi) ASCK, ASCK2 (Asynchronous Serial Clock)

These are external baud rate clock input pins.

(vii) $\overline{\text{SCK1}}$, $\overline{\text{SCK2}}$ (Serial Clock)

These are serial clock I/O pins of the 3-wire serial I/O.

(5) P40 to P47 (Port 4) ... 3-state I/O

Port 4 is an 8-bit I/O port with an output latch. This port can be set in the input or output mode in 1-bit units by using port 4 mode register (PM4). Each pin is provided with a software programmable pull-up resistor.

Port 4 functions as the low-order multiplexed address/data bus (AD0 to AD7) if so specified by memory expansion mode register (MM) when an external memory or I/O is connected, in addition to the I/O port function.

When $\overline{\text{RESET}}$ is input, this port is set in the input mode (output high-impedance status), and the contents of the output latch are undefined.

(6) P50 to P57 (Port 5) ... 3-state I/O

Port 5 is an 8-bit I/O port with an output latch. This port can be set in the input or output mode in 1-bit units by using port 5 mode register (PM5). Each pin is provided with a software programmable pull-up resistor.

This port functions as follows if so specified by memory expansion mode register (MM) when an external memory or I/O is connected:

- When external 8-bit bus is specified
  As the high-order address bus (AD8 to AD15)
- When external 16-bit bus is specified
  As the high-order multiplexed address/data bus (AD8 to AD15).

When $\overline{\text{RESET}}$ is input, this port is set in the input mode (output high-impedance status), and the contents of the output latch are undefined.

(7) P60 to P63 (Port 6) ... 3-state I/O

Port 6 is a 4-bit I/O port with an output latch. This port can be set in the input or output mode in 1-bit units by using port 6 mode register (PM6). Each pin is provided with a software programmable pull-up resistor.

In addition to as an I/O port, this port also functions as the high-order address bus (A16 to A19) if so specified by the memory expansion mode register when an external memory or I/O is connected.

When $\overline{\text{RESET}}$ is input, this port is set in the input mode (output high-impedance status), and the contents of the output latch are undefined.

(8) P70 to P77 (Port 7) ... Input

Port 7 is an 8-bit input port. In addition to as input port pins, its pins also function as an A/D converter analog input (low-order 8 channels) pins (ANI0 to ANI7), and can always input analog signals. This port is set in the analog input mode by using A/D converter mode register (ADM).

The level of each pin of this port can always be read or tested, regardless of the multiplexed function.

(9) P80 to P87 (Port 8) ... Input

Port 8 is an 8-bit input port. In addition to functioning as input port pins, its pins also functions as an A/D converter analog input (high-order 8 channels) pins (ANI8 to ANI15), and can always input analog signals. This port is set in the analog input mode by using A/D converter mode register (ADM).

The level of each pin of this port can always be read or tested, regardless of the multiplexed function.

(10) P90 to P94 (Port 9) ... 3-state I/O

Port 9 is a 5-bit I/O port with an output latch. This port can be set in the input or output mode in 1-bit units by using port 9 mode register (PM9). Each pin is provided with a software programmable pull-up resistor.

In addition to the I/O port function, port 9 also functions as control signal pins (refer to **Table 2-3**). P90 to P93 function as read/write strobe signals and an address strobe signal if so specified by the memory extension mode register (MM) when an external memory or I/O is connected. P94 functions as a wait signal input pin if so specified by port 9 mode control register (PMC9).

When $\overline{\text{RESET}}$ is input, this port is set in the input mode (output high-impedance status), and the contents of the output latch are undefined.

Table 2-3. Operation Mode of Port 9

Pin Name	Port Mode	Control Signal I/O Mode	Manipulation to Use Port 9 as Control Pins
P90	I/O Port	$\overline{\text{RD}}$	Specifying external memory expansion mode by MM0 to MM3 bits of MM
P91		$\overline{\text{LWR}}$	
P92		$\overline{\text{HWR}}$	
P93		ASTB	
P94		$\overline{\text{WAIT}}$	Setting of PMC94 bit of PMC9 to 1

Remark For details, refer to **CHAPTER 15 LOCAL BUS INTERFACE FUNCTION**.

(a) Port mode

Each port pin not set in the control mode can be set in the input or output mode by using the port 9 mode register (PM9).

(b) Control signal I/O mode

(i) $\overline{RD}$ (Read Strobe)

This pin outputs a strobe signal to read an external memory. The operation of this pin is specified by the memory extension mode register (MM).

(ii) $\overline{LWR}$, $\overline{HWR}$ (Low/High Write Strobe)

These pins output strobe signals to write an external memory. The operations of these pins are specified by the memory extension mode register (MM).

(iii) ASTB (Address Strobe)

This is a timing signal output pin to latch the address information output from the AD0 to AD15 pins to access the external memory. The operation of this pin is specified by the memory extension mode register (MM).

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

This pin inputs a wait signal. The operation of this pin is specified by the port 9 mode control register (PMC9).

(11) BWD (Bus Width Definition) ... Input

This pin specifies the width of the bus. Depending on the setting of this pin, the value of the bus width specification register (BW) at reset differs as follows:

BWD	External Bus Width	Value of BW at Reset
0	8 bits	0000H
1	16 bits	00FFH

(12) MODE (Mode) ... Input

This pin is used by NEC Electronics for testing IC. Be sure to directly connect this pin to V_{SS}.

(13) MODE1 (Mode) ... Input

This pin specifies the standby function invalid mode.

Connect this pin to V_{SS} when this mode is not used.

(14) CLKOUT (Clock Output) ... Output

Clock output. Low level is output in the IDLE mode or STOP mode, otherwise f_{XX} (oscillation frequency) is always output.

(15) X1, X2 (Crystal)

These pins are used to connect a crystal for internal clock oscillation. To supply an external clock, input the clock to the X1 pin. For the processing of the X2 pin at this time, refer to **CHAPTER 4 CLOCK GENERATOR**.

(16) $\overline{RESET}$ (Reset) ... Input

Active-low reset input

(17) AV_{REF} (Analog Reference Voltage)

This pin inputs a reference voltage to the A/D converter.

(18) AV_{DD} (Analog Power Supply)

This is the power supply pin of the A/D converter. Keep the potential at this pin same as that of the V_{DD} pin.

(19) AV_{SS} (Analog Ground)

This is the GND pin of the A/D converter. Keep the potential at this pin same as that of the V_{SS} pin.

(20) V_{DD} (Power Supply)

This is a positive power supply. Connect all the V_{DD} pins to a positive power supply.

(21) V_{SS} (Ground)

This is a GND pin. Ground all the V_{SS} pins.

2.3 I/O Circuits of Pins and Processing of Unused Pins

Table 2-4 shows the I/O circuit type of each pin and recommended processing of the unused pins.
For the I/O circuit type, refer to **Figure 2-1**.

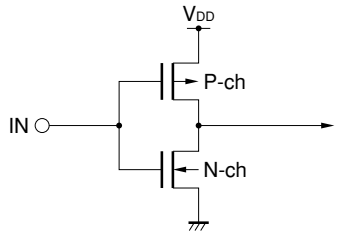
Table 2-4. I/O Circuit Type of Each Pin and Recommended Processing of Unused Pins

Pin Name	I/O Circuit Type	I/O	Recommended Connection of Unused Pins
P00-P03	5-A	I/O	Input: Individually connect to V_{DD} or V_{SS} via resistor.
P10-P12	5		Output: Leave unconnected.
P20/NMI	2	Input	Connect to V_{SS}.
P21/INTP0/TO00	8	I/O	Input: Individually connect to V_{DD} or V_{SS} via resistor.
P22/INTP1/TO01			Output: Leave unconnected.
P23/INTP2/TO02			
P24/INTP3/TO03			
P25/INTP4			
P26/INTP5			
P27/INTP6			
P30/TO10	5		
P31/TO11			
P32/RxD/SI1			
P33/TxD/SO1			
P34/ASCK/$\overline{SCK1}$	8		
P35/RxD2/SI2	5		
P36/TxD2/SO2			
P37/ASCK2/$\overline{SCK2}$	8		
P40/AD0-P47/AD7	5-A		
P50/AD8-P57/AD15			
P60/A16-P63/A19			
P70/ANI0-P77/ANI7	9	Input	Connect to V_{SS}.
P80/ANI8-P87/ANI15			
P90/$\overline{RD}$	5-A	I/O	Input: Individually connect to V_{DD} or V_{SS} via resistor.
P91/$\overline{LWR}$			Output: Leave unconnected.
P92/$\overline{HWR}$			
P93/ASTB			
P94/$\overline{WAIT}$			
★ BWD	1	Input	Connect to V_{DD} or V_{SS}.
MODE, MODE1			Directly connect to V_{SS}.
$\overline{RESET}$	2		–
CLKOUT	3	Output	Leave unconnected.
AV_{REF}	–	–	Connect to V_{SS}.
AV_{SS}			
AV_{DD}			Connect to V_{DD}.

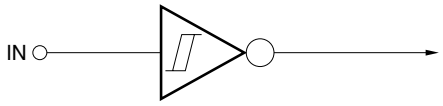
Remark The circuit type numbers are serial in the 78K series but are not always so with some models (because some models are not provided with particular circuits).

Figure 2-1. I/O Circuits of Pins

Type 1

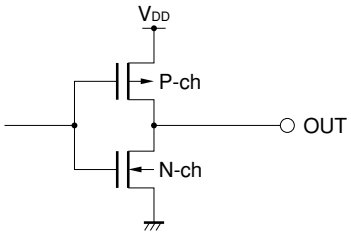


Type 2

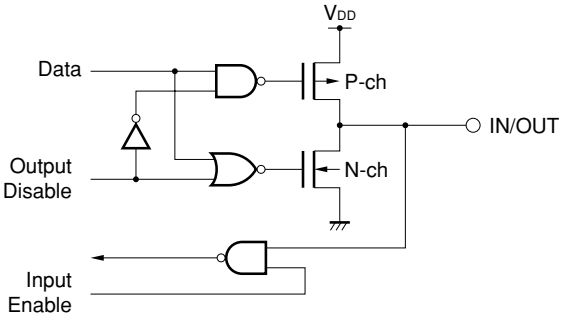


Schmitt-trigger input with hysteresis characteristics

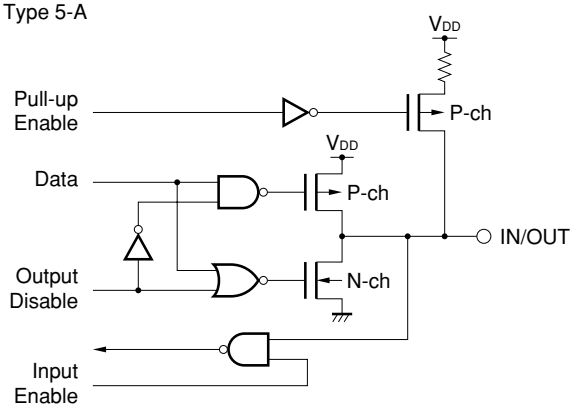
Type 3



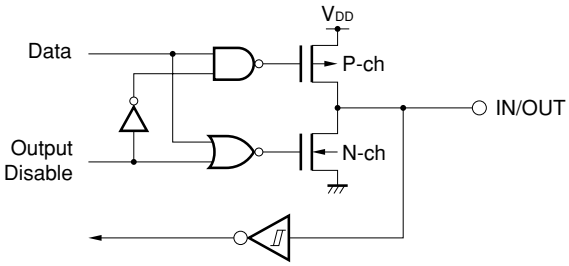
Type 5



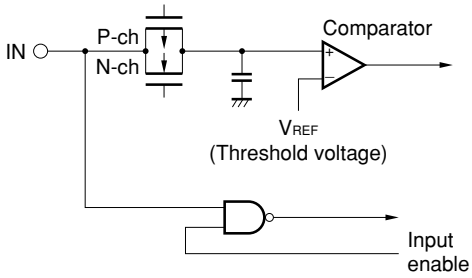
Type 5-A



Type 8



Type 9



CHAPTER 3 CPU ARCHITECTURE

3.1 Memory Space

The μPD784054 can access a 1 M-byte memory space. The mapping of the internal data area (special function registers and internal RAM) depends on the LOCATION instruction. A LOCATION instruction must be executed after reset release, and can only be used once.

The program after reset release must be as follows:

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

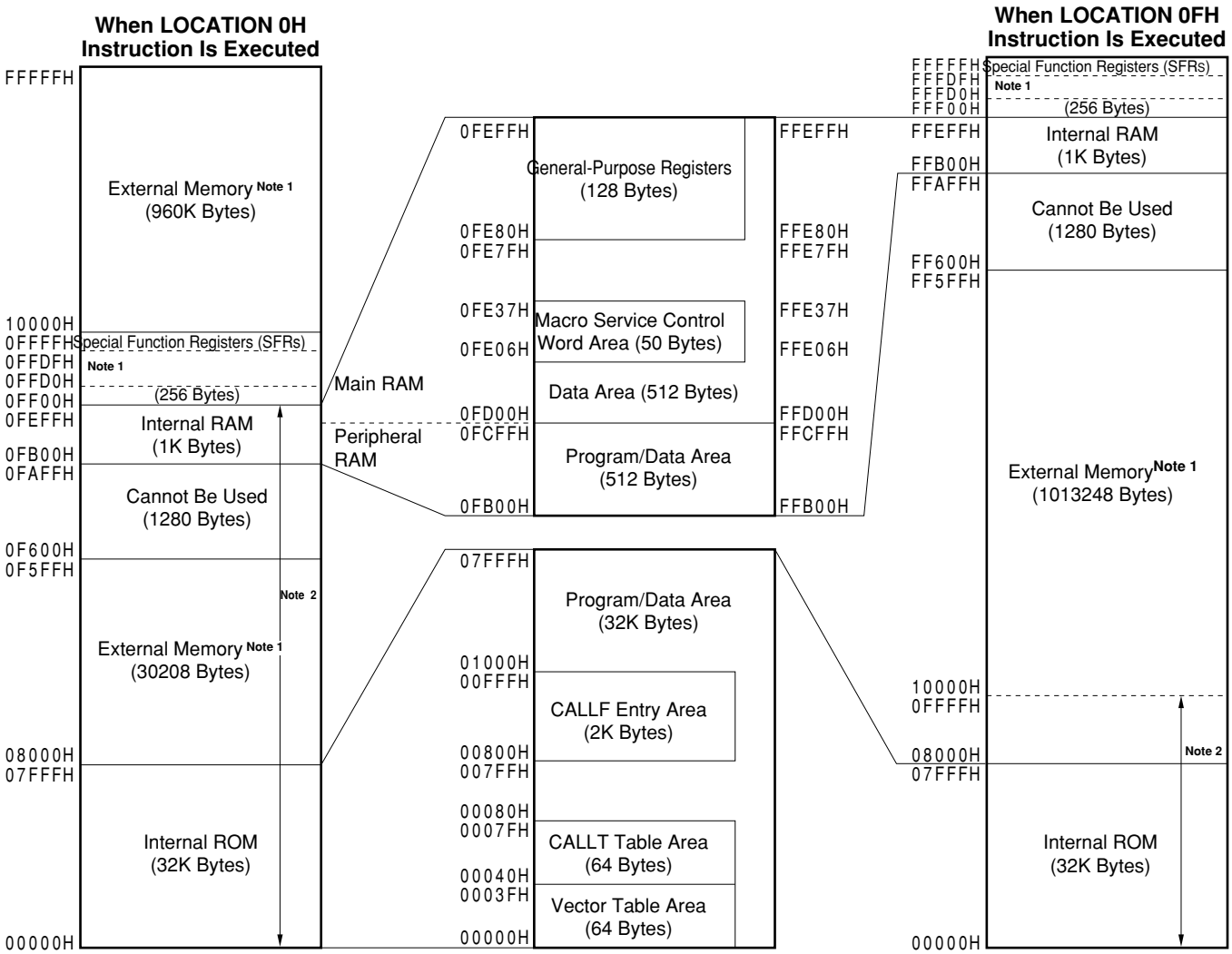
(1) When LOCATION 0H instruction is executed

The internal data area is mapped onto addresses 0FB00H to 0FFFFH.
Internal ROM is mapped onto addresses 0 to 07FFFH.
External memory is accessed in external memory extension mode.

(2) When LOCATION 0FH instruction is executed

The internal data area is mapped onto addresses FFB00H to FFFFFH.
Internal ROM is mapped onto addresses 0 to 07FFFH.
External memory is accessed in external memory extension mode.

Figure 3-1. Memory Map



Notes 1. Accessed in the external memory extension mode.

2. Base area or entry area by reset or interrupt. The internal RAM is not reset.

3.2 Internal ROM Area

The μPD784054 incorporates ROM which is used to store programs, table data, etc.

Table 3-1. Internal ROM Area

Product Name	Internal ROM	Address Space	
		Location 0H Instruction	Location 0FH Instruction
μPD784054	32 K × 8 bits	00000H-07FFFH	00000H-07FFFH

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

When the instruction execution cycle equal to an external ROM fetch is selected, wait insertion is performed by the wait function, but when high-speed fetches are used, wait insertion is not performed for internal ROM.

$\overline{\text{RESET}}$ input sets the instruction execution cycle equal to the external ROM fetch cycle.

3.3 Base Area

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

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

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

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

3.3.1 Vector table area

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

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

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

Table 3-2. Vector Table

Vector Table Address	Interrupt Cause
0003CH	Operand error
0003EH	BRK
00000H	Reset ($\overline{\text{RESET}}$ input)
00002H	NMI
00004H	INTWDT
00006H	INTOV0
00008H	INTOV1
0000AH	INTOV4
0000CH	INTP0/INTCC00
0000EH	INTP1/INTCC01
00010H	INTP2/INTCC02
00012H	INTP3/INTCC03
00014H	INTP4
00016H	INTP5
00018H	INTP6
0001AH	INTCM10
0001CH	INTCM11
00026H	INTCM40
00028H	INTCM41
0002AH	INTSER
0002CH	INTSR/INTCSI1
0002EH	INTST
00030H	INTSER2
00032H	INTSR2/INTCSI2
00034H	INTST2
00036H	INTAD

3.3.2 CALLT instruction table area

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

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

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

3.3.3 CALLF instruction entry area

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

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

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

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

3.4 Internal Data Area

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

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

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

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

3.4.1 Internal RAM area

The μPD784054 incorporates general-purpose static RAM.

This area is configured as follows:

- Internal RAM area
  - Peripheral RAM (PRAM)
  - Internal high-speed RAM (IRAM)

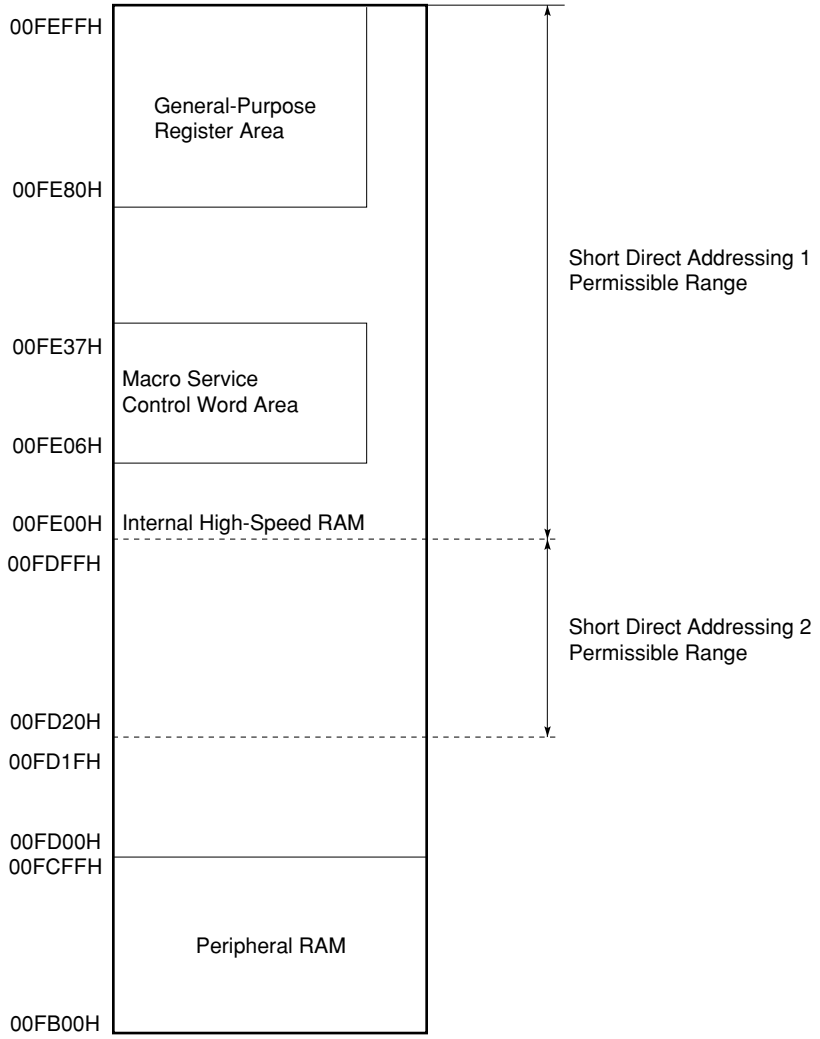
Table 3-3. Internal RAM Area

Internal RAM / Product Name	Internal RAM Area		
		Peripheral RAM: PRAM	Internal High-Speed RAM: IRAM
μPD784054	1024 bytes (0FB00H-0FEFFH)	512 bytes (0FB00H-0FCFFH)	512 bytes (0FD00H-0FEFFH)

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

The internal RAM memory map is shown in Figure 3-2.

Figure 3-2. Internal RAM Memory Map



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

(1) Internal high-speed RAM (IRAM)

The internal high-speed RAM (IRAM) allows high-speed accesses to be made. The short direct addressing mode for high-speed accesses can be used on FD20H to FEFFH in this area. There are two kinds of short direct addressing mode, short direct addressing 1 and short direct addressing 2, according to the target address. The function is the same in both of these addressing modes. With some instructions, the word length is shorter with short direct addressing 2 than with short direct addressing 1. Refer to the **78K/IV Series User's Manual - Instruction** for details.
A program fetch cannot be performed from IRAM. If a program fetch is performed from an address onto which IRAM is mapped, CPU inadvertent loop will result.
The following areas are reserved in IRAM.

- General-purpose register area : FE80H to FEFFH
- Macro service control word area : FE06H to FE37H
- Macro service channel area : FE00H to FEFFH (the address is specified by the macro service control word)

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

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

(2) Peripheral RAM (PRAM)

The peripheral RAM (PRAM) is used as ordinary program memory or data memory. When used as program memory, the program must be written to the peripheral RAM beforehand by a program.
Program fetches from peripheral RAM are fast, with a 2-byte fetch being executed in 2 clocks.

3.4.2 Special function register (SFR) area

The on-chip peripheral hardware special function registers (SFRs) are mapped onto the area from 0FF00H to 0FFFFH (refer to **Figure 3-1**).

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

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

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

3.4.3 External SFR area

In μPD784054, the 16-byte area from 0FFD0H to 0FFDFH in the SFR area (when the LOCATION 0H is executed; 0FFFD0H to 0FFFDFH when the LOCATION 0FH instruction is executed) is mapped as an external SFR area. When the external memory extension mode is externally connected peripheral I/Os, etc., can be accessed using the address bus or address/data bus, etc.

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

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

3.5 External Memory Space

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

3.6 Control Registers

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

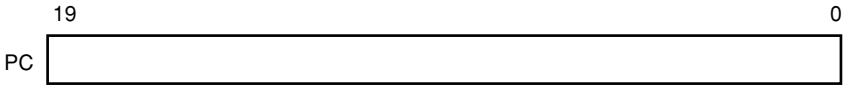
3.6.1 Program counter (PC)

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

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

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

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



3.6.2 Program status word (PSW)

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

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

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

$\overline{\text{RESET}}$ input resets (0) all bits.

"0" must always be written to the bits written as "0" in Figure 3-4. The contents of bits written as "-" are undefined when read.

Figure 3-4. Format of Program Status Word (PSW)

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

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

The flags are described below.

(1) Carry flag (CY)

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

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

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

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

The P/V flag performs the following two kinds of operation associated with execution of an operation instruction. The status of the P/V flag can be tested with a conditional branch instruction.

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

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

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

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

```
    78H (+120)    =      0111  1000
+)  69H (+105)    = +)   0110  1001
                     0   1110  0001  =  –31   P/V = 1
                     ↑
                     CY
```

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

```
    FBH (–5)      =      1111  1011
+)  F0H (–16)     = +)   1111  0000
                     1   1110  1011  =  –21   P/V = 0
                     ↑
                     CY
```

(3) Interrupt request enable flag (IE)

This flag controls CPU interrupt request acknowledgment operations.
When "0", interrupts are disabled, and only non-maskable interrupts and unmasked macro service can be acknowledged. All other interrupts are disabled.
When "1", the interrupt enabled state is set, and enabling of interrupt request acknowledgment is controlled by the interrupt mask flags corresponding to the individual interrupt requests and the priority of the individual interrupts.
The IE flag is set (1) by execution of an EI instruction, and reset (0) by execution of a DI instruction or acknowledgment of an interrupt.

(4) Auxiliary carry flag (AC)

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

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

(5) Register set selection flag (RSS)

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

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

(6) Zero flag (Z)

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

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

(7) Sign flag (S)

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

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

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

This is a 3-bit flag used to select one of the 8 register banks (register bank 0 to register bank 7) (refer to **Table 3-4**).

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

Table 3-4. Register Bank Selection

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

(9) User flag (UF)

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

3.6.3 Use of RSS bit

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

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

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

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

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

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

(1) RSS bit recommendations

- Registers used by instructions for which the A, X, B, C and AX registers are directly entered in the operand column of the operation list (refer to **19.2.**)

- Registers specified as implied by instructions that use the A, AX, B and C registers by means of implied addressing

- Registers used in addressing by instructions that use the A, B and C registers in indexed addressing and based indexed addressing

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

- When RSS = 0
  A→R1, X→R0, B→R3, C→R2, AX→RP0, BC→RP1

- When RSS = 1
  A→R5, X→R4, B→R7, C→R6, AX→RP2, BC→RP3

Registers used other than those mentioned above are always the same irrespective of the value of the RSS bit. With the NEC Electronics assembler (RA78K4), the register operation code generated when the A, X, B, C, AX and BC registers are described by those names is determined by the assembler RSS pseudo-instruction.
When the RSS bit is set or reset, an RSS pseudo-instruction must be written immediately before (or immediately after) the relevant instruction (refer to **example** below).

<Program example>

- When RSS is set to 0

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

- When RSS is set to 1

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

(2) Operation code generation method with RA78K4

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

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

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

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

- If R0 to R7 or RP0 to RP4 is written as r, r', rp or rp' in the operand column, an operation code in accordance with that specification is output (an operation code for which A or AX is directly entered in the operand column is not output.)

- R1, R3, R2 or R5, R7, R6 cannot be used for registers A, B and C used in indexed addressing and based indexed addressing.

(3) Operating precautions

Switching the RSS bit has the same effect as having two register sets. However, when writing a program, care must be taken to ensure that the static program code and dynamic RSS bit changes at the time of program execution always coincide.

Also, a program that sets RSS to 1 cannot be used by a program that uses the context switching function, and therefore program usability is poor. Moreover, since different registers are used with the same name, program readability is poor and debugging is difficult. Therefore, if it is necessary to set RSS to 1, these disadvantages must be fully taken into consideration when writing a program.

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

3.6.4 Stack pointer (SP)

The stack pointer is a 24-bit register that holds the start address of the stack area (LIFO type: 00000H to FFFFFFH) (refer to **Figure 3-5**). It is used to address the stack area when subroutine processing or interrupt processing is performed. Be sure to write “0” in the high-order 4 bits.

The contents of the SP are decremented before a write to the stack area and incremented after a read from the stack area (refer to **Figures 3-6** and **3-7**).

The SP is accessed by dedicated instructions.

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

Example SP initialization

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

Figure 3-5. Format of Stack Pointer (SP)

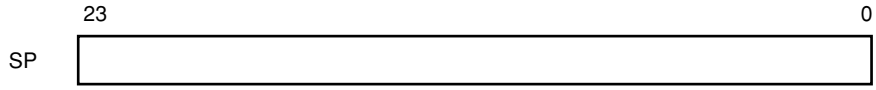


Figure 3-6. Data Saved to Stack Area

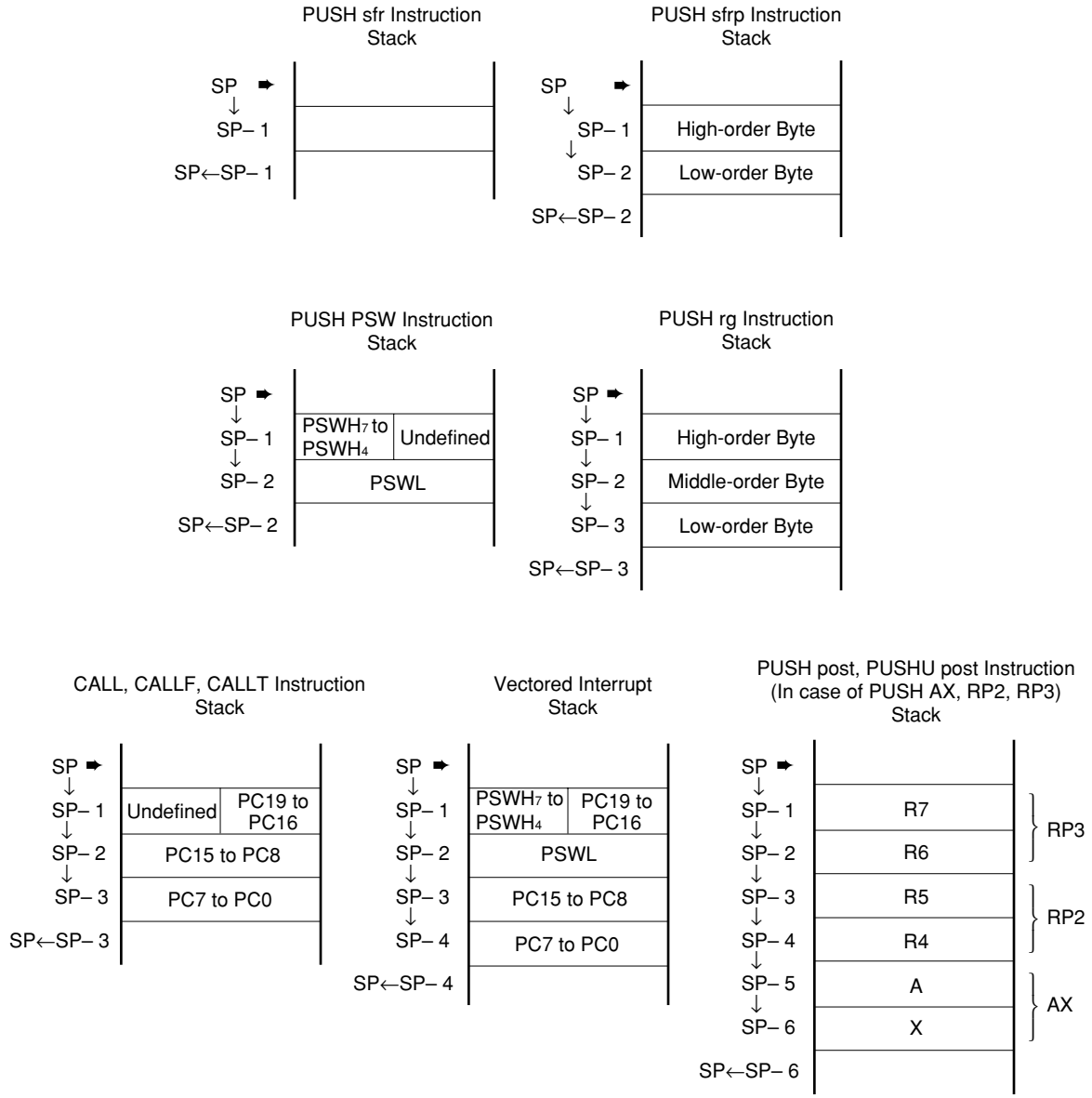
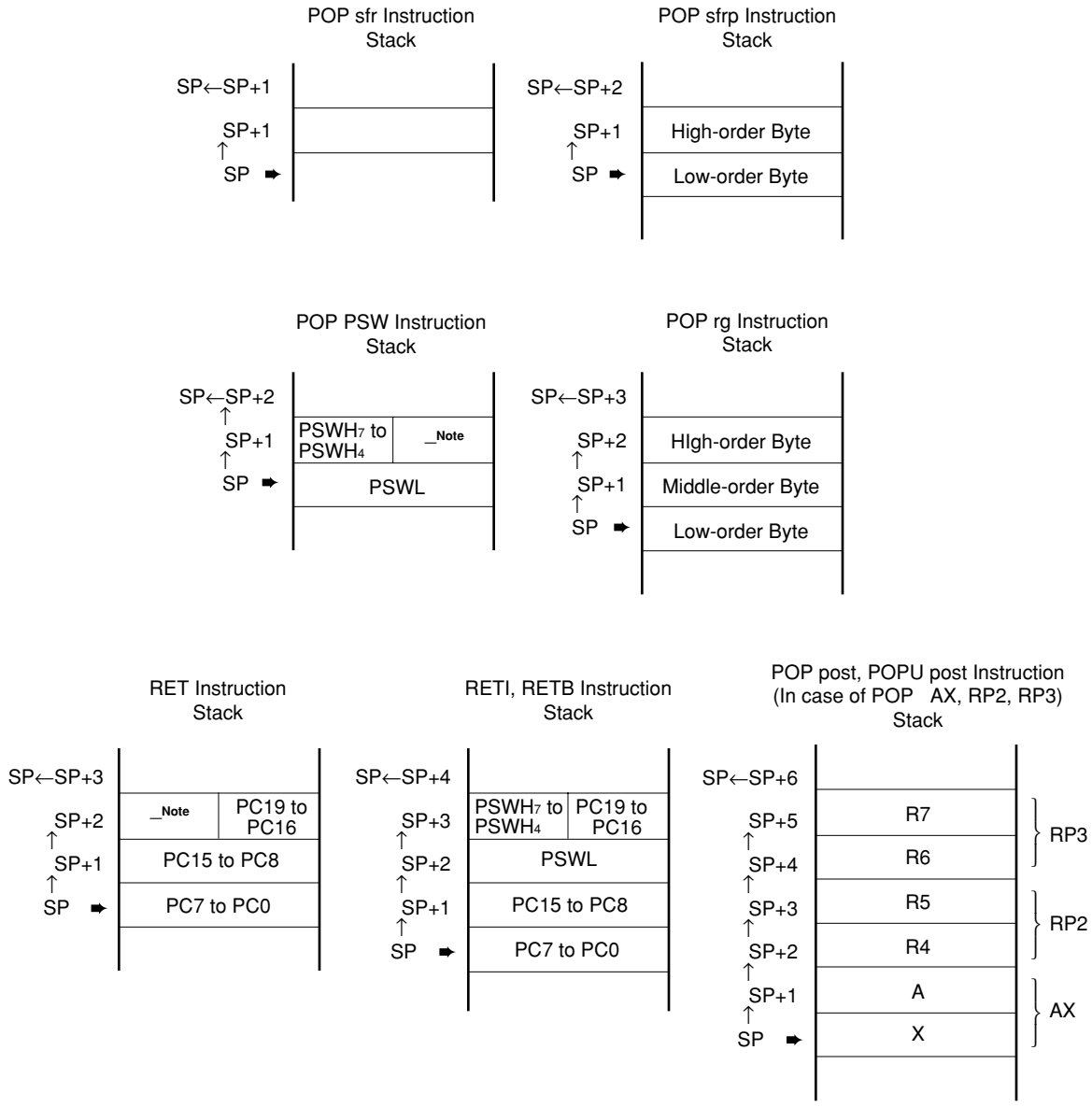


Figure 3-7. Data Restored from Stack Area



Note This 4-bit data is ignored.

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

2. The stack pointer (SP) is undefined after $\overline{\text{RESET}}$ input. Moreover, non-maskable interrupts can still be acknowledged when the SP is in an undefined state. An unanticipated operation may therefore be performed if a non-maskable interrupt request is generated when the SP is in the undefined state directly after reset release. To avoid this risk, the program after reset release must be written as follows.

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

3.7 General-Purpose Registers

3.7.1 Configuration

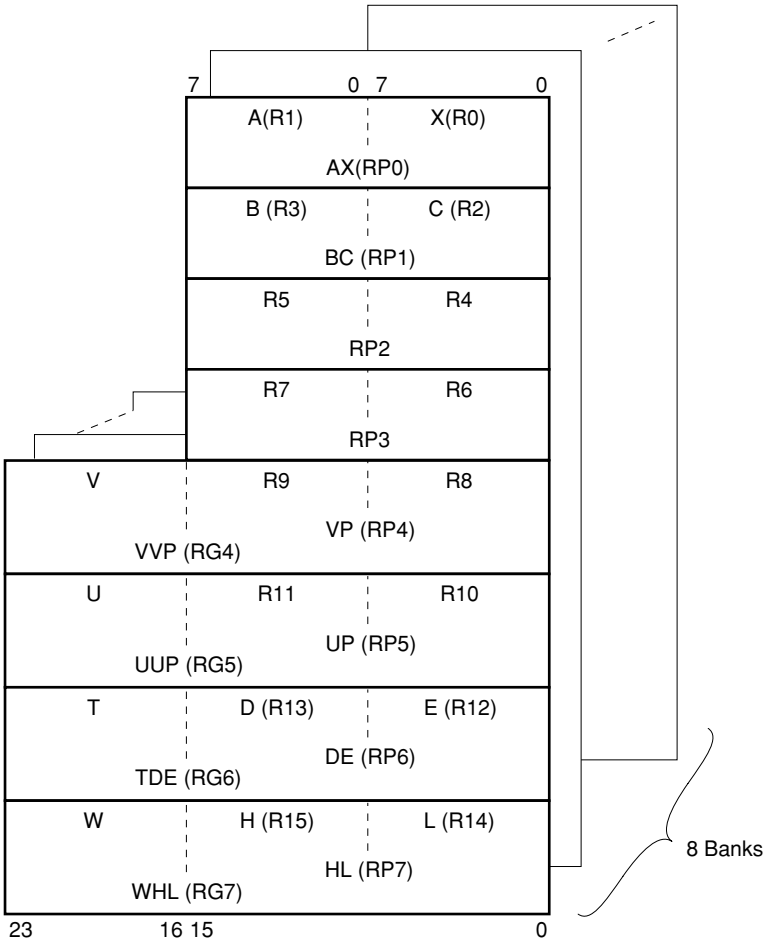
There are sixteen 8-bit general-purpose registers, and two 8-bit general-purpose registers can be used together as a 16-bit general-purpose register. In addition, four of the 16-bit general-purpose registers can be combined with an 8-bit register for address extension, and used as 24-bit address specification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

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

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

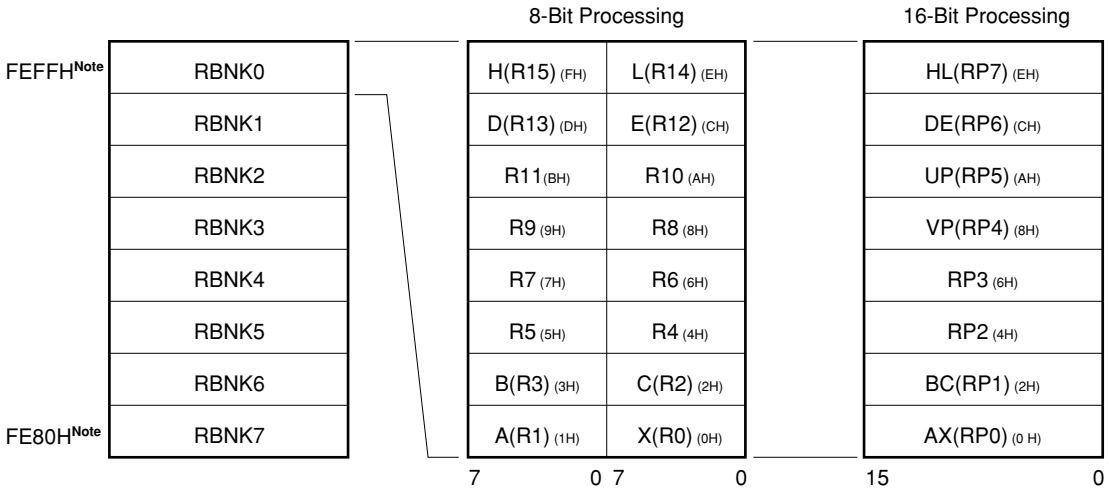
Upon $\overline{\text{RESET}}$ input, register bank 0 is selected. The register bank used during program execution can be checked by reading the register bank selection flag (RBS0, RBS1, RBS2) in the PSW.

Figure 3-8. Format of General-Purpose Register



Remark Absolute names are shown in parentheses.

Figure 3-9. General-Purpose Register Addresses



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

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

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

Example When register bank 2 is specified

```
      .
      .
      .
PUSH PSW
SEL RB2
      .
      .
      .                    Operations in register bank 2
POP PSW
      .
      .
      .                    Operations in original register bank
```

3.7.2 Functions

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

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

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

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

The registers have the following specific functions.

A (R1):

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

X (R0):

- Can be used for bit data storage.

AX (RP0):

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

AXDE:

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

B (R3):

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

C (R2):

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

RP2:

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

RP3:

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

VVP (RG4):

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

UUP (RG5):

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

DE (RP6), HL (RP7):

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

TDE (RG6):

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

WHL (RG7):

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

In addition to the function name that emphasizes the specific function of the register (X, A, C, B, E, D, L, H, AX, BC, VP, UP, DE, HL, VVP, UUP, TDE, WHL), each register can also be written by its absolute name (R0 to R15, RP0 to RP7, RG4 to RG7). The correspondence between these names is shown in Table 3-5.

Table 3-5. Correspondence between Function Names and Absolute Names

(a) 8-bit registers

Absolute Name	Function Name	
	RSS = 0	RSS = 1 Note
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 Note
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 RSS should only be set to 1 when a 78K/III series program is used.

Remark R8 to R11 have no function name.

3.8 Special Function Registers (SFRs)

These are registers to which a special function is assigned, such as on-chip peripheral hardware mode registers, control registers, etc. They are mapped onto the 256-byte space from 0FF00H to 0FFFFH[Note].

Note When the LOCATION 0H instruction is executed. When the LOCATION 0FH instruction is executed, the area is FFF00H to FFFFFH.

Caution Addresses onto which SFRs are not assigned should not be accessed in this area. If such an address is as accessed by mistake, the μPD784054 may become deadlocked. A deadlock can only be released by reset input.

A list of special function registers (SFRs) is given in Table 3-6. The meaning of the items in the table is as explained below.

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

Table 3-6. Special Function Registers (SFRs) List (1/4)

Address[Note 1]	Special Function Register (SFR) Name	Symbol	R/W	Bit Units for Manipulation			On Reset
				1 bit	8 bits	16 bits	
0FF00H	Port 0	P0	R/W	○	○	–	Undefined
0FF01H	Port 1	P1		○	○	–	
0FF02H	Port 2	P2	Note 2	○	○	–	
0FF03H	Port 3	P3	R/W	○	○	–	
0FF04H	Port 4	P4		○	○	–	
0FF05H	Port 5	P5		○	○	–	
0FF06H	Port 6	P6		○	○	–	
0FF07H	Port 7	P7	R	○	○	–	
0FF08H	Port 8	P8		○	○	–	
0FF09H	Port 9	P9	R/W	○	○	–	
0FF10H	Timer register 0	TM0	R	–	–	○	0000H
0FF11H							
0FF12H	Capture/compare register 00	CC00	R/W	–	–	○	Undefined
0FF13H							
0FF14H	Capture/compare register 01	CC01		–	–	○	
0FF15H							
0FF16H	Capture/compare register 02	CC02		–	–	○	
0FF17H							
0FF18H	Capture/compare register 03	CC03		–	–	○	
0FF19H							
0FF1AH	Timer register 1	TM1	R	–	–	○	0000H
0FF1BH							
0FF1CH	Compare register 10	CM10	R/W	–	–	○	Undefined
0FF1DH							
0FF1EH	Compare register 11	CM11		–	–	○	
0FF1FH							
0FF20H	Port 0 mode register	PM0		○	○	–	FFH
0FF21H	Port 1 mode register	PM1		○	○	–	
0FF22H	Port 2 mode register	PM2[Note 3]		○	○	–	
0FF23H	Port 3 mode register	PM3		○	○	–	
0FF24H	Port 4 mode register	PM4		○	○	–	
0FF25H	Port 5 mode register	PM5		○	○	–	
0FF26H	Port 6 mode register	PM6		○	○	–	
0FF29H	Port 9 mode register	PM9		○	○	–	
0FF2FH	Port read control register	PRDC		○	○	–	00H
0FF30H	Timer unit mode register 0	TUM0		○	○	–	
0FF31H	Timer mode control register	TMC		○	○	–	

Notes 1. When the LOCATION 0H instruction is executed. Add “F0000H” to this value when the LOCATION 0FH instruction is executed.

2. Bit 0 of P2 can only be read. Bits 1 to 7 can be read/written.

3. Bit 0 of PM2 is fixed to “1” by hardware.

Table 3-6. Special Function Registers (SFRs) List (2/4)

Address[Note 1]	Special Function Register (SFR) Name	Symbol	R/W	Bit Units for Manipulation			On Reset
				1 bit	8 bits	16 bits	
0FF32H	Timer output control register 0	TOC0	R/W	○	○	–	00H
0FF33H	Timer output control register 1	TOC1		○	○	–	
0FF37H	Timer mode control register 4	TMC4		○	○	–	
0FF38H	Prescaler mode register	PRM		–	○	–	
0FF3AH	Prescaler mode register 4	PRM4		–	○	–	
0FF3BH	Noise protection control register	NPC		○	○	–	
0FF3CH	External interrupt mode register 0	INTM0		○	○	–	
0FF3DH	External interrupt mode register 1	INTM1		○	○	–	
0FF3EH	Interrupt valid edge flag register 1	IEF1		○	○	–	Undefined
0FF3FH	Interrupt valid edge flag register 2	IEF2		○	○	–	
0FF42H	Port 2 mode control register	PMC2[Note 2]		○	○	–	00H
0FF43H	Port 3 mode control register	PMC3		○	○	–	
0FF49H	Port 9 mode control register	PMC9		○	○	–	
0FF4EH	Pull-up resistor option register L	PUOL		○	○	–	
0FF4FH	Pull-up resistor option register H	PUOH		○	○	–	
0FF60H	Timer register 4	TM4	R	–	–	○	0000H
0FF61H							
0FF62H	Compare register 40	CM40	R/W	–	–	○	Undefined
0FF63H							
0FF64H	Compare register 41	CM41		–	–	○	
0FF65H							
0FF6EH	A/D converter mode register	ADM		○	○	–	00H
0FF70H	A/D conversion result register 0	ADCR0	R	–	–	○	Undefined
0FF71H							
0FF71H	A/D conversion result register 0H	ADCR0H		–	○	–	
0FF72H	A/D conversion result register 1	ADCR1		–	–	○	
0FF73H							
0FF73H	A/D conversion result register 1H	ADCR1H		–	○	–	
0FF74H	A/D conversion result register 2	ADCR2		–	–	○	
0FF75H							
0FF75H	A/D conversion result register 2H	ADCR2H		–	○	–	
0FF76H	A/D conversion result register 3	ADCR3		–	–	○	
0FF77H							
0FF77H	A/D conversion result register 3H	ADCR3H		–	○	–	
0FF78H	A/D conversion result register 4	ADCR4		–	–	○	
0FF79H							
0FF79H	A/D conversion result register 4H	ADCR4H		–	○	–	

Notes 1. When the LOCATION 0H instruction is executed. Add “F0000H” to this value when the LOCATION 0FH instruction is executed.

2. Bits 0, and 5 to 7 of PMC2 are fixed to “0” by hardware.

Table 3-6. Special Function Registers (SFRs) List (3/4)

Address[Note 1]	Special Function Register (SFR) Name	Symbol	R/W	Bit Units for Manipulation			On Reset
				1 bit	8 bits	16 bits	
0FF7AH	A/D conversion result register 5	ADCR5	R	–	–	○	Undefined
0FF7BH							
0FF7BH	A/D conversion result register 5H	ADCR5H		–	○	–	
0FF7CH	A/D conversion result register 6	ADCR6		–	–	○	
0FF7DH							
0FF7DH	A/D conversion result register 6H	ADCR6H		–	○	–	
0FF7EH	A/D conversion result register 7	ADCR7		–	–	○	
0FF7FH							
0FF7FH	A/D conversion result register 7H	ADCR7H		–	○	–	
0FF84H	Clocked serial interface mode register 1	CSIM1	R/W	○	○	–	00H
0FF85H	Clocked serial interface mode register 2	CSIM2		○	○	–	
0FF88H	Asynchronous serial interface mode register	ASIM		○	○	–	
0FF89H	Asynchronous serial interface mode register 2	ASIM2		○	○	–	
0FF8AH	Asynchronous serial interface status register	ASIS	R	○	○	–	
0FF8BH	Asynchronous serial interface status register 2	ASIS2		○	○	–	
0FF8CH	Serial receive buffer: UART0	RXB		–	○	–	Undefined
	Serial transmit shift register: UART0	TXS	W	–	○	–	
	Serial shift register: IOE1	SIO1	R/W	–	○	–	
0FF8DH	Serial receive buffer: UART2	RXB2	R	–	○	–	
	Serial transmit shift register: UART2	TXS2	W	–	○	–	
	Serial shift register: IOE2	SIO2	R/W	–	○	–	
0FF90H	Baud rate generator control register	BRGC		–	○	–	00H
0FF91H	Baud rate generator control register 2	BRGC2		–	○	–	
0FFA8H	In-service priority register	ISPR	R	○	○	–	
0FFAAH	Interrupt mode control register	IMC	R/W	○	○	–	80H
0FFACH	Interrupt mask register 0L	MK0L		○	○	–	FFH
0FFACH	Interrupt mask register 0	MK0		–	–	○	FFFFH
0FFADH							
0FFADH	Interrupt mask register 0H	MK0H		○	○	–	FFH
0FFAEH	Interrupt mask register 1L	MK1L		○	○	–	
0FFAEH	Interrupt mask register 1	MK1		–	○	○	FFFFH
0FFAFH							
0FFAFH	Interrupt mask register 1H	MK1H		○	○	–	FFH
0FFC0H	Standby control register[Note 2]	STBC		–	○	–	30H
0FFC2H	Watchdog timer mode register[Note 2]	WDM		–	○	–	00H
0FFC4H	Memory expansion mode register	MM		○	○	–	20H
0FFC7H	Programmable wait control register 1	PWC1		–	○	–	AAH

Notes 1. When the LOCATION 0H instruction is executed. Add “F0000H” to this value when the LOCATION 0FH instruction is executed.

2. These registers can be written only by using dedicated instructions MOV STBC, #byte and MOV WDM, #byte, and cannot be written by any other instructions.

Table 3-6. Special Function Registers (SFRs) List (4/4)

Address[Note 1]	Special Function Register (SFR) Name	Symbol	R/W	Bit Units for Manipulation			On Reset
				1 bit	8 bits	16 bits	
0FFC8H	Programmable wait control register 2	PWC2	R/W	–	–	○	AAAAH
0FFC9H							
0FFCAH	Bus width specification register	BW		–	–	○	**Note 3**
0FFCBH							
0FFCFH	Oscillation stabilization time specification register	OSTS		–	○	–	00H
0FFD0H-0FFDFH	External SFR area	–		○	○	–	Undefined
0FFE0H	Interrupt control register (INTOV0)	OVIC0		○	○	–	43H
0FFE1H	Interrupt control register (INTOV1)	OVIC1		○	○	–	
0FFE2H	Interrupt control register (INTOV4)	OVIC4		○	○	–	
0FFE3H	Interrupt control register (INTP0)	PIC0		○	○	–	
0FFE4H	Interrupt control register (INTP1)	PIC1		○	○	–	
0FFE5H	Interrupt control register (INTP2)	PIC2		○	○	–	
0FFE6H	Interrupt control register (INTP3)	PIC3		○	○	–	
0FFE7H	Interrupt control register (INTP4)	PIC4		○	○	–	
0FFE8H	Interrupt control register (INTP5)	PIC5		○	○	–	
0FFE9H	Interrupt control register (INTP6)	PIC6		○	○	–	
0FFEAH	Interrupt control register (INTCM10)	CMIC10		○	○	–	
0FFEBH	Interrupt control register (INTCM11)	CMIC11		○	○	–	
0FFF0H	Interrupt control register (INTCM40)	CMIC40		○	○	–	
0FFF1H	Interrupt control register (INTCM41)	CMIC41		○	○	–	
0FFF2H	Interrupt control register (INTSER)	SERIC		○	○	–	
0FFF3H	Interrupt control register (INTSR)	SRIC		○	○	–	
	Interrupt control register (INTCSI1)	CSIIC1		○	○	–	
0FFF4H	Interrupt control register (INTST)	STIC		○	○	–	
0FFF5H	Interrupt control register (INTSER2)	SERIC2		○	○	–	
0FFF6H	Interrupt control register (INTSR2)	SRIC2		○	○	–	
	Interrupt control register (INTCSI2)	CSIIC2		○	○	–	
0FFF7H	Interrupt control register (INTST2)	STIC2		○	○	–	
0FFF8H	Interrupt control register (INTAD)	ADIC		○	○	–	
0FFFCH	Internal memory size select register[Note 2]	IMS		–	○	–	CDH

Notes 1. When the LOCATION 0H instruction is executed. Add “F0000H” to this value when the LOCATION 0FH instruction is executed.

2. Writing to IMS is valid only with the flash memory model (μPD78F4046). When writing to IMS with mask ROM models (μPD784054), the value is not changed and remains the same as the value on reset.

3. The value of this register on reset differs depending on the setting of the BWD pin.

BWD = 0: 0000H

BWD = 1: 00FFH

3.9 Cautions

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

(2) Special function registers (SFRs)
Addresses onto which SFRs are not assigned should not be accessed in the area 0FF00H to 0FFFFH[Note]. If such an address is accessed by mistake, the μPD784054 may become deadlocked. A deadlock can only be released by reset input.

Note When the LOCATION 0H instruction is executed; FFF00H to FFFFFH when the LOCATION 0FH instruction is executed.

(3) Stack pointer (SP) operation
With stack addressing, the entire 1 M-byte space can be accessed, but a stack area cannot be reserved in the SFR area or internal ROM area.

(4) Stack pointer (SP) initialization
The SP is undefined after $\overline{\text{RESET}}$ input, while non-maskable interrupts can be acknowledged directly after reset release. Therefore, an unforeseen operation may be performed if a non-maskable interrupt request is generated while the SP is in the undefined state directly after reset release. To minimize this risk, the following program should be coded without fail after reset release.

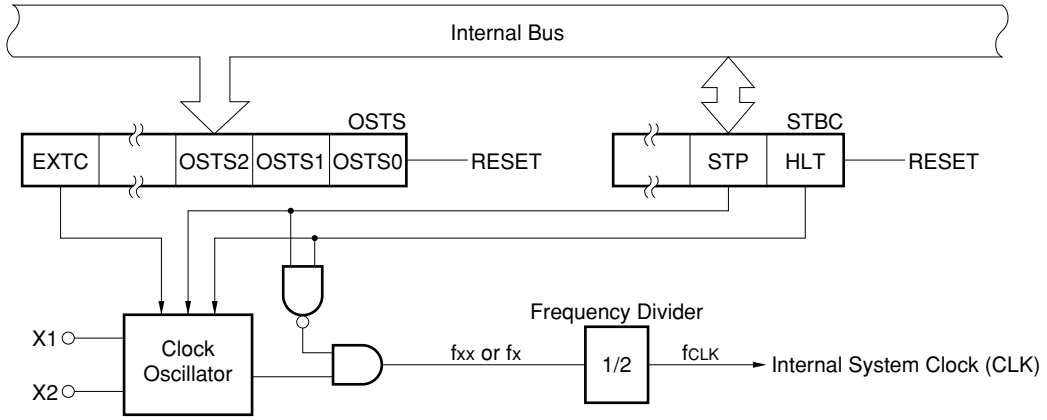
```
RSTVCT      CSEG   AT     0
            DW     RSTSTRT
              to
INITSEG     CSEG   BASE
RSTSTRT :   LOCATION 0H ; or LOCATION 0FH
            MOVG SP, #STKBGN
```

CHAPTER 4 CLOCK GENERATOR

4.1 Configuration and Function

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

Figure 4-1. Block Diagram of Clock Generator



Remark f_{XX} : crystal/ceramic oscillation frequency
f_X : external clock frequency
f_{CLK} : internal system clock frequency

The clock oscillator oscillates by means of a crystal resonator/ceramic resonator connected to the X1 and X2 pins. When standby mode (STOP) is set, oscillation stops (refer to **CHAPTER 16 STANDBY FUNCTION**).

An external clock can also be input. In this case, input the clock signal to the X1 pin.

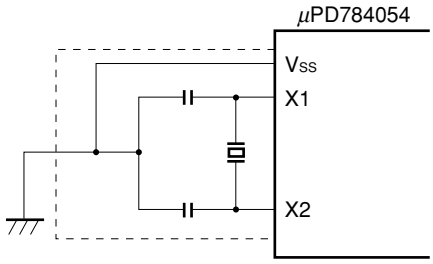
The processing of the X2 pin differs depending on the setting of the EXTC bit of the oscillation stabilization time specification register (OSTS), as follows:

EXTC bit = 1: Input a clock in reverse phase to the clock input to X1 pin to the X2 pin.
EXTC bit = 0: Leave the X2 pin unconnected.

The frequency divider circuit divides the output (f_{XX} or f_X) of the clock oscillator by two, to generate an internal system clock (f_{CLK}).

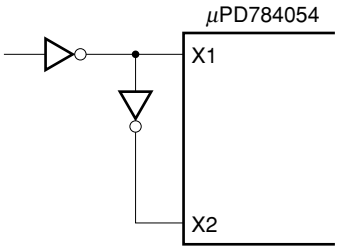
Figure 4-2. Clock Oscillator External Circuitry

(a) Crystal/ceramic oscillation

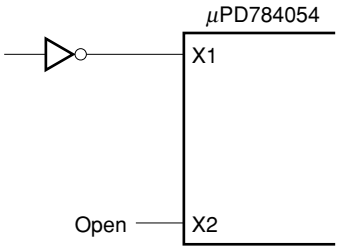


(b) External clock

• EXTC bit of OSTS = 1



• EXTC bit of OSTS = 0



Cautions 1. The oscillator should be as close as possible to the X1 and X2 pins.

2. No other signal lines should pass through the area enclosed by the dotted line.

Remark Use of crystal resonator and ceramic resonator

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

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

4.2 Control Registers

4.2.1 Standby control register (STBC)

STBC is a register used to set the standby mode. Refer to **CHAPTER 16 STANDBY FUNCTION** for details of the standby modes.

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

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

Since an operand error interrupt is only generated in the event of an inadvertent program loop (with the NEC Electronics assembler RA78K4, only the correct dedicated instruction is generated when the MOV STBC, #byte instruction 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 no operation is performed. That is, a write is not performed on the STBC, and an interrupt such as an operand error interrupt is not generated. The STBC can be read at any time with a data transfer instruction.

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

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

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

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

	7	6	5	4	3	2	1	0
STBC	0	0	1	1	0	0	STP	HLT

STP	HLT	CPU Operating Mode Control
0	0	Normal mode
0	1	HALT mode
1	0	STOP mode
1	1	IDLE mode

Caution If the STOP mode is used when external clock input is used, the EXTC bit of the oscillation stabilization time specification register (OSTS) must be set (1) before setting the STOP mode. If the STOP mode is used when the EXTC bit of the OSTS is in the cleared (0) state when external clock input is used, the μPD784054 may be damaged or suffer reduced reliability.

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

4.2.2 Oscillation stabilization time specification register (OSTS)

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

The OSTS can be read/written to by an 8-bit manipulation instruction.

$\overline{\text{RESET}}$ input clears the OSTS register contents to 00H.

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

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

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

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

EXTC	Selects External Clock
0	Opens X2 pin when crystal/ceramic oscillation is used or when external clock is used.
1	Inputs clock in reverse phase to clock input X1 pin to X2 pin.

OSTS2	OSTS1	OSTS0	Selects oscillation stabilization time (for details, refer to **Figure 16-4**).

Cautions 1. When using a crystal/ceramic oscillation, the EXTC bit must be cleared (0). If the EXTC bit is set (1), oscillation will stop.

2. If the STOP mode is used with external clock input, the EXTC bit must be set (1) before setting the STOP mode. If the STOP mode is used when the EXTC bit is in the cleared (0) state, the μPD784054 may be damaged or suffer reduced reliability.

3. When setting the EXTC bit to 1 during external clock input, be sure to input a clock in phase reverse to that of the clock input to the X1 pin, to the X2 pin. When the EXTC bit is set to 1, the μPD784054 operates on only the clock input to the X2 pin.

4.3 Clock Generator Operation

4.3.1 Clock oscillator

(1) When using crystal/ceramic oscillation

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

(2) When using external clock

The clock oscillator supplies the clock input from the X1 pin to the internal circuitry when the $\overline{\text{RESET}}$ signal is input. The oscillator operates as follows when the EXTC bit of the oscillation stabilization time specification register (OSTS) is set to 1.

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

Cautions 1. When using a crystal/ceramic oscillation, the EXTC bit of the Oscillation stabilization time specification register (OSTS) must be cleared (0). If the EXTC bit is set (1), oscillation will stop.

2. If the STOP mode is used with external clock input, the EXTC bit of the OSTS must be set (1) before setting the STOP mode. If the STOP mode is used when the EXTC bit is in the cleared (0) state, not only will the clock generator consumption current not be reduced, but the μPD784054 may also be damaged or suffer reduced reliability.

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

4.3.2 Frequency divider

The frequency divider divides the output from the clock oscillator by two, and supplies the result to the CPU and peripheral hardware.

4.4 Cautions

The following cautions apply to the clock generator.

4.4.1 When an external clock is input

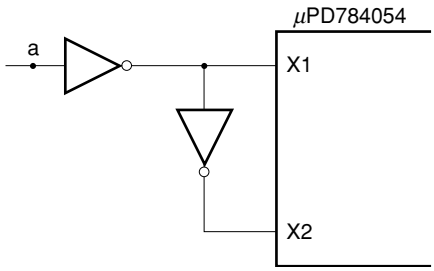
(1) If the STOP mode is used with external clock input, the EXTC bit of the oscillation stabilization time specification register (OSTS) must be set (1). If the STOP mode is used when the EXTC bit is in the cleared (0) state, the μPD784054 may be damaged or suffer reduced reliability.

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

(3) When an external clock is input, this should be performed with a HCMOS device, or a device with the equivalent drive capability.

(4) A signal should not be extracted from the X1 and X2 pins. If a signal is extracted, it should be extracted from point a in Figure 4-5.

Figure 4-5. Signal Extraction with External Clock Input



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

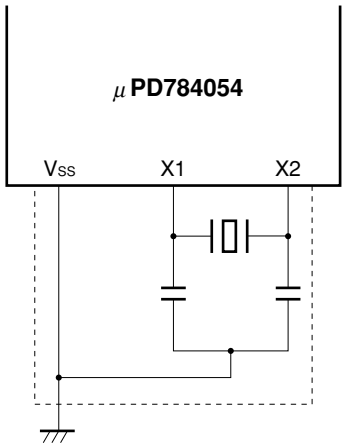
4.4.2 When crystal/ceramic oscillation is used

(1) As the oscillator is a high-frequency analog circuit, considerable care is required.
The following points, in particular, require attention.

- The wiring should be kept as short as possible.
- No other signal lines should be crossed.
- Avoid lines carrying a high fluctuating current.
- The oscillator capacitor grounding point should always be at the same potential as the V_{SS} pin. Do not ground to a ground pattern carrying a high current.
- A signal should not be taken from the oscillator.

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

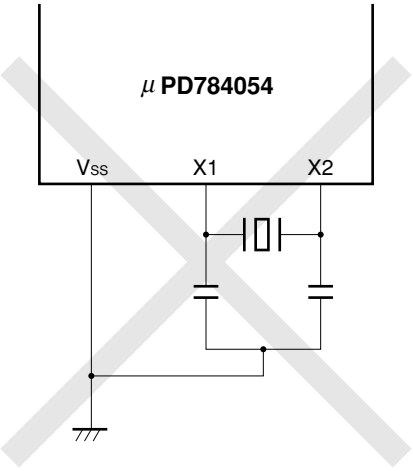
Figure 4-6. Cautions on Resonator Connection



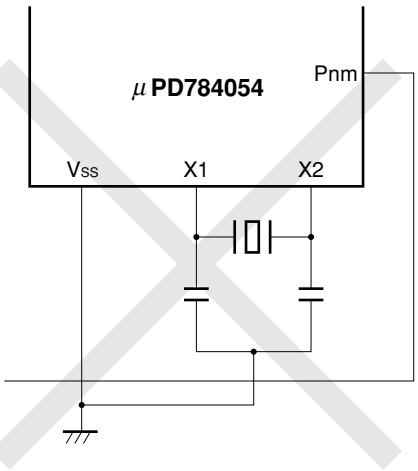
Cautions 1. The oscillator should be as close as possible to the X1 and X2 pins.
2. No other signal lines should pass through the area enclosed by the dotted line.

Figure 4-7. Incorrect Example of Resonator Connection

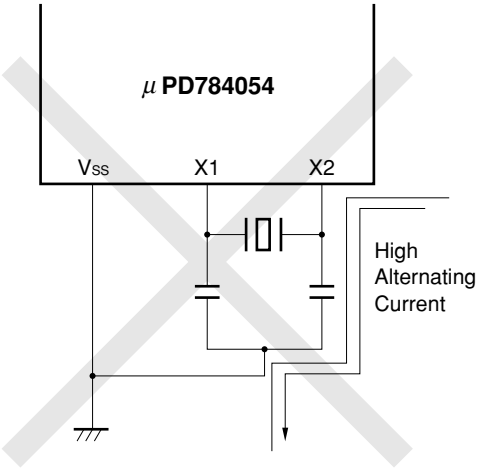
(a) Wiring of connected circuits is too long



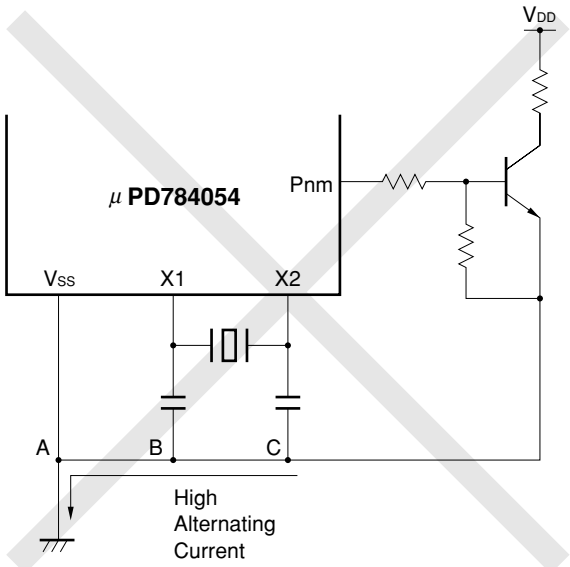
(b) Crossed signal lines



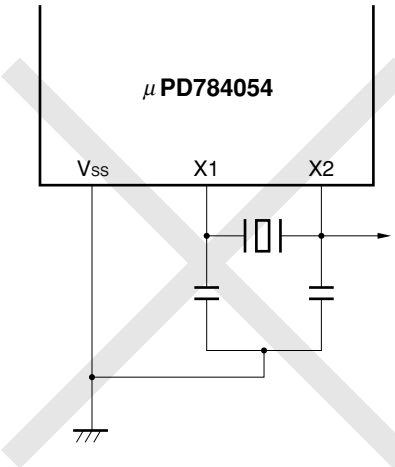
(c) Wiring near high alternating current



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



(e) Signal extracted



(2) When the device is powered on, and when restoring from the STOP mode, sufficient time must be allowed for the oscillation to stabilize. Generally speaking, the time required for oscillation stabilization is several milliseconds when a crystal resonator is used, and several hundred microseconds when a ceramic resonator is used.
An adequate oscillation stabilization period should be secured by the following means:

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

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

CHAPTER 5 PORT FUNCTIONS

5.1 Digital Input/Output Port

The μPD784054 is provided with the ports shown in Figure 5-1, enabling various kinds of control to be performed. The function of each port is shown in Table 5-1. For port 0, ports 4 to 6, and port 9, connection of an internal pull-up resistor can be specified by software when used as input ports.

Figure 5-1. Port Configuration

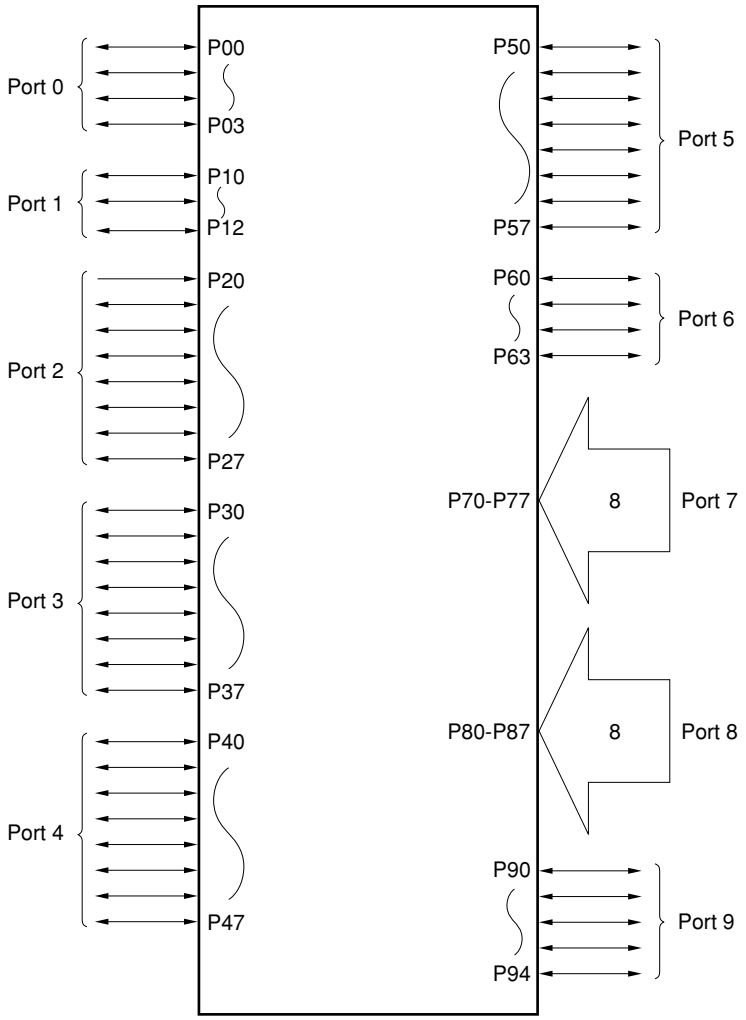


Table 5-1. Port Function

Port Name	Pin Name	Function	Specification of Pull-Up Resistor by Software
Port 0	P00-P03	Can be set in input or output mode bit-wise.	All pins can be set in input mode
Port 1	P10-P12		–
Port 2	P20-P27	Can be set in input or output mode bit-wise (however, P20 is input-only).	
Port 3	P30-P37	Can be set in input or output mode bit-wise.	
Port 4	P40-P47		All pins can be set in input mode
Port 5	P50-P57		
Port 6	P60-P63		
Port 7	P70-P77	Input port	–
Port 8	P80-P87		
Port 9	P90-P94	Can be set in input or output mode bit-wise.	All pins can be set in input mode

5.2 Port 0

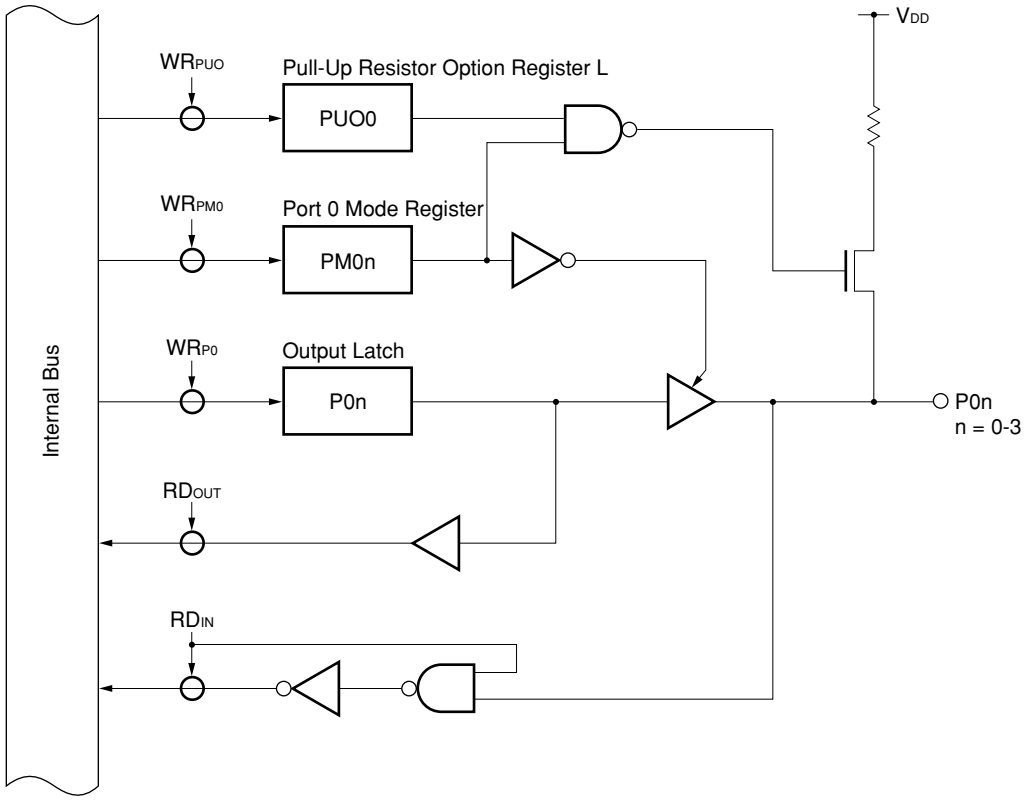
Port 0 is a 4-bit input/output port with an output latch. Input/output can be specified bit-wise by means of the port 0 mode register (PM0). Each pin incorporates a software programmable pull-up resistor.

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

5.2.1 Hardware configuration

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

Figure 5-2. Block Diagram of Port 0



5.2.2 Input/output mode/control mode setting

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

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

Address : 0FF20H On reset : FFH R/W

	7	6	5	4	3	2	1	0
PM0	1	1	1	1	PM03	PM02	PM01	PM00

PM0n	Specifies I/O Mode of P0n Pin (n = 0 to 3)
0	Output mode (output buffer on)
1	Input mode (output buffer off)

5.2.3 Operating status

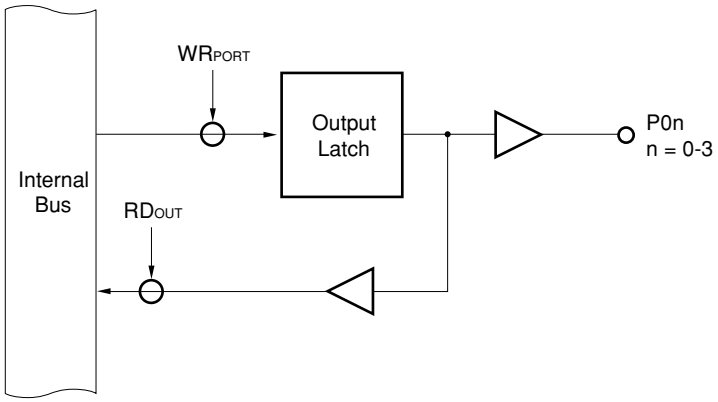
Port 0 is an input/output port

(1) When set as an output port

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

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

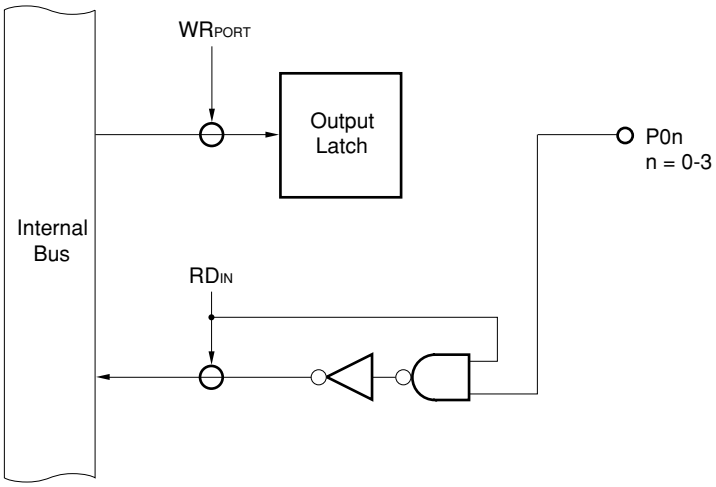
Figure 5-4. Port Specified as Output Port



(2) When set as an input port

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

Figure 5-5. Port Specified as Input Port



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

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

5.2.4 Internal pull-up resistors

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

Whether or not an internal pull-up resistor is to be used can be specified for each pin by means of the PUO0 bit of the pull-up resistor option register L (PUOL) and the port 0 mode register (PM0).

When PUO0 bit is 1, the internal pull-up resistor of only the pin set in the input mode by the PM0 is valid when the PUO bit is 1.

Figure 5-6. Pull-Up Resistor Option Register L (PUOL) Format

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

	7	6	5	4	3	2	1	0
PUOL	0	PUO6	PUO5	PUO4	0	0	0	PUO0

PUO6	Specifies Pull-up Resistor of Port 6 (refer to **Figure 5-44**).

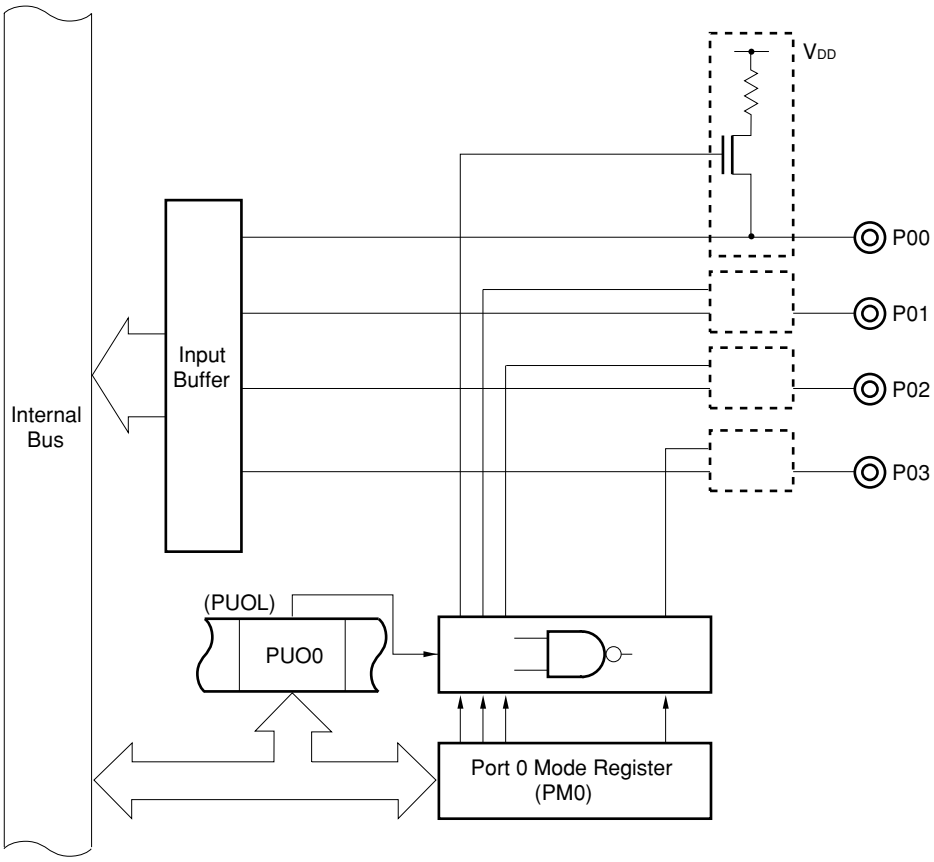
PUO5	Specifies Pull-up Resistor of Port 5 (refer to **Figure 5-38**).

PUO4	Specifies Pull-up Resistor of Port 4 (refer to **Figure 5-32**).

PUO0	Specifies Pull-up Resistor of Port 0
0	Not used with port 0
1	Used with port 0

Remark When STOP mode is entered, setting 00H in PUOL is effective in reducing the current consumption.

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



5.3 Port 1

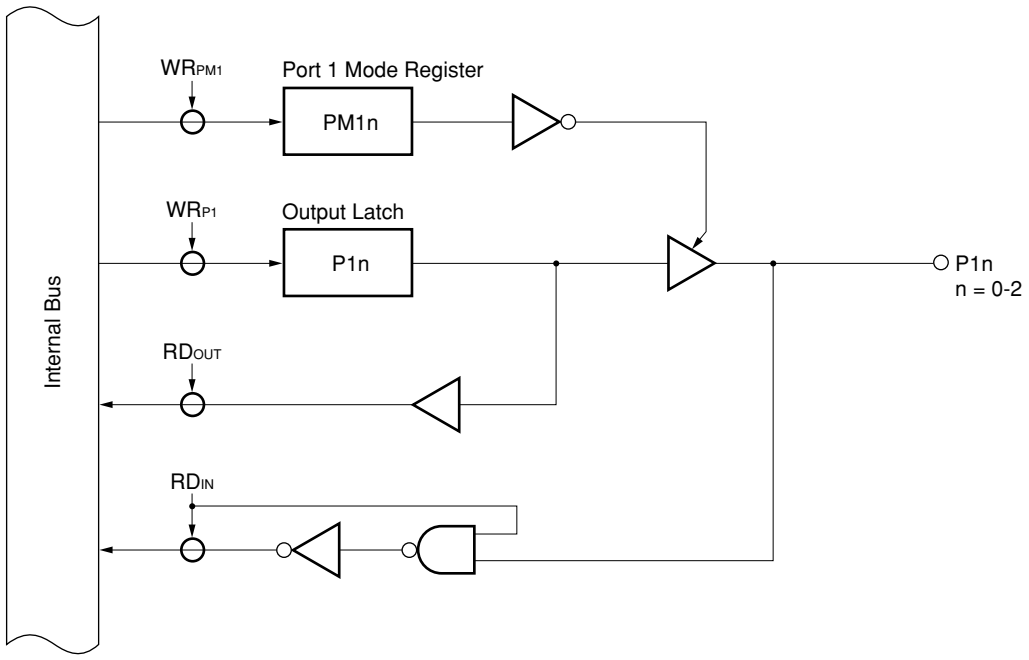
Port 1 is a 3-bit input/output port with an output latch. Input/output can be specified bit-wise by means of the port 1 mode register (PM1).

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

5.3.1 Hardware configuration

The port 1 hardware configuration is shown in Figure 5-8.

Figure 5-8. Block Diagram of Port 1



5.3.2 Setting I/O mode/control mode

The input/output mode of port 1 is set by using the port 1 mode register (PM1) per pin, as shown in Figure 5-9.

Figure 5-9. Format of Port 1 Mode Register (PM1)

Address : 0FF21H On reset : FFH R/W

	7	6	5	4	3	2	1	0
PM1	1	1	1	1	1	PM12	PM11	PM10

PM1n	Specifies I/O mode of P1n pin (n = 0 to 2)
0	Output mode (output buffer ON)
1	Input mode (output buffer OFF)

5.3.3 Operating status

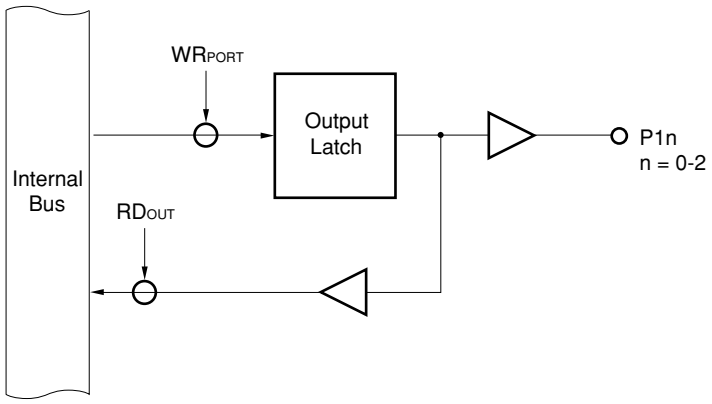
Port 1 is an input/output port.

(1) When set as an output port

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

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

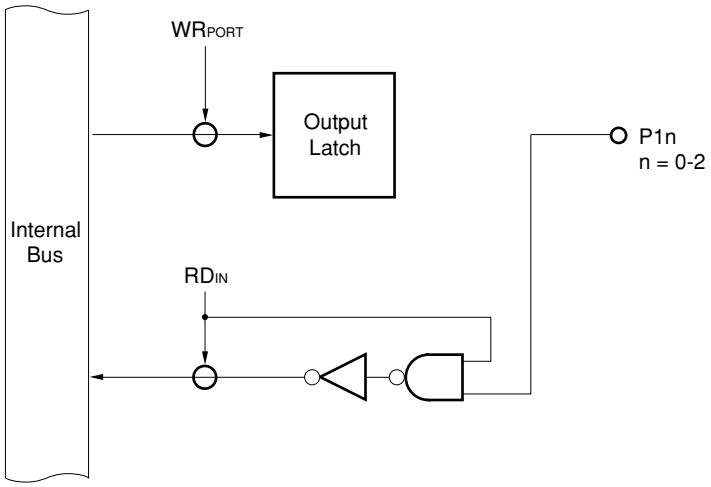
Figure 5-10. Port Specified as Output Port



(2) When set as an input port

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

Figure 5-11. Port Specified as Input Port



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

5.4 Port 2

Port 2 is an 8-bit I/O port with an output latch. This port can be set in the input or output mode in 1-bit units by using port 2 mode register (PM2) (however, P20 is input-only).

In addition to the input/output port function, port 2 also has a function to input control signals such as external interrupt signals, and output the timer signal of timer 0 (refer to **Table 5-2**). P21 to P24 serve as the timer output pins of timer 0 if so specified by port 2 mode control register (PMC2). The level of each pin of this port can always be read or tested regardless of the multiplexed function.

All the eight pins are Schmitt-trigger input pins to prevent malfunctioning due to noise.

When $\overline{\text{RESET}}$ is input, this port is set in the input mode (output high-impedance status), and the contents of the output latch are undefined.

Table 5-2. Operation Mode of Port 2

(n = 0 to 7)

Mode	Port Mode		Control Signal Output Mode
Set condition	PMC2n = 0		PMC2n = 1
	PM2n = 0	PM2n = 1	PM2n = ×
P20	–	Input port/NMI input[Note]	–
P21	Output port	Input port/INTP0 input	TO00 output
P22		Input port/INTP1 input	TO01 output
P23		Input port/INTP2 input	TO02 output
P24		Input port/INTP3 input	TO03 output
P25		Input port/INTP4 input	–
P26		Input port/INTP5 input	
P27		Input port/INTP6 input	

Note The NMI input pin accepts an interrupt request regardless of whether interrupts are enabled or disabled.

Remark ×: don't care

(1) Port mode

(a) Function as port pin

Each port pin set in the port mode by the port 2 mode control register (PMC2) can be set in the input or output mode in 1-bit units by the port 2 mode register (PM2) (however, P20 is fixed in the input mode).

(b) Function as control signal input pins

If PMC2n (n = 0 to 7) bit of PMC2 is "0" and if PM2n (n = 0-7) bit of PM2 is "1", the pins of port 2 can be used as the following control signal input pins.

(i) NMI (Non-maskable Interrupt)

This pin inputs an external non-maskable interrupt request. Whether the interrupt request is detected at the rising or falling edge can be specified by using external interrupt mode register 0 (INTM0).

(ii) INTP0 to INTP6 (Interrupt from Peripherals)

These pins input external interrupt requests. When the valid edge specified by external interrupt mode registers (INTM0 and INTM1) is detected on the INTP0 to INTP6 pins, an interrupt occurs (refer to **CHAPTER 13 EDGE DETECTION FUNCTION**).

The INTP0 to INTP4 pins can also be used as external trigger input pins of each function, as follows:

- INTP0 ... Capture trigger input pin of capture/compare register 00 (CC00) of timer 0
- INTP1 ... Capture trigger input pin of capture/compare register 01 (CC01) of timer 0
- INTP2 ... Capture trigger input pin of capture/compare register 02 (CC02) of timer 0
- INTP3 ... Capture trigger input pin of capture/compare register 03 (CC03) of timer 0
- INTP4 ... External trigger input pin of A/D converter

(2) Control signal output mode

The P21 to P24 pins can be used as the timer output pins (TO00 to TO03) of timer 0 in 1-bit units if so specified by the port 2 mode control register (PMC2).

5.4.1 Hardware configuration

The port 2 hardware configuration is shown Figure 5-12 to 5-14.

Figure 5-12. Block Diagram of P20 (Port 2)

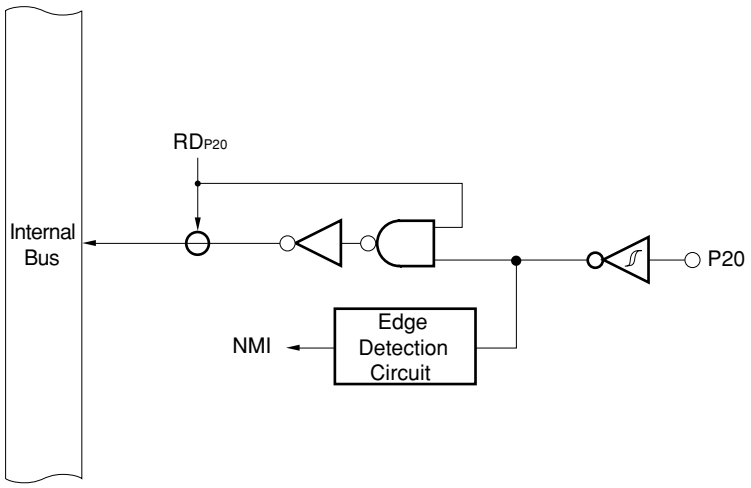


Figure 5-13. Block Diagram of P21 to P24 (Port 2)

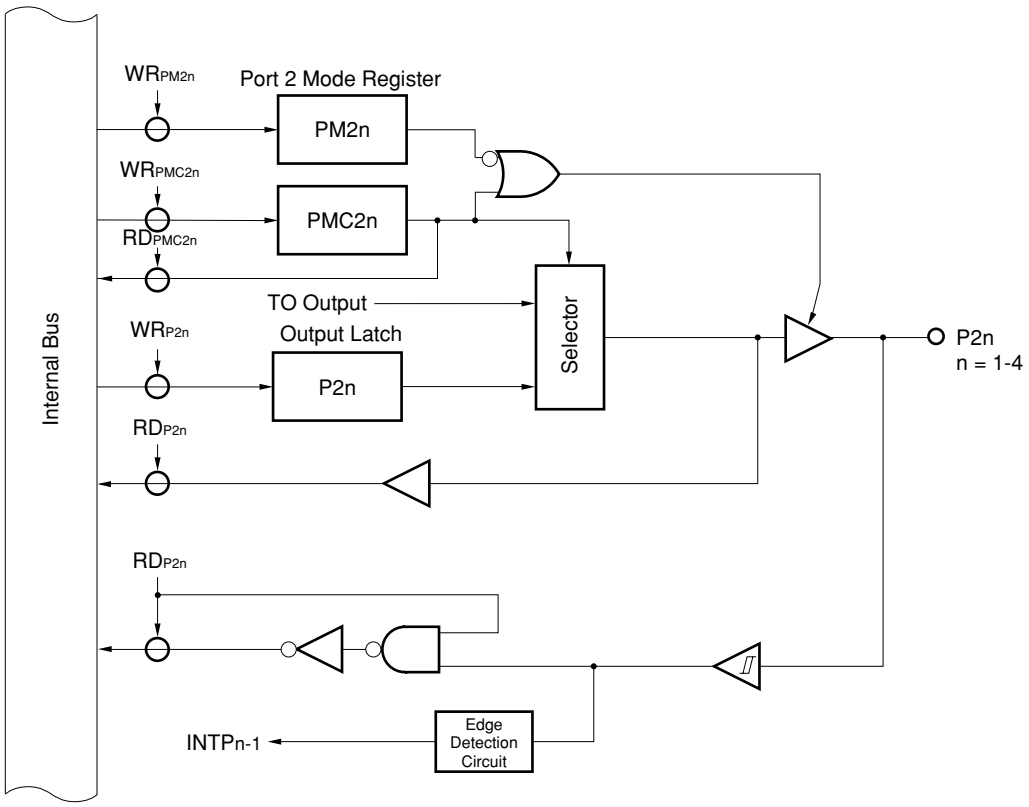
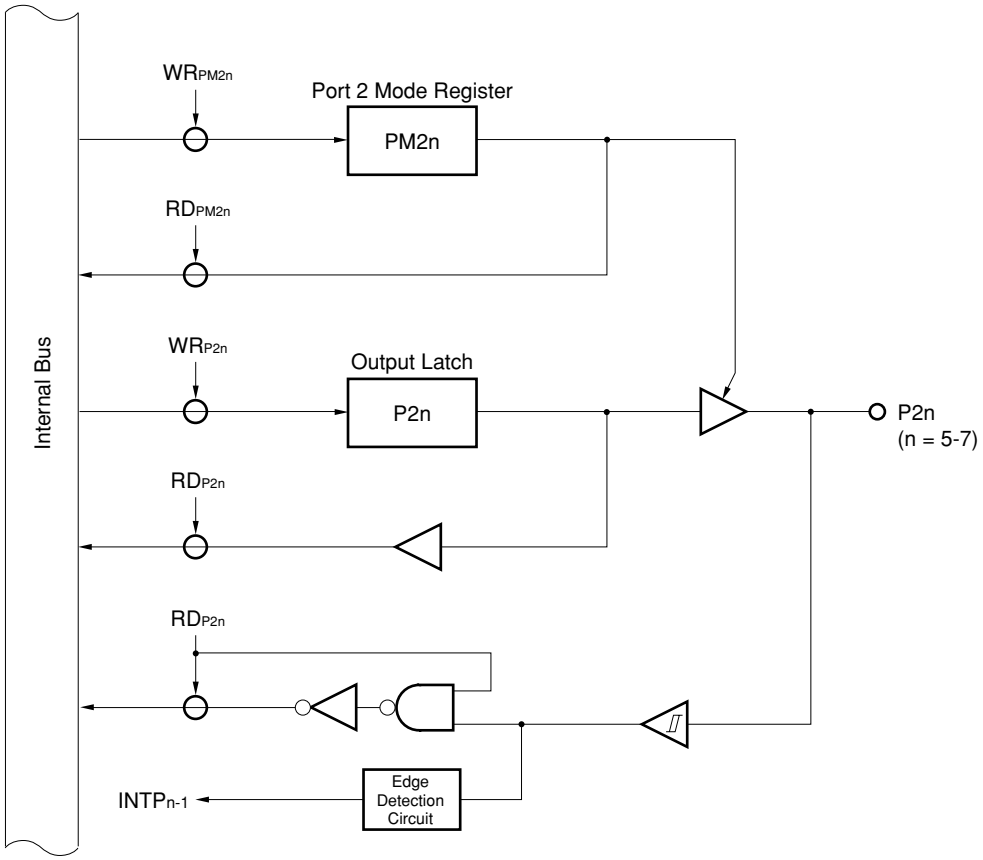


Figure 5-14. Block Diagram of P25 to P27 (Port 2)



5.4.2 Setting I/O mode/control mode

The input/output mode of P21 to P27 is set per pin by using the port 2 mode register (PM2), as shown in Figure 5-15. P20 is input-only.

P21 to P24 also functions as timer output pins of timer 0, in addition to as input/output port pins. To use these pins as timer output pins, set them in the control mode by using the port 2 mode control register (PMC2) as shown in Figure 5-16.

Figure 5-15. Format of Port 2 Mode Register (PM2)

Address : 0FF22H On reset : FFH R/W

	7	6	5	4	3	2	1	0
PM2	PM27	PM26	PM25	PM24	PM23	PM22	PM21	1

PM2n	Specifies input/output mode of P2n pin (n = 1 to 7)
0	Output mode (output buffer ON)
1	Input mode (output buffer OFF)

Figure 5-16. Format of Port 2 Mode Control Register (PMC2)

Address : 0FF42H On reset : 00H R/W

	7	6	5	4	3	2	1	0
PMC2	0	0	0	PMC24	PMC23	PMC22	PMC21	0

PMC24	Specifies Control Mode of P24 Pin
0	I/O port mode/INTP3 input mode
1	TO03 output mode

PMC23	Specifies Control Mode of P23 Pin
0	I/O port mode/INTP2 input mode
1	TO02 output mode

PMC22	Specifies Control Mode of P22 Pin
0	I/O port mode/INTP1 input mode
1	TO01 output mode

PMC21	Specifies Control Mode of P21 Pin
0	I/O port mode/INTP0 input mode
1	TO00 output mode

Caution Even when using the P21 to P27 pins in the output port mode or timer output mode, INTPn (n = 0 to 6) interrupt occurs depending on edge detection of the pin level. Therefore, mask the interrupt before using the pins.

5.4.3 Operating status

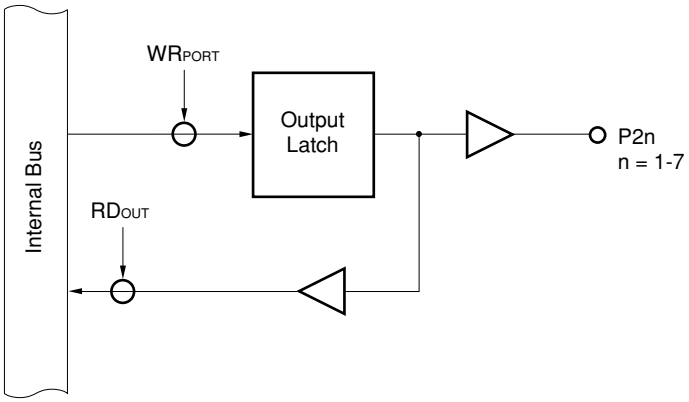
Port 2 is an I/O port (however, the P20 pin is input-only). The P21 to P24 pins can also be used as timer output pins of timer 0.

(1) In output port mode

The output latch is valid, and data is transferred between the output latch and accumulator by a transfer instruction. The contents of the output latch can be freely set by a logical operation instruction. Data that has been written to the output latch[Note].

Note Including when the other bits of the same port are manipulated by a bit manipulation instruction.

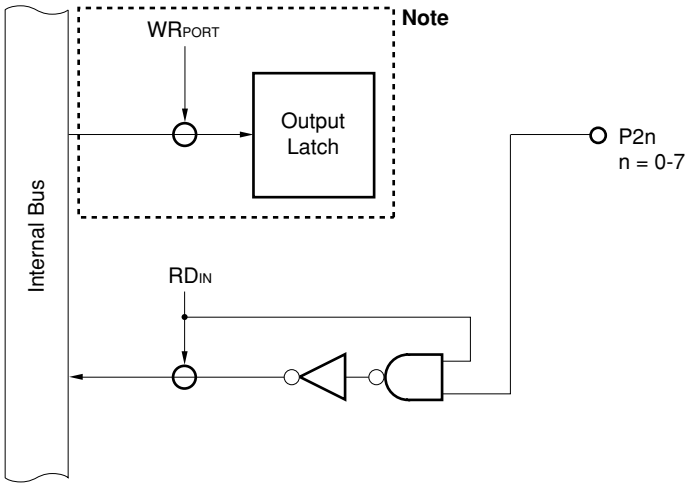
Figure 5-17. Port in Output Port Mode



(2) In input port mode

The level of a port pin can be loaded to the accumulator by using a transfer instruction. Even in this case, data can be written to the output latch. Data transferred from the accumulator by a transfer instruction is stored to all the output latches regardless of whether the input or output mode is specified. However, because the output buffer of a bit (pin) set in the input mode is in the high-impedance state, its contents are not output to the port pin (the contents of the output latch are output to the port pin when the mode of the pin is changed from input to output). The contents of the output latch of the pin set in the input port cannot be loaded to the accumulator.

Figure 5-18. Port in Input Port Mode



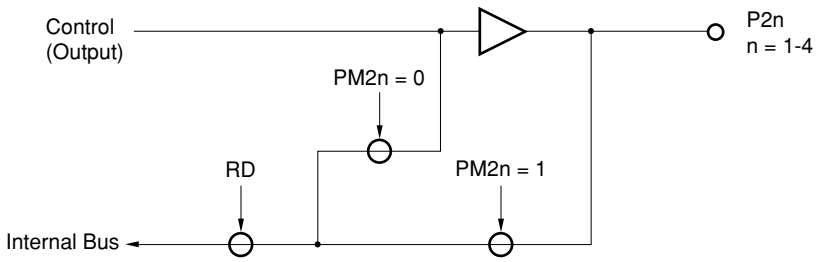
Note P20 does not have the circuit enclosed by the dotted line in the above figure.

Caution Although the result of a bit manipulation instruction is ultimately 1 bit manipulation, it accesses a port in 8-bit units. If such an instruction is executed to manipulate a port with some pins set in the input mode and the others in the control mode, the contents of the output latch are undefined (except when a pin is manipulated by the SET1 or CLR1 instruction). Especially, care must be exercised if the mode of some pins must be changed between input and output.
The same applies when manipulating the port by using the other 8-bit operation instructions.

(3) Pin in control mode

P21 to P24 can be used to output control signals in 1-bit units regardless of the setting of the port 2 mode register (PM2), if the corresponding bit of the port 2 mode control register (PMC2) is set (1). When using each pin as a control signal pin, the status of the control signal can be checked by executing an instruction that reads the port.

Figure 5-19. Port in Control Mode



If the PM2n (n = 1 to 4) bit of PM2 is set (1), and if an instruction that reads the port is executed, the level of the corresponding control signal pin can be read.
If the port read instruction is executed when the PM2n bit is reset (0), the status of the control signal in the μPD784054 can be read.

5.5 Port 3

Port 3 is an 8-bit input/output port with an output latch. Input/output can be specified bit-wise by means of the port 3 mode register (PM3).

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

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

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

Table 5-3. Port 3 Operating Modes

(n = 0 to 7)

Mode	Port Mode	Control Signal Input/Output Mode
Setting Condition	PMC3n = 0	PMC3n = 1
P30	Input/output port	TO10 output
P31		TO11 output
P32		RxD/SI1 input
P33		TxD/SO1 output
P34		ASCK input/$\overline{\text{SCK1}}$ input/output
P35		RxD2/SI2 input
P36		TxD2/SO2 output
P37		ASCK2 input/$\overline{\text{SCK2}}$ input/output

(a) Port mode

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

(b) Control signal input/output mode

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

(i) TO10, TO11 (Timer Output)

These are timer output pins of timer 1.

(ii) RxD, RxD2 (Receive Data)

These are serial data input pins of the asynchronous serial interface.

(iii) TxD, TxD2 (Transmit Data)

These are serial data output pins of the asynchronous serial interface.

(iv) SI1, SI2 (Serial Input)

These are serial data input pins of the 3-wire serial I/O.

(v) SO1, SO2 (Serial Output)

These are serial data output pins of the 3-wire serial I/O.

(vi) ASCK, ASCK2 (Asynchronous Serial Clock)

These are external baud rate clock input pins.

(vii) $\overline{\text{SCK1}}$, $\overline{\text{SCK2}}$ (Serial Clock)

These are serial clock I/O pins of the 3-wire serial I/O.

5.5.1 Hardware configuration

The port 3 hardware configuration is shown in Figures 5-20 to 5-22.

Figure 5-20. Block Diagram of P30, P31, P33 and P36 (Port 3)

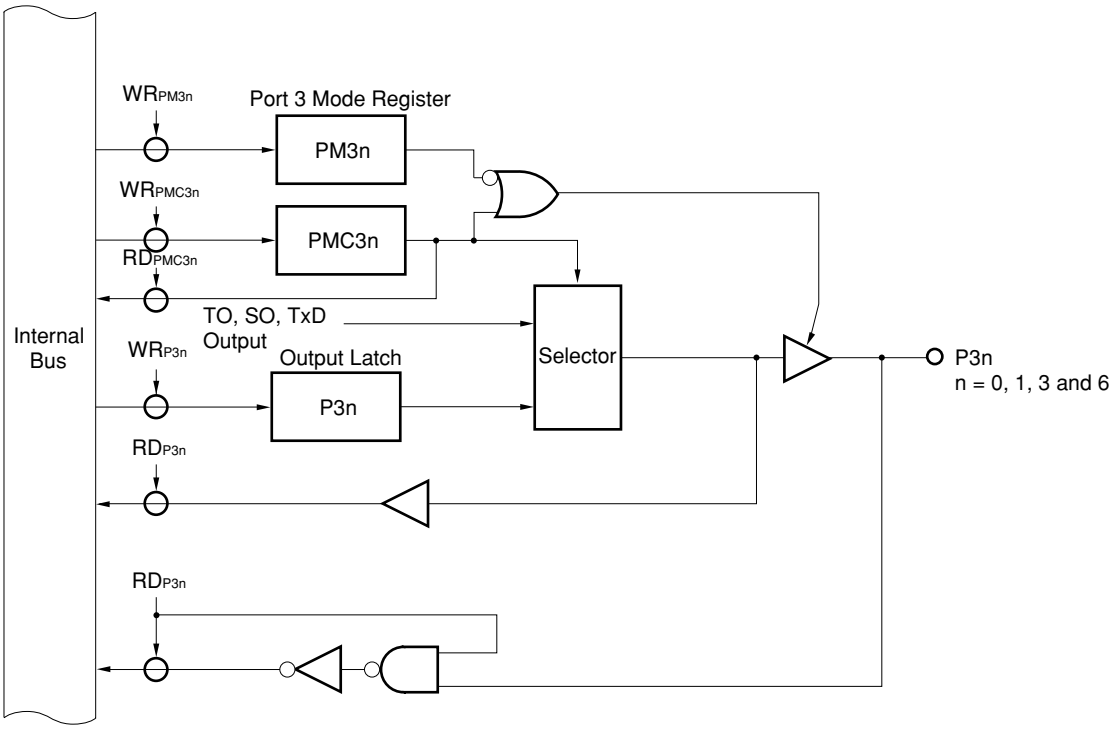


Figure 5-21. Block Diagram of P32 and P35 (Port 3)

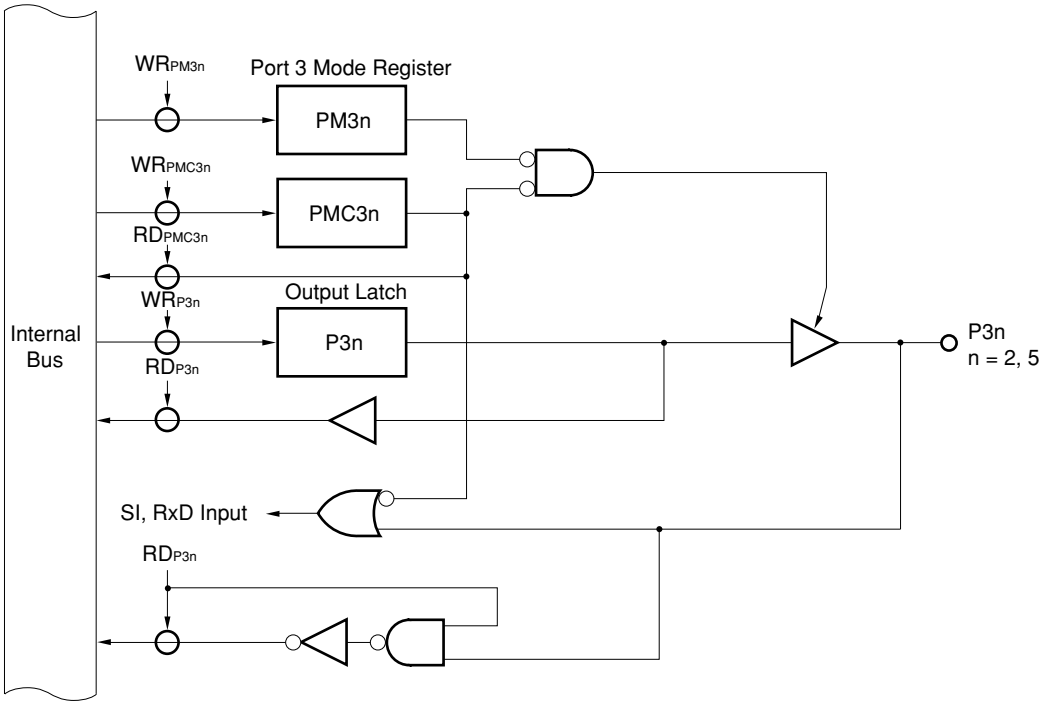
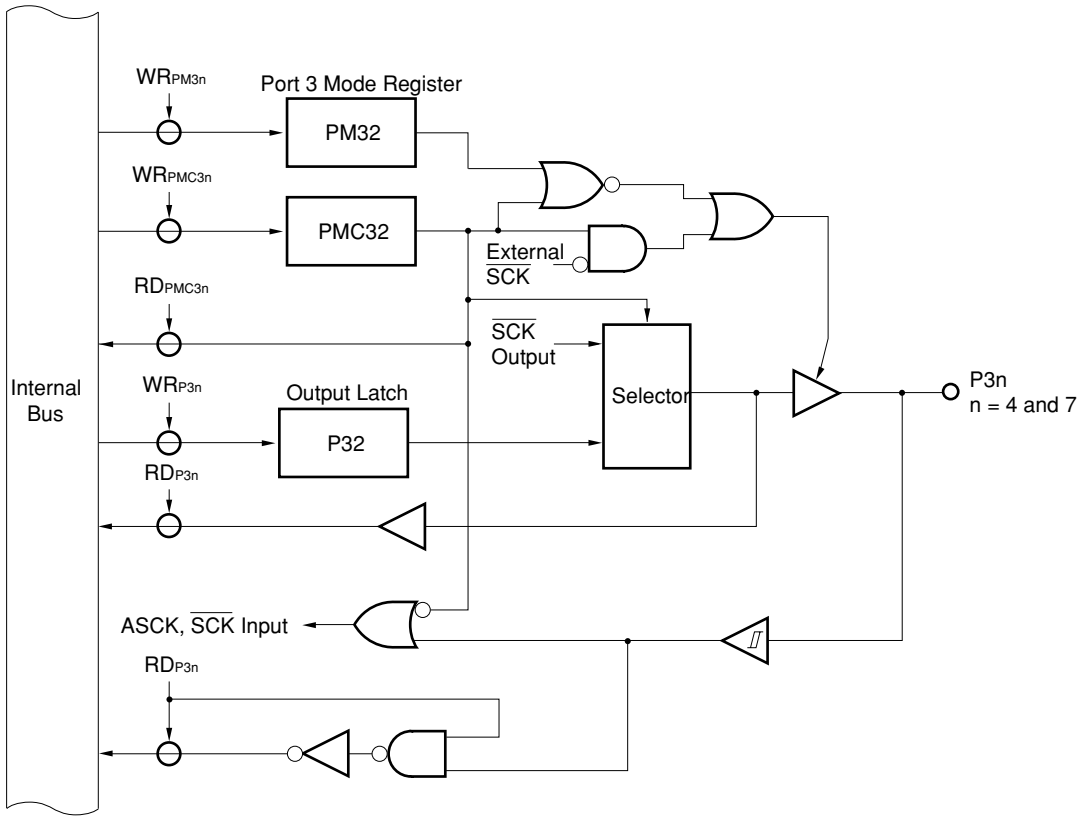


Figure 5-22. Block Diagram of P34 and P37 (Port 3)



5.5.2 Input/output mode/control mode setting

The port 3 input/output mode is set for each pin by means of the port 3 mode register (PM3) as shown in Figure 5-23.

In addition to their input/output port function, port 3 pins also have a dual function as various control signal pins, and the control mode is specified by means of the port 3 mode control register (PMC3) as shown in Figure 5-24.

Figure 5-23. Format of Port 3 Mode Register (PM3)

Address : 0FF23H On reset : FFH R/W

	7	6	5	4	3	2	1	0
PM3	PM37	PM36	PM35	PM34	PM33	PM32	PM31	PM30

PM3n	Specifies I/O Mode of P3n Pin (n = 0 to 7)
0	Output mode (output buffer ON)
1	Input mode (output buffer OFF)

Figure 5-24. Format of Port 3 Mode Control Register (PMC3)

Address : 0FF43H On reset : 00H R/W

	7	6	5	4	3	2	1	0
PMC3	PMC37	PMC36	PMC35	PMC34	PMC33	PMC32	PMC31	PMC30

PMC37	Specifies Control Mode of P37 Pin
0	I/O port mode
1	ASCK2/$\overline{\text{SCK2}}$ I/O mode

PMC36	Specifies Control Mode of P36 Pin
0	I/O port mode
1	TxD2/SO2 output mode

PMC35	Specifies Control Mode of P35 Pin
0	I/O port mode
1	RxD2/SI2 input mode

PMC34	Specifies Control Mode of P34 Pin
0	I/O port mode
1	ASCK/$\overline{\text{SCK1}}$ I/O mode

PMC33	Specifies Control Mode of P33 Pin
0	I/O port mode
1	TxD/SO1 output mode

PMC32	Specifies Control Mode of P32 Pin
0	I/O port mode
1	RxD/SI1 input mode

PMC31	Specifies Control Mode of P31 Pin
0	I/O port mode
1	TO11 output mode

PMC30	Specifies Control Mode of P30 Pin
0	I/O port mode
1	TO10 output mode

5.5.3 Operating status

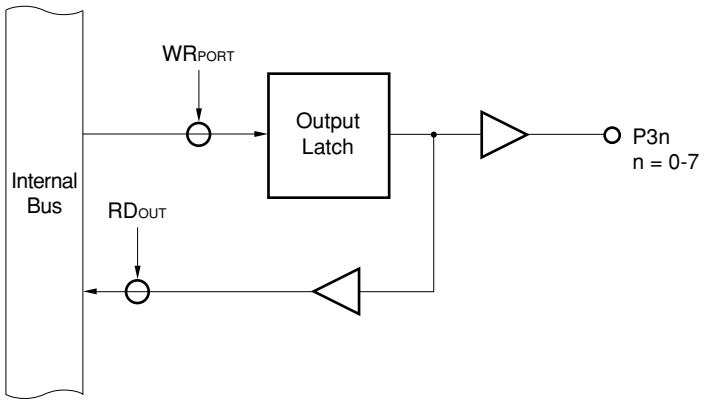
Port 3 is an input/output port, with a dual function as various control pins.

(1) When set as an output port

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

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

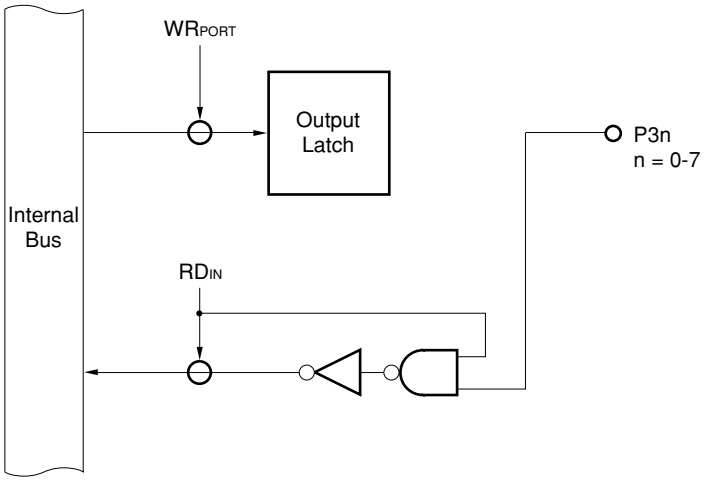
Figure 5-25. Port Specified as Output Port



(2) When set as an input port

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

Figure 5-26. Port Specified as Input Port



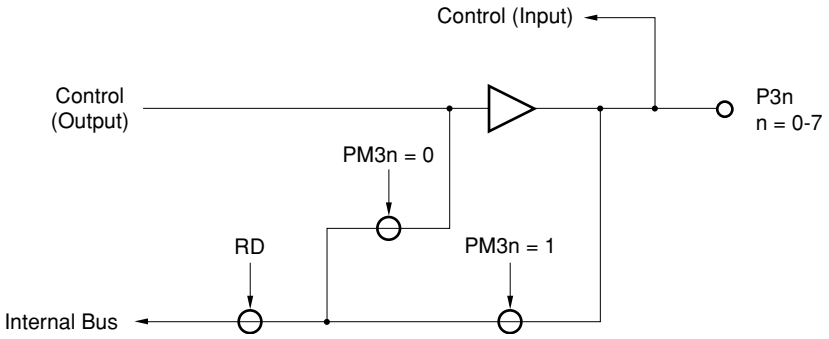
Caution A bit manipulation instruction manipulates one bit as the result, but accesses the port in 8-bit units. Therefore, if a bit manipulation instruction is used on a port with a mixture of input and output pins or port mode and control mode, the contents of the output latch of pins specified as inputs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

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

(3) When specified as control signal input/output

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

Figure 5-27. Control Specification



(a) When port is control signal output

When PM3n (n = 0 to 7) bits of the port 3 mode register (PM3) is set (1), the control signal pin level can be read by executing a port read instruction.

When PM3n bit is reset (0), the μPD784054 internal control signal status can be read by executing a port read instruction.

(b) When port is control signal input

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

Caution Pins that function as input pins in the control mode may malfunction if the corresponding bits of the port 3 mode control register (PMC3) are rewritten while the pins are operating. Therefore, write PMC3 on initializing the system.

5.6 Port 4

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

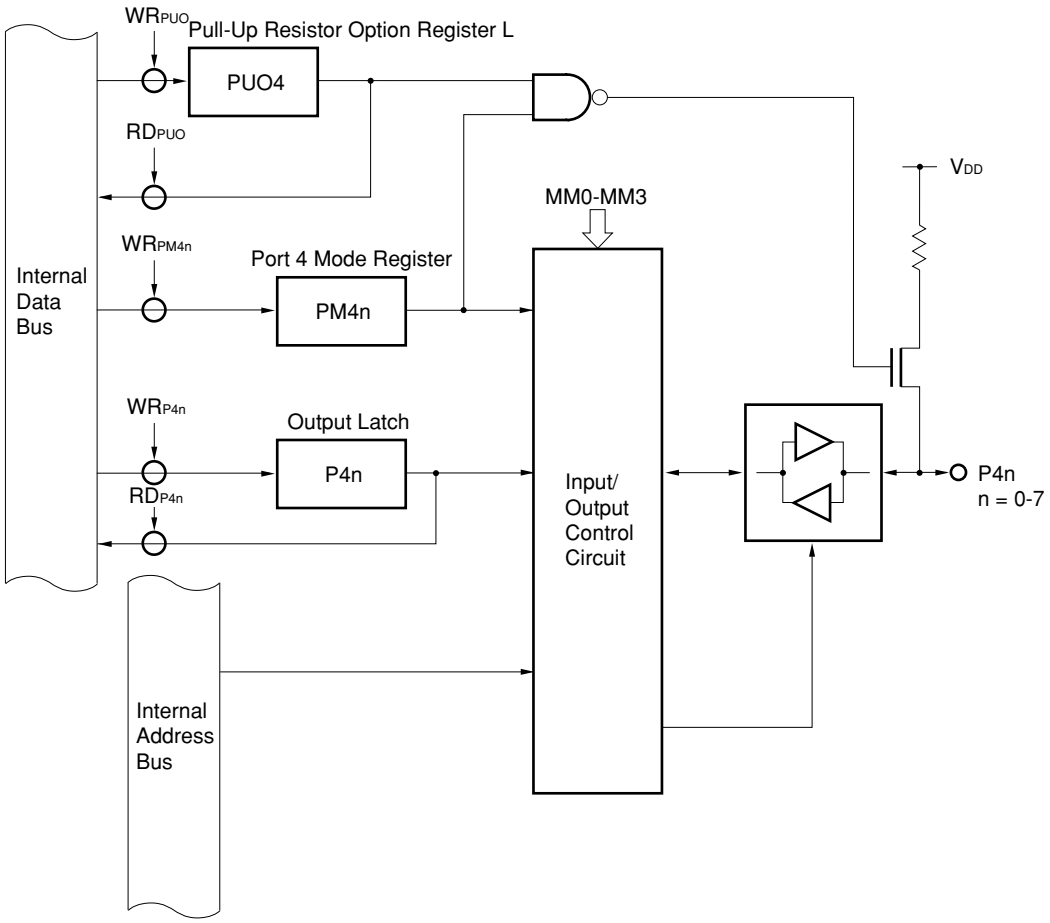
In addition to its function as input/output port, port 4 also functions as the low-order multiplexed address/data bus (AD0 to AD7) when external memory or I/Os are extended.

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

5.6.1 Hardware configuration

The port 4 hardware configuration is shown in Figure 5-28.

Figure 5-28. Block Diagram of Port 4



5.6.2 Input/output mode/control mode setting

The port 4 input/output mode is set for each pin by means of the port 4 mode register (PM4) as shown in Figure 5-29.

When port 4 is used as the address/data bus, it is set by means of the memory extension mode register (MM: Refer to **Figure 15-1**) as shown in Table 5-4.

Figure 5-29. Format of Port 4 Mode Register (PM4)

Address : 0FF24H On reset : FFH R/W

	7	6	5	4	3	2	1	0
PM4	PM47	PM46	PM45	PM44	PM43	PM42	PM41	PM40

PM4n	Specifies I/O Mode of P4n Pin (n = 0 to 7)
0	Output mode (output buffer ON)
1	Input mode (output buffer OFF)

Table 5-4. Operation Mode of Port 4

Bits of MM				Operation Mode	Remark
MM3	MM2	MM1	MM0		
0	0	0	0	Port (P40-P47)	—
0	0	1	1	Address/data bus (AD0-AD7)	Setting prohibited when external 16-bit bus specified
0	1	0	0		
0	1	0	1		
0	1	1	0		
0	1	1	1		—
1	0	0	0		
1	0	0	1		

5.6.3 Operating status

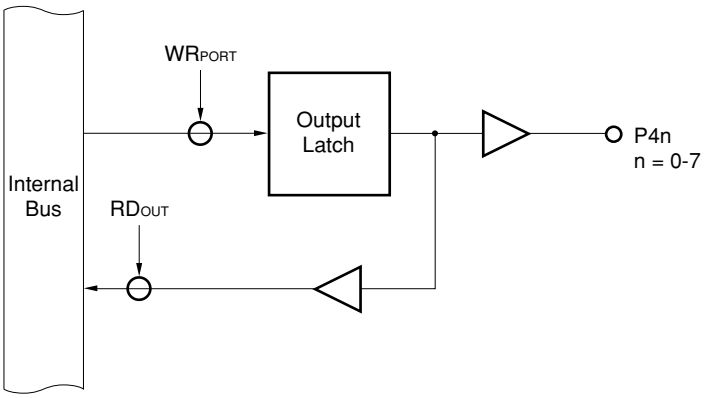
Port 4 is an input/output port, with a dual function as the address/data bus (AD0 to AD7).

(1) When set as an output port

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

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

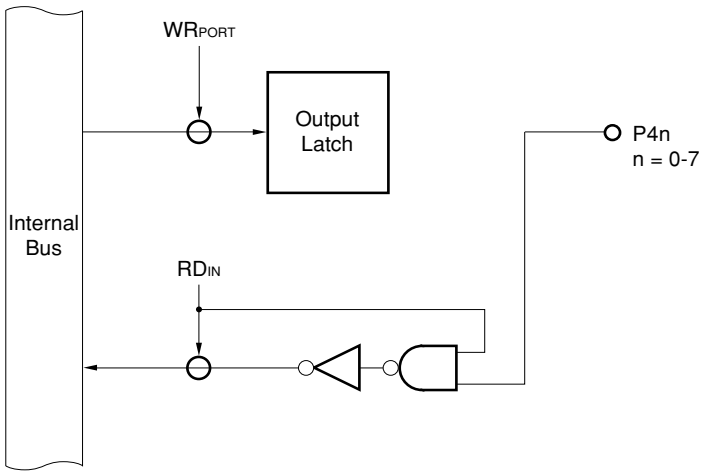
Figure 5-30. Port Specified as Output Port



(2) When set as an input port

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

Figure 5-31. Port Specified as Input Port



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

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

Used automatically when an external access is performed.
Input/output instructions should not be executed on port 4.

5.6.4 Internal pull-up resistors

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

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

When the PUO4 bit is 1, the internal pull-up resistor of only the pin set in the input mode by the memory expansion mode register (MM) and PM4 is valid.

Figure 5-32. Format of Pull-up Resistor Option Register L (PUOL)

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

	7	6	5	4	3	2	1	0
PUOL	0	PUO6	PUO5	PUO4	0	0	0	PUO0

PUO6	Specifies Pull-up Resistor of Port 6 (refer to **Figure 5-44**).

PUO5	Specifies Pull-up Resistor of Port 5 (refer to **Figure 5-38**).

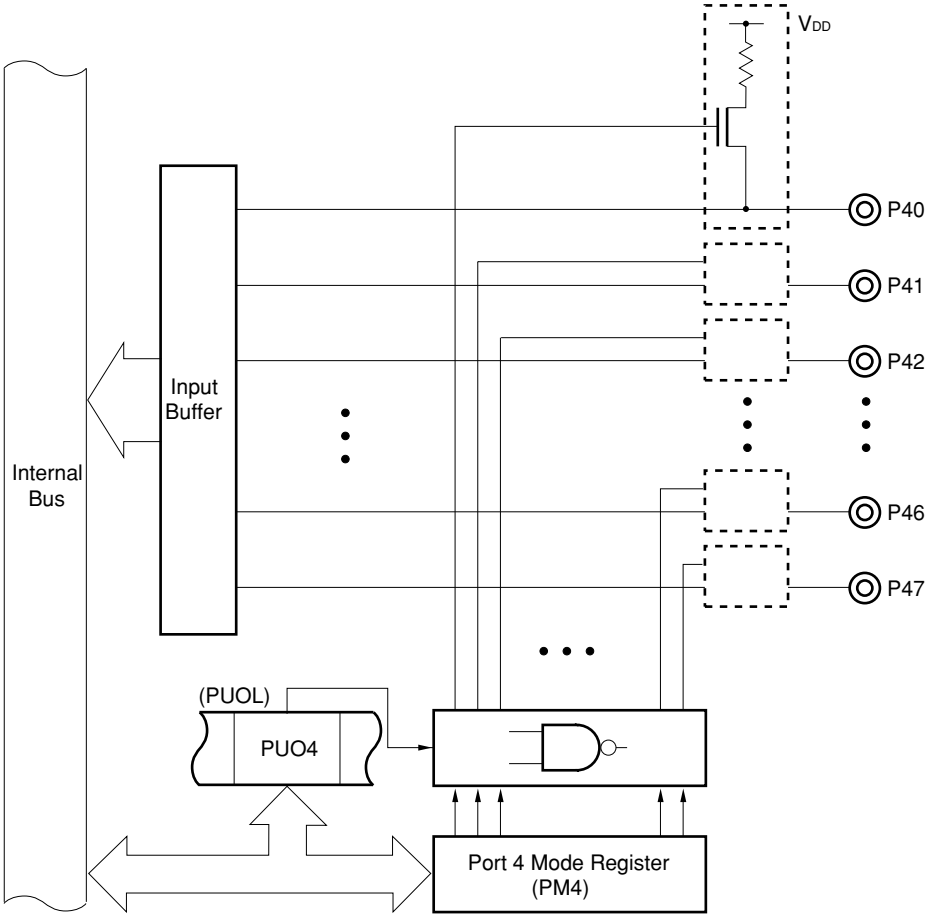
PUO4	Specifies Pull-up Resistor of Port 4.
0	Not used with port 4
1	Used with port 4

PUO0	Specifies Pull-up Resistor of Port 0 (refer to **Figure 5-6**).

Caution When using port 4 as the address/data bus, be sure to reset the PUO4 bit to "0" to not connect the internal pull-up resistor.

Remark When STOP mode is entered, setting 00H in PUOL is effective in reducing the current consumption.

Figure 5-33. Pull-Up Resistor Specification (Port 4)



5.7 Port 5

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

In addition to as an I/O port, port 5 also functions as follows when an external memory or I/O is connected:

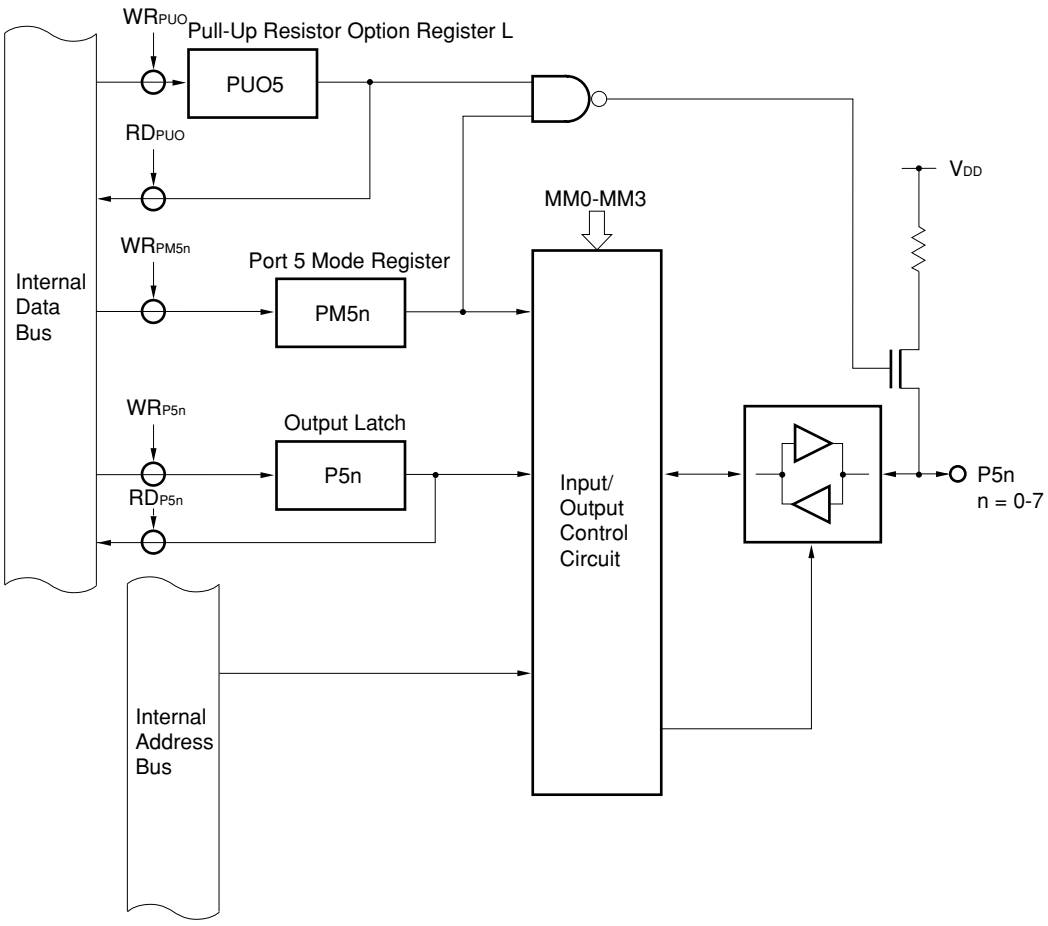
- When external 8-bit bus is specified
  As the high-order address bus (AD8 to AD15)
- When external 16-bit bus is specified
  As the high-order multiplexed address/data bus (AD8 to AD15)

When $\overline{\text{RESET}}$ is input, this port is set in the input mode (output high-impedance status), and the contents of the output latch are undefined.

5.7.1 Hardware configuration

The port 5 hardware configuration is shown in Figure 5-34.

Figure 5-34. Block Diagram of Port 5



5.7.2 Input/output mode/control mode setting

The port 5 input/output mode is set for each pin by means of the port 5 mode register (PM5) as shown in Figure 5-35.

When port 5 pins can be used as port or address pins in 2-bit units, the setting is performed by means of the memory extension mode register (MM: Refer to **Figure 15-1**) as shown in Table 5-5.

Figure 5-35. Format of Port 5 Mode Register (PM5)

Address : 0FF25H On reset : FFH R/W

	7	6	5	4	3	2	1	0
PM5	PM57	PM56	PM55	PM54	PM53	PM52	PM51	PM50

PM5n	Specifies I/O Mode of P5n Pin (n = 0 to 7)
0	Output mode (output buffer ON)
1	Input mode (output buffer OFF)

Table 5-5. Operation Mode of Port 5

Bits of MM				Operation mode								Remark
MM3	MM2	MM1	MM0	P50	P51	P52	P53	P54	P55	P56	P57	
0	0	0	0	Port (P50-P57)								—
0	0	1	1									Setting prohibited when external 16-bit bus specified. AD8-AD13 used as address bus
0	1	0	0	AD8	AD9	Port						
0	1	0	1	AD8	AD9	AD10	AD11	Port				
0	1	1	0	AD8	AD9	AD10	AD11	AD12	AD13	Port		
0	1	1	1	AD8	AD9	AD10	AD11	AD12	AD13	AD14	AD15	—
1	0	0	0									
1	0	0	1									

5.7.3 Operating status

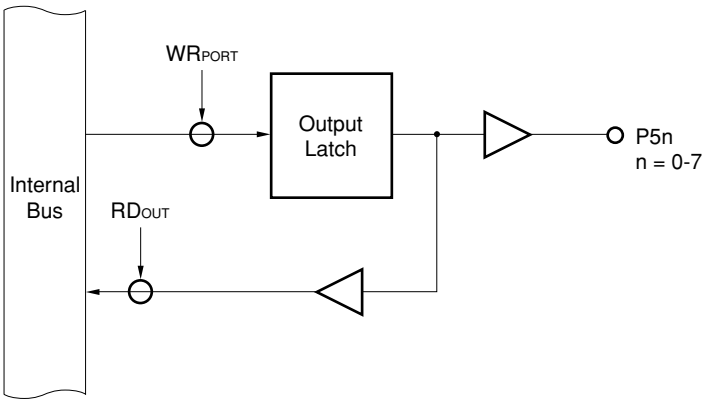
Port 5 is an input/output port, with a dual function as the address/data bus (AD8 to AD15).

(1) When set as an output port

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

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

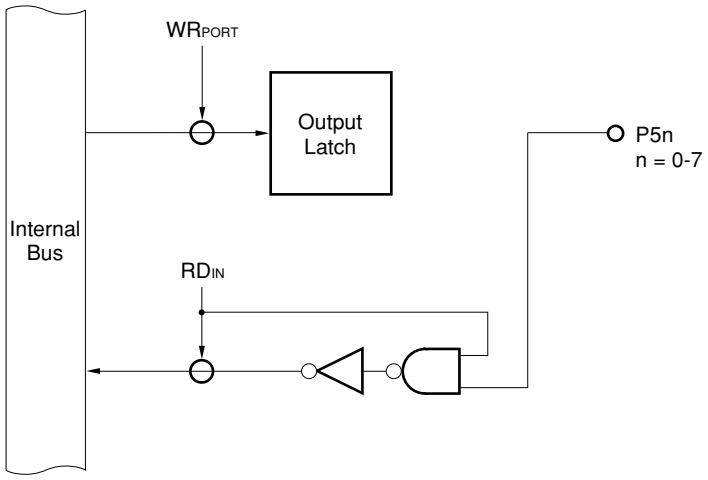
Figure 5-36. Port Specified as Output Port



(2) When set as an input port

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

Figure 5-37. Port Specified as Input Port



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

(3) When used as address/data bus (AD8 to AD15)

Port 5 is automatically used when an external address/data bus is accessed.
At this time, do not execute an I/O instruction to port 5.

5.7.4 Internal pull-up resistors

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

Whether or not an internal pull-up resistor is to be used can be specified for each pin by means of the PUO5 bit of the pull-up resistor option register L (PUOL) and the port 5 mode register (PM5).

When PUO5 bit is 1, the internal pull-up resistor of only the pin set in the input port by the memory expansion mode register (MM) and PM5 is valid.

Figure 5-38. Format of Pull-Up Resistor Option Register L (PUOL)

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

	7	6	5	4	3	2	1	0
PUOL	0	PUO6	PUO5	PUO4	0	0	0	PUO0

PUO6	Specifies Pull-up Resistor of Port 6 (refer to **Figure 5-44**).

PUO5	Specifies Pull-up Resistor of Port 5.
0	Not used with port 5
1	Used with port 5

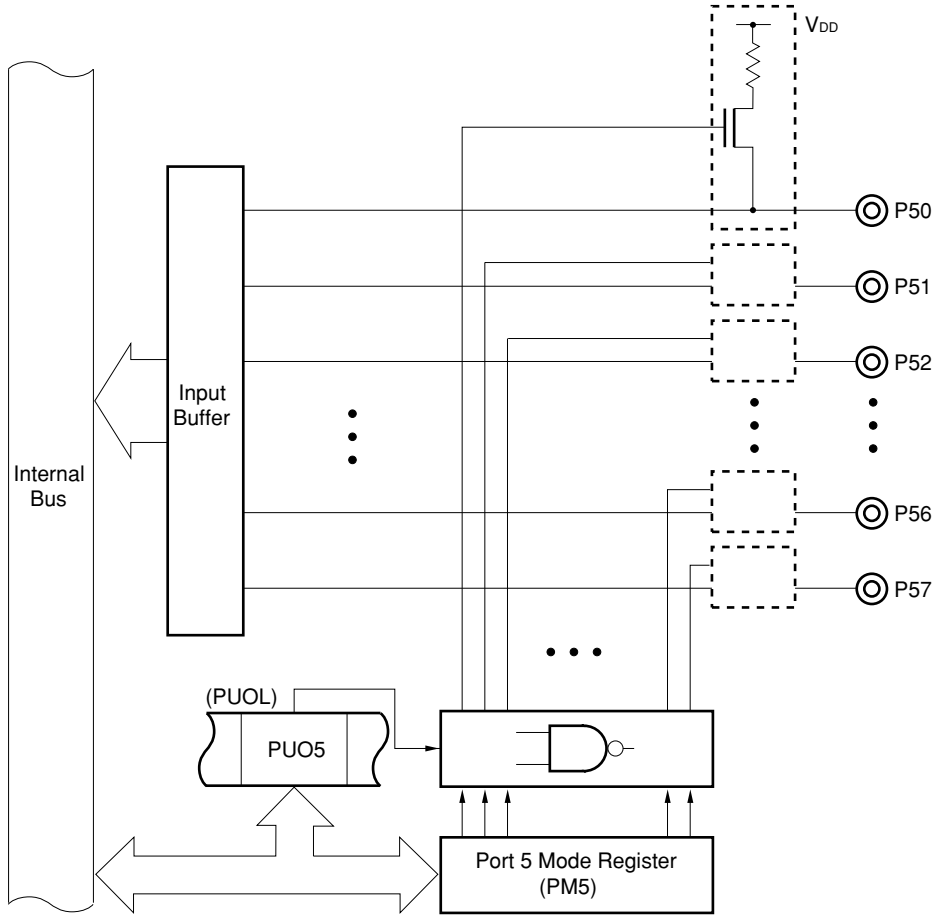
PUO4	Specifies Pull-up Resistor of Port 4 (refer to **Figure 5-32**).

PUO0	Specifies Pull-up Resistor of Port 0 (refer to **Figure 5-6**).

Caution When port 5 is used as the address/data bus, and "0" must be set in PUO5 bit so that internal pull-up resistor connection is not performed.

Remark When STOP mode is entered, setting 00H in PUOL is effective in reducing the current consumption.

Figure 5-39. Pull-Up Resistor Specification (Port 5)



5.8 Port 6

Port 6 is a 4-bit I/O port with an output latch. This port can be set in the input or output mode in 1-bit units by using port 6 mode register (PM6). Each pin is provided with a software programmable pull-up resistor.

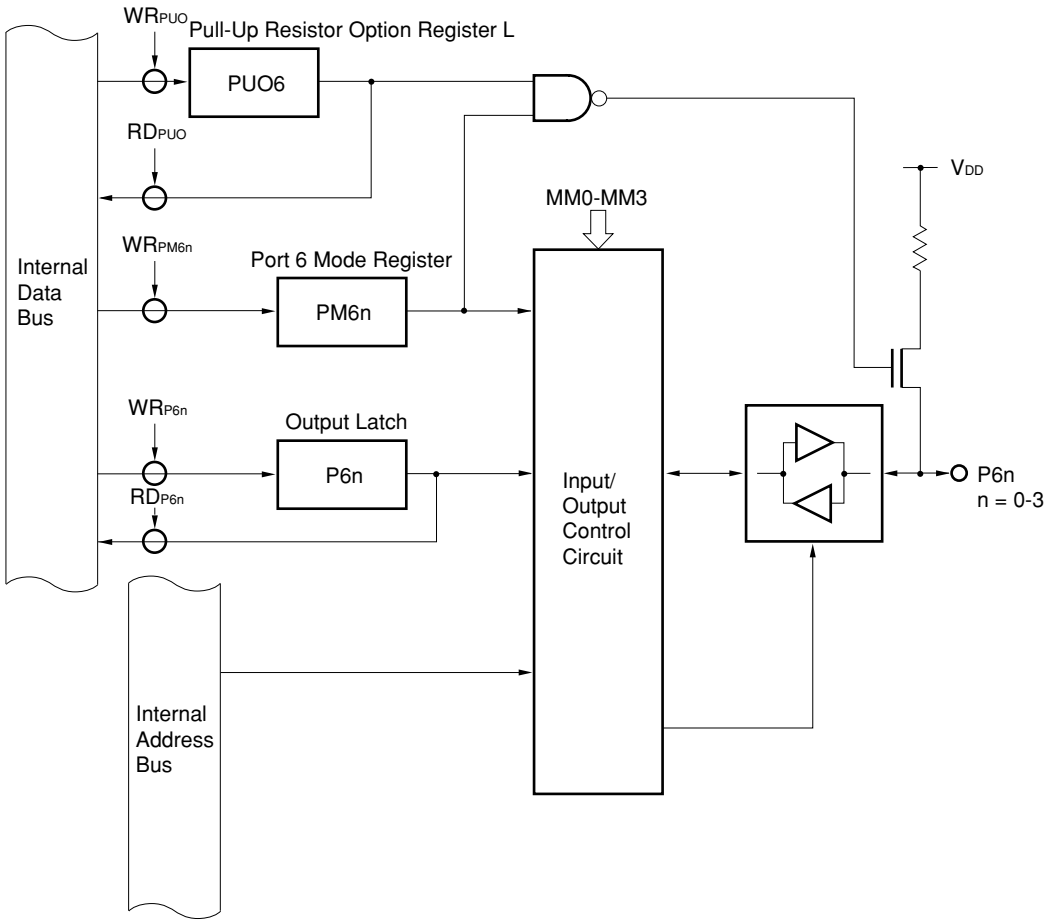
In addition to as an I/O port, this port also functions as the high-order address bus (A16 to A19) if so specified when an external memory or I/O is connected.

When $\overline{\text{RESET}}$ is input, this port is set in the input mode (output high-impedance status), and the contents of the output latch are undefined.

5.8.1 Hardware configuration

The port 6 hardware configuration is shown in Figures 5-40.

Figure 5-40. Block Diagram of Port 6



5.8.2 Setting of I/O mode/control mode

The input/output mode of port 6 is set in 1-bit units by using the port 6 mode register (PM6) as shown in Figure 5-41.

Port 6 can be used as port pins or address pins in 2-bit units. Whether it is used as port pins or address pins is specified by using the memory extension mode register (MM: refer to **Figure 15-1**), as shown in Table 5-6.

Figure 5-41. Format of Port 6 Mode Register (PM6)

Address : 0FF26H On reset : FFH R/W

	7	6	5	4	3	2	1	0
PM6	1	1	1	1	PM63	PM62	PM61	PM60

PM6n	Specifies I/O Mode of P6n Pin (n = 0 to 3)
0	Output mode (output buffer ON)
1	Input mode (output buffer OFF)

Table 5-6. Operation Mode of Port 6

Bits of MM				Operation mode				Remark
MM3	MM2	MM1	MM0	P60	P61	P62	P63	
0	0	0	0	Port (P60-P63)				—
0	0	1	1					Setting prohibited when external 16-bit bus specified.
0	1	0	0					
0	1	0	1					
0	1	1	0					
0	1	1	1					—
1	0	0	0	A16	A17	Port		
1	0	0	1	A16	A17	A18	A19	

5.8.3 Operating status

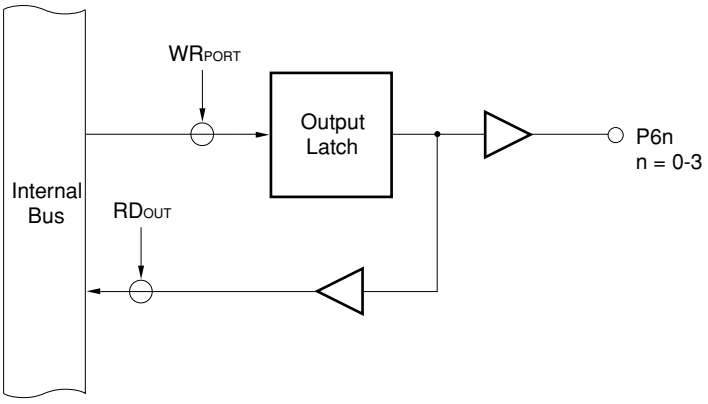
Port 6 is an input/output port, with a dual function as the address bus (A16 to A19).

(1) When set as an output port

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

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

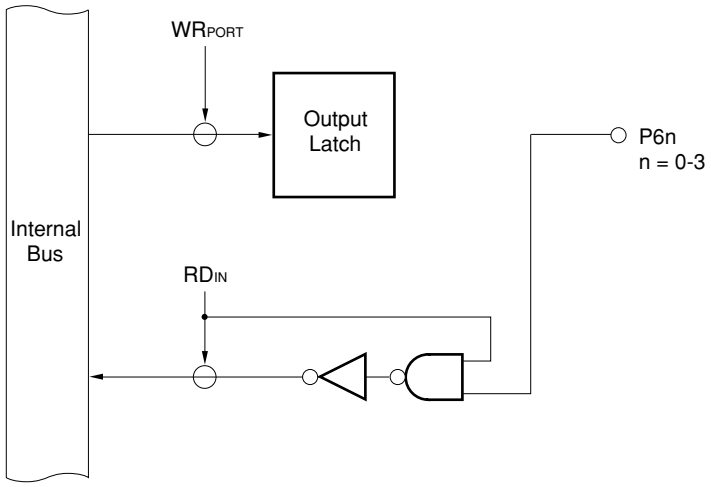
Figure 5-42. Port Specified as Output Port



(2) When set as an input port

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

Figure 5-43. Port Specified as Input Port



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

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

(3) When used as address bus (A16 to A19)

Port 6 is automatically used for external access.

At this time, do not execute an I/O instruction to port 6.

5.8.4 Internal pull-up resistors

Port 6 incorporates pull-up resistors. Use of these internal resistors when pull-up is necessary enables the number of parts and the mounting area to be reduced.

Whether or not an internal pull-up resistor is to be used can be specified for each pin by means of the PUO6 bit of the pull-up resistor option register L (PUOL) and the port 6 mode register (PM6).

The internal pull-up resistor of only the pin set in the input mode by PM6 is valid when the PUO6 bit is 1.

Even when port 6 is specified as the address bus, specifying the use of the internal pull-up resistor is valid. To not connect the internal pull-up resistor, either set the output mode by using the port 6 mode register (PM6) (PM6n = 0: n = 0 to 3), or reset PUO6 to 0.

Figure 5-44. Format of Pull-Up Resistor Option Register L (PUOL)

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

	7	6	5	4	3	2	1	0
PUOL	0	PUO6	PUO5	PUO4	0	0	0	PUO0

PUO6	Specifies Pull-up Resistor of Port 6
0	Not used with port 6
1	Used with port 6

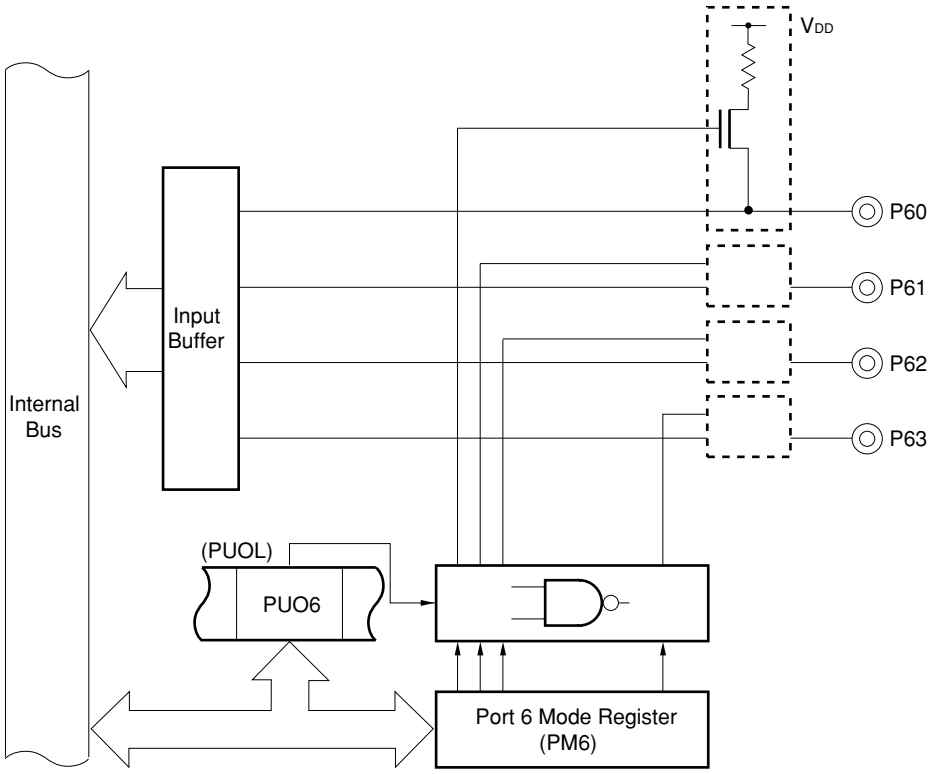
PUO5	Specifies Pull-up Resistor of Port 5 (refer to **Figure 5-38**).

PUO4	Specifies Pull-up Resistor of Port 4 (refer to **Figure 5-32**).

PUO0	Specifies Pull-up Resistor of Port 0 (refer to **Figure 5-6**).

Remark When STOP mode is entered, setting 00H in PUOL is effective in reducing the current consumption.

Figure 5-45. Pull-Up Resistor Specification (Port 6)



5.9 Port 7

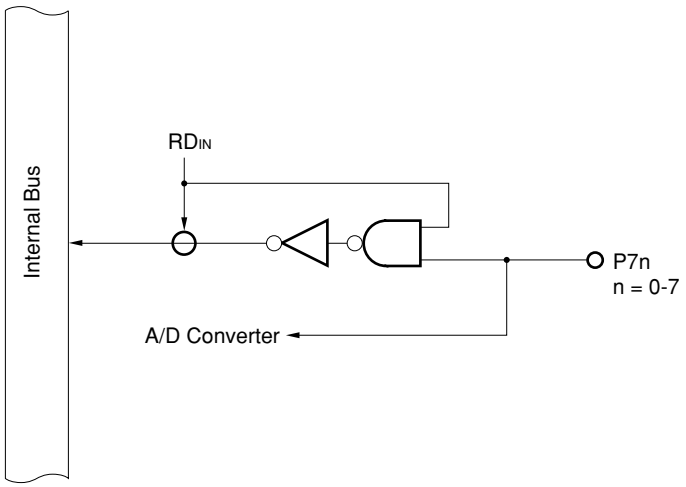
Port 7 is an 8-bit input port. In addition to functioning as input port pins, its pins also function as an A/D converter analog input (low-order 8 channels) pins (ANI0 to ANI7), and can always input analog signals. This port is set in the analog input mode by using A/D converter mode register (ADM) (refer to **Figure 11-3**).

The level of each pin of this port can always be read or tested, regardless of the multiplexed function.

5.9.1 Hardware configuration

Figure 5-46 shows the hardware configuration of port 7.

Figure 5-46. Block Diagram of Port 7



5.9.2 Notes

(1) Do not apply a voltage outside the range of AV_{SS} to AV_{REF} to the P70 to P77 pins when they are used as ANI0 to ANI7. For details, refer to **11.6 Cautions** in **CHAPTER 11 A/D CONVERTER**.

(2) If some pins of port 7 are used for analog input and the others are used for digital input, and if the digital input changes at analog input sampling timing, the A/D conversion accuracy is affected. When a high accuracy is necessary, do not use analog input and digital input simultaneously.

5.10 Port 8

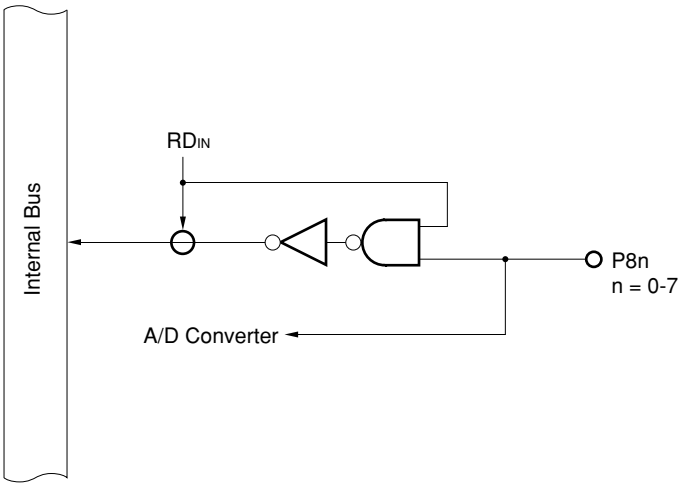
Port 8 is an 8-bit input port. In addition to functioning as input port pins, its pins also function as an A/D converter analog input (high-order 8 channels) pins (ANI8 to ANI15), and can always input analog signals. This port is set in the analog input mode by using A/D converter mode register (ADM) (refer to **Figure 11-3**).

The level of each pin of this port can always be read or tested, regardless of the multiplexed function.

5.10.1 Hardware configuration

Figure 5-47 shows the hardware configuration of port 8.

Figure 5-47. Block Diagram of Port 8



5.10.2 Cautions

(1) Do not apply a voltage outside the range of AV_{SS} to AV_{REF} to the P80 to P87 pins when they are used as ANI8 to ANI15. For details, refer to **11.6 Cautions** in **CHAPTER 11 A/D CONVERTER**.

(2) If some pins of port 8 are used for analog input and the others are used for digital input, and if the digital input changes at analog input sampling timing, the A/D conversion accuracy is affected. When a high accuracy is necessary, do not use analog input and digital input simultaneously.

5.11 Port 9

Port 9 is a 5-bit I/O port with an output latch. This port can be set in the input or output mode in 1-bit units by using port 9 mode register (PM9). Each pin is provided with a software programmable pull-up resistor.

In addition to the I/O port function, port 9 also functions as control signal pins (refer to **Table 5-7**). P90 to P93 functions as a read/write strobe signals and address strobe signal when an external memory or I/O is connected. P94 functions as a wait signal input pin if so specified by port 9 mode control register (PMC9).

When $\overline{\text{RESET}}$ is input, this port is set in the input mode (output high-impedance status), and the contents of the output latch are undefined.

Table 5-7. Operation Mode of Port 9

Pin Name	Port Mode	Control Signal I/O Mode	Manipulation to Use Port 9 as Control Pins
P90	I/O Port	$\overline{\text{RD}}$	Specifying external memory expansion mode by MM0 to MM3 bits of memory expansion mode register (MM)
P91		$\overline{\text{LWR}}$	
P92		$\overline{\text{HWR}}$	
P93		ASTB	
P94		$\overline{\text{WAIT}}$	Setting of PMC94 bit of PMC9 to 1

Remark For details, refer to **CHAPTER 15 LOCAL BUS INTERFACE FUNCTION**.

(a) Port mode

Each port pin not set in the control mode can be set in the input or output mode in 1-bit units by using the port 9 mode register (PM9).

(b) Control signal I/O mode

(i) $\overline{\text{RD}}$ (Read Strobe)

This pin outputs a strobe signal to read an external memory. The operation of this pin is specified by the memory expansion mode register (MM).

(ii) $\overline{\text{LWR}}$, $\overline{\text{HWR}}$ (Low/High Write Strobe)

These pins output strobe signals to write an external memory. The operations of these pins are specified by the memory expansion mode register (MM).

(iii) ASTB (Address Strobe)

This is a timing signal output pin to latch the address information output from the AD0 to AD15 pins to access the external memory. The operation of this pin is specified by the memory expansion mode register (MM).

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

This pin inputs a wait signal. The operation of this pin is specified by the port 9 mode control register (PMC9).

5.11.1 Hardware configuration

Figure 5-48 and Figure 5-49 show the hardware configuration of port 9.

Figure 5-48. Block Diagram of P90 to P93 (Port 9)

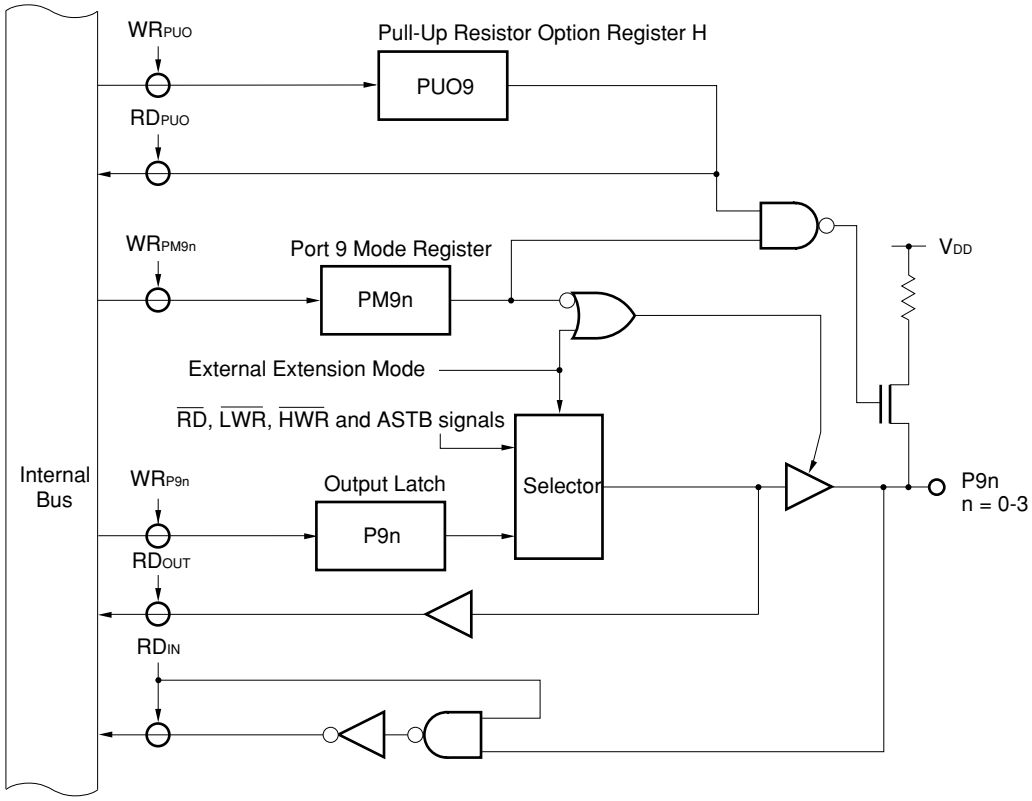
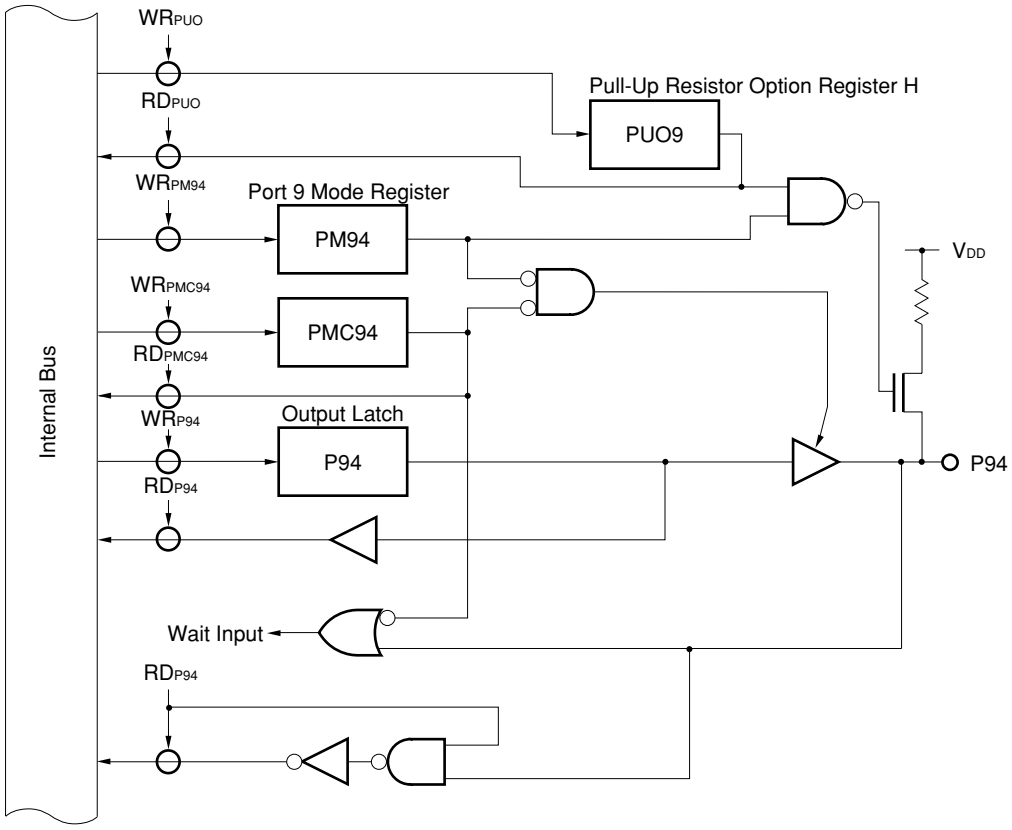


Figure 5-49. Block Diagram of P94 (Port 9)



5.11.2 Setting of I/O mode/control mode

The input/output mode of port 9 is set per pin by using the port 9 mode register (PM9) as shown in Figure 5-50.

In addition to as an I/O port function, port 9 also has the following functions. P90 to P93 can be used as $\overline{RD}$, $\overline{LWR}$, $\overline{HWR}$, and ASTB pins, if so specified by the memory extension mode register (MM: refer to **Figure 15-1**), as shown in Table 5-8. P94 can be used as a $\overline{WAIT}$ pin if so specified by the port 9 mode control register (PMC9) as shown in Figure 5-51.

Figure 5-50. Format of Port 9 Mode Register (PM9)

Address : 0FF29H On reset : FFH R/W

	7	6	5	4	3	2	1	0
PM9	1	1	1	PM94	PM93	PM92	PM91	PM90

PM9n	Specifies I/O Mode of P9n Pin (n = 0 to 4)
0	Output mode (output buffer ON)
1	Input mode (output buffer OFF)

Table 5-8. Operation Mode of P90 to P93

Bits of MM				Operation mode				Remark
MM3	MM2	MM1	MM0	P90	P91	P92	P93	
0	0	0	0	Port (P90-P93)				—
0	0	1	1	$\overline{RD}$	$\overline{LWR}$	$\overline{HWR}$	ASTB	Setting prohibited when external 16-bit bus specified.
0	1	0	0					
0	1	0	1					
0	1	1	0					
0	1	1	1					—
1	0	0	0					
1	0	0	1					

Figure 5-51. Format of Port 9 Mode Control Register (PMC9)

Address : 0FF49H On reset : 00H R/W

	7	6	5	4	3	2	1	0
PMC9	0	0	0	PMC94	0	0	0	0

PMC94	Specifies Control Mode of P94 Pin
0	I/O port mode
1	$\overline{WAIT}$ input mode

5.11.3 Operating status

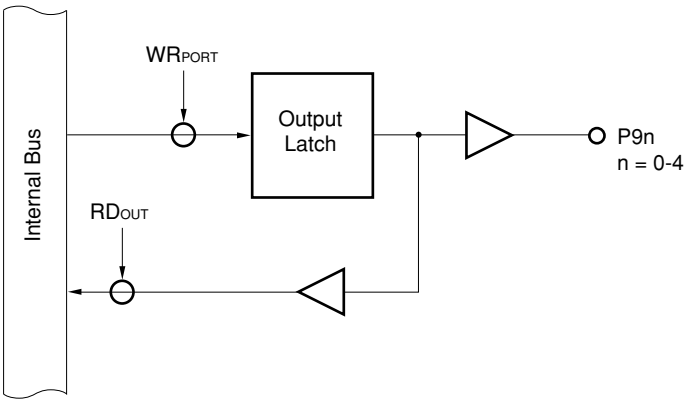
Port 9 is an input/output port and is multiplexed with control pins.

(1) In output port mode

The output latch is valid, and data is transferred between the output latch and accumulator by a transfer instruction. The contents of the output latch can be freely set by a logical operation instruction. Data that has been written to the output latch is retained until new data is written to the output latch[Note].

Note Including when the other bits of the same port are manipulated by a bit manipulation instruction.

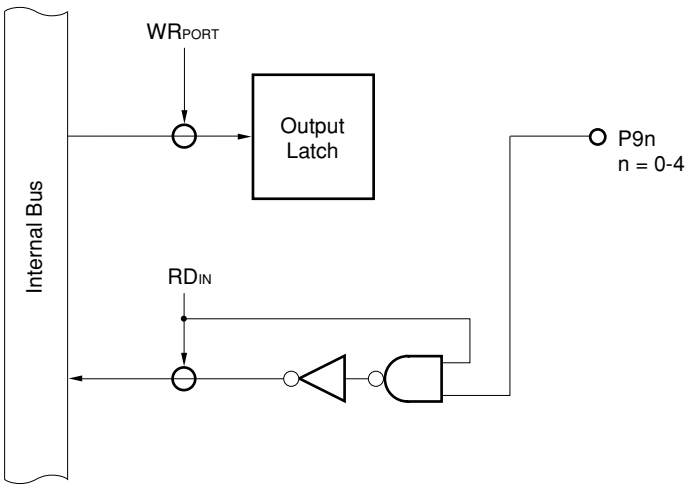
Figure 5-52. Port in Output Port Mode



(2) In input port mode

The level of a port pin can be loaded to the accumulator by using a transfer instruction. Even in this case, data can be written to the output latch. Data transferred from the accumulator by a transfer instruction is stored to all the latches regardless of whether the input or output mode is specified. However, because the output buffer of a bit (pin) set in the input mode is in the high-impedance state, its contents are not output to the port pin (the contents of the output latch are output to the port pin when the mode of the pin is changed from input to output). The contents of the output latch of the pin set in the input port cannot be loaded to the accumulator.

Figure 5-53. Port in Input Port Mode



Caution Although the result of a bit manipulation instruction is ultimately 1 bit manipulation, it accesses a port in 8-bit units. If such an instruction is executed to manipulate a port with some pins set in the input mode and the others in the control mode, the contents of the output latch are undefined (except when a pin is manipulated by the SET1 or CLR1 instruction). Especially, care must be exercised if the mode of some pins must be changed between input and output.
The same applies when manipulating the port by using the other 8-bit operation instructions.

(3) Pin in control mode

- **P90 to P93**
  These pins are automatically used as the $\overline{RD}$, $\overline{LWR}$, $\overline{HWR}$, and ASTB pins when the external memory or I/O is accessed.
  At this time, do not execute an I/O instruction to P90 to P93.

- **P94**
  This pin can be used as the $\overline{WAIT}$ pin, regardless of the setting of the port 9 mode register (PM9), if the PMC94 bit of the port 9 mode control register (PMC9) is set (1). When using P94 as the $\overline{WAIT}$ pin, the status of the $\overline{WAIT}$ pin can be read by executing an instruction that reads the port only when the PM94 bit of PM9 is set (1).

Caution The pin that functions as an input pin in the control mode (P94) may malfunction if the PMC94 bit of the port 9 mode control register (PMC9) is rewritten while the pin is operating. Therefore, write PMC9 on initializing the system.

5.11.4 Internal pull-up resistor

Port 9 is provided with pull-up resistors. When the port must be pulled up, the number of components and the mounting area can be reduced by using these internal pull-up resistor.

Whether the internal pull-up resistors are used or not is specified per pin by using the PUO9 bit of the pull-up resistor option register H (PUOH) and port 9 mode register (PM9).

When the PUO9 bit is 1, the internal pull-up resistor of only the pin specified as follows is valid.

- P90 to P93 : Set in input port mode by memory extension mode register (MM) and PM9
- P94 : Set in input mode by PM9

Even when P94 is specified as the $\overline{WAIT}$ pin, specifying its use as a pull-up resistor is valid. In order not to connect the internal pull-up resistor, either specify the output mode by using PM9 (PM94 = 0), or reset (0) in PUO9.

Figure 5-54. Format of Pull-up Resistor Option register H (PUOH)

Address : 0FF4FH On reset : 00H R/W

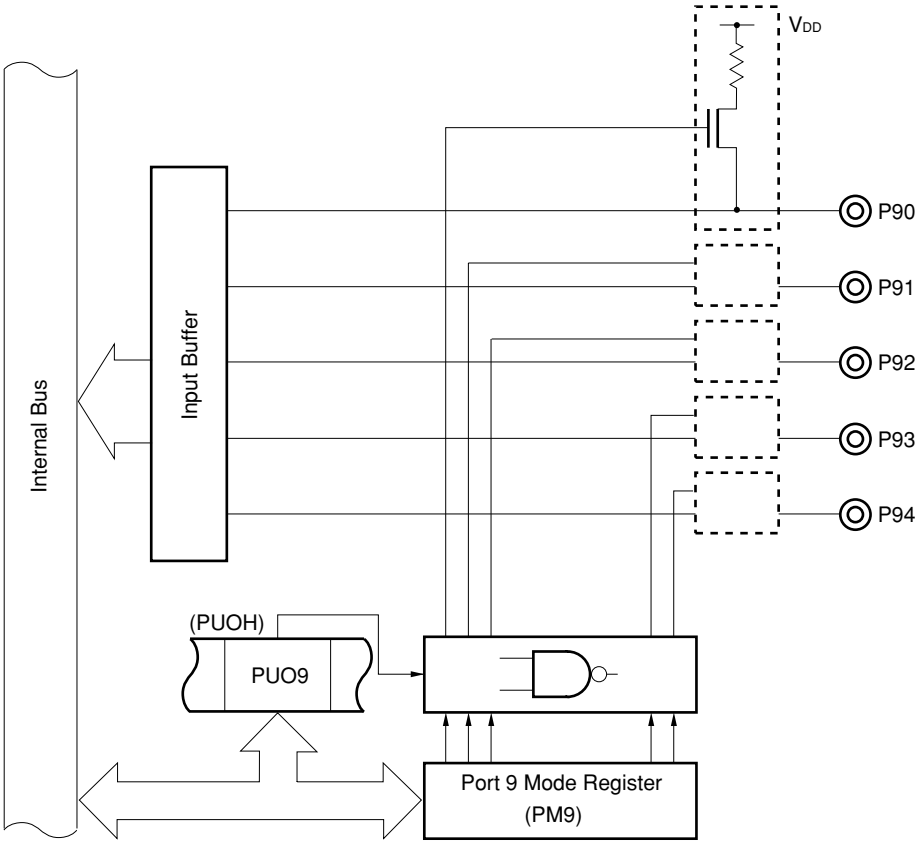
	7	6	5	4	3	2	1	0
PUOH	0	0	0	0	0	0	PUO9	0

PUO9	Specifies Pull-up Resistor of Port 9
0	Not used with port 9
1	Used with port 9

Caution When using P90 to P93 as the $\overline{RD}$, $\overline{LWR}$, $\overline{HWR}$, and ASTB pins, be sure to reset the PUO9 bit to "0" in order not to connect the internal pull-up resistor.

Remark Resetting PUOH to 00H is effective for decreasing the current consumption when the STOP mode is set.

Figure 5-55. Specifying Pull-up Resistor (port 9)



5.12 Port Output Data Check Function

The μPD784054 has a function to read the status of a port pin even in the output mode, to improve the reliability of the system (pin access mode). Therefore, the output data and the actual pin status can be checked as necessary.

To read the pin status, set (1) bit 0 of the port read control register (PRDC), and then read the port.

When $\overline{\text{RESET}}$ is input, PRDC is reset to 00H.

Figure 5-56. Format of Port Read Control Register (PRDC)

Address : 0FF2FH On reset : 00H R/W

	7	6	5	4	3	2	1	0
PRDC	0	0	0	0	0	0	0	PRDC0

PRDC0	Specifies Operation Mode
0	Normal mode
1	Pin access mode

Example To check the output data of ports 0 (P0), 4 (P4), and 5 (P5) by using the pin access mode.

```
TEST:   DI              ; Disables interrupts
        MOV     A, #5AH ; Test data = 5AH
        MOV     P0, A   ; Sets 5AH to output latch
        MOV     P4, A
        MOV     P5, A
        SET1    PRDC.0  ; Sets pin access mode (sets PRDC)
        CMP     A, P0   ; Compares pin level and output latch contents
        BNE     $ERR0   ; Error if unmatch
        CMP     A, P4
        BNE     $ERR4
        CMP     A, P5
        BNE     $ERR5
        CLR1    PRDC.0  ; Returns to normal mode (resets PRDC)
        EI              ; Enables interrupts
```

Cautions 1. If a bit manipulation instruction is executed to manipulate the port, it is not executed normally in the pin access mode (PRDC0 = 1). After checking the port, be sure to reset the mode to the normal mode (PRDC0 = 0).

2. If an interrupt occurs in the pin access mode (PRDC0 = 1), a bit manipulation instruction may be executed with this mode maintained, causing malfunctioning. Be sure to set the DI status before checking the port.
Do not use a macro service that manipulates the port.

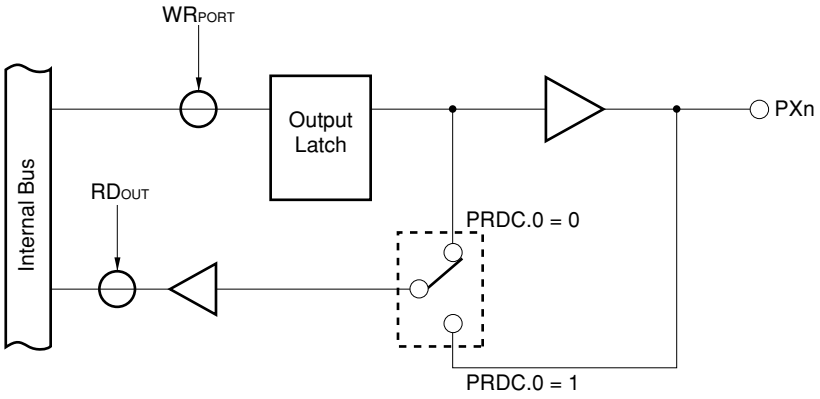
3. Occurrence of the non-maskable interrupt cannot be prevented. Take the following measures in the program, depending on the system:

- Do not manipulate the port in the non-maskable interrupt routine.
- Save the level of PRDC.0 at the beginning of the non-maskable interrupt routine, and restore it on returning execution from the interrupt routine.

If PRDC.0 is set (1), the switch enclosed by the dotted line in the figure below is connected to the pin, and the pin level is read. If a bit manipulation instruction is executed in this status, the pin level is read and the bit is manipulated, affecting the value of the output latch.

When PRDC.0 is reset (0), the normal operation is performed.

Figure 5-57. Concept of Control (in output port mode)



Dedicated instructions (CHKL, CHKLA) that are used to frequently check the port status are available. These instructions compare the pin status with the contents of the output latch (in port mode), or the pin status with the level of the internal control output signal (in control mode) through exclusive OR.

Example To check the pin status and the contents of the output latch using the CHKL or CHKLA instruction.

```
TEST:   SET1    P0.3        ; Sets bit 3 of port 0
        CHKL    P0          ; Checks port 0
        BNE     $ERR1       ; Branches to error processing (ERR1) if contents of output
                            ; latch do not match pin status
          .
          .
          .
ERR1:   CHKLA   P0          ; Checks defective bit
        BT      A.3, $BIT03 ; Bit 3?
        BT      A.2, $BIT02 ; Bit 2?
        BT      A.1, $BIT01 ; Bit 1?
        BR      $BIT00      ; Bit 0 is defective if all other bits are valid.
```

Cautions 1. Use the CHKL or CHKLA instruction when the PRDC0 bit of the port read control register (PRDC) is "0" (normal mode).

2. The result of comparison by the CHKL or CHKLA instruction always matches, regardless of whether the pin set in the input port mode is set in the port mode or control mode. Because the input level of the input-only pin is read when the CHKL or CHKLA instruction is executed, because this pin does not have an output latch. In other words, executing the CHKL or CHKLA instruction to the input-only pin is practically invalid, therefore, do not use the instruction to manipulate such a pin.

3. To check the output level of a port with some of its bits set in the control output mode and others in the port output mode, using the CHKL or CHKLA instruction, execute the instruction after changing the input/output mode of the control output pin to the input mode (the output level of the control output pin changes asynchronously and therefore cannot be checked by the CHKL or CHKLA instruction).

5.13 Cautions

(1) All the port pins go into a high-impedance state when the $\overline{\text{RESET}}$ signal is input (the internal pull-up resistor is also disconnected from the pin).
If it is necessary to prevent a pin from going into a high-impedance state during $\overline{\text{RESET}}$ input, use an external circuit.

(2) Bits 1, 3, and 7 of the pull-up resistor option register L (PUOL) that specifies connection of the internal pull-up resistor, and bits 0 and 2 to 7 of the pull-up resistor option register H (PUOH) are fixed to “0”. However, if “1” is written to these bits, 1 can be read with an in-circuit emulator.

(3) The contents of the output latch are not initialized by $\overline{\text{RESET}}$ input. To use a port as an output port, be sure to initialize the output latch before turning ON the output buffer. Unless the output buffer is initialized before the output buffer is turned ON, unexpected data is output to the output port.
In the same way, when using a port as control pins, be sure to initialize the internal peripheral hardware, and then set the port in the control mode.

(4) Although the result of a bit manipulation instruction is ultimately 1 bit manipulation, it accesses a port in 8-bit units. If such an instruction is executed to manipulate a port with some pins set in the input/output mode and the others in the port mode and in the control mode, the contents of the output latch are undefined (except when a pin is manipulated by the SET1 or CLR1 instruction). Especially, care must be exercised if the mode of some pins must be changed between input and output.
The same applies when manipulating the port by using the other 8-bit operation instructions.

(5) Even when using the P21 to P27 pins in the output port mode or timer output mode, INTPn (n = 0 to 6) interrupt occurs depending on the edge detection of the pin level. Mask the interrupt before using these pins.

(6) The pins used as input pins in the control mode (P32, P34, P35, P37, and P94) may malfunction if the corresponding bit of the port n mode control register (PMCn: n = 3, 9) while these pins are operating. Therefore, write PMCn on initializing the system.

(7) When using ports 4 and 5, and P90 to P93 as pins in the external memory extension mode, be sure to reset the corresponding bits of the pull-up resistor option registers (PUOL, PUOH) to “0”, in order not to connect the internal pull-up resistor.

(8) Do not apply a voltage outside the range of AV_{SS} to AV_{REF} to P70 to P77 and P80 to P87 used as ANI0 to ANI15. For details, refer to **11.6 Cautions** in **CHAPTER 11 A/D CONVERTER**.

(9) If some pins of ports 7 and 8 are used for analog input and the others are used for digital input, and if the digital input changes at analog input sampling timing, the A/D conversion accuracy is affected. When a high accuracy is necessary, do not use analog input and digital input simultaneously.

(10) A bit manipulation instruction executed to manipulate the port is not executed normally in the pin access mode (PRDC0 of port read control register (PRDC) = 1). After checking the port, be sure to reset the mode to the normal mode (PRDC0 = 0).

(11) If an interrupt occurs in the pin access mode (PRDC0 of PRDC = 1), a bit manipulation instruction may be executed with this mode maintained, causing malfunctioning. Be sure to set the DI status before checking the port. Do not use a macro service that manipulates the port.

(12) Occurrence of the non-maskable interrupt cannot be prevented in the pin access mode (PRDC0 of PRDC = 1). Take the following measures by using the program, depending on the system:

- Do not manipulate the port in the non-maskable interrupt routine.
- Save the level of PRDC.0 at the beginning of the non-maskable interrupt routine, and restore it on returning execution from the interrupt routine.

(13) Use the CHKL or CHKLA instruction when the PRDC0 bit of PRDC is “0” (normal mode).

(14) The result of comparison by the CHKL or CHKLA instruction always matches, regardless of whether the pin set in the input port mode is set in the port mode or control mode.
Because the input level of the input-only pin is read when the CHKL or CHKLA instruction is executed, because this pin does not have an output latch. In other words, executing the CHKL or CHKLA instruction to the input-only pin is practically invalid, and therefore, do not use the instruction to manipulate such a pin.

(15) To check the output level of a port with some of its bits set in the control output mode and others in the port output mode, by using the CHKL or CHKLA instruction, execute the instruction after changing the input/output mode of the control output pin to the input mode (the output level of the control output pin changes asynchronously and therefore cannot be checked by the CHKL or CHKLA instruction).

CHAPTER 6 OUTLINE OF TIMER

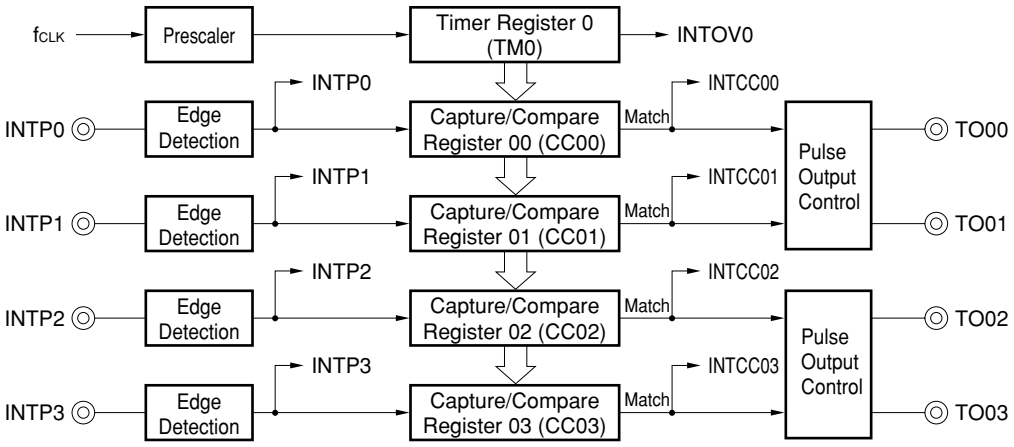
The μPD784054 incorporates three 16-bit time units.
These timer units can be used as eleven units of timers because the μPD784054 supports eleven interrupt requests.

Table 6-1. Operations of Timer

Item \ Name		Timer 0	Timer 1	Timer 4
Operation mode	Interval timer	4 ch	2 ch	2 ch
Function	Timer output	4 ch	2 ch	—
	Toggle output	○	○	—
	Set/reset output	○	○	—
	Overflow interrupt	○	○	○
	Number of interrupt requests	5	3	3

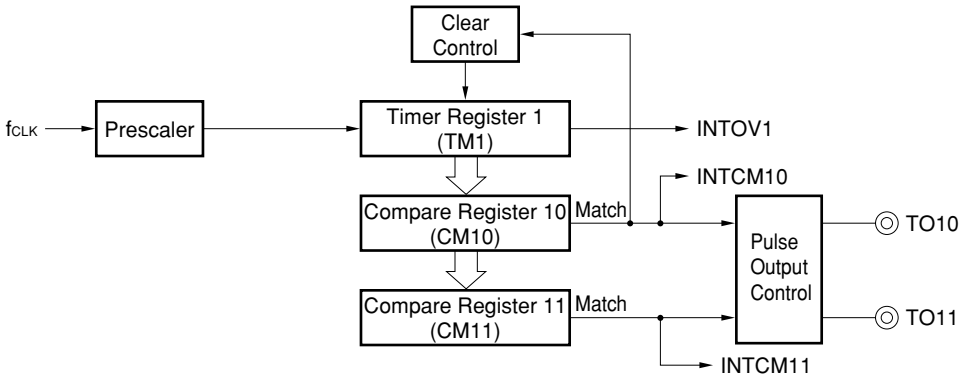
Figure 6-1. Block Diagram of Timer

Timer 0



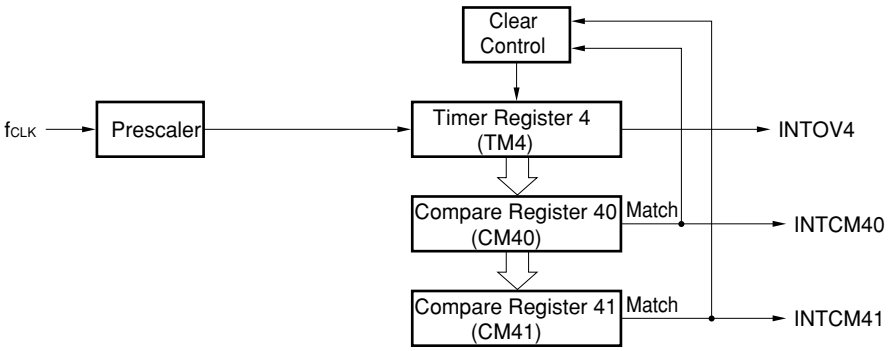
Prescaler: $f_{CLK}/4$, $f_{CLK}/8$, $f_{CLK}/16$, $f_{CLK}/32$, $f_{CLK}/64$

Timer 1



Prescaler: $f_{CLK}/8$, $f_{CLK}/16$, $f_{CLK}/32$, $f_{CLK}/64$, $f_{CLK}/128$

Timer 4



Prescaler: $f_{CLK}/4$, $f_{CLK}/8$, $f_{CLK}/16$, $f_{CLK}/32$, $f_{CLK}/64$

CHAPTER 7 TIMER 0

7.1 Function

Timer 0 is a 16-bit free running timer.

Because this timer has four capture/compare registers and a toggle and set/reset timer output functions, it can be used as an interval timer or to measure pulse width.

(1) Interval timer

When timer 0 is used as an interval timer, it generates an internal interrupt at interval set in advance.

Table 7-1. Interval Time of Timer 0

Minimum Interval Time[Note]	Maximum Interval Time	Resolution
$4/f_{CLK}$ (0.25 μs)	$2^{16} \times 4/f_{CLK}$ (16.4 ms)	$4/f_{CLK}$ (0.25 μs)
$8/f_{CLK}$ (0.5 μs)	$2^{16} \times 8/f_{CLK}$ (32.8 ms)	$8/f_{CLK}$ (0.5 μs)
$16/f_{CLK}$ (1.0 μs)	$2^{16} \times 16/f_{CLK}$ (65.5 ms)	$16/f_{CLK}$ (1.0 μs)
$32/f_{CLK}$ (2.0 μs)	$2^{16} \times 32/f_{CLK}$ (131 ms)	$32/f_{CLK}$ (2.0 μs)
$64/f_{CLK}$ (4.0 μs)	$2^{16} \times 64/f_{CLK}$ (262 ms)	$64/f_{CLK}$ (4.0 μs)

( ): at f_{CLK} = 16 MHz

Note The minimum interval time is limited by the data transfer processing time. Consider the interrupt processing time or macro service processing time used (refer to **Table 14-11 Interrupt Acceptance Processing Time** and **Table 14-12 Macro Service Processing Time**).

(2) Pulse width measurement

Timer 0 can be used to detect the pulse width of a signal input to an external interrupt request input pin (INTP0 to INTP3).

Table 7-2. Pulse Width Measurement Range of Timer 0

Measurable Pulse Width[Note 1]	Resolution
$4/f_{CLK}$ (0.25 μs)[Note 2] – $2^{16} \times 4/f_{CLK}$ (16.4 ms)	$4/f_{CLK}$ (0.25 μs)
$8/f_{CLK}$ (0.5 μs)[Note 2] – $2^{16} \times 8/f_{CLK}$ (32.8 ms)	$8/f_{CLK}$ (0.5 μs)
$16/f_{CLK}$ (1.0 μs)[Note 2] – $2^{16} \times 16/f_{CLK}$ (65.5 ms)	$16/f_{CLK}$ (1.0 μs)
$32/f_{CLK}$ (2.0 μs)[Note 2] – $2^{16} \times 32/f_{CLK}$ (131 ms)	$32/f_{CLK}$ (2.0 μs)
$64/f_{CLK}$ (4.0 μs)[Note 2] – $2^{16} \times 64/f_{CLK}$ (262 ms)	$64/f_{CLK}$ (4.0 μs)

( ): at f_{CLK} = 16 MHz

Notes 1. The minimum measurable pulse width changes depending on the sampling clock selected by the noise protection control register (NPC). The minimum measurable pulse width is either of the values in the above table and the table below, whichever greater.

Sampling Clock	Minimum Pulse Width
f_{CLK}	$4/f_{CLK}$ (0.25 μs)
$f_{CLK}/4$	$16/f_{CLK}$ (1.0 μs)

2. This value is limited by the data transfer processing time. Consider the interrupt processing time or macro service processing time used (refer to **Table 14-11 Interrupt Acceptance Processing Time** and **Table 14-12 Macro Service Processing Time**).

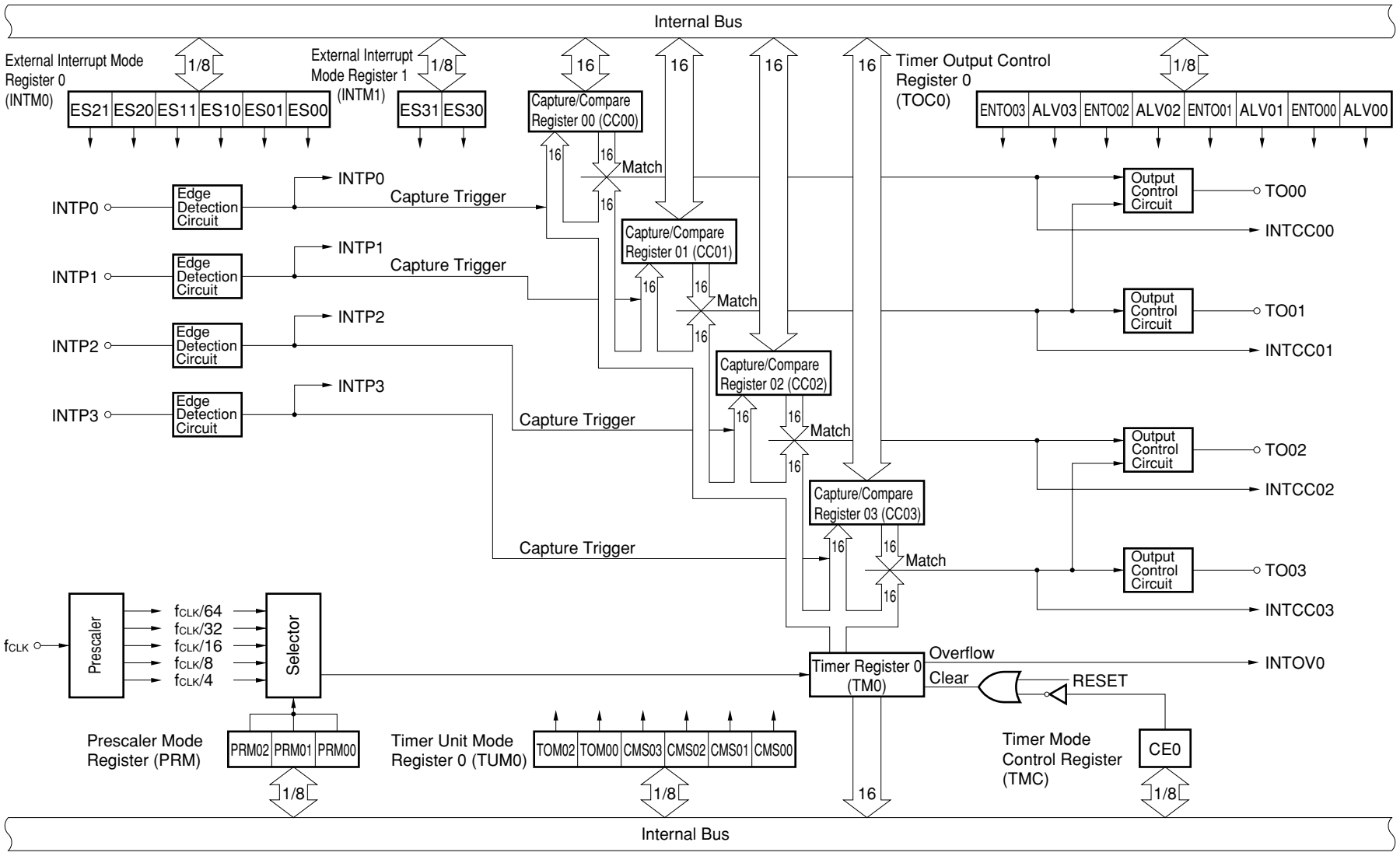
7.2 Configuration

Timer 0 consists of the following registers:

- Timer register (TM0) × 1
- Capture/compare register (CC0n) × 4 (n = 0 to 3)

Figure 7-1 shows the block diagram of timer 0.

Figure 7-1. Block Diagram of Timer 0



(1) Timer register 0 (TM0)

TM0 is a timer register that counts up the count clock specified by the prescaler mode register (PRM).

Counting of this timer register is enabled or disabled by the timer mode control register (TMC).

The timer register can be only read by using a 16-bit manipulation instruction. When $\overline{\text{RESET}}$ is input, TM0 is cleared to 0000H and stops counting.

(2) Capture/compare registers (CC00 to CC03)

CC0n (n = 0 to 3) is a 16-bit register that can be used as a compare register to detect match between its value and the count value of TM0 or as a capture register to capture the count value of TM0. Whether CC0n is used as a compare register or capture register is specified by the timer unit mode register 0 (TUM0).

This register can be read or written by using a 16-bit manipulation instruction.

When $\overline{\text{RESET}}$ is input, the value of this register is undefined.

When the CE0 bit of the timer mode control register (TMC) is 0 and timer 0 is stopped, the capture operation is not performed.

(a) As compare register

When used as a compare register, CC0n functions as a 16-bit register that holds the value determining the cycle of the interval timer operation.

When the contents of CC0n matches with the contents of TM0, an interrupt request (INTCC0n: n = 0 to 3) and a timer output control signal are generated.

(b) As capture register

When used as a capture register, CC0n functions as a 16-bit register that captures the contents of TM0 in synchronization with the valid edge (capture trigger) input from an external interrupt input pin (INTPn: n = 0 to 3).

The contents of CC0n are retained until the next capture trigger is generated.

(3) Edge detection circuit

The edge detection circuit detects the valid edge of an external input.

It detects the valid edge of the INTP0 to INTP3 pin inputs, and generates an external interrupt request (INTP0 to INTP3) and capture trigger. The valid edge is specified by the external interrupt mode registers (INTM0 and INTM1) (for the details of INTM0 and INTM1, refer to **Figures 13-1** and **13-2**).

(4) Output control circuit

When the contents of CC0n (n = 0 to 3) and the contents of TM0 match, the timer output can be inverted. A square wave can be output from a timer output pin (TO00 to TO03) if so specified by the timer output control register 0 (TOC0). The TO00 and TO02 pins can also output a set and reset signals if so specified by the timer unit mode register 0 (TUM0).

The timer output can be enabled or disabled by TOC0. When the timer output is disabled, a fixed level is output to the TO0n (n = 0 to 3) pin (the output level is fixed by TOC0).

(5) Prescaler

The prescaler generates a count clock by dividing the internal system clock. The clock generated by the prescaler is selected by the selector, and TM0 performs the count operation by using this clock as a count clock.

(6) Selector

The selector selects one of the five signals generated by dividing the internal system clock as the count clock of TM0.

7.3 Timer 0 Control Register

(1) Timer unit mode register 0 (TUM0)

TUM0 is a register that specifies the output mode of the timer output pins (TO00, TO02, and TO10) of timers 0 and 1, controls the clear operation of the timer register 1 (TM1), and specifies the operations of the capture/compare registers (CC00 to CC03) of timer 0.

This register can be read or written by using an 8-bit manipulation instruction or a bit manipulation instruction. Figure 7-2 shows the format of TUM0.

When $\overline{\text{RESET}}$ is input, the value of TUM0 is cleared to 00H.

Figure 7-2. Format of Timer Unit Mode Register 0 (TUM0)

Address : 0FF30H On reset : 00H R/W

	7	6	5	4	3	2	1	0
TUM0	TOM10	CLR1	TOM02	TOM00	CMS03	CMS02	CMS01	CMS00

TOM10	Specifies Output Mode of TO10 Pin (refer to **Figure 8-2**).

CLR1	Controls Clear Operation of TM1 (refer to **Figure 8-2**).

TOM0n	Specifies Output Mode of TO0n Pin (n = 0, 2)
0	Toggle output
1	Set/reset output

CMS0n	Specifies Operation of CC0n (n = 0 to 3)
0	Capture register
1	Compare register

(2) Timer mode control register (TMC)

TMC is a register that controls the count operation of timer registers 0 and 1 (TM0 and TM1).

This register can be read or written by using an 8-bit manipulation instruction or a bit manipulation instruction. Figure 7-3 shows the format of TMC.

When $\overline{\text{RESET}}$ is input, the value of this register is cleared to 00H.

Figure 7-3. Format of Timer Mode Control Register (TMC)

Address : 0FF31H On reset : 00H R/W

	⑦	6	5	4	③	2	1	0
TMC	CE1	0	0	0	CE0	0	0	0

CE1	Controls Count Operation of TM1 (refer to **Figure 8-3**).

CE0	Controls Count Operation of TM0
0	Clears and stops counting
1	Enables counting

(3) Timer output control register 0 (TOC0)

TOC0 is a register that specifies the operation and active level of the timer output pins (TO00 to TO03) of timer 0.

This register can be read or written by using an 8-bit manipulation instruction or a bit manipulation instruction. Figure 7-4 shows the format of TOC0.

When $\overline{\text{RESET}}$ is input, the value of this register is cleared to 00H.

Figure 7-4. Format of Timer Output Control Register 0 (TOC0)

Address : 0FF32H On reset : 00H R/W

	⑦	6	⑤	4	③	2	①	0
TOC0	ENTO03	ALV03	ENTO02	ALV02	ENTO01	ALV01	ENTO00	ALV00

ENTO0n	Specifies Operation of TO0n Pin (n = 0 to 3)
0	Outputs $\overline{\text{ALV0n}}$
1	Enables pulse output

ALV0n	Specifies Active Level of TO0n Pin (n = 0 to 3)
0	Low level
1	High level

(4) Prescaler mode register (PRM)

PRM is a register that specifies the count clock of timer registers 0 and 1 (TM0 and TM1).

This register can be read or written by using an 8-bit manipulation instruction. Figure 7-5 shows the format of PRM.

When $\overline{\text{RESET}}$ is input, the value of this register is cleared to 00H.

Figure 7-5. Format of Prescaler Mode Register (PRM)

Address : 0FF38H On reset : 00H R/W

	7	6	5	4	3	2	1	0
PRM	0	PRM12	PRM11	PRM10	0	PRM02	PRM01	PRM00

PRM12	PRM11	PRM10	Specifies Count Clock of TM1 (refer to **Figure 8-5**).

(f_{CLK} = 16 MHz)

PRM02	PRM01	PRM00	Specifies Count Clock of TM0.	
			Count Clock [Hz]	Resolution [μs]
0	0	0	f_{CLK}/4	0.25
0	0	1	f_{CLK}/8	0.5
0	1	0	f_{CLK}/16	1.0
0	1	1	f_{CLK}/32	2.0
1	0	0	f_{CLK}/64	4.0
Other			Setting prohibited	

Remark f_{CLK}: internal system clock

7.4 Operation of Timer Register 0 (TM0)

7.4.1 Basic operation

Timer 0 counts up by using the count clock specified by the prescaler mode register (PRM).

Counting is enabled or disabled by the CE0 bit of the timer mode control register (TMC). When the CE0 bit is set (1) by software, TM0 is set to 0001H at the first count clock, and starts counting up. When the CE0 bit is cleared (0) by software, TM0 is immediately cleared to 0000H, and stops the capture operation and generation of the match signal.

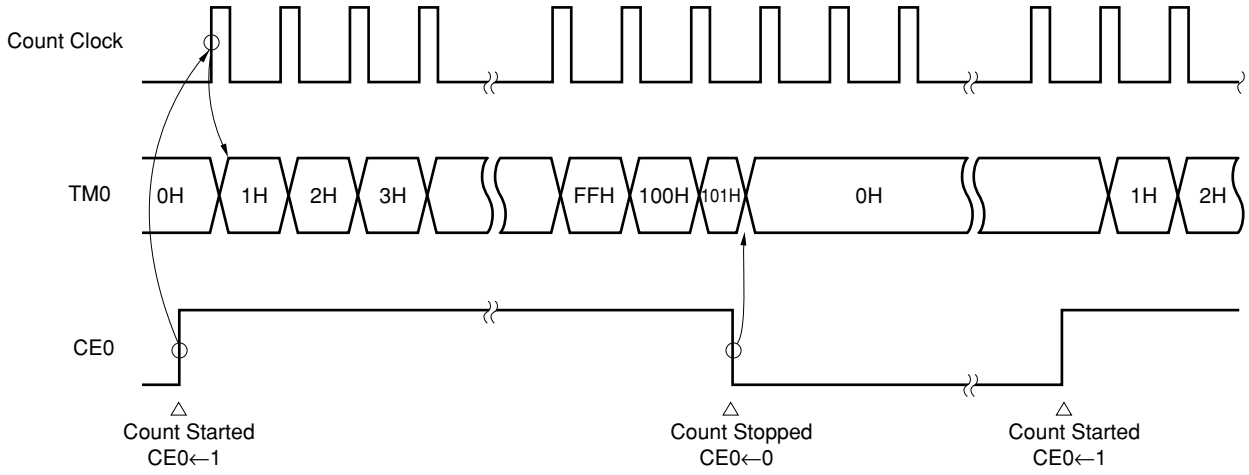
If the CE0 bit is set (1) while it has been already set (1), TM0 is not cleared but continues counting.

If a count clock is input when TM0 reaches FFFFH, TM0 is cleared to 0000H, and an overflow interrupt (INTOV0) occurs, but TM0 continues counting.

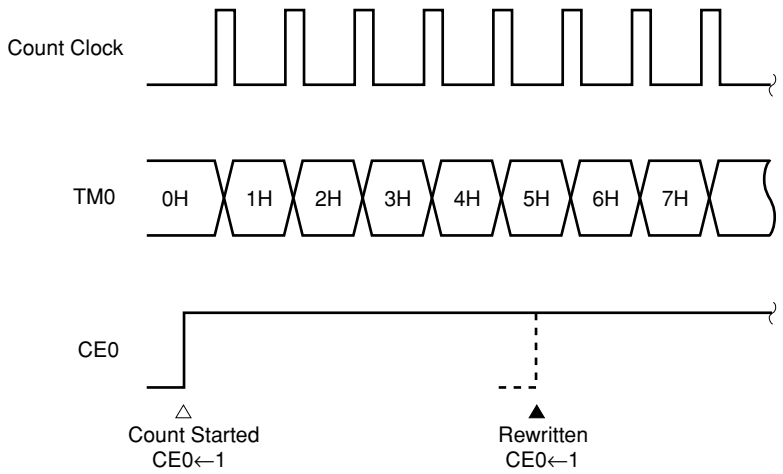
When $\overline{\text{RESET}}$ is input TM0 is cleared to 0000H and stops counting.

Figure 7-6. Basic Operation of Timer Register 0 (TM0)

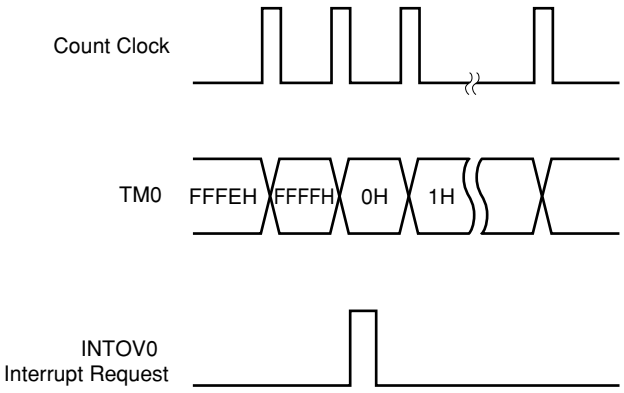
(a) Count started → count stopped → count started



(b) When "1" is written to the CE0 bit again after the count starts



(c) Operation when TM0 = FFFFH

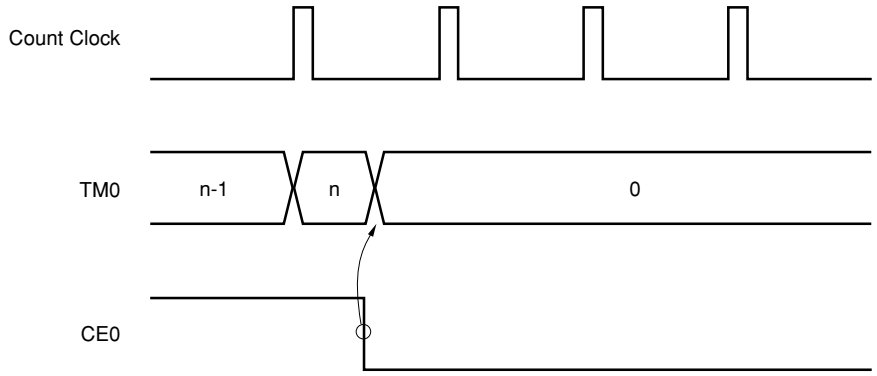


7.4.2 Clear operation

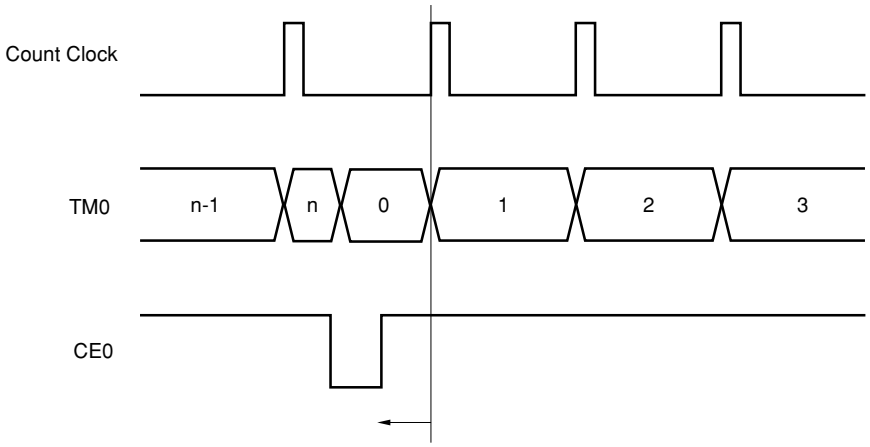
Timer register 0 (TM0) is cleared by clearing (0) the CE0 bit of the timer mode control register (TMC). TM0 is cleared as soon as the CE0 bit has been cleared (0).

Figure 7-7. Clear Operation of Timer Register 0 (TM0)

(a) Basic operation

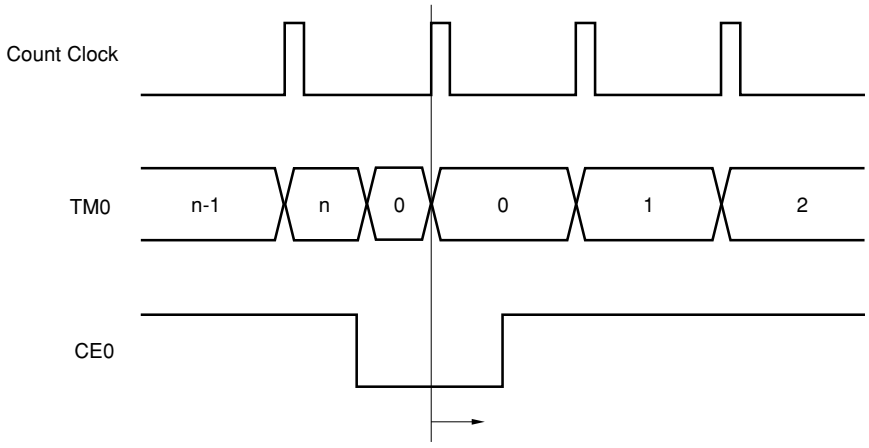


(b) Restart before count clock input after clearance



If the CE0 bit is set (1) before this count clock, the count starts from 1 on the count clock.

(c) Restart after count clock input after clearance



If the CE0 bit is set (1) from this count clock onward, the count starts from 1 on the count clock after the CE0 bit is set (1).

7.5 Operation of Capture/Compare Register

7.5.1 Compare operation

Timer 0 performs a compare operation by comparing the value set to a capture/compare register (CC00 to CC03) specified as a compare register with the count value of a timer register 0 (TM0).

If the count value of TM0 matches with the value set in advance to CC0n (n = 0 to 3) as a result of counting by TM0, the timer sends a match signal to the output control circuit, and at the same time, generates an interrupt request signal (INTCC0n: n = 0 to 3).

Table 7-3. Interrupt Request Signal from Compare Register (timer 0)

Compare Register	Interrupt Request Signal
CC00	INTCC00
CC01	INTCC01
CC02	INTCC02
CC03	INTCC03

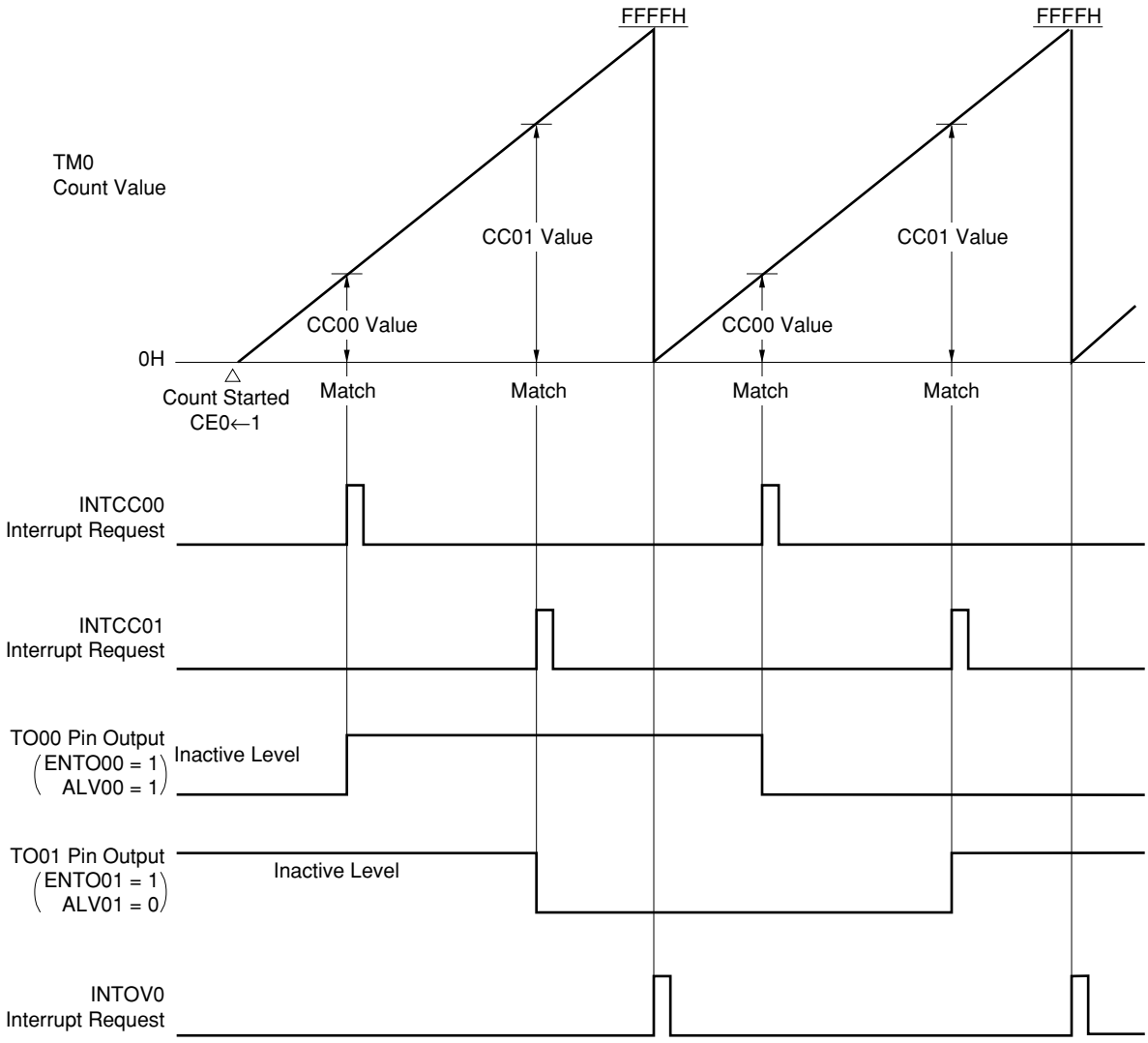
Remark CC00 to CC03 are capture/compare registers. Whether these registers are used as capture registers or compare registers is specified by the timer unit mode register 0 (TUM0).

Timer 0 has four timer output pins (TO00 to TO03). Table 7-4 shows the operation mode of each of these pins (for details, refer to **7.6 Basic Operation of Output Control Circuit**).

Table 7-4. Operation Mode of Timer Output Pin (timer 0)

Timer Output Pin	Output Operation Mode		Specification of Operation Mode
TO00	Toggle	Set/reset	TOM00 bit of TUM0
TO01	Toggle	—	—
TO02	Toggle	Set/reset	TOM02 bit of TUM0
TO03	Toggle	—	—

Figure 7-8. Compare Operation (timer 0)



7.5.2 Capture operation

In synchronization with an external trigger, timer 0 also performs a capture operation that captures and retains the count value of timer register 0 (TM0) to a capture register.

As an external trigger, the valid edge detected from an external interrupt request input pin (INTP0 to INTP3) is used (capture trigger). In synchronization with this capture trigger, the count value of TM0 is captured to a capture/compare register (CC0n: n = 0 to 3) specified as a capture operation in synchronization with INTPn (n = 0 to 3).

The contents of CC00 to CC03 are retained until the following capture triggers each corresponding to CC00 to CC03 are generated.

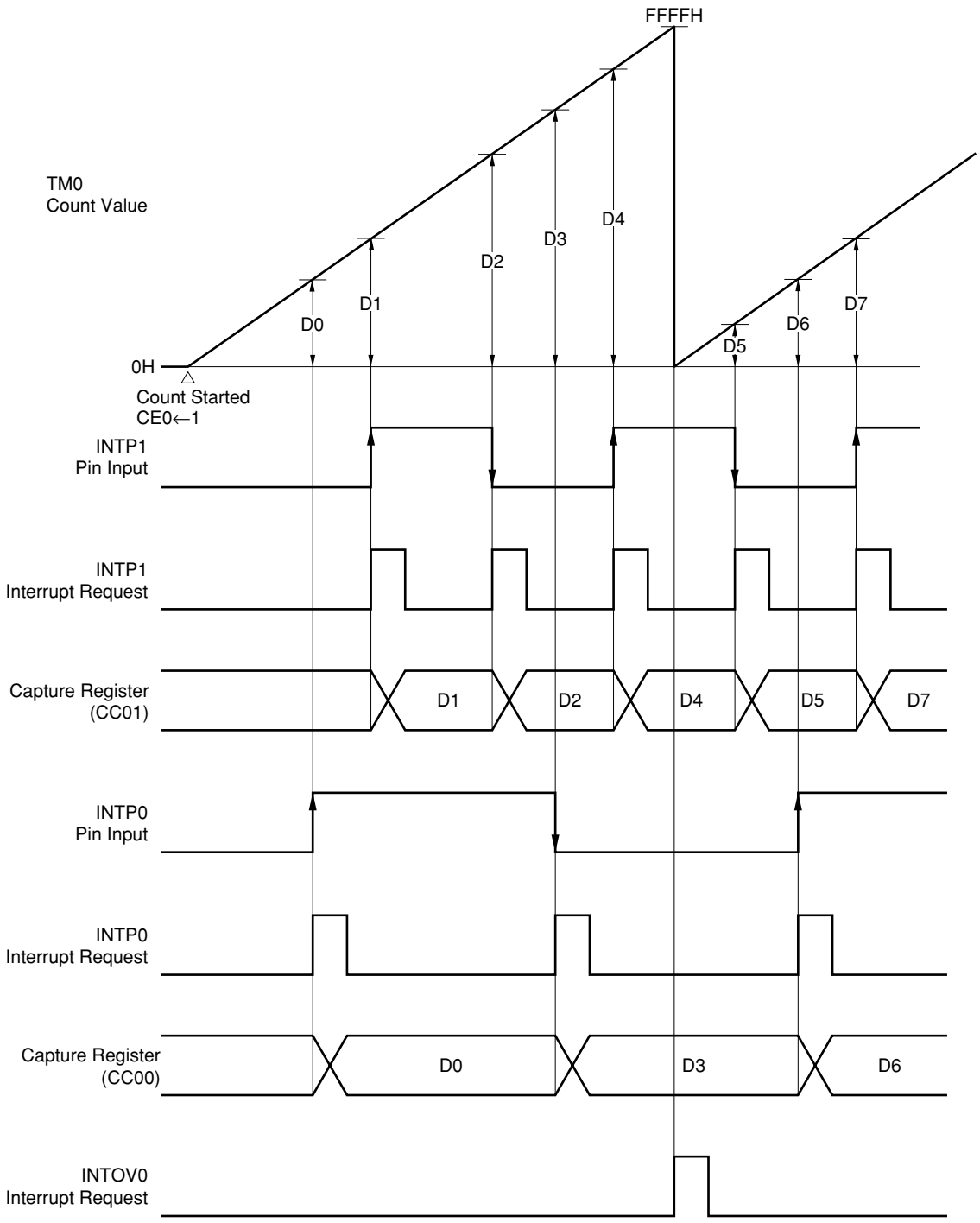
Table 7-5. Capture Trigger Signal to Capture Register (timer 0)

Capture Register	Capture Trigger Signal
CC00	INTP0
CC01	INTP1
CC02	INTP2
CC03	INTP3

Remark CC00 to CC03 are capture/compare registers. Whether these registers are used as capture registers or compare registers is specified by the timer unit mode register 0 (TUM0).

The valid edge of the capture trigger is specified by external interrupt mode registers (INTM0 and INTM1) If the capture trigger is specified so that both the rising and falling edges are valid, the width of an externally input pulse can be measured. If the capture trigger is generated with either of the edges specified as valid, the cycle of an input pulse can be measured.

Figure 7-9. Capture Operation (timer 0)



Remark Dn: TM0 count value (n = 0, 1, 2, ...)

7.6 Basic Operation of Output Control Circuit

The output control circuit controls the levels of the timer output pins (TO00 to TO03) by using the match signals from the compare registers (CC00 to CC03). The operation of the output control circuit is determined by the timer output control register 0 (TOC0). Note that the TO01 and TO03 pin outputs can be used for toggle operation only. The TO00 and TO02 pin outputs can be used for toggle or set/reset operation, according to the specification by the timer unit mode register 0 (TUM0).

To output the signals TO00 to TO03 to pins, the corresponding pins must be set in the control mode by using the port 2 mode control register (PMC2).

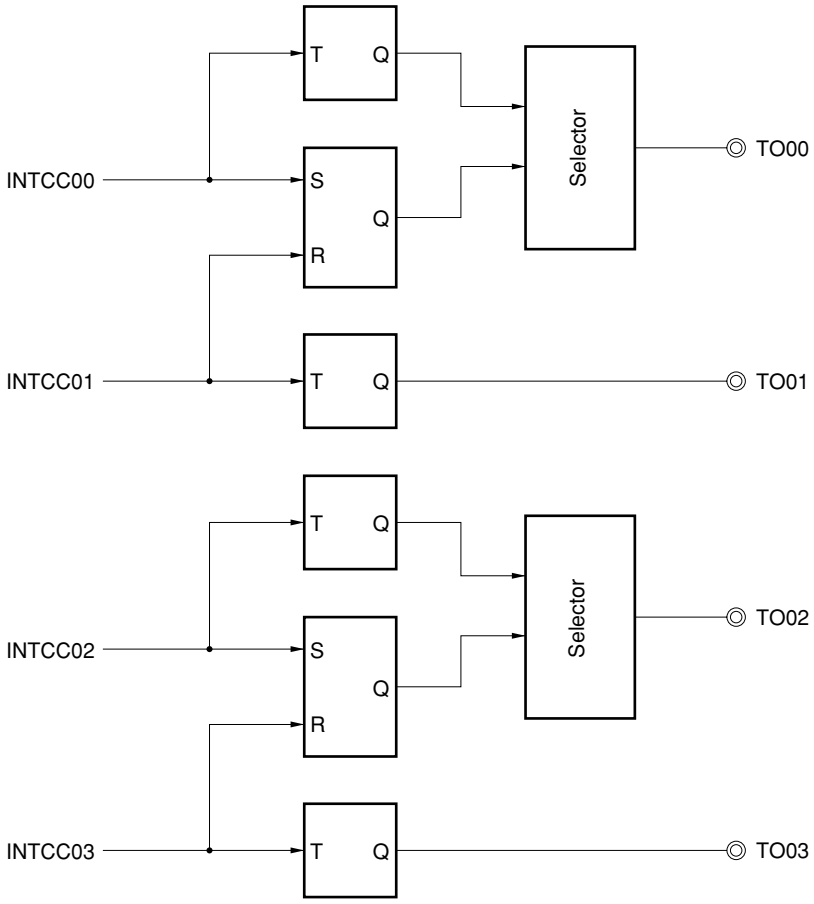
Table 7-6. Toggle Signal of Timer Output Pin (timer 0)

Timer Output	Toggle Signal
TO00	INTCC00
TO01	INTCC01
TO02	INTCC02
TO03	INTCC03

Table 7-7. Set/Reset Signal of Timer Output Pin (timer 0)

Timer Output	Set Signal	Reset Signal
TO00	INTCC00	INTCC01
TO02	INTCC02	INTCC03

Figure 7-10. Block Diagram of Timer Output Operation of Timer 0



7.6.1 Basic operation

By setting (1) the ENTO0n (n = 0 to 3) bit of the timer output control register 0 (TOC0), a pulse can be output from the TO0n (n = 0 to 3) pin.

Clearing (0) ENTO0n bit sets the TO0n to a fixed level. The fixed level is determined by the ALV0n (n = 0 to 3) bit of the TOC0. The level is high when ALV0n bit is 0, and low when 1.

7.6.2 Toggle output

Toggle output is an operating mode in which the output level is inverted each time the compare register (CC0n: n = 0 to 3) value coincides with the timer register 0 (TM0) value. The output level of timer output (TO0n: n = 0 to 3) is inverted by a match between CC0n and TM0.

When timer 0 is stopped by clearing (0) the CE0 bit of the timer mode control register (TMC), the output level at the time it was stopped is retained as is.

Figure 7-11. Operation of Toggle Output

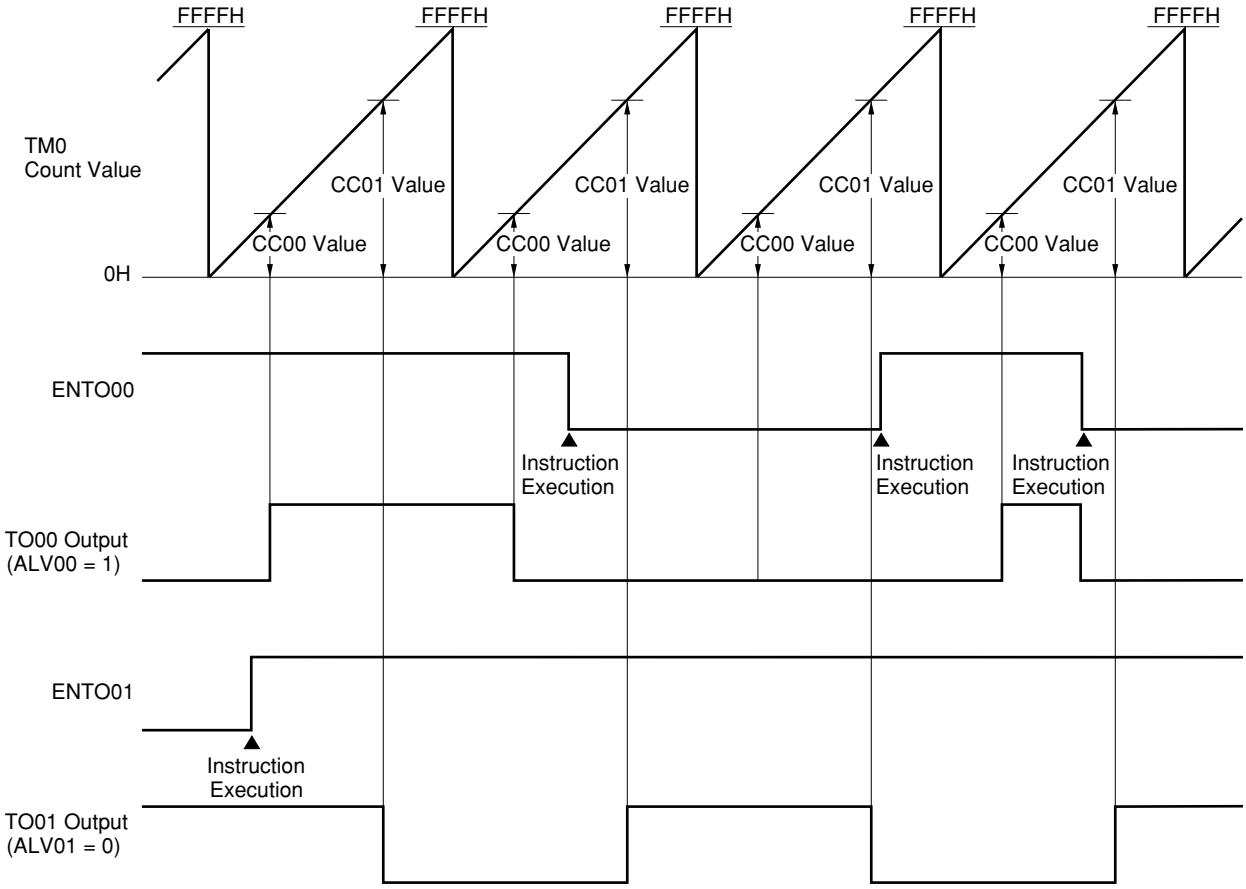


Table 7-8. Toggle Output of TO00 to TO03 (f_{CLK} = 16 MHz)

Count Clock	Minimum Pulse Width[Note]	Maximum Pulse Width
$f_{CLK}/4$	$4/f_{CLK}$ (0.25 μs)	$2^{16} \times 4/f_{CLK}$ (16.4 ms)
$f_{CLK}/8$	$8/f_{CLK}$ (0.5 μs)	$2^{16} \times 8/f_{CLK}$ (32.8 ms)
$f_{CLK}/16$	$16/f_{CLK}$ (1.0 μs)	$2^{16} \times 16/f_{CLK}$ (65.5 ms)
$f_{CLK}/32$	$32/f_{CLK}$ (2.0 μs)	$2^{16} \times 32/f_{CLK}$ (131 ms)
$f_{CLK}/64$	$64/f_{CLK}$ (4.0 μs)	$2^{16} \times 64/f_{CLK}$ (262 ms)

Note The minimum interval time is limited by the data transfer processing time. Consider the interrupt processing time or macro service processing time used (refer to **Table 14-11 Interrupt Acceptance Processing Time** and **Table 14-12 Macro Service Processing Time**).

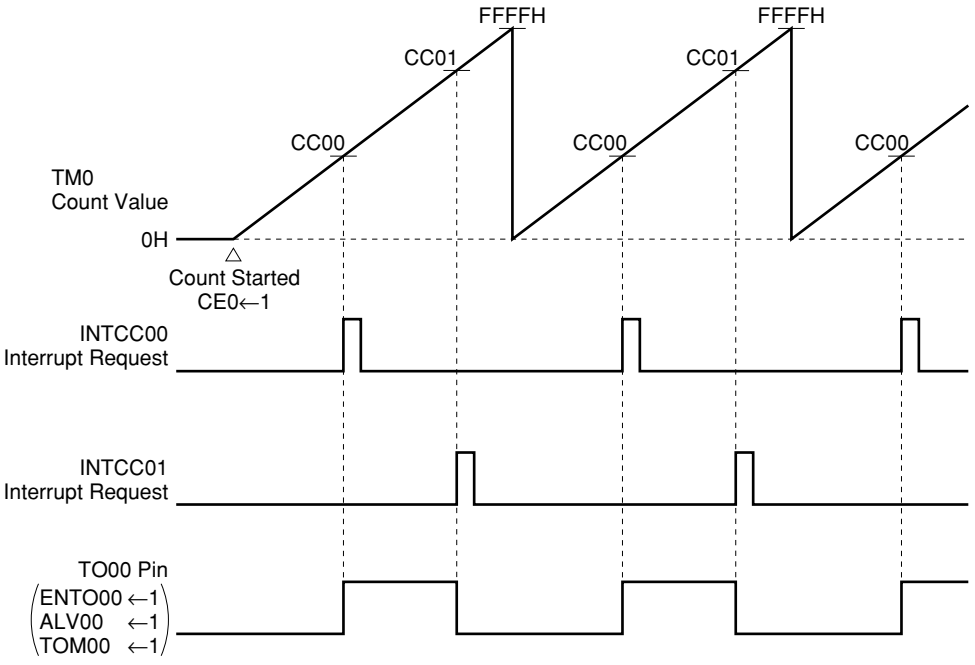
7.6.3 Set/reset output

The set/reset output is an operation mode in which the timer output is set or reset each time the value of the compare register (CC0n: n = 0 to 3) matches with the value of timer register 0 (TM0).

If CC00 = CC01 and CC02 = CC03, interrupt requests are simultaneously generated, and timer outputs (TO00 and TO02) are used as $\overline{ALV00}$ and $\overline{ALV02}$.

When timer 0 is stopped by clearing (0) the CE0 bit of the timer mode control register (TMC), the output level at which the timer stops is retained as is.

Figure 7-12. Operation of Set/Reset Output (timer 0)



7.7 Examples of Use

7.7.1 Operation as interval timer

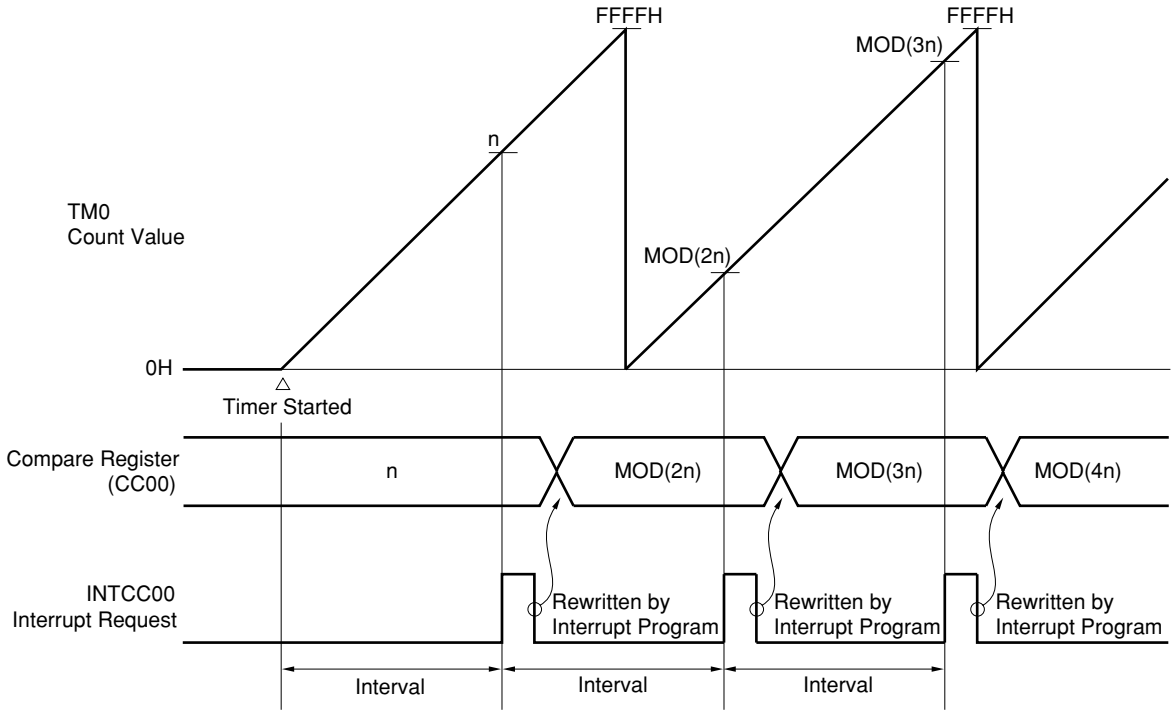
When timer register 0 (TM0) is made free-running and a fixed value is added to the compare register (CC0n: n = 0 to 3) in the interrupt processing routine, TM0 operates as an interval timer with the added fixed value as the cycle (refer to **Figure 7-13**).

This interval timer can count within the range shown in Table 7-1 (internal system clock f_{CLK} = 16 MHz).

Since TM0 has four capture compare registers, four interval timers with different cycles can be constructed.

Taking an example where compare register CC00 is used, the control register settings are shown in Figure 7-14, the setting procedure in Figure 7-15, and the processing in the interrupt processing routine in Figure 7-16.

Figure 7-13. Timing of Interval Timer Operation



Remark Interval time = $n \times x/f_{CLK}$

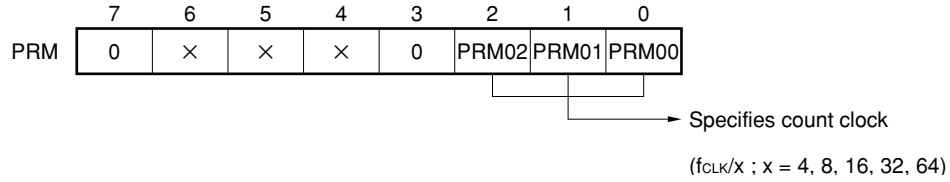
$y \leq n \leq FFFFH$

x = 4, 8, 16, 32, 64

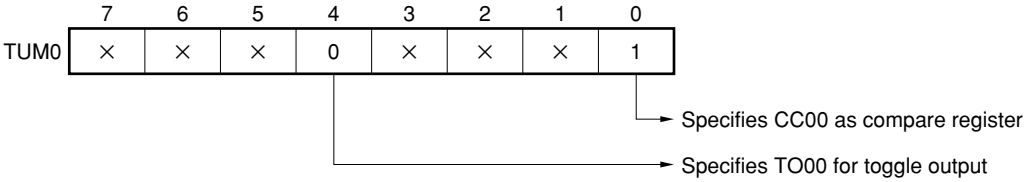
y is limited by the data transfer processing time. Consider the processing time of the interrupt used or the macro service processing time (refer to **Table 14-11 Interrupt Acceptance Processing Time** and **Table 14-12 Macro Service Processing Time**).

Figure 7-14. Set Contents of Control Register for Interval Timer Operation

(a) Prescaler mode register (PRM)



(b) Timer unit mode register 0 (TUM0)



× : don't care

Figure 7-15. Setting Procedure of Interval Timer Operation

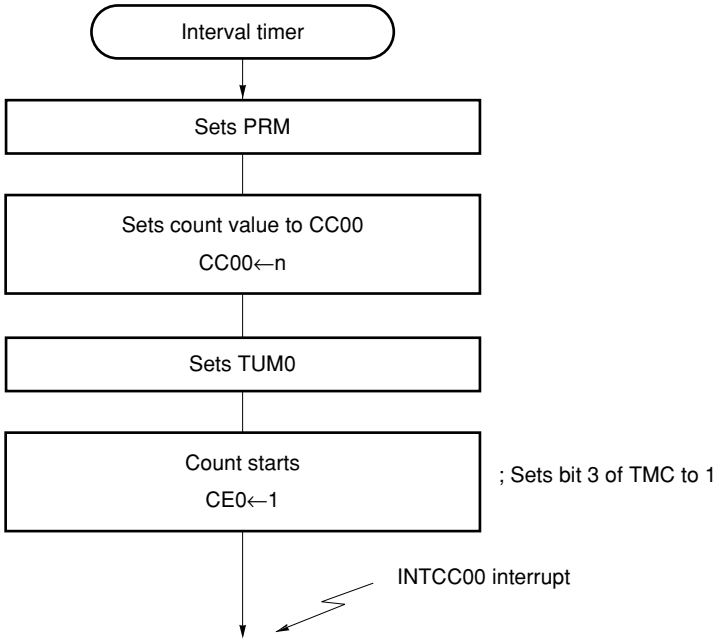
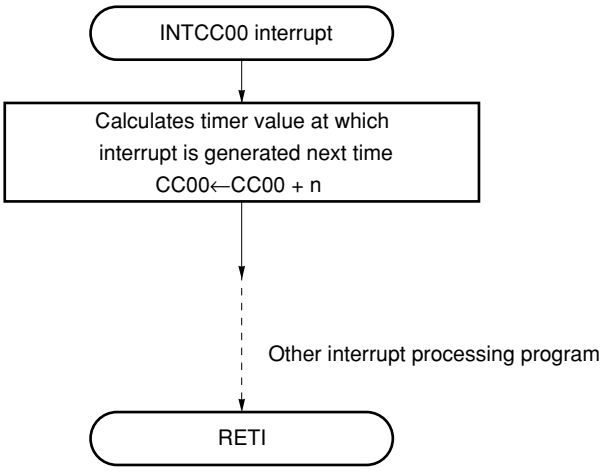


Figure 7-16. Interrupt Request Processing of Interval Timer Operation



7.7.2 Pulse width measurement operation

In pulse width measurement, the high-level or low-level width of external pulses input to the external interrupt request input pin (INTP0 to INTP3) is measured.

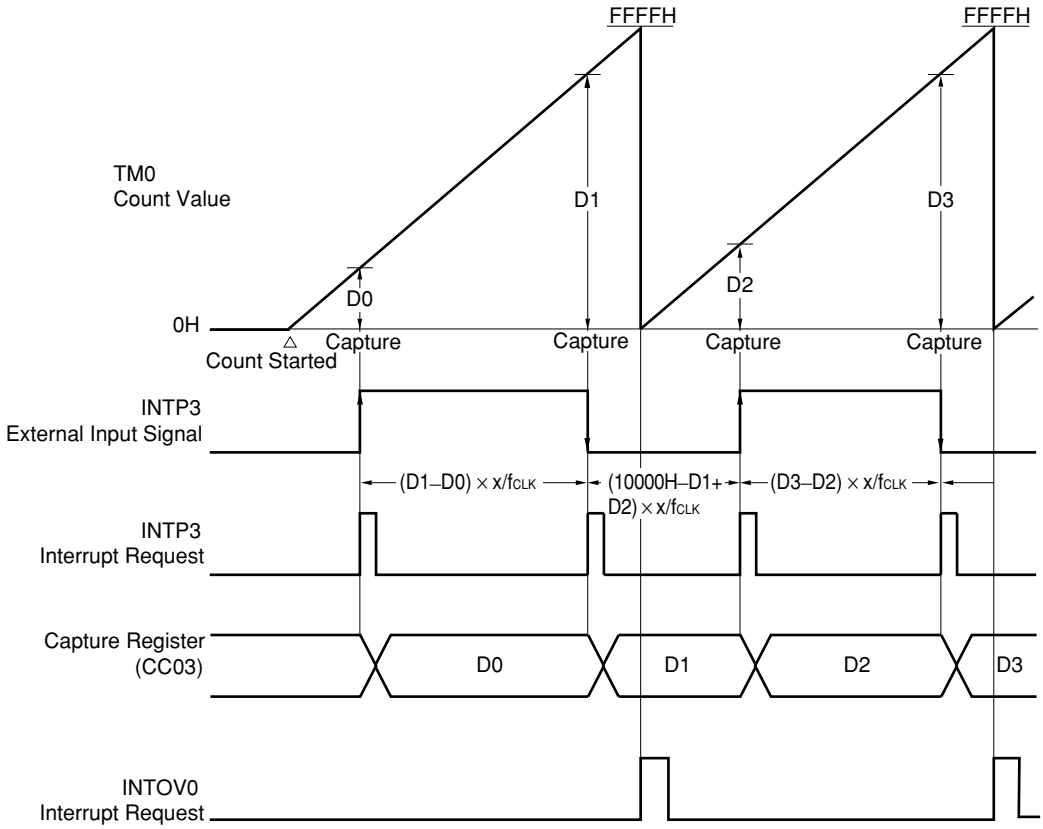
When the sampling clock is f_{CLK} both the high-level and low-level widths of pulses input to the INTPn (n = 0 to 3) pin must be at least 4 system clocks (0.25 μs: f_{CLK} = 16 MHz); if shorter than this, the valid edge will not be detected and a capture operation will not be performed.

When a pulse width is measured, the pulse width in a range shown in Table 7-2 can be measured (f_{CLK} = 16 MHz). How a pulse width is measured is explained below where the INTP3 pin is used as an external input pin.

As shown in Figure 7-17, the timer register 0 (TM0) value being counted is fetched into the capture register (CC03) in synchronization with a valid edge (specified as both rising and falling edges) in the INTP3 pin input, and held there. The pulse width is obtained from the product of the difference between the TM0 count value (Dn) fetched into and held in the CC03 on detection of the nth valid edge and the count value (D_{n-1}) fetched and held on detection of valid edge n-1, and the number of count clocks (x/f_{CLK}; x = 4, 8, 16, 32, 64).

The control register settings are shown in Figure 7-18, the setting procedure in Figure 7-19, and the processing at interrupt processing routine in Figure 7-20.

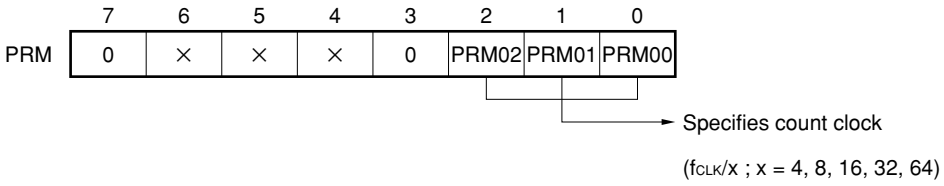
Figure 7-17. Timing of Pulse Width Measurement



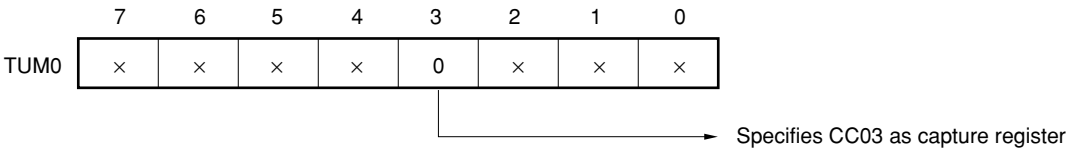
Remark Dn: TM0 count value (n = 0, 1, 2, ...)
x = 4, 8, 16, 32, 64

Figure 7-18. Control Register Settings for Pulse Width Measurement

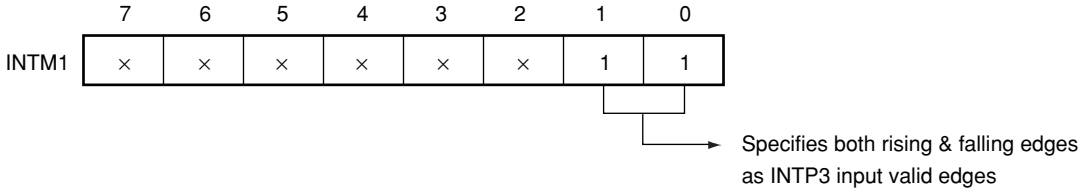
(a) Prescaler mode register (PRM)



(b) Timer unit mode register 0 (TUM0)



(c) External interrupt mode register 1 (INTM1)



× : Don't care

Figure 7-19. Pulse Width Measurement Setting Procedure

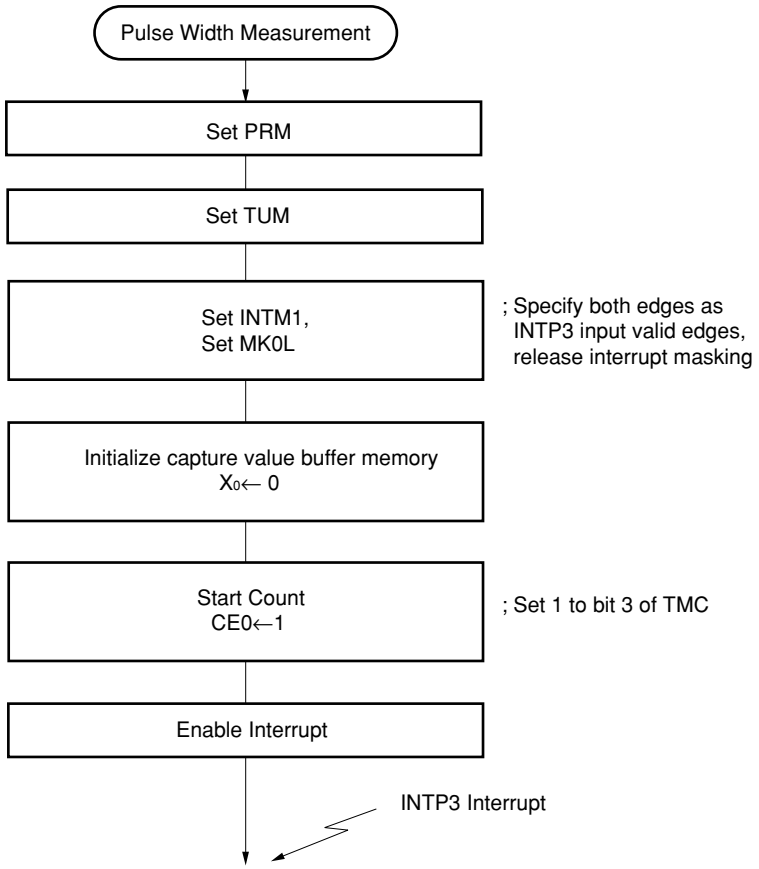
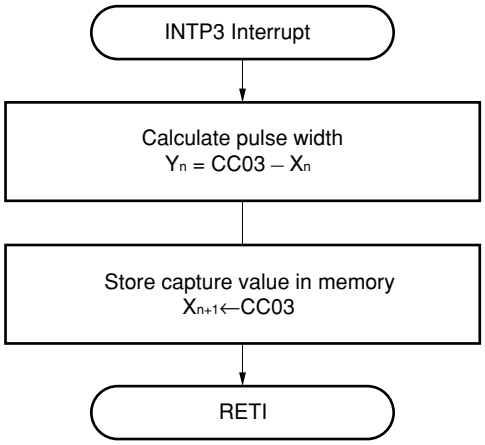


Figure 7-20. Interrupt Request Processing that Calculates Pulse Width

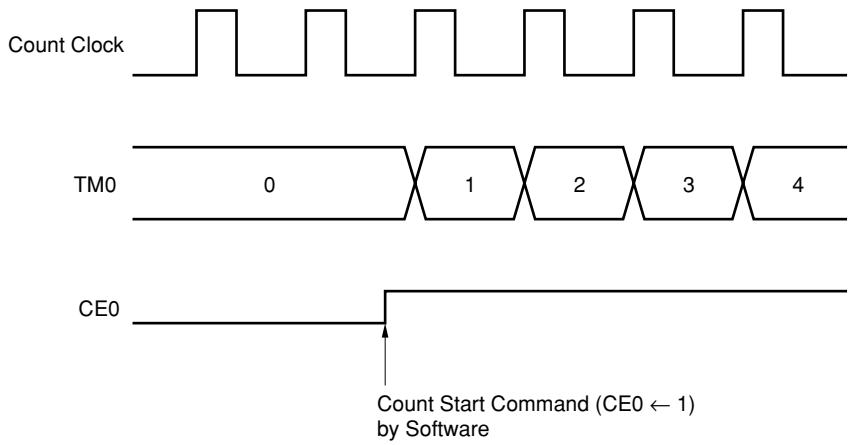


7.8 Cautions

(1) The prescaler uses one time base in common with all the timers (timers 0 and 1, timers/counters 2 and 3, and timer 4). If one of the timers sets the CE bit to “1”, the time base starts counting. If another timer sets the CE bit to “1” while one timer is operating, the first count clock of the timer may be shortened because the time base has already started counting.

For example, when using timer/counter 0 as an interval timer, the first interval time is shortened by up to 1 count clock. The second and those that follow are at the specified interval.

Figure 7-21. Operation When Counting Is Started



(2) While timer 0 is operating (while the CE0 bit of the timer mode control register (TMC) is set), malfunctioning may occur if the contents of the following registers are rewritten. This is because it is undefined which takes precedence in a contention the change in the hardware functions due to rewriting the register, or the change in the status because of the function before rewriting.

Therefore, be sure to stop the counter operation for the sake of safety before rewriting the contents of the following registers.

- Timer unit mode register 0 (TUM0)
- Timer output control register 0 (TOC0)
- Prescaler mode register (PRM)

(3) If the contents of the compare register (CC0n: n = 0 to 3) match with those of TM0 operation when an instruction that stops timer register 0 (TM0) operation is executed, the counting operation of TM0 stops, but an interrupt request is generated.

In order not to generate the interrupt when stopping the operation of TM0, mask the interrupt in advance by using the interrupt mask register before stopping TM0.

Example

Program that may generate interrupt request

```
CLR1   CE0                ← Interrupt request
OR     MK0L, #78H           from timer 0
                            occurs between
                            these instructions
       ⋮
```

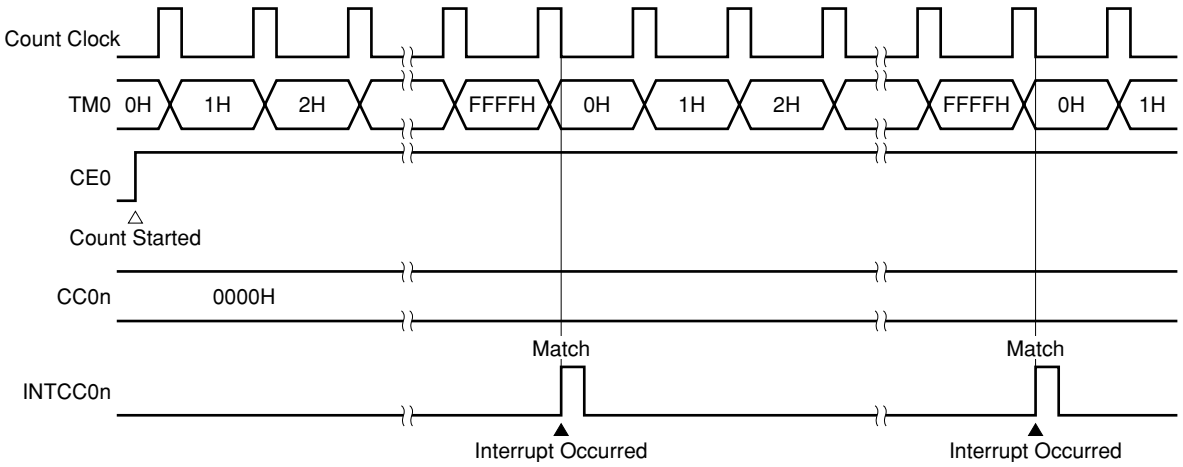
Program that does not generate interrupt request

```
OR     MK0L, #78H         ← Disables interrupt
CLR1   CE0                  from timer 0
CLR1   PIF0               ← Clears interrupt request
CLR1   PIF1                 flag from timer 0
CLR1   PIF2
CLR1   PIF3
⋮
```

(4) Match between timer register 0 (TM0) and compare register (CC0n: n = 0 to 3) is detected only when TM0 is incremented. Therefore, the interrupt request is not generated even if the same value as TM0 is written to CC0n, and the timer output (TO0n: n = 0 to 3) does not change.

(5) When the compare register (CC00 to CC03) is set to 0000H, the compare operation is performed after counting by TM0. Therefore, the match interrupt (INTCC00 to INTCC03) does not occur immediately after counting has been started. If CC0n (n = 0 to 3) is set to 0000H, TM0 counts up to FFFFH, the timer overflows, and match interrupt INTCC0n (n = 0 to 3) occurs.

Figure 7-22. Operation When Compare Register (CC00 to CC03) Is Set to 0000H



Remark n = 0 to 3

(6) If the timer output is enabled when the active level is changed, the output level of pins may change momentarily. To prevent this, enable the timer output after the active level have been changed.

(7) To change the active level specification (ALV0n bit (n = 0 to 3) of the timer output control register 0 (TOC0)), change the active level specification after the timer output of the corresponding timer output pins has been disabled.

CHAPTER 8 TIMER 1

8.1 Function

Timer 1 is a 16-bit timer.

In addition to a function as an interval timer, this timer has a toggle and set/reset function as timer output.

When used as an interval timer, timer 1 generates an internal interrupt at interval determined in advance.

Table 8-1. Interval Time of Timer 1

Minimum Interval Time	Maximum Interval Time	Resolution
$8/f_{CLK}$ (0.5 μs)	$2^{16} \times 8/f_{CLK}$ (32.8 ms)	$8/f_{CLK}$ (0.5 μs)
$16/f_{CLK}$ (1.0 μs)	$2^{16} \times 16/f_{CLK}$ (65.5 ms)	$16/f_{CLK}$ (1.0 μs)
$32/f_{CLK}$ (2.0 μs)	$2^{16} \times 32/f_{CLK}$ (131 ms)	$32/f_{CLK}$ (2.0 μs)
$64/f_{CLK}$ (4.0 μs)	$2^{16} \times 64/f_{CLK}$ (262 ms)	$64/f_{CLK}$ (4.0 μs)
$128/f_{CLK}$ (8.0 μs)	$2^{16} \times 128/f_{CLK}$ (524 ms)	$128/f_{CLK}$ (8.0 μs)

( ): at f_{CLK} = 16 MHz

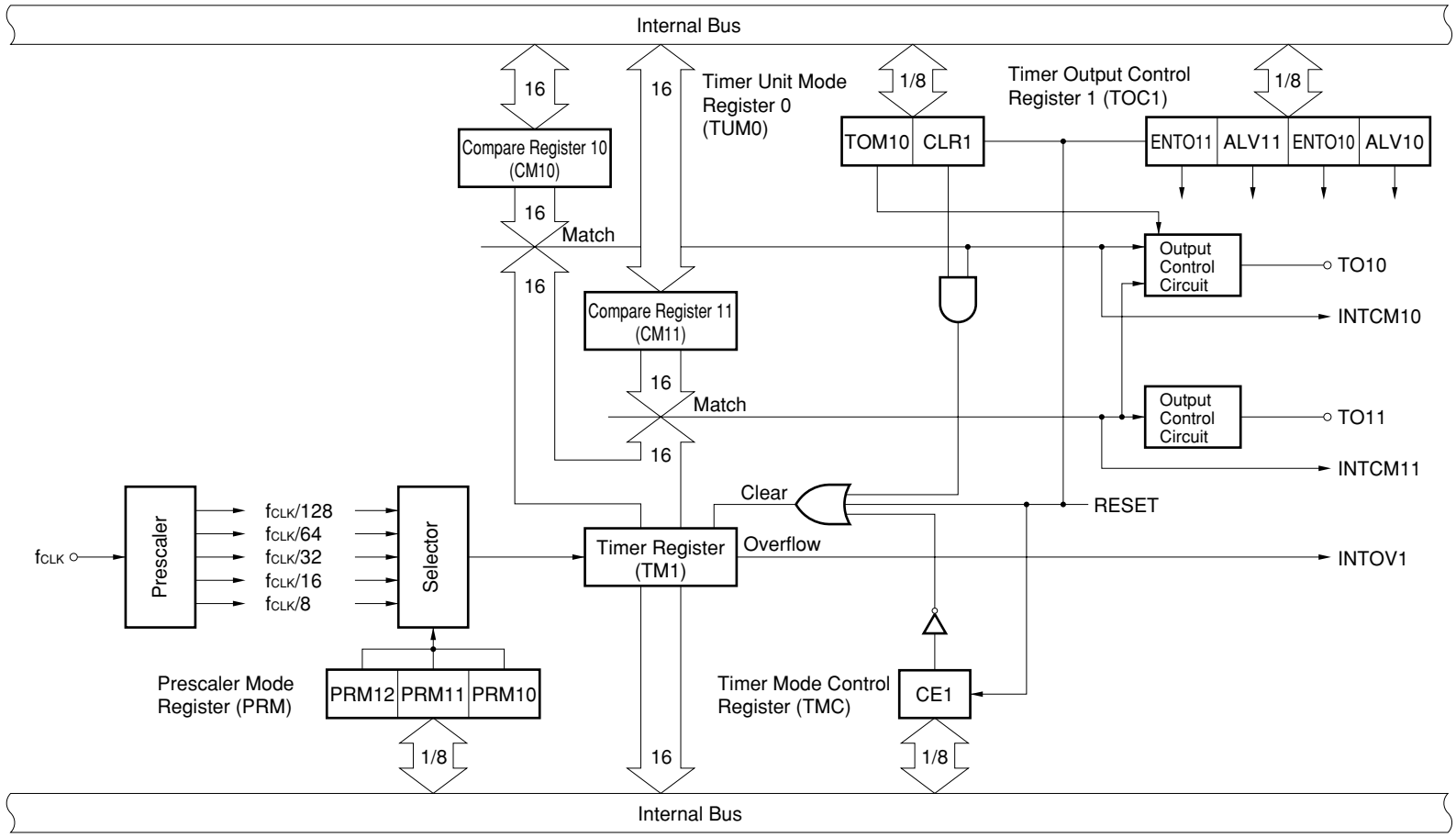
8.2 Configuration

Timer 1 consists of the following registers:

- Timer register (TM1) × 1
- Compare register (CM1n) × 2 (n = 0, 1)

Figure 8-1 shows the block diagram of timer 1.

Figure 8-1. Block Diagram of Timer 1



(1) Timer register 1 (TM1)

TM1 is a timer register that counts up the count clock specified by the prescaler mode register (PRM).

Counting of this timer register is enabled or disabled by the timer mode control register (TMC).

The timer register can be only read by using a 16-bit manipulation instruction. When $\overline{\text{RESET}}$ is input, TM1 is cleared to 0000H and stops counting.

(2) Compare registers (CM10, CM11)

CM1n (n = 0, 1) is a 16-bit register that holds the value determining the cycle of the interval timer operation.

When the contents of CM1n matches with the contents of TM1, an interrupt request (INTCM1n: n = 0, 1) and a timer output control signal are generated. The count value of TM1 can be cleared when its value matches with the contents of CM10.

These compare registers can be read or written by using 16-bit manipulation instructions. When $\overline{\text{RESET}}$ is input, their contents are undefined.

(3) Output control circuit

When the contents of CM1n (n = 0, 1) and the contents of TM1 match, the timer output can be inverted. A square wave can be output from a timer output pin (TO10, TO11) if so specified by the timer output control register 1 (TOC1).

The TO10 pin can also output a set and reset signals if so specified by the timer unit mode register 0 (TUM0).

The timer output can be enabled or disabled by TOC1. When the timer output is disabled, a fixed level is output to the TO1n (n = 0, 1) pin (the output level is fixed by TOC1).

(4) Prescaler

The prescaler generates a count clock by dividing the internal system clock. The clock generated by the prescaler is selected by the selector, and TM1 performs the count operation by using this clock as a count clock.

(5) Selector

The selector selects one of the five signals generated by dividing the internal system clock as the count clock of TM1.

8.3 Timer 1 Control Register

(1) Timer unit mode register 0 (TUM0)

TUM0 is a register that specifies the output mode of the timer output pins (TO00, TO02, and TO10) of timers 0 and 1, controls the clear operation of the timer register 1 (TM1), and specifies the operations of the capture/compare registers (CC00 to CC03) of timer 0.

This register can be read or written by using an 8-bit manipulation instruction or a bit manipulation instruction. Figure 8-2 shows the format of TUM0.

When $\overline{\text{RESET}}$ is input, the value of TUM0 is cleared to 00H.

Figure 8-2. Format of Timer Unit Mode Register 0 (TUM0)

Address : 0FF30H On reset : 00H R/W

	7	6	5	4	3	2	1	0
TUM0	TOM10	CLR1	TOM02	TOM00	CMS03	CMS02	CMS01	CMS00

TOM10	Specifies Output Mode of TO10 Pin
0	Toggle output
1	Set/reset output

CLR1	Controls Clear Operation of TM1 by match with CM10
0	Disabled (free running mode)
1	Enabled (interval timer mode)

TOM0n	Specifies Output Mode of TO0n Pin (n = 0, 2) (refer to **Figure 7-2**).

CMS0n	Specifies Operation of CC0n (n = 0 to 3) (refer to **Figure 7-2**).

(2) Timer mode control register (TMC)

TMC is a register that controls the count operation of timer registers 0 and 1 (TM0 and TM1).

This register can be read or written by using an 8-bit manipulation instruction or a bit manipulation instruction. Figure 8-3 shows the format of TMC.

When $\overline{\text{RESET}}$ is input, the value of this register is cleared to 00H.

Figure 8-3. Format of Timer Mode Control Register (TMC)

Address : 0FF31H On reset : 00H R/W

	⑦	6	5	4	③	2	1	0
TMC	CE1	0	0	0	CE0	0	0	0

CE1	Controls Count Operation of TM1
0	Clears and stops counting
1	Enables counting operation

CE0	Controls Count Operation of TM0 (refer to **Figure 7-3**).

(3) Timer output control register 1 (TOC1)

TOC1 is a register that specifies the operation and active level of the timer output pins (TO10, TO11) of timer 1.

This register can be read or written by using an 8-bit manipulation instruction or a bit manipulation instruction. Figure 8-4 shows the format of TOC1.

When $\overline{\text{RESET}}$ is input, the value of this register is cleared to 00H.

Figure 8-4. Format of Timer Output Control Register 1 (TOC1)

Address : 0FF33H On reset : 00H R/W

	7	6	5	4	③	2	①	0
TOC1	0	0	0	0	ENTO11	ALV11	ENTO10	ALV10

ENTO1n	Specifies Operation of TO1n Pin (n = 0, 1)
0	Outputs $\overline{\text{ALV1n}}$
1	Enables pulse output

ALV1n	Specifies Active Level of TO1n Pin (n = 0, 1)
0	Low level
1	High level

(4) Prescaler mode register (PRM)

PRM is a register that specifies the count clock of timer registers 0 and 1 (TM0, TM1).

This register can be read or written by using an 8-bit manipulation instruction. Figure 8-5 shows the format of PRM.

When $\overline{\text{RESET}}$ is input, the value of this register is cleared to 00H.

Figure 8-5. Format of Prescaler Mode Register (PRM)

Address : 0FF38H On reset : 00H R/W

	7	6	5	4	3	2	1	0
PRM	0	PRM12	PRM11	PRM10	0	PRM02	PRM01	PRM00

(f_{CLK} = 16 MHz)

PRM12	PRM11	PRM10	Specifies Count Clock of TM1	
			Count Clock [Hz]	Resolution [μs]
0	0	0	f_{CLK}/8	0.5
0	0	1	f_{CLK}/16	1.0
0	1	0	f_{CLK}/32	2.0
0	1	1	f_{CLK}/64	4.0
1	0	0	f_{CLK}/128	8.0
Other			Setting prohibited	

PRM02	PRM01	PRM00	Specifies Count Clock of TM0 (refer to **Figure 7-5**).

Remark f_{CLK}: internal system clock

8.4 Operation of Timer Register 1 (TM1)

8.4.1 Basic operation

Timer 1 counts up by using the count clock specified by the prescaler mode register (PRM).

Counting is enabled or disabled by the CE1 bit of the timer mode control register (TMC). When the CE1 bit is set (1) by software, TM1 is set to 0001H at the first count clock, and starts counting up. When the CE1 bit is cleared (0) by software, TM1 is immediately cleared to 0000H, and stops the generation of the match signal.

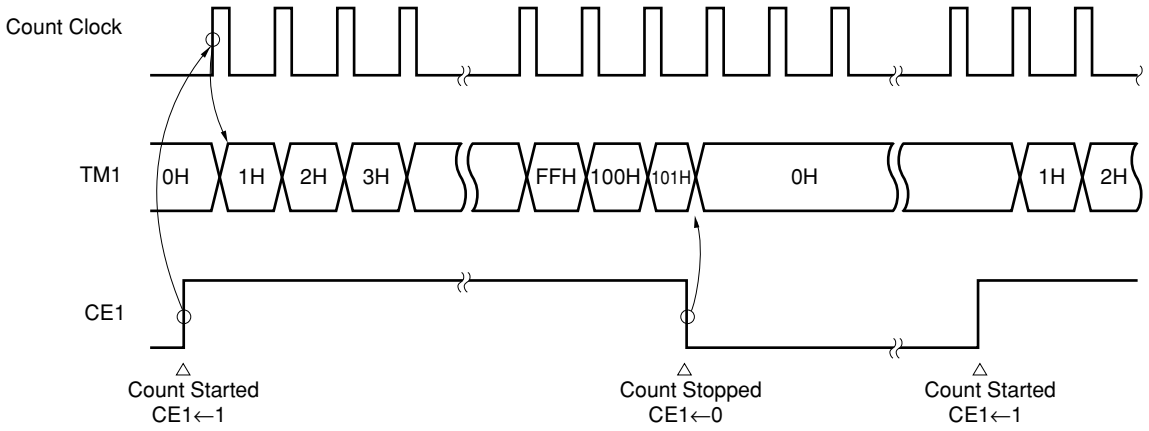
If the CE1 bit is set (1) while it has been already set (1), TM1 is not cleared but continues counting.

If a count clock is input when TM1 reaches FFFFH, TM1 is cleared to 0000H, and an overflow interrupt (INTOV1) occurs.

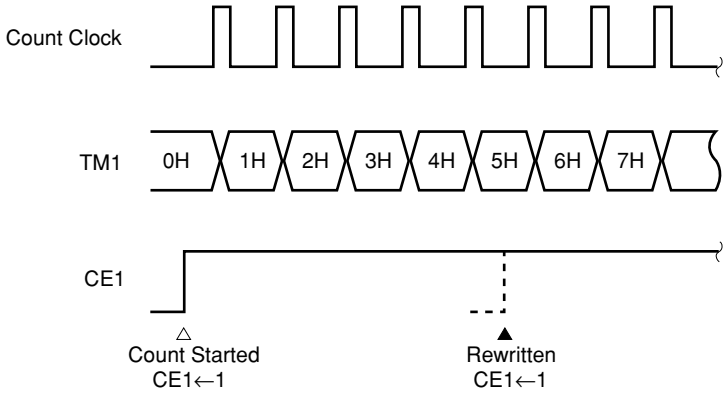
When $\overline{\text{RESET}}$ is input, TM1 is cleared to 0000H and stops counting.

Figure 8-6. Basic Operation of Timer Register 1 (TM1)

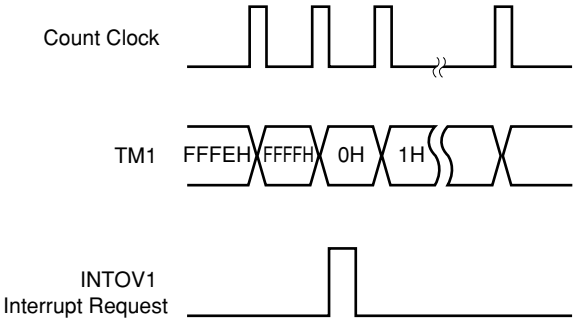
(a) Count started → count stopped → count started



(b) When "1" is written to the CE1 bit again after the count starts



(c) Operation when TM1 = FFFFH

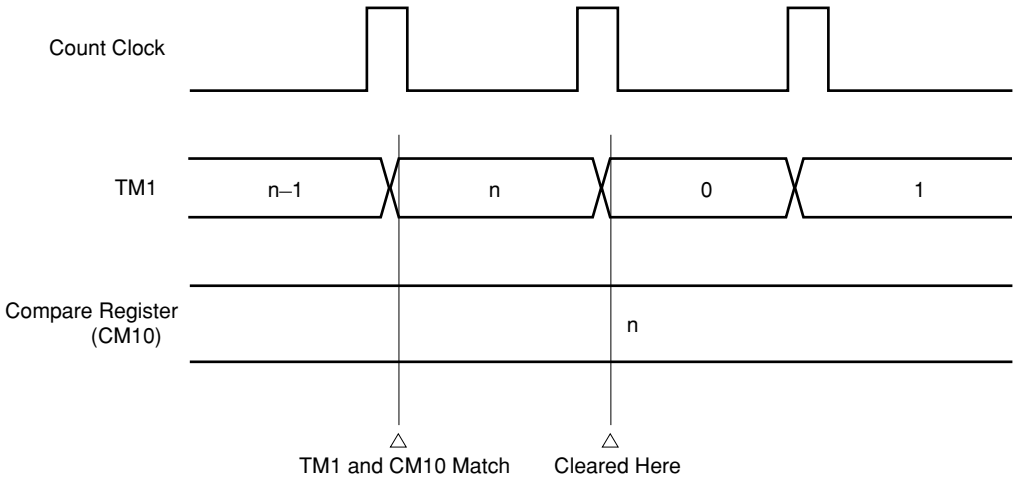


8.4.2 Clear operation

(1) Clear operation after match with compare register

Timer register 1 (TM1) can be cleared automatically after a match with the compare register (CM10). When a clearance source arises, TM1 is cleared to 000H on the next count clock. Therefore, even if a clearance source arises, the value at the point at which the clearance source arose is retained until the next count clock arrives.

Figure 8-7. TM1 Clear Operation by Match with Compare Register (CM10)

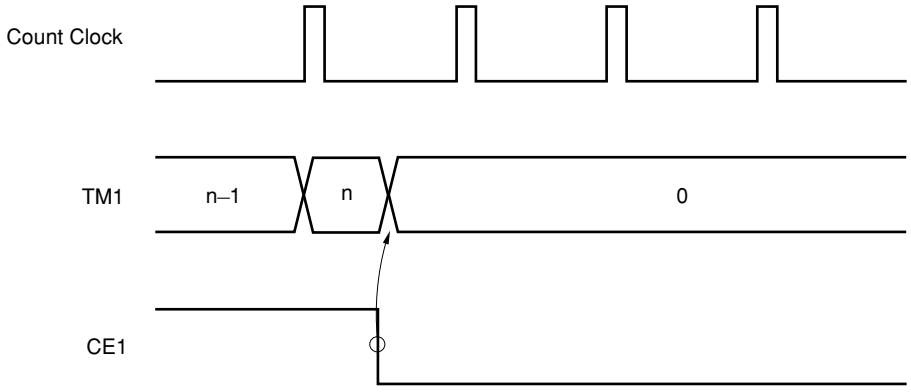


(2) Clear operation by CE1 bit of timer mode control register (TMC)

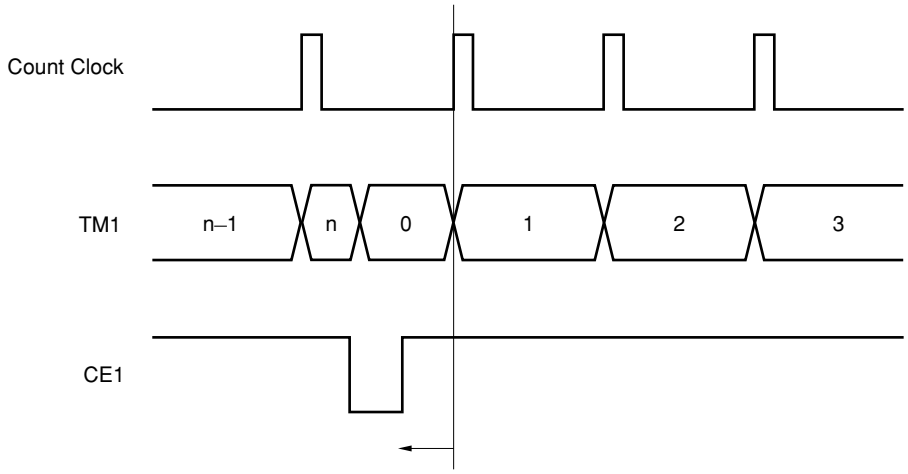
Timer register 1 (TM1) is also cleared when the CE1 bit of TMC is cleared (0) by software. The clear operation is performed immediately after the clearance (0) of the CE1 bit.

Figure 8-8. TM1 Clear Operation When CE1 Bit is Cleared (0)

(a) Basic operation

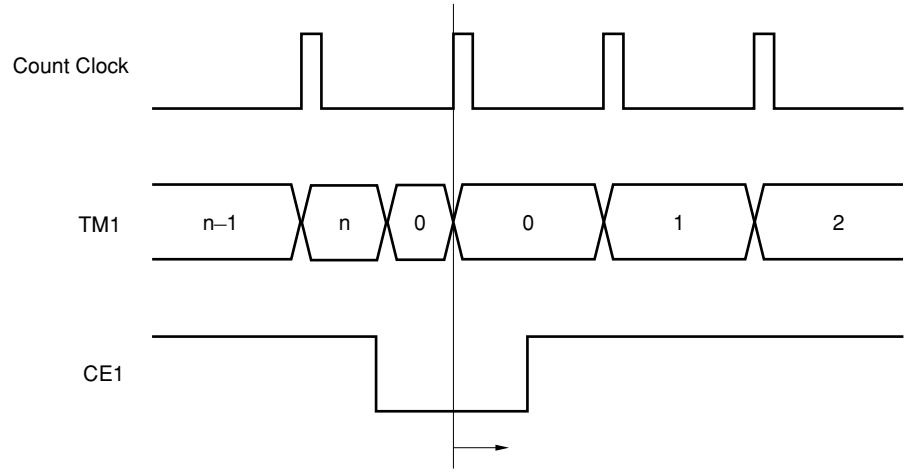


(b) Restart before count clock is input after clearance



If the CE1 bit is set (1) before this count clock, this count clock starts counting from 1.

(c) Restart after count clock is input after clearance



If the CE1 bit is set (1) from this count clock onward, the count clock starts counting from 1 after the CE1 bit is set (1).

8.5 Operation of Compare Register

Timer 1 performs a compare operation by comparing the value set to a compare register (CM10, CM11) specified as a compare register with the count value of a timer register 1 (TM1).

If the count value of TM1 matches with the value set in advance to CM1n (n = 0, 1) as a result of counting by TM1, the timer sends a match signal to the output control circuit, and at the same time, generates an interrupt request signal (INTCM10, INTCM11).

After the value of TM1 has matched with the value of CM10, the count value of TM1 can be cleared, so that TM1 can be used as an interval timer that repeatedly counts the value set to CM10.

Table 8-2. Interrupt Request Signal from Compare Register (timer 1)

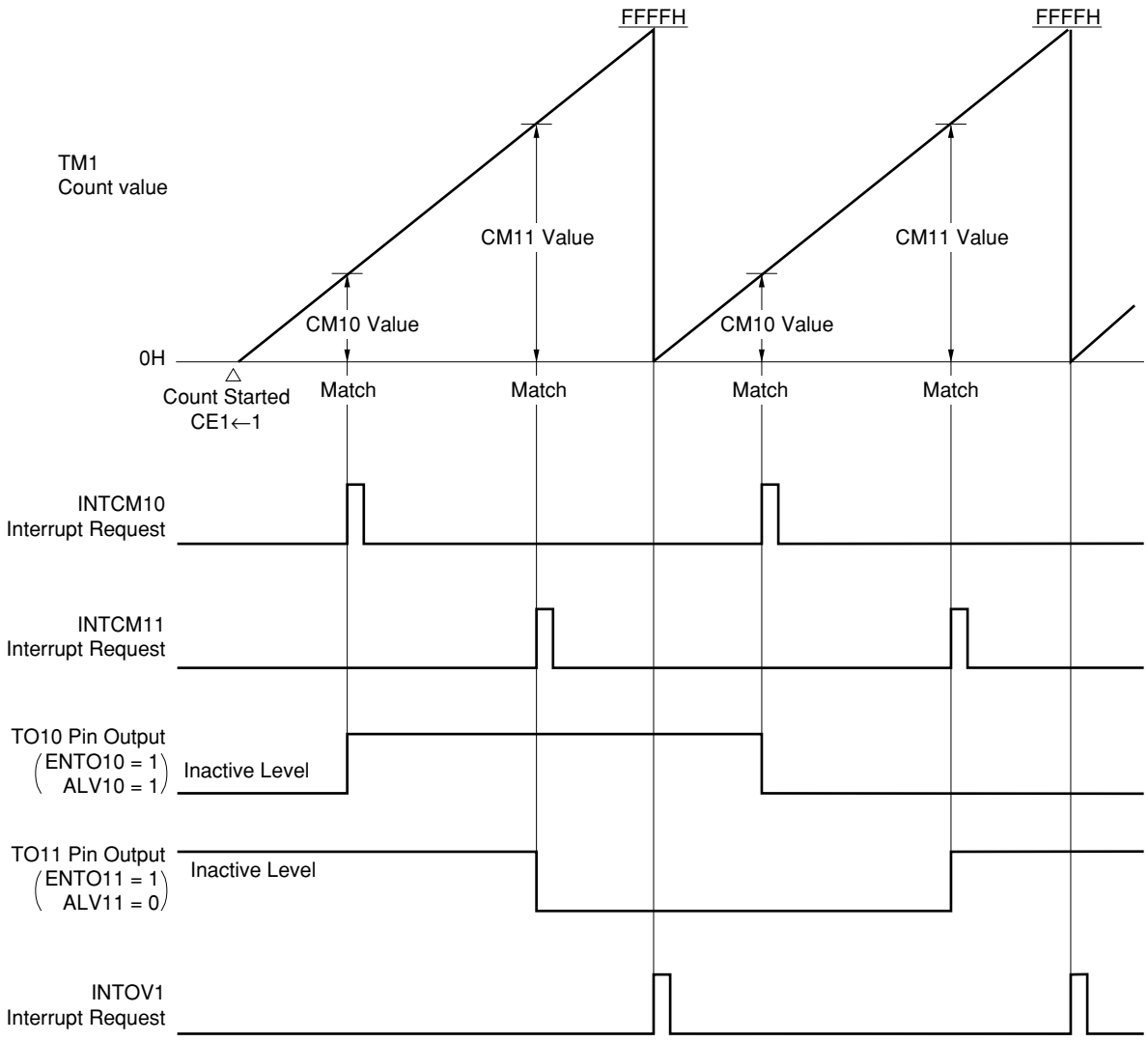
Compare Register	Interrupt Request Signal
CM10	INTCM10
CM11	INTCM11

Timer 1 has two timer output pins (TO10, TO11). Table 8-3 shows the operation mode of each of these pins (for details, refer to **8.6 Basic Operation of Output Control Circuit**).

Table 8-3. Operation Mode of Timer Output Pin (timer 1)

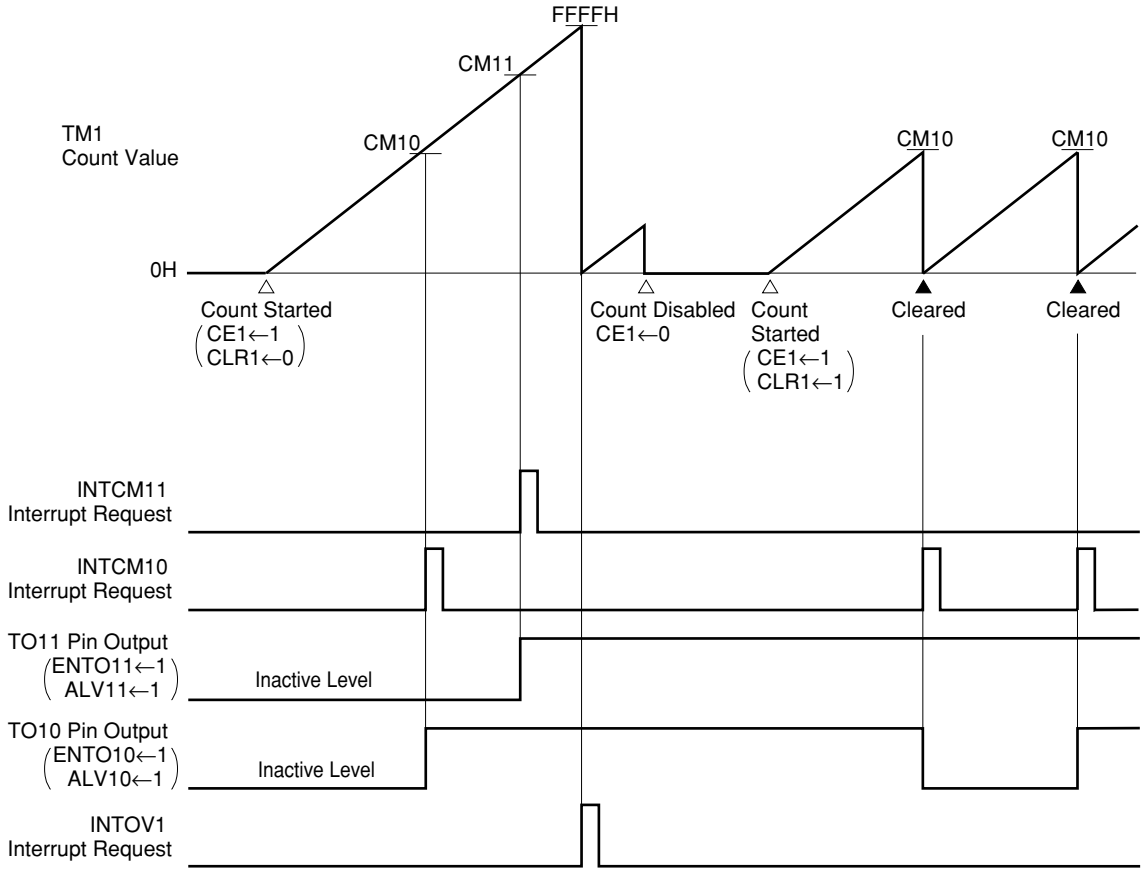
Timer Output Pin	Output Operation Mode		Specification of Operation Mode
TO10	Toggle	Set/reset	TUM10 bit of TUM0
TO11	Toggle	–	–

Figure 8-9. Compare Operation (timer 1)



Remark CLR1 = 0

Figure 8-10. Clearing TM1 after Detection of Match



8.6 Basic Operation of Output Control Circuit

The output control circuit controls the levels of the timer output pins (TO10, TO11) by using the coincidence signals from the compare registers (CM10, CM11). The operation of the output control circuit is determined by the timer output control register 1 (TOC1). Note that the TO11 pin output can be used for toggle operation only. The TO10 pin output can be used for toggle or set/reset operation, according to the specification by the timer unit mode register 0 (TUM0).

To output the TO10 and TO11 signals to pins, the corresponding pins must be set in the control mode by using the port 3 mode control register (PMC3).

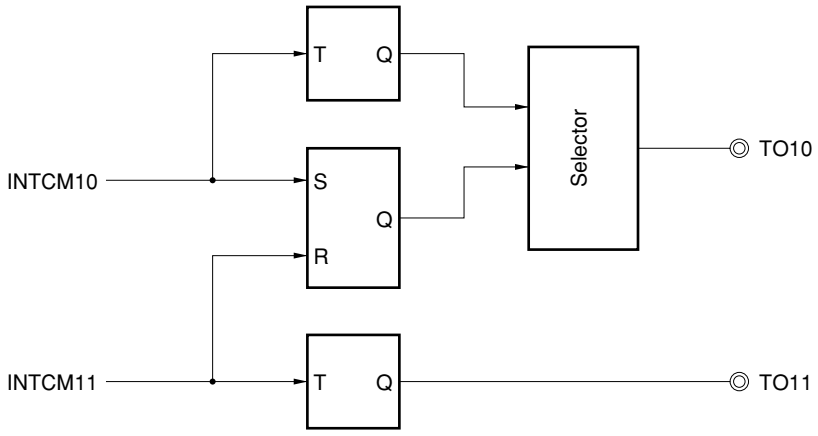
Table 8-4. Toggle Signal of Timer Output Pin (timer 1)

Timer Output	Toggle Signal
TO10	INTCM10
TO11	INTCM11

Table 8-5. Set/Reset Signal of Timer Output Pin (timer 1)

Timer Output	Set Signal	Reset Signal
TO10	INTCM10	INTCM11

Figure 8-11. Block Diagram of Timer Output Operation of Timer 1



8.6.1 Basic operation

By setting (1) the ENTO1n (n = 0, 1) bit of the timer output control register 1 (TOC1), a pulse can be output from the TO1n (n = 0, 1) pin.

By clearing (0) the ENTO1n bit, the level of TO1n is fixed. The level to which TO1n is fixed is determined by the ALV1n (n = 0, 1) bit of TOC1. When the ALV1n bit is 0, TO1n is fixed to the high level; when ALV1n bit is 1, it is fixed to the low level.

8.6.2 Toggle output

Toggle output is an operation mode in which the output level is inverted each time the value of the compare register (CM10, CM11) matches with the value of timer register 1 (TM1). The output level of the timer output TO10 is inverted when the value of CM10 matches with TM1, and the output level of TO11 is inverted when the value of CM11 matches with the value of TM1.

When timer 1 is stopped by clearing (0) the CE1 bit of the timer mode control register (TMC), the output level is retained as is.

Figure 8-12. Operation of Toggle Output

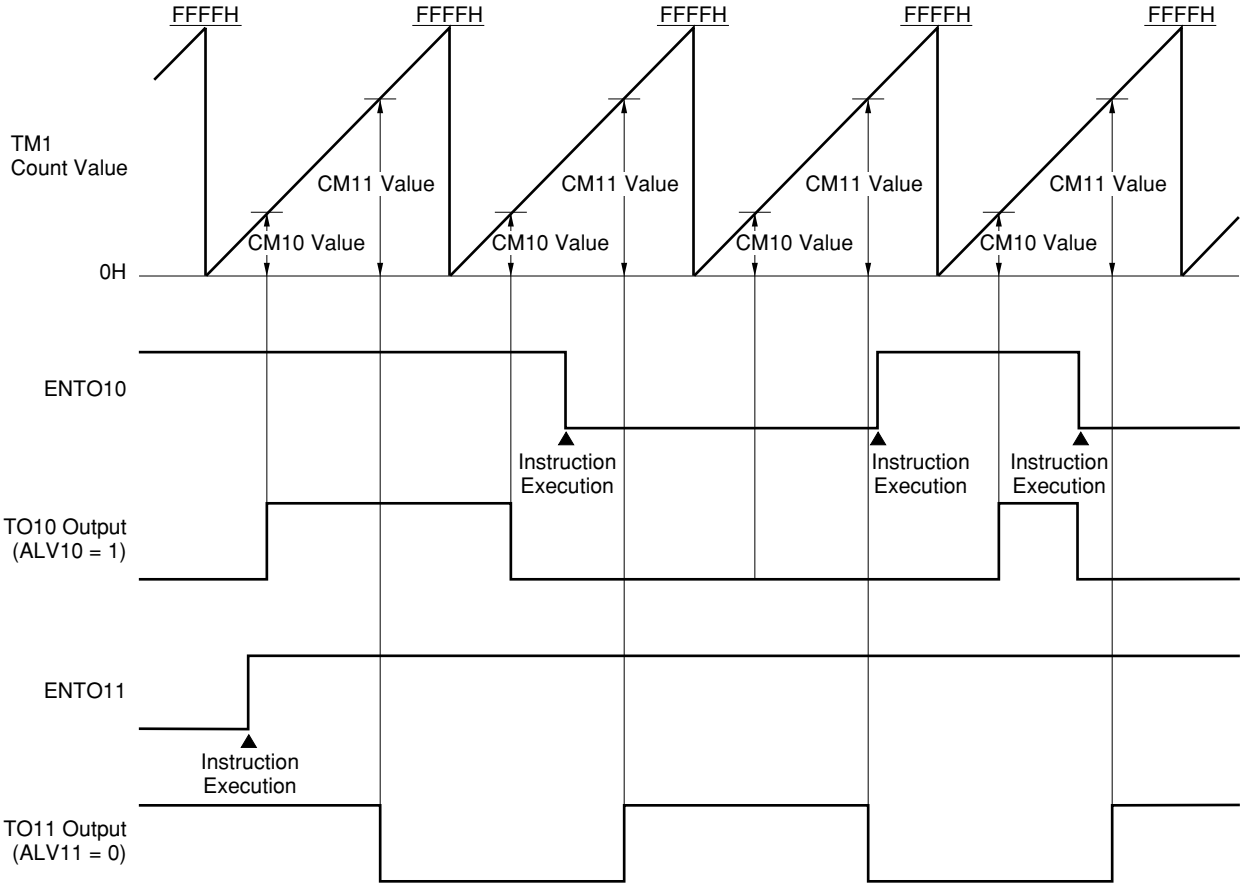


Table 8-6. Toggle Output of TO10 and TO11 (f_{CLK} = 16 MHz)

Count Clock	Minimum Pulse Width	Maximum Pulse Width
$f_{CLK}/8$	$8/f_{CLK}$ (0.5 μs)	$2^{16} \times 8/f_{CLK}$ (32.8 ms)
$f_{CLK}/16$	$16/f_{CLK}$ (1.0 μs)	$2^{16} \times 16/f_{CLK}$ (65.5 ms)
$f_{CLK}/32$	$32/f_{CLK}$ (2.0 μs)	$2^{16} \times 32/f_{CLK}$ (131 ms)
$f_{CLK}/64$	$64/f_{CLK}$ (4.0 μs)	$2^{16} \times 64/f_{CLK}$ (262 ms)
$f_{CLK}/128$	$128/f_{CLK}$ (8.0 μs)	$2^{16} \times 128/f_{CLK}$ (524 ms)

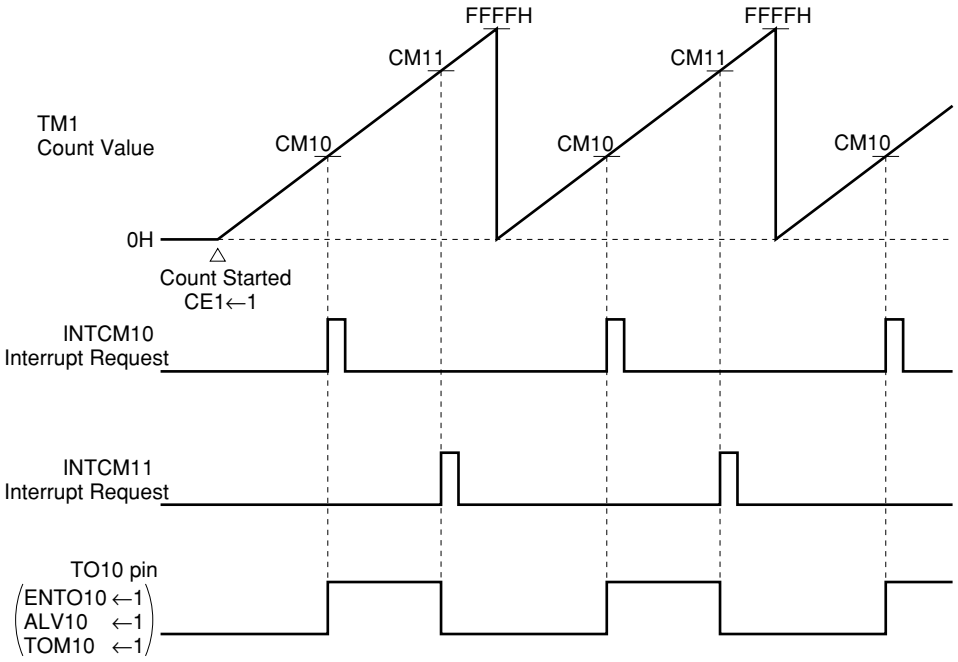
8.6.3 Set/reset output

The set/reset output is an operation mode in which the timer output is set or reset each time the value of the compare register (CM1n: n = 0, 1) matches with the value of timer register 1 (TM1).

If CM10 = CM11, interrupt requests are simultaneously generated, and timer output (TO10) is used as $\overline{ALV10}$.

When timer 1 is stopped by clearing (0) the CE1 bit of the timer mode control register (TMC), the output level at which the timer stops is retained as is.

Figure 8-13. Operation of Set/Reset Output (timer 1)



8.7 Examples of Use

8.7.1 Operation as interval timer (1)

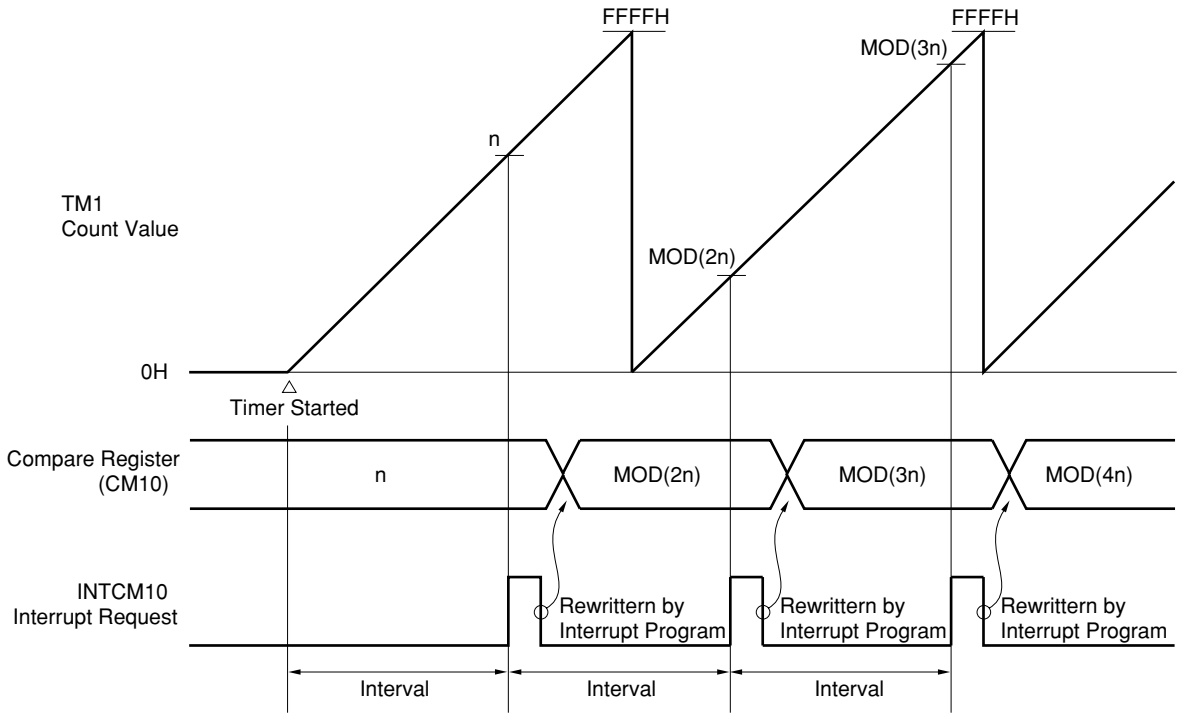
When timer register 1 (TM1) is made free-running and a fixed value is added to the compare register (CM1n: n = 0, 1) in the interrupt processing routine, TM1 operates as an interval timer with the added fixed value as the cycle (refer to **Figure 8-14**).

This interval timer can count in the range shown in Table 8-1 (internal system clock f_{CLK} = 16 MHz).

Because TM1 has two compare registers, interval timers of two types of cycles can be created.

Figure 8-15 shows the set contents of the control registers, Figure 8-16 shows how to set the registers, and Figure 8-17 shows the processing in an interrupt routine, where compare register CM10 is used.

Figure 8-14. Timing of Interval Timer Operation (1)



Remark Interval time = n × x/f_{CLK}

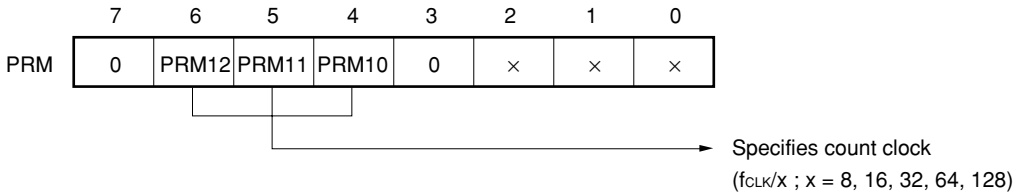
y ≤ n ≤ FFFFH

x = 4, 8, 16, 32, 64

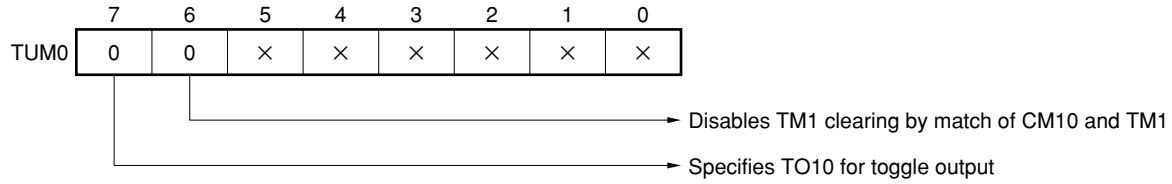
y is limited by the data transfer processing time. Consider the processing time of the interrupt used or the macro service processing time (refer to **Table 14-11 Interrupt Acceptance Processing Time** and **Table 14-12 Macro Service Processing Time**).

Figure 8-15. Control Register Settings for Interval Timer Operation (1)

(a) Prescaler mode register (PRM)



(b) Timer unit mode register 0 (TUM0)



× : don't care

Figure 8-16. Setting Procedure of Interval Timer Operation (1)

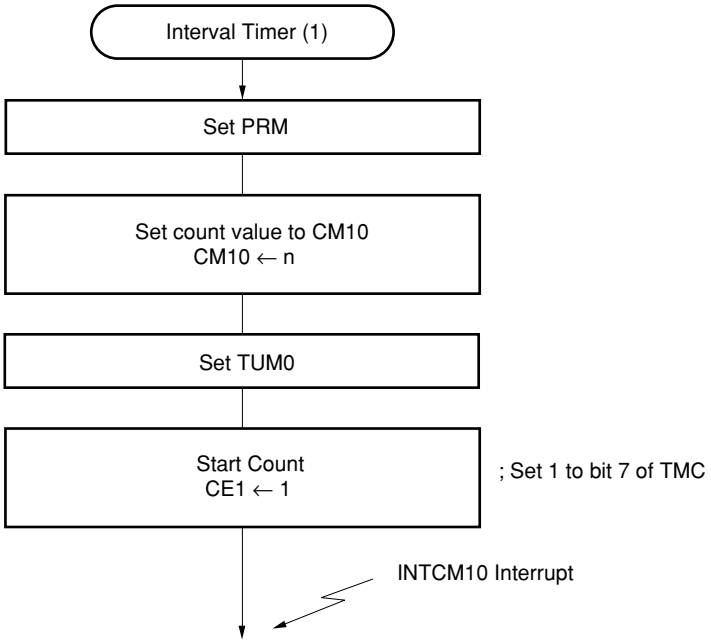
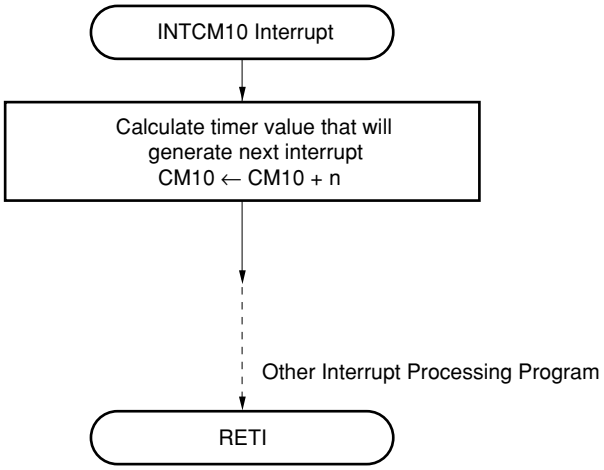


Figure 8-17. Interrupt Request Processing of Interval Timer Operation (1)



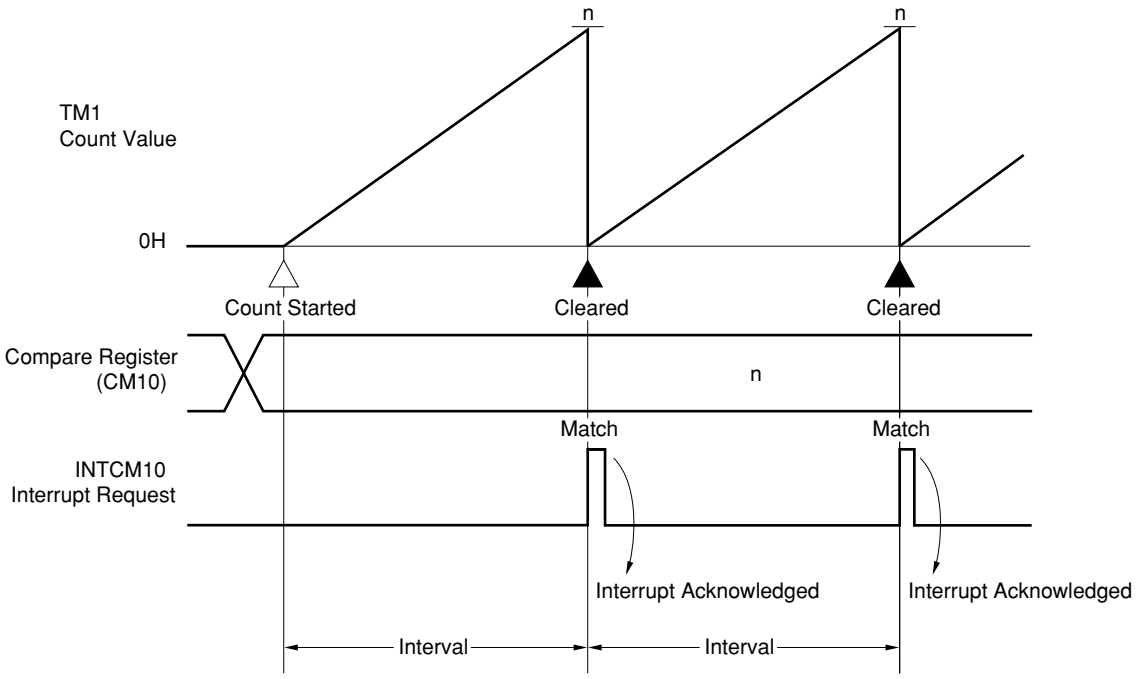
8.7.2 Operation as interval timer (2)

TM1 operates as an interval timer that generates interrupts repeatedly with the preset count time as the interval (refer to **Figure 8-18**).

This interval timer can count in the range shown in Table 8-1 (internal system clock f_{CLK} = 16 MHz)

The control register settings are shown in Figure 8-19, and the setting procedure in Figure 8-20.

Figure 8-18. Timing of Interval Timer Operation (2)



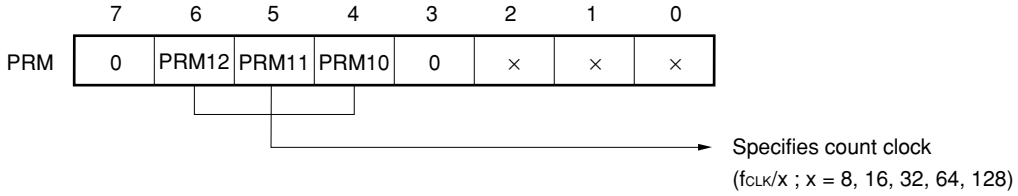
Remark Interval = $(n+1) \times x/f_{CLK}$

$0 \leq n \leq$ FFFFH

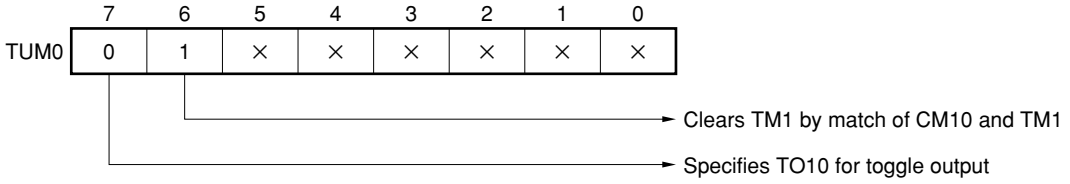
x = 8, 16, 32, 64, 128

Figure 8-19. Control Register Settings for Interval Timer Operation (2)

(a) Prescaler mode register (PRM)

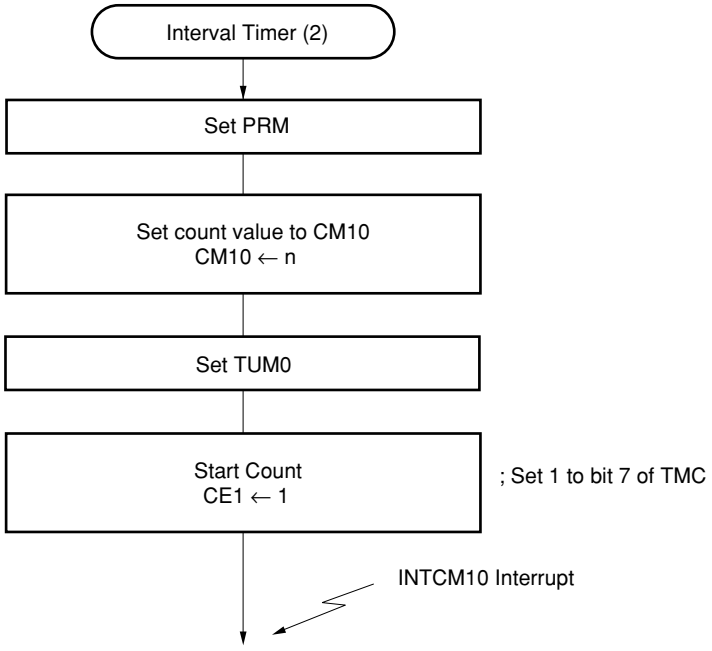


(b) Timer unit mode register 0 (TUM0)



× : don't care

Figure 8-20. Setting Procedure of Interval Timer Operation (2)

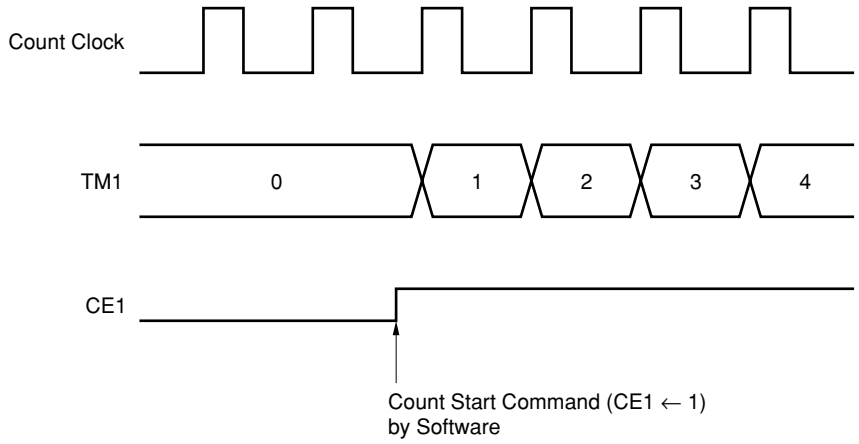


8.8 Cautions

(1) The prescaler uses one time base in common with all the timers (timers 0 and 1, timers/counters 2 and 3, and timer 4). If one of the timers sets the CE bit to “1”, the time base starts counting. If another timer sets the CE bit to “1” while one timer operates, the first count clock of the timer may be shortened because the time base has already started counting.

For example, when using timer/counter 1 as an interval timer, the first interval time is shortened by up to 1 count clock. The second and those that follow are at the specified interval.

Figure 8-21. Operation When Counting Is Started



(2) While timer 1 is operating (while the CE1 bit of the timer mode control register (TMC) is set), malfunctioning may occur if the contents of the following registers are rewritten. This is because it is undefined which takes precedence in a contention, the change in the hardware functions due to rewriting the register, or the change in the status because of the function before rewriting.

Therefore, be sure to stop the counter operation for the sake of safety before rewriting the contents of the following registers.

- Timer unit mode register 0 (TUM0)
- Timer output control register 1 (TOC1)
- Prescaler mode register (PRM)

(3) If the contents of the compare register (CM1n: n = 0, 1) matches with those of TM1 when an instruction that stops timer register 1 (TM1) operation is executed, the counting operation of TM1 stops, but an interrupt request is generated.
In order not to generate the interrupt when stopping the operation of TM1, mask the interrupt in advance by using the interrupt mask register before stopping TM1.

Example

Program that may generate interrupt request

```
        ⋮
CLR1  CE1
OR    MK0H, #0CH   ← Interrupt request from timer 1 occurs between these instructions
        ⋮
```

Program that does not generate interrupt request

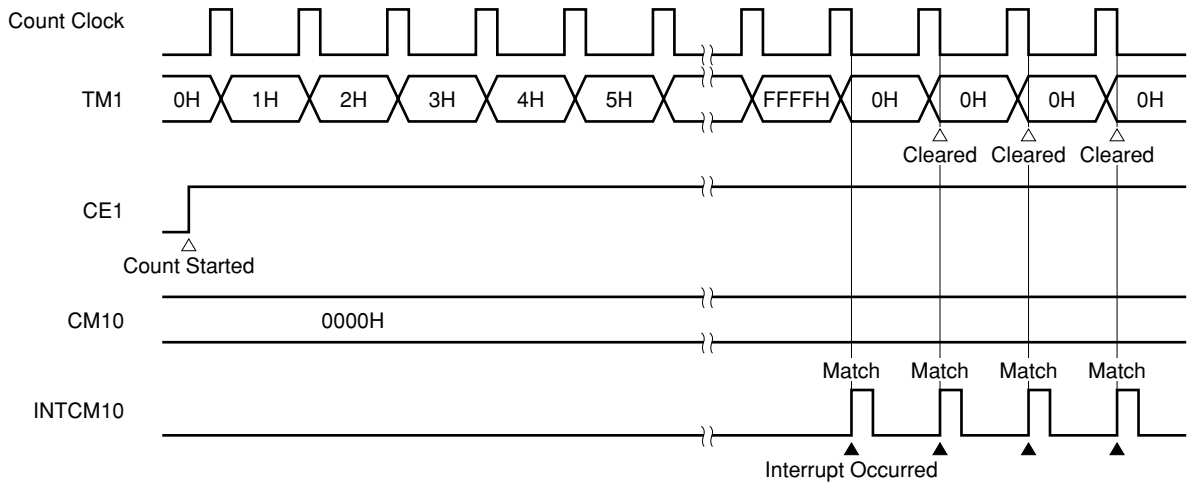
```
        ⋮
OR    MK0H, #0CH   ← Disables interrupt from timer 1
CLR1  CE1
CLR1  CMIF10       ← Clears interrupt request flag from timer 1
CLR1  CMIF11
        ⋮
```

(4) A match between the timer register 1 (TM1) and compare register (CM1n: n = 0, 1) is detected only when TM1 is incremented. Therefore, the interrupt request is not generated even if the same value as TM1 is written to CM1n, and the timer output (TO1n: n = 0, 1) does not change.

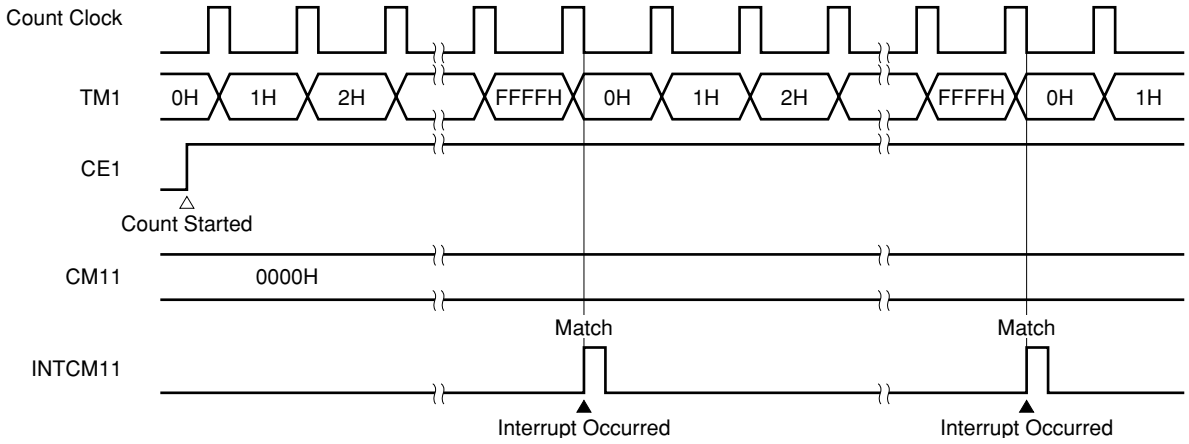
(5) When the compare register (CM10, CM11) is set to 0000H, the compare operation is performed after counting by TM1. Therefore, the interrupt due to a match (INTCM10, INTCM11) does not occur immediately after counting has been started. If CM1n (n = 0, 1) is set to 0000H, TM1 counts up to FFFFH, the timer overflows, and the interrupt due to a match INTCM1n (n = 0, 1) occurs.

Figure 8-22. Operation When Compare Register (CM10, CM11) Is Set to 0000H

(a) CM10



(b) CM11



(6) If the timer output is enabled when the active level is changed, the output level of pins may change momentarily. To prevent this, enable the timer output after the active level have been changed.

(7) To change the active level specification (ALV1n bit (n = 0, 1) of the timer output control register 1 (TOC1)), change the active level specification after the timer output of the corresponding timer output pins has been disabled.

CHAPTER 9 TIMER 4

9.1 Function

Timer 4 is a 16-bit timer.

This timer functions as an interval timer.

When used as an interval timer, timer 4 generates an internal interrupt at a predetermined interval.

Table 9-1. Interval Time of Timer 4

Minimum Interval Time	Maximum Interval Time	Resolution
$4/f_{CLK}$ (0.25 μs)	$2^{16} \times 4/f_{CLK}$ (16.4 ms)	$4/f_{CLK}$ (0.25 μs)
$8/f_{CLK}$ (0.5 μs)	$2^{16} \times 8/f_{CLK}$ (32.8 ms)	$8/f_{CLK}$ (0.5 μs)
$16/f_{CLK}$ (1.0 μs)	$2^{16} \times 16/f_{CLK}$ (65.5 ms)	$16/f_{CLK}$ (1.0 μs)
$32/f_{CLK}$ (2.0 μs)	$2^{16} \times 32/f_{CLK}$ (131 ms)	$32/f_{CLK}$ (2.0 μs)
$64/f_{CLK}$ (4.0 μs)	$2^{16} \times 64/f_{CLK}$ (262 ms)	$64/f_{CLK}$ (4.0 μs)

( ): at f_{CLK} = 16 MHz

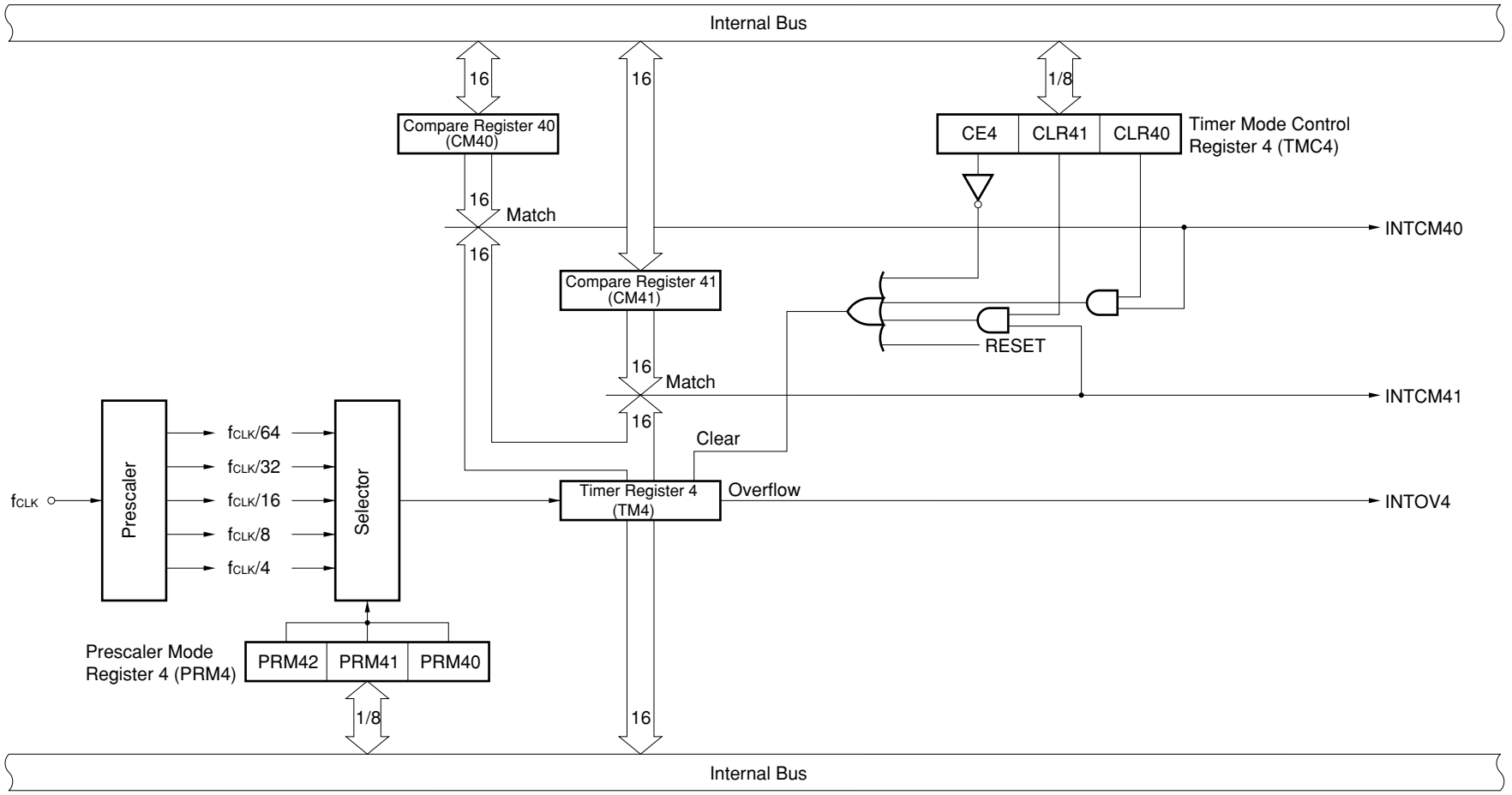
9.2 Configuration

Timer 4 consists of the following registers:

- Timer register (TM4) × 1
- Compare register (CM4n) × 2 (n = 0, 1)

Figure 9-1 shows the block diagram of timer 4.

Figure 9-1. Block Diagram of Timer 4



(1) Timer register 4 (TM4)

TM4 is a timer register that counts up the count clock specified by the prescaler mode register 4 (PRM4).
Counting of this timer register is enabled or disabled by the timer mode control register 4 (TMC4).
The timer register can be only read by using a 16-bit manipulation instruction. When $\overline{\text{RESET}}$ is input, TM4 is cleared to 0000H and stops counting.

(2) Compare registers (CM40, CM41)

CM4n (n = 0, 1) is a 16-bit register that holds the contents determining the cycle of the interval timer operation.
When the contents of CM4n matches with the contents of TM4, an interrupt request (INTCM4n: n = 0, 1) is generated.
The count value of TM4 can be cleared when its value matches with the contents of CM4n.
These compare registers can be read or written by using 16-bit manipulation instructions. When $\overline{\text{RESET}}$ is input, their contents are undefined.

(3) Prescaler

The prescaler generates a count clock by dividing the internal system clock. The clock generated by the prescaler is selected by the selector, and TM4 performs the count operation by using this clock as a count clock.

(4) Selector

The selector selects one of the five signals generated by dividing the internal system clock as the count clock of TM4.

9.3 Timer 4 Control Register

(1) Timer mode control register 4 (TMC4)

TMC 4 is a register that controls the count and clear operations of timer register 4 (TM4).

This register can be read or written by using an 8-bit manipulation instruction or a bit manipulation instruction. Figure 9-2 shows the format of TMC4.

When $\overline{\text{RESET}}$ is input, the value of this register is cleared to 00H.

Figure 9-2. Format of Timer Mode Control Register 4 (TMC4)

Address: 0FF37H On reset: 00H R/W

	7	6	5	4	③	2	1	0
TMC4	0	0	0	0	CE4	0	CLR41	CLR40

CE4	Controls Count Operation of TM4
0	Clears and stops counting
1	Enables counting operation

CLR41	Clear Operation of TM4 by Match with CM41
0	Disables (free running mode)
1	Enables (interval timer mode)

CLR40	Clear Operation of TM4 by Match with CM40
0	Disables (free running mode)
1	Enables (interval timer mode)

(2) Prescaler mode register 4 (PRM4)

PRM4 is a register that specifies the count clock of timer register 4 (TM4).

This register can be read or written by using an 8-bit manipulation instruction. Figure 9-3 shows the format of PRM4.

When $\overline{\text{RESET}}$ is input, the value of this register is cleared to 00H.

Figure 9-3. Format of Prescaler Mode Register 4 (PRM4)

Address: 0FF3AH On reset: 00H R/W

	7	6	5	4	3	2	1	0
PRM4	0	0	0	0	0	PRM42	PRM41	PRM40

(f_{CLK} = 16 MHz)

PRM42	PRM41	PRM40	Specifies Count Clock of TM4.	
			Count Clock [Hz]	Resolution [μs]
0	0	0	f_{CLK}/4	0.25
0	0	1	f_{CLK}/8	0.5
0	1	0	f_{CLK}/16	1.0
0	1	1	f_{CLK}/32	2.0
1	0	0	f_{CLK}/64	4.0
Other			Setting prohibited	

Remark f_{CLK}: internal system clock

9.4 Operation of Timer Register 4 (TM4)

9.4.1 Basic operation

Timer 4 counts up by using the count clock specified by the prescaler mode register 4 (PRM4).

Counting is enabled or disabled by the CE4 bit of the timer mode control register 4 (TMC4). When the CE4 bit is set (1) by software, TM4 is set to 0001H at the first count clock, and starts counting up. When the CE4 bit is cleared (0) by software, TM4 is immediately cleared to 0000H, and stops generation of the match signal.

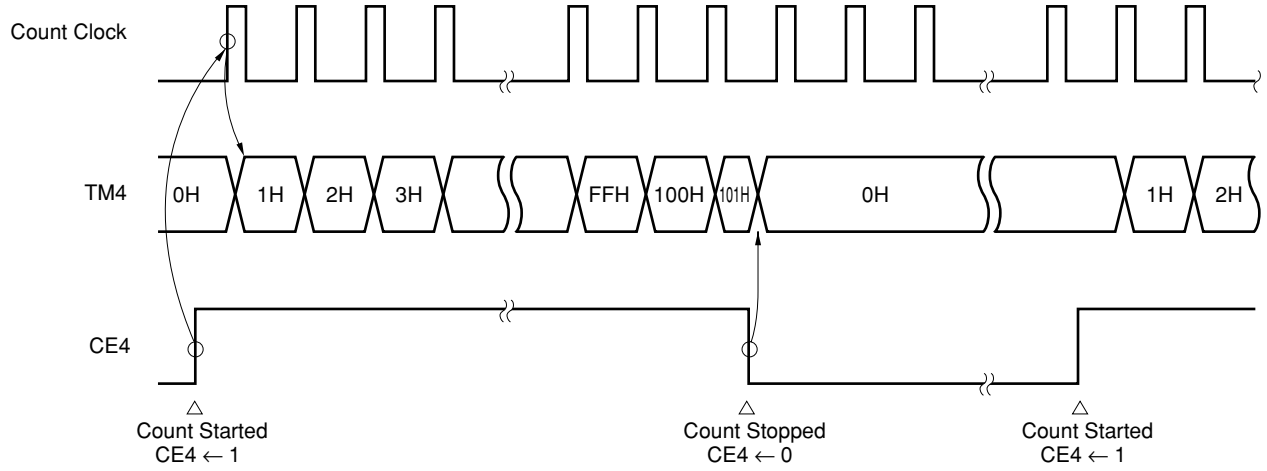
If the CE4 bit is set (1) while it has been already set (1), TM4 is not cleared but continues counting.

If a count clock is input when TM4 reaches FFFFH, TM4 is cleared to 0000H, and an overflow interrupt (INTOV4) occurs.

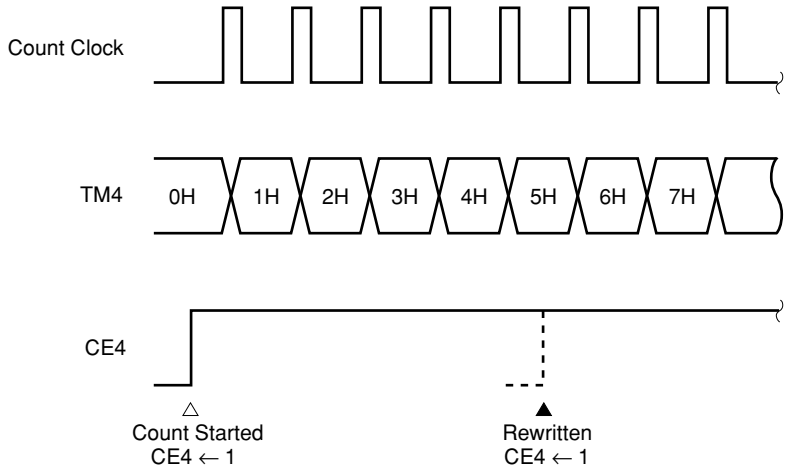
When $\overline{\text{RESET}}$ is input, TM4 is cleared to 0000H and stops counting.

Figure 9-4. Basic Operation of Timer Register 4 (TM4)

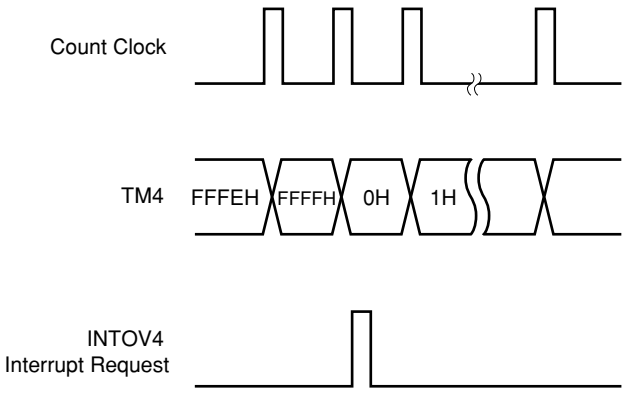
(a) When counting starts, stops, and then starts again



(b) If CE4 bit is set to "1" again after counting has been started



(c) Operation when TM4 is FFFFH

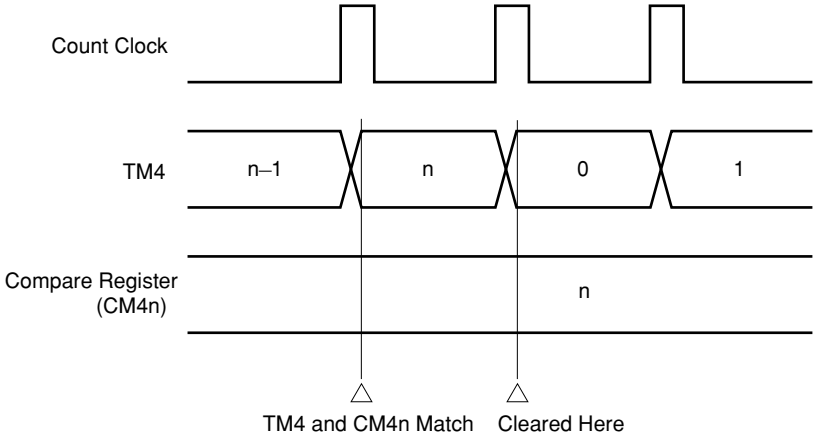


9.4.2 Clear operation

(1) Clear operation by match with compare register

Timer register 4 (TM4) can be automatically cleared when its value matches with the value of a compare register (CM4n: n = 0, 1). When a clearance source arises, TM3 is cleared to 0000H on the next count clock. Therefore, even if a clearance source arises, the value at the point at which the clearance source arose is retained until the next count clock arrives.

Figure 9-5. TM4 Clear Operation by Match with Compare Register (CM40, CM41)



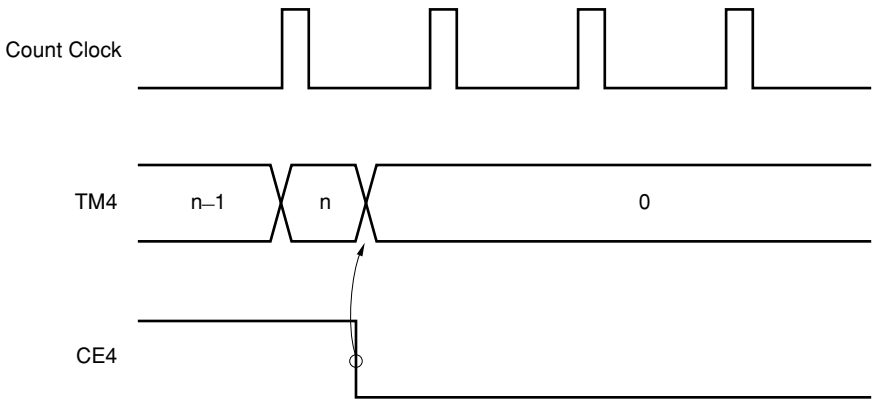
Remark n = 0, 1

(2) Clear operation by CE4 bit of timer mode control register 4 (TMC4)

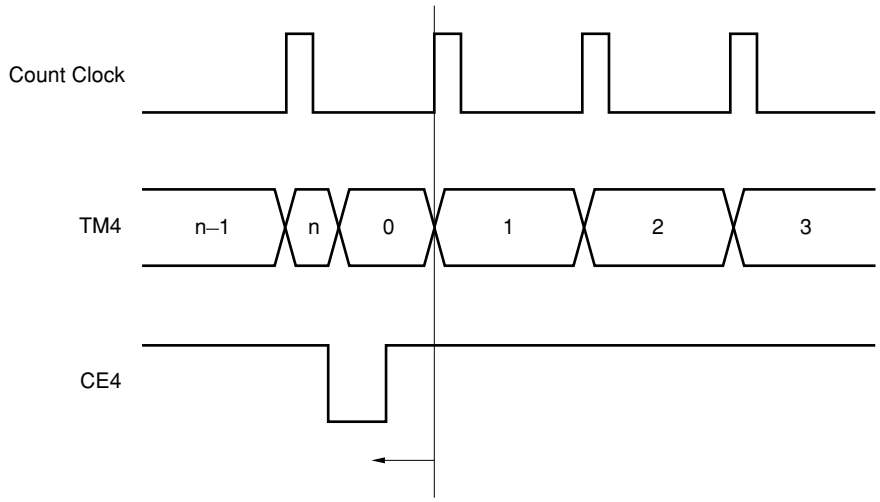
Timer register 4 (TM4) is also cleared when the CE4 bit of TMC4 is cleared (0) by software. The clear operation is performed following clearance (0) of the CE4 bit in the same way.

Figure 9-6. Clear Operation of TM4 When CE4 Bit is Cleared (0)

(a) Basic operation

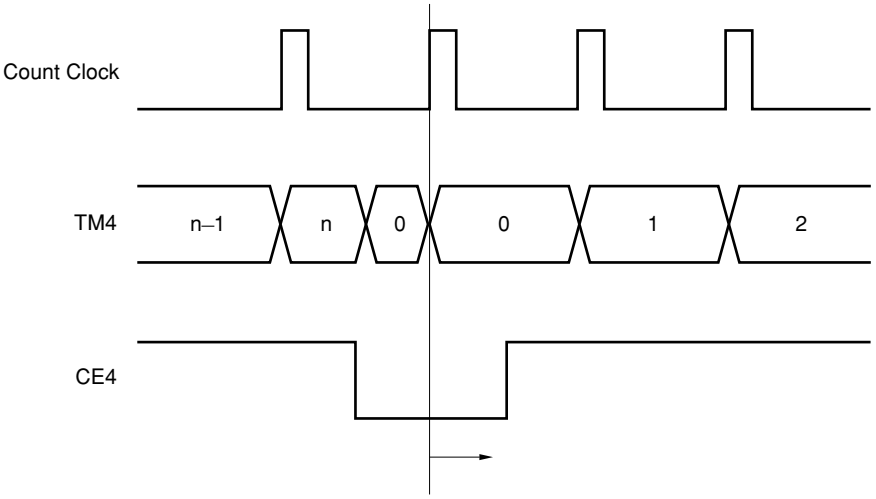


(b) Restart before count clock is input after clearance



If the CE4 bit is set (1) before this count clock, the count starts from 1 on this count clock

(c) Restart when count clock is input after clearance



If the CE4 bit is set (1) from this count clock onward, the count starts from 1 on the count clock after the CE4 bit is set (1).

9.5 Operation of Compare Register

Timer 4 performs a compare operation by comparing the value set to a compare register (CM40, CM41) with the count value of a timer register 4 (TM4).

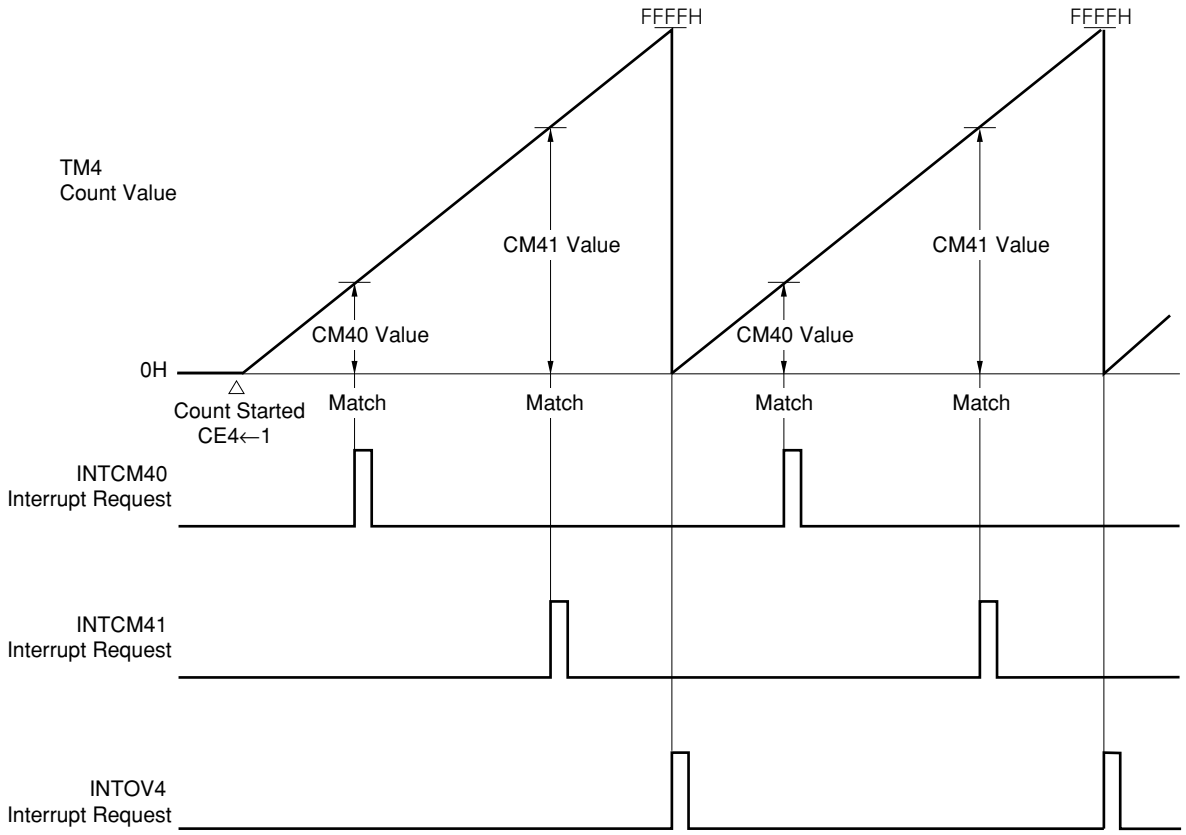
If the count value of TM4 matches with the value set in advance to CM4n (n = 0, 1) as a result of counting by TM4, an interrupt request (INTCM4n: n = 0, 1) is generated.

Moreover, the contents of TM4 can be cleared after it has matched with the value of CM4n, so that TM4 can operate as an interval timer that repeatedly counts the value set to CM4n.

Table 9-2. Interrupt Request Signal from Compare Register (timer 4)

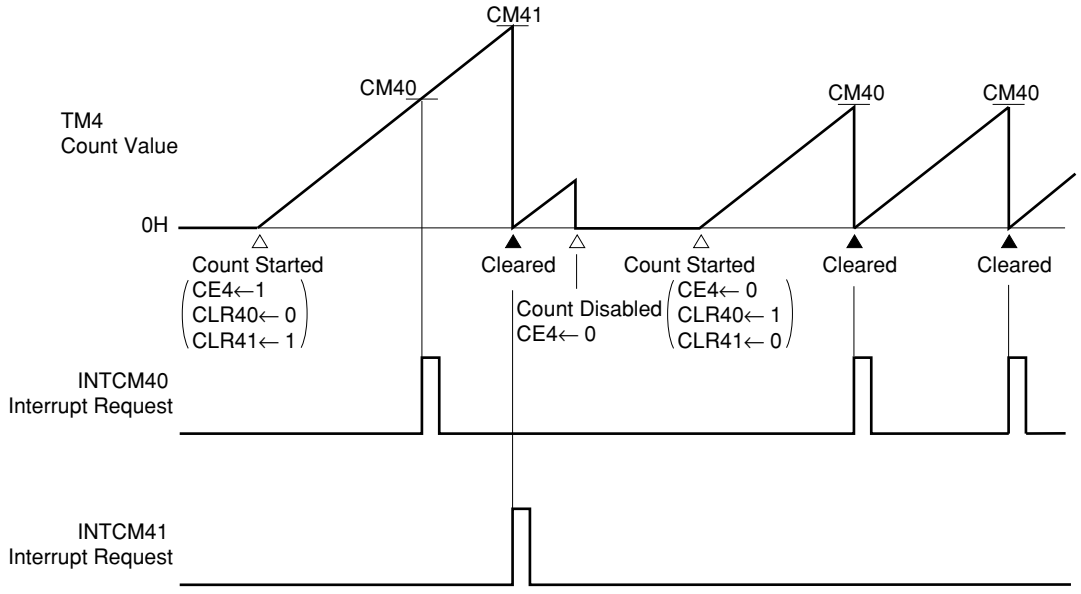
Compare Register	Interrupt Request Signal
CM40	INTCM40
CM41	INTCM41

Figure 9-7. Compare Operation (timer 4)



Remark CLR40 = 0, CLR41 = 0

Figure 9-8. TM4 Clearance after Match Detection



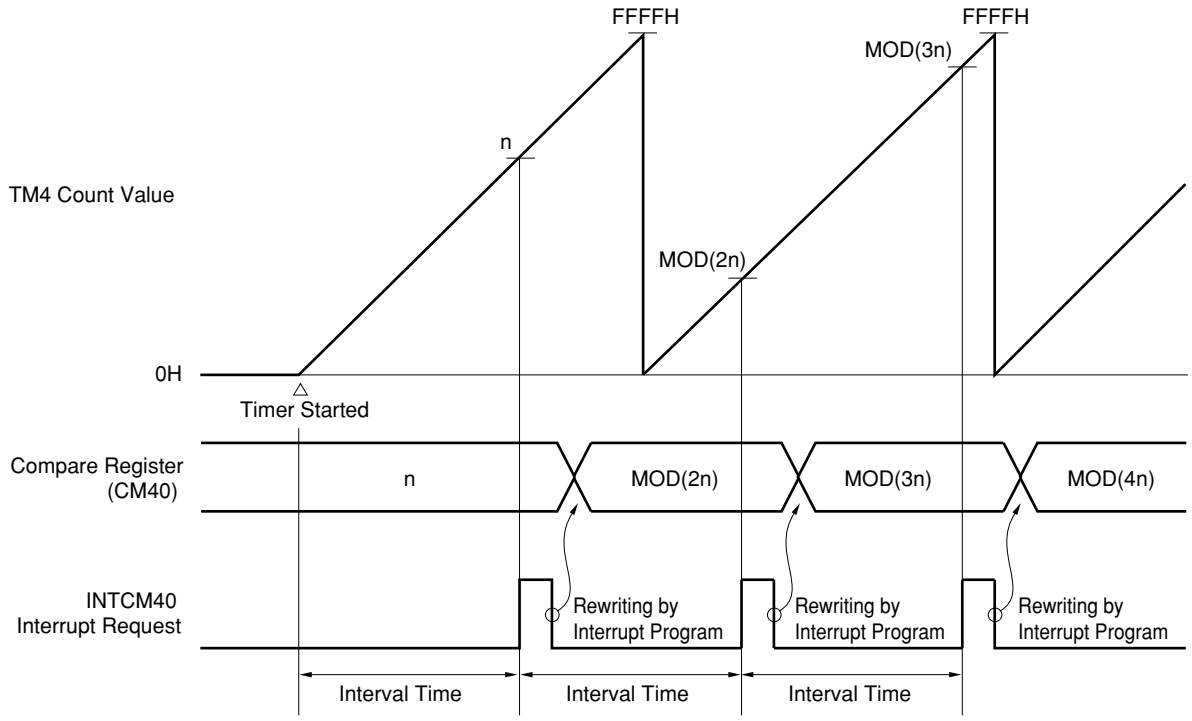
9.6 Example of Use

9.6.1 Operation as interval timer (1)

By setting the timer register 4 (TM4) in the free running mode and adding a specific value to a compare register (CM4n: n = 0, 1) in an interrupt processing routine, TM4 can be used as an interval timer whose cycle is determined by the specific value to be added (refer to **Figure 9-9**).

Figure 9-10 shows the set contents of the control registers, Figure 9-11 shows how to set the control registers, and Figure 9-12 shows the processing in the interrupt routine, where compare register CM40 is used.

Figure 9-9. Timing of Interval Timer Operation (1)



Remark Interval time = $n \times x/f_{CLK}$

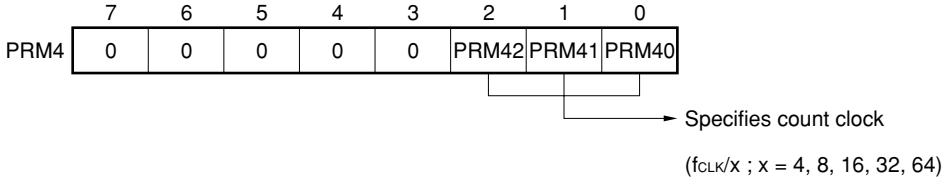
$y \leq n \leq FFFFH$

x = 4, 8, 16, 32, 64

y is limited by the data transfer processing time. Consider the processing time of the interrupt used or the macro service processing time (refer to **Table 14-11 Interrupt Acceptance Processing Time** and **Table 14-12 Macro Service Processing Time**).

Figure 9-10. Set Contents of Control Registers for Interval Timer Operation (1)

(a) Prescaler mode register 4 (PRM4)



(b) Timer mode control register 4 (TMC4)

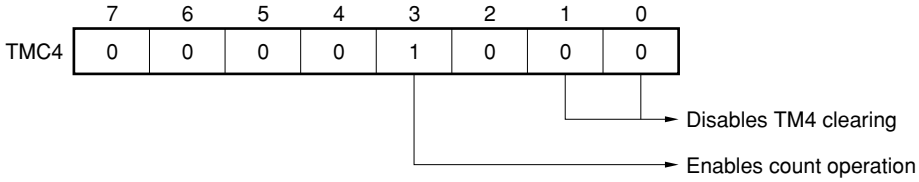


Figure 9-11. Setting Procedure of Interval Timer Operation (1)

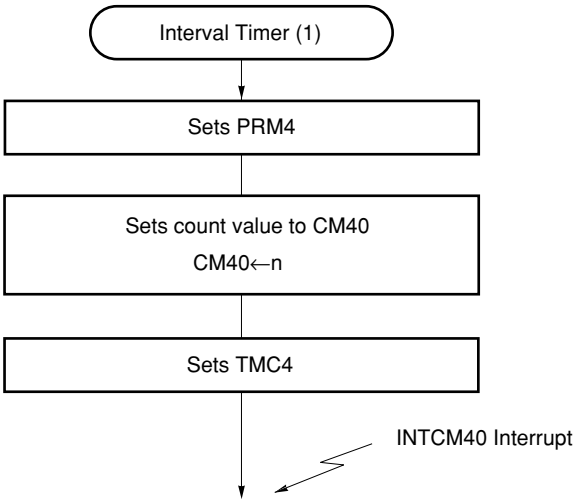
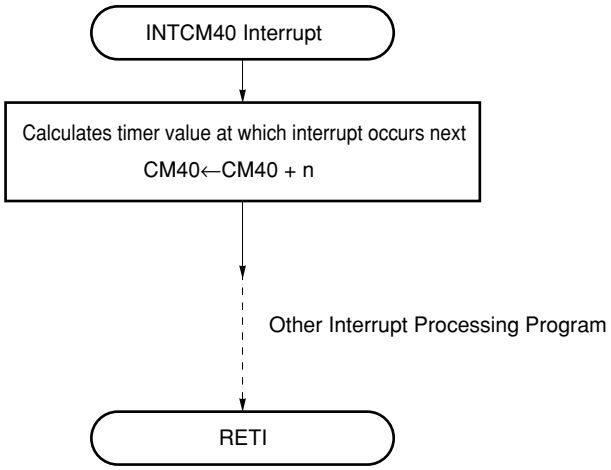


Figure 9-12. Interrupt Request Processing of Interval Timer Operation (1)

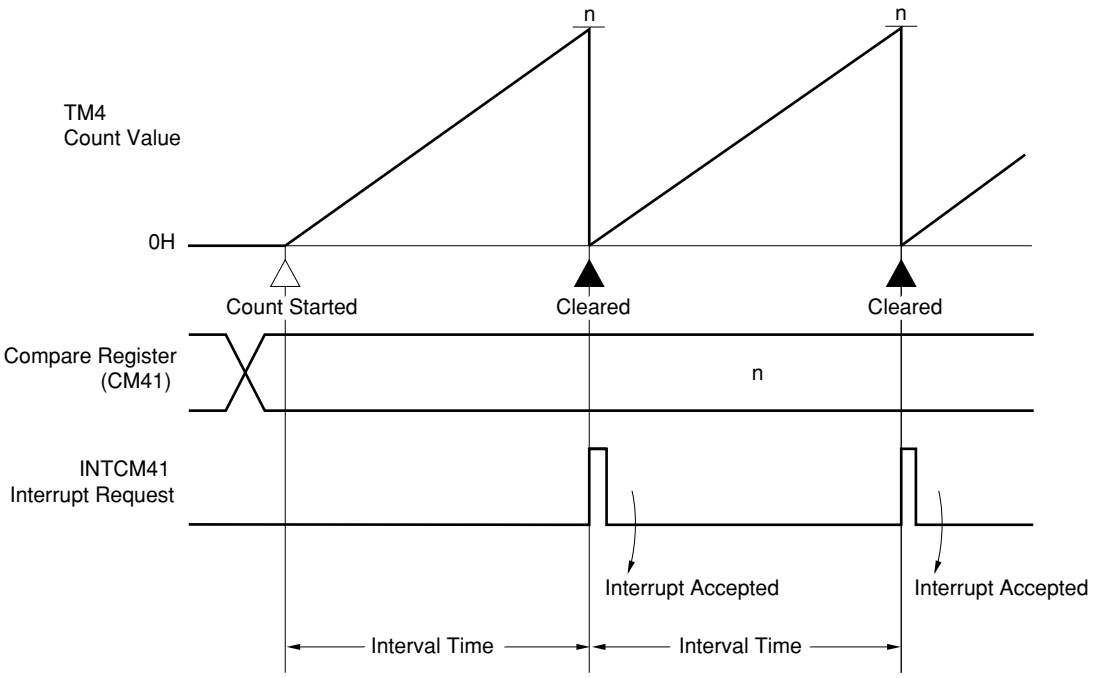


9.6.2 Operation as interval timer (2)

TM4 can be used as an interval timer that repeatedly generates an interrupt at interval determined by the count value set in advance (refer to **Figure 9-13**).

Figure 9-4 shows the set contents of the control registers, and Figure 9-15 shows how to set the control registers, where compare register CM41 is used.

Figure 9-13. Timing of Interval Timer Operation (2)



Remark Interval time = (n+1) × x/f_{CLK}

$0 \leq n \leq FFFFH$

x = 4, 8, 16, 32, 64

Figure 9-14. Set Contents of Control Register for Interval Timer Operation (2)

(a) Prescaler mode register 4 (PRM4)

(b) Timer mode control register 4 (TMC4)

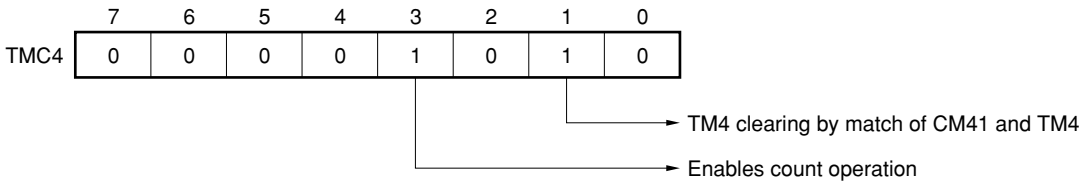
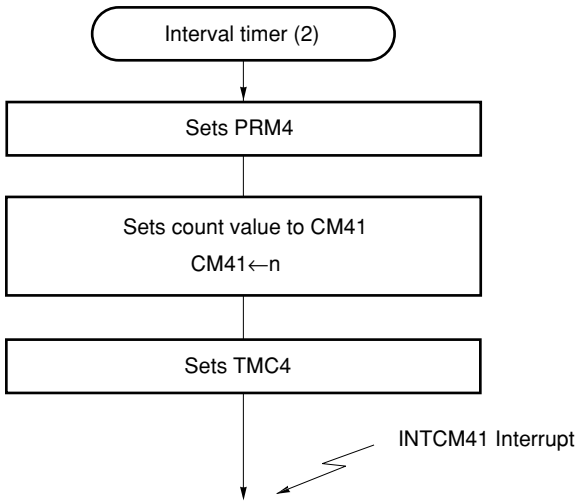


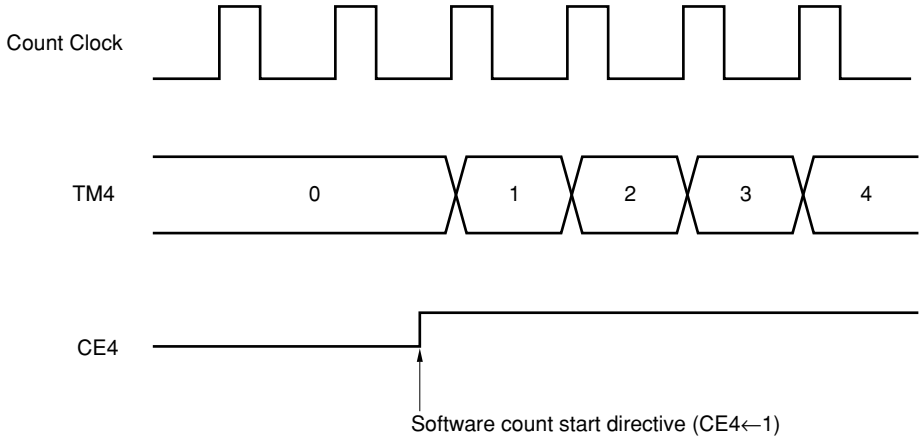
Figure 9-15. Setting Procedure of Interval Timer Operation (2)



9.7 Cautions

(1) The prescaler uses one time base commonly with all the timers (timers 0 and 1, timers/counters 2 and 3, and timer 4). If one of the timers sets the CE bit to “1”, the time base starts counting. If another timer sets the CE bit to “1” while one timer operates, the first count clock of the timer may be shortened because the time base has already started counting.
For example, if a timer/counter is used as an interval timer, the first interval will be shortened by up to one count clock. The second and subsequent intervals will be as specified.

Figure 9-16. Operation When Count Starts



(2) There is a possibility of misoperation if the next register contents are rewritten while the timer 4 is running (when the CE4 bit of the timer mode control register 4 (TMC4) is set). The misoperation occurs as there is no defined order of priority in the event of contention between the timings at which the hardware function changes due to a register rewrite and the status changes in the function prior to the rewrite.
When the contents of following registers are rewritten, counter operations must be stopped first to ensure stability.

- CLR40 and CLR41 bits of timer mode control register 4 (TMC4)
- Prescaler mode register 4 (PRM4)

(3) If the compare register (CM4n: n = 0, 1) and TM4 contents match when an instruction that stops timer register 4 (TM4) operation is executed, the TM4 count operation stops, but an interrupt request is generated.
If you do not want an interrupt to be generated when TM4 operation is stopped, interrupts should be masked by means of interrupt the mask register before stopping the TM4.

Example

Program in which an interrupt request may be generated

```
        ⋮
CLR1  CE4
OR    MK1L, #03H   ← Interrupt request generated by timer 4 here
        ⋮
```

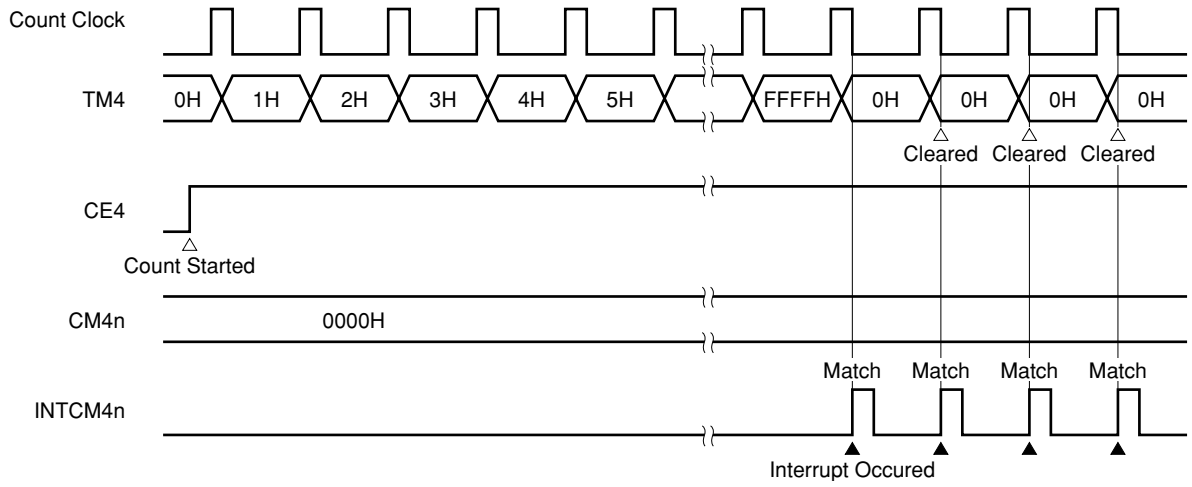
Program in which an interrupt request is not generated

```
        ⋮
OR    MK1L, #03H   ← Disables interrupts from timer 4
CLR1  CE4
CLR1  CMIF40       ← Clears timer 4 interrupt request flag
CLR1  CMIF41
        ⋮
```

(4) Match between timer register 4 (TM4) and compare register (CM4n: n = 0, 1) is detected only when TM4 is incremented. Therefore, the interrupt request is not generated even if the same value as TM4 is written to CM4n.

(5) If a compare register (CM40, CM41) is set to 0000H, the compare operation is performed after counting has been completed. Therefore, the interrupt due to a match (INTCM40, INTCM41) does not occur immediately after counting has been started. If CM4n (n = 0, 1) is set to 0000H, TM4 counts up to FFFFH, overflows, and then the interrupt due to a match INTCM4n (n = 0, 1) occurs.

Figure 9-17. Operation When Compare Register (CM40, CM41) Is Set to 0000H



Remark n = 0, 1

CHAPTER 10 WATCHDOG TIMER FUNCTION

The watchdog timer is a timer that detects inadvertent program loops.

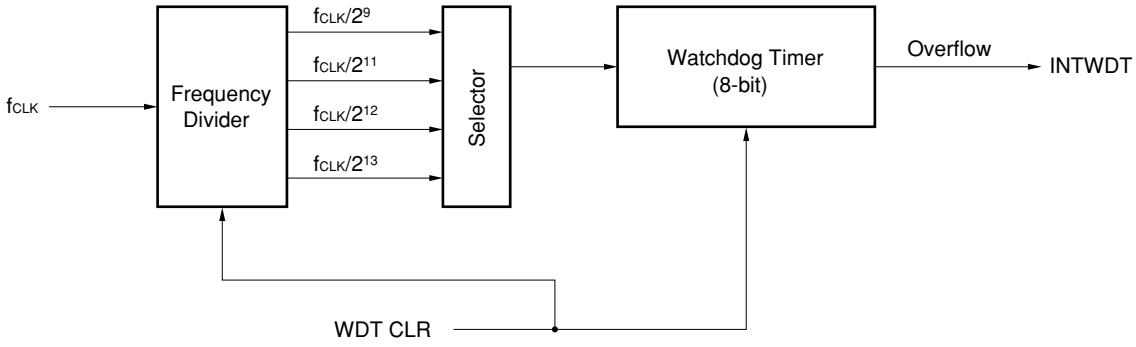
Watchdog timer interrupts are used to detect system or program errors. For this purpose, instructions that clear the watchdog timer (start the count) within a given period are inserted at various places in a program.

If an instruction that clears the watchdog timer is not executed within the set time and the watchdog timer overflows, a watchdog timer interrupt (INTWDT) is generated and a program error is reported.

10.1 Configuration

The watchdog timer block diagram is shown in Figure 10-1.

Figure 10-1. Block Diagram of Watchdog Timer



10.2 Watchdog Timer Mode Register (WDM)

The WDM is an 8-bit register that controls the watchdog timer operation.

To prevent erroneous clearing of the watchdog timer by an inadvertent program loop, writing can only be performed by a dedicated instruction. This dedicated instruction, MOV WDM,#byte, has a special code configuration (4 bytes), and a write is not performed unless the 3rd and 4th bytes of the operation code are mutual complements.

If the 3rd and 4th bytes of the operation code are not complements, a write is not performed and an operand error interrupt is generated. In this case, the return address saved in the stack area is the address of the instruction that was the source of the error, and thus the address that was the source of the error can be identified from the return address saved in the stack area.

If recovery from an operand error is simply performed by means of an RETB instruction, an endless loop will result.

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

Other write instructions (MOV WDM, A, AND WDM, #byte, SET1 WDM.7, etc.) are ignored and do not perform any operation. That is, a write is not performed to the WDM, and an interrupt such as an operand error interrupt is not generated.

After a system reset ($\overline{\text{RESET}}$ input), once the watchdog timer has been started (by setting (1) the RUN bit), the WDM contents cannot be changed. The watchdog timer can only be stopped by a reset, but can be cleared at any time with a dedicated instruction.

The WDM can be read at any time by a data transfer instruction.

$\overline{\text{RESET}}$ input clears the WDM to 00H.

The WDM format is shown in Figure 10-2.

Figure 10-2. Format of Watchdog Timer Mode Register (WDM)

Address: 0FFC2H On reset: 00H R/W

	7	6	5	4	3	2	1	0
WDM	RUN	0	0	PRC	0	WDI2	WDI1	0

RUN	Specifies Operation of Watchdog Timer
0	Stops watchdog timer
1	Clears watchdog timer to start counting

PRC	Priority of Interrupt Request of Watchdog Timer
0	Interrupt request of watchdog timer < interrupt request of NMI pin input
1	Interrupt request of watchdog timer > interrupt request of NMI pin input

WDI2	WDI1	Count Clock	Overflow Time [ms]	
			f_{CLK} = 12.5 MHz	f_{CLK} = 16.0 MHz
0	0	$f_{CLK}/2^9$	10.5	8.2
0	1	$f_{CLK}/2^{11}$	41.9	32.8
1	0	$f_{CLK}/2^{12}$	83.9	65.5
1	1	$f_{CLK}/2^{13}$	167.8	131.1

Remark f_{CLK}: internal system clock

Cautions 1. The watchdog timer mode register (WDM) can only be written to with a dedicated instruction (MOV WDM, #byte).

2. The same value should be written each time in writes to the WDM to set (1) the RUN bit. The contents written the first time cannot be changed even if a different value is written.

3. Once the RUN bit has been set (1), it cannot be reset (0) by software.

10.3 Operation

10.3.1 Count operation

The watchdog timer is cleared, and the count started, by setting (1) the RUN bit of the watchdog timer mode register (WDM). When overflow time specified by the WDI2 and WDI1 bits of WDM has elapsed after the RUN bit has been set (1), a non-maskable interrupt (INTWDT) is generated.

If the RUN bit is set (1) again before the overflow time elapses, the watchdog timer is cleared and the count operation is started again.

10.3.2 Interrupt priorities

The watchdog timer interrupt (INTWDT) is a non-maskable interrupt. Other non-maskable interrupts are interrupts from the NMI pin (NMI). The order of acknowledgment when an INTWDT interrupt and NMI interrupt are generated simultaneously can be specified by the setting of bit 4 of the watchdog timer mode register (WDM).

Even if INTWDT is generated while the NMI processing program is executed when NMI acknowledgement is specified to take precedence, INTWDT is not acknowledged until completion of execution of the NMI processing program.

10.4 Cautions

10.4.1 General cautions on use of watchdog timer

(1) The watchdog timer is one means of detecting inadvertent program loops, but it cannot detect all inadvertent program loops. Therefore, in equipment that requires a high level of reliability, you should not rely on the on-chip watchdog timer alone, but should use external circuitry for early detection of inadvertent program loops, to enable processing to be performed that will restore the normal state or establish a stable state and then stop the operation.

(2) The watchdog timer cannot detect inadvertent program loops in the following cases.

<1> If watchdog timer clearance is performed in the timer interrupt processing program
<2> If cases where an interrupt request or macro service is held pending (refer to **14.9**) occur consecutively
<3> If the watchdog timer is cleared periodically when the program is looping inadvertently due to an error in the program logic (if each module of the program functions normally but the overall program does not)
<4> If the watchdog timer is periodically cleared by a group of instructions executed when an inadvertent program loop occurs
<5> If the STOP mode, HALT mode, or IDLE mode is entered as the result of an inadvertent program loop
<6> If an inadvertent program loop of watchdog timer also occurs in the event of CPU hang up due to external noise

In cases <1>, <2> and <3> the program can be amended to allow detection to be performed.
In case <4>, the watchdog timer can only be cleared by a 4-byte dedicated instruction. Similarly, in case <5>, the STOP mode, HALT mode, or IDLE mode cannot be set unless a 4-byte dedicated instruction is used. For state <2> to be entered as the result of an inadvertent program loop, 3 or more consecutive bytes of data must comprise a specific pattern (e.g. BT PSWL. bit, $$, etc.). Therefore, the establishment of state <2> as the result of <4>, <5> or an inadvertent program loop is likely to be extremely rare.

10.4.2 Cautions on μPD784054 watchdog timer

(1) The watchdog timer mode register (WDM) can only be written to with a dedicated instruction (MOV WDM, #byte).

(2) The same value should be written each time in writes to the watchdog timer mode register (WDM) to set (1) the RUN bit. The contents written the first time cannot be changed even if a different value is written.

(3) Once the RUN bit has been set (1), it cannot be reset (0) by software.

CHAPTER 11 A/D CONVERTER

The μPD784054 incorporates an analog/digital (A/D) converter with 16 multiplexed analog inputs (ANI0 to ANI15).

The successive approximation conversion method is used, and the conversion result is held in the 10-bit A/D conversion result register (ADCR0 to ADCR7). This allows fast, high-precision conversion to be performed (conversion time of 13 μs when f_{CLK} = 16 MHz and high-speed conversion is used).

There are two modes for starting A/D conversion, as follows:

- Hardware start : Conversion started by trigger input (INTP4).
- Software start : Conversion started in accordance with A/D converter mode register (ADM) bit setting.

After start-up, there are two operating modes, as follows:

- Scan mode : Multiple analog inputs are selected in order, and conversion data is obtained from all pins.
- Select mode : One pin is used as the analog input, and conversion values are obtained in succession.

Stoppage of all the above modes and conversion operations is specified by the ADM register.

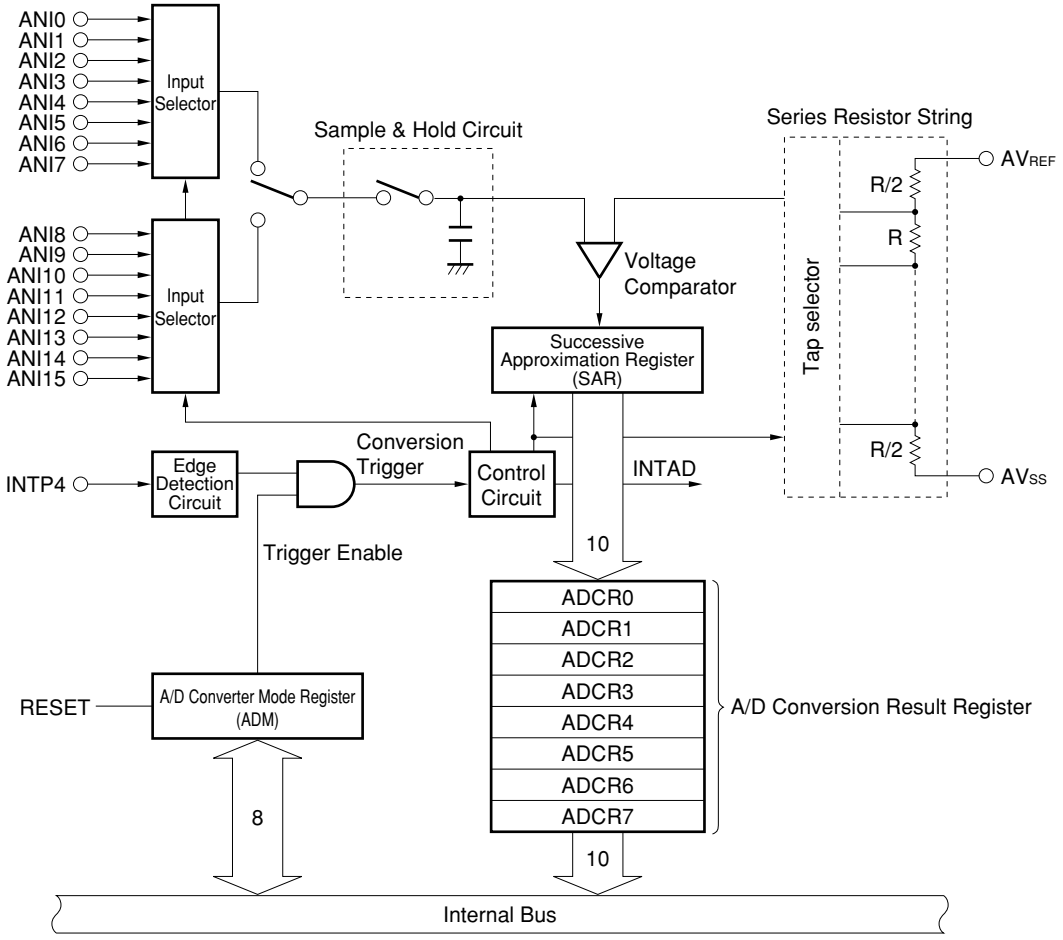
In each mode, the conversion result is held in ADCRn (n = 0 to 7) each time A/D conversion has been completed. When A/D conversion has been completed, an A/D conversion end interrupt request (INTAD) is generated. This interrupt can start a macro service that automatically transfers data by hardware.

11.1 Configuration

Figure 11-1 shows the block diagram of the A/D converter.

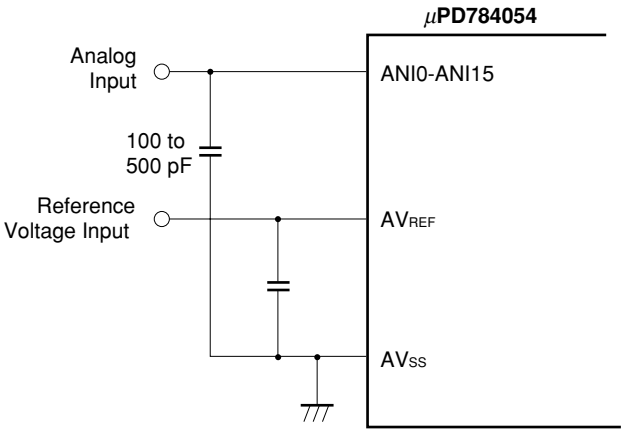
The high-order 8 channels (ANI8 to ANI15) and low-order 8 channels (ANI0 to ANI7) of the A/D converter are selected by using the A/D converter mode register (ADM).

Figure 11-1. Block Diagram of A/D Converter



Cautions 1. A capacitor should be connected between the analog input pins (ANI0 to ANI15) and AV_{SS} and between the reference voltage input pin (AV_{REF}) and AV_{SS} to prevent misoperation due to noise. Be sure to connect the capacitor as closely to ANI0 to ANI15 and AV_{REF} as possible.

Figure 11-2. Example of Capacitor Connection on A/D Converter Pins



2. A voltage outside the range AV_{SS} to AV_{REF} should not be applied to pins used as A/D converter input pins. Refer to 11.6 Cautions for details.

(1) Input circuit

The input circuit selects the analog input in accordance with the specification of the A/D converter mode register (ADM), and sends the analog input to the sample & hold circuit according to the operating mode,

(2) Sample & hold circuit

The sample & hold circuit samples the analog inputs arriving sequentially one by one and holds the analog input in the process of A/D conversion.

(3) Voltage comparator

The voltage comparator determines the voltage difference between the analog input and the series resistor string value tap.

(4) Series resistor string

The series resistor string is used to generate voltages that match the analog inputs.

The series resistor string is connected between the A/D converter reference voltage pin (AV_{REF}) and the A/D converter GND pin (AV_{SS}). To provide 1024 equal voltage steps between the two pins, it is made up of 1023 equal resistors and two resistors with half that resistance value.

The series resistor string voltage tap is selected by a tap selector controlled by the successive approximation register (SAR).

(5) SAR: Successive Approximation Register

The SAR is a 10-bit register in which the data for which the series resistor string voltage tap value matches the analog input voltage value is set bit by bit starting from the most significant bit (MSB).

When data has been set up to the least significant bit (LSB) of the SAR (when A/D conversion is completed), the SAR contents (conversion result) are stored in the A/D conversion result register (ADCRn: n = 0-7).

(6) Edge detection circuit

The edge detection circuit detects a valid edge from the interrupt request input pin (INTP4) input, and generates an external interrupt request signal (INTP4) and A/D conversion operation external trigger.

The INTP4 pin input valid edge is specified by external interrupt mode register 1 (INTM1) (refer to **Figure 13-2**).

External trigger enabling/disabling is set by means of the A/D converter mode register (ADM) (refer to **11.2 A/D Converter Mode Register (ADM)**).

11.2 A/D Converter Mode Register (ADM)

ADM is an 8-bit register that controls A/D converter operations.

The ADM register can be read or written to with an 8-bit manipulation instruction or bit manipulation instruction. Its format is shown in Figure 11-3.

Bits 0 to 2 (ANIS0 to ANIS2) select input analog signals to be converted. Bit 3 (PS) selects whether ANI0 to ANI7 (low-order 8 channels) or ANI8 to ANI15 (high-order 8 channels) are used as analog input pins. The low-order 8 channels and high-order 8 channels have identical functions.

Bit 5 (AM0) and bit 6 (AM1) control the operation mode of A/D conversion. If the AM0 and AM1 bits are cleared (0), all conversion operations under execution are stopped. At this time, ADCRn (n = 0 to 7) is not updated, nor is the INTAD interrupt request generated. Moreover, power supply to the voltage comparator is stopped to reduce the current consumption of the A/D converter.

Bit 7 (TRG) enables external synchronization of the A/D conversion operation. If the TRG bit is set (1) when the AM0 or AM1 bits are set, the conversion operation is initialized each time the valid edge is input to the INTP4 pin as an external trigger. If the TRG bit is cleared (0), the conversion operation is performed regardless of the INTP4 pin input.

If data is written to ADM during conversion, the conversion operation is initialized and started from the beginning again.

When $\overline{\text{RESET}}$ is input, the value of ADM is reset to 00H.

Caution When the STOP mode or IDLE mode is used, the consumption current should be reduced by clearing (0) the AM0 bit and AM1 bit before entering the STOP or IDLE mode. If the AM0 bit or AM1 bit remains set (1), the conversion operation will be stopped by entering the STOP or IDLE mode, but the power supply to the voltage comparator will not be stopped, and therefore the A/D converter consumption current will not be reduced.

Figure 11-3. Format of A/D Converter Mode Register (ADM)

Address: 0FF6EH On reset: 00H R/W

	7	6	5	4	③	2	1	0
ADM	TRG	AM1	AM0	FR	PS	ANIS2	ANIS1	ANIS0

TRG	Controls External Trigger
0	Disables external trigger
1	Enables external trigger

AM1	AM0	Specifies A/D Conversion Operation Mode	
0	0	Stops conversion	
0	1	Scan mode	
1	0	Select mode	1-buffer mode
1	1		4-buffer mode

FR	Selects Conversion Time
0	208 clocks (f_{CLK} > 12.5 MHz)
1	169 clocks (f_{CLK} ≤ 12.5 MHz)

PS	Selects Analog Input Pin
0	ANI0 to ANI7 (port 7)
1	ANI8 to ANI15 (port 8)

ANIS2	ANIS1	ANIS0	Selects Analog Input	
			In select mode	In scan mode
0	0	0	ANI0/ANI8	ANI0/ANI8
0	0	1	ANI1/ANI9	ANI0/ANI8, ANI1/ANI9
0	1	0	ANI2/ANI10	ANI0/ANI8- ANI2/ANI10
0	1	1	ANI3/ANI11	ANI0/ANI8- ANI3/ANI11
1	0	0	ANI4/ANI12	ANI0/ANI8- ANI4/ANI12
1	0	1	ANI5/ANI13	ANI0/ANI8- ANI5/ANI13
1	1	0	ANI6/ANI14	ANI0/ANI8- ANI6/ANI14
1	1	1	ANI7/ANI15	ANI0/ANI8- ANI7/ANI15

Remark f_{CLK}: internal system clock

Table 11-1. Conversion Time Set by FR Bit

Internal system clock: f_{CLK} (MHz)	16	14	12.5	10
FR bit	0	0	1	1
Conversion time (μs)	13	14.9	13.5	16.9

Caution Once the A/D converter starts operating, conversion operations are performed repeatedly until the AM0 bit and AM1 bit of the A/D converter mode register (ADM) is cleared (0). Therefore, a superfluous interrupt may be generated if ADM setting is performed after interrupt-related registers, etc., when A/D converter mode conversion, etc., is performed. The result of this superfluous interrupt is that the conversion result storage address appears to have been shifted when the scan mode is used. Also, when the select mode is used, the first conversion result appears to have been an abnormal value, such as the conversion result for the other channel. It is therefore recommended that A/D converter mode conversion be carried out using the following procedure.

<1> Write to the ADM
<2> Interrupt request flag (ADIF) clearance (0)
<3> Interrupt mask flag setting

Operations <1> to <3> should not be divided by an interrupt or macro service.

Alternatively, the following procedure is recommended.

<1> Stop the A/D conversion operation by clearing (0) the AM0 bit and AM1 bit of the ADM.
<2> Interrupt request flag (ADIF) clearance (0).
<3> Interrupt mask flag setting
<4> Write to the ADM

11.3 A/D Conversion Result Registers (ADCR0 to ADCR7)

The μPD784054 has eight 10-bit A/D conversion result registers (ADCR0 to ADCR7) that store the results of A/D conversion.

Each ADCRn (n = 0 to 7) can be only read by using a 16-bit manipulation instruction or an 8-bit manipulation instruction.

The conversion result can be read from ADCRn in the following two ways:

(1) Word access (by execution of 16-bit manipulation instruction)

Of the word data read, the low-order 10 bits are valid.

The high-order 6 bits are always “0” when read.

Figure 11-4 illustrates word access to ADCRn.

Figure 11-4. Word Access to A/D Conversion Result Register

Symbol	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	R/W
ADCRn	0	0	0	0	0	0	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0	R

(n = 0-7)

Symbol	Address	On reset
ADCR0	0FF70H	Undefined
ADCR1	0FF72H	
ADCR2	0FF74H	
ADCR3	0FF76H	
ADCR4	0FF78H	
ADCR5	0FF7AH	
ADCR6	0FF7CH	
ADCR7	0FF7EH	

Remark AD0-AD9: A/D conversion result

(2) Byte access (by execution of 8-bit manipulation instruction)

Of the 10-bit data of the A/D conversion result, the high-order 8 bits are read.

Figure 11-5 illustrates byte access to ADCRn

Figure 11-5. Byte Access to A/D Conversion Result Register

Symbol	7	6	5	4	3	2	1	0	R/W
ADCRnH	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2	R

(n = 0-7)

Symbol	Address	On reset
ADCR0H	0FF71H	Undefined
ADCR1H	0FF73H	
ADCR2H	0FF75H	
ADCR3H	0FF77H	
ADCR4H	0FF79H	
ADCR5H	0FF7BH	
ADCR6H	0FF7DH	
ADCR7H	0FF7FH	

Remark AD2-AD9: A/D conversion result (high-order 8 bits of 10 bits)

11.4 Operation

11.4.1 Basic A/D converter operation

(1) A/D Conversion Operation procedure

A/D conversion is performed by means of the following procedure:

(a) Analog pin selection and operating mode specification are set with the A/D converter mode register (ADM), and the A/D conversion is started.

(b) When conversion starts, the MSB (bit 9) of the successive approximation register (SAR) is set (1) automatically.

(c) When bit 9 of the SAR is set (1), the tap selector sets the series resistor string voltage tap to

$$\frac{1023}{2048}\ AV_{REF} \ (\fallingdotseq 1/2\ AV_{REF}).$$

(d) The voltage difference between the series resistor string voltage tap and the analog input is determined by the voltage comparator. If the analog input is greater than (1/2) AV_{REF}, the MSB of the SAR remains set (1), and if it is less than (1/2) AV_{REF}, the MSB is cleared (0).

(e) Next, bit 8 of the SAR is set (1) automatically, and the next comparison is performed. Here, the series resistor string voltage tap is selected according to the value of bit 9 for which the result has already been set, as shown below.

- Bit 9 = 1 ..... $\frac{1535}{2048}\ AV_{REF} \fallingdotseq \frac{3}{4}\ AV_{REF}$
- Bit 9 = 0 ..... $\frac{511}{2048}\ AV_{REF} \fallingdotseq \frac{1}{4}\ AV_{REF}$

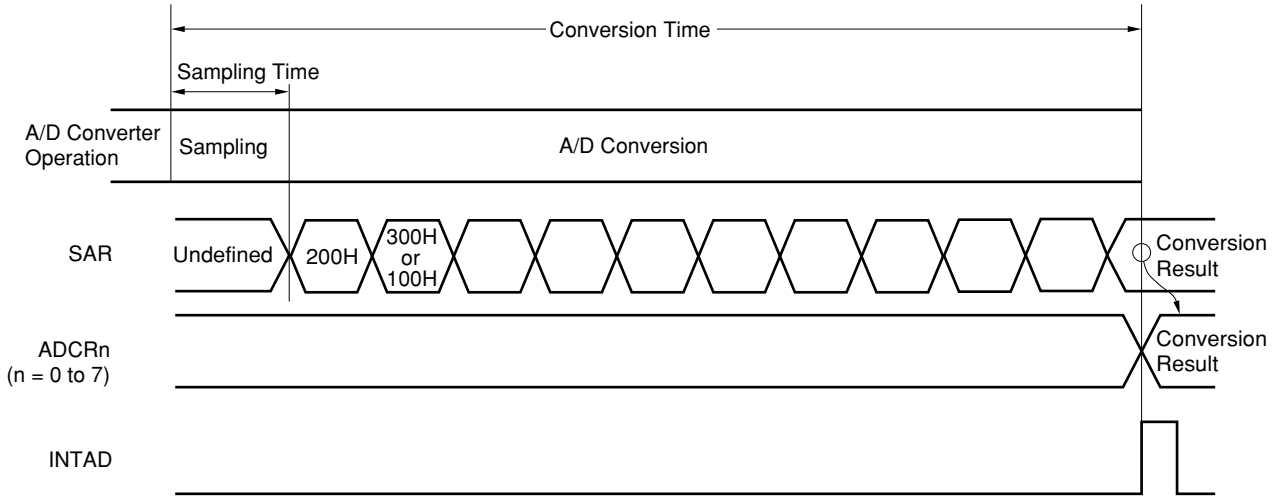
This voltage tap is compared with the analog input voltage, and bit 8 of the SAR is manipulated as follows according to the result:

- Analog input voltage ≥ voltage tap: Bit 8 = 1
- Analog input voltage < voltage tap: Bit 8 = 0

(f) The same kind of comparison is continued up to the LSB (bit 0) of the SAR (binary search method).

(g) When comparison of the 10 bits is completed, a valid digital result is left in the SAR, and that value is transferred to the A/D conversion result register (ADCR0 to ADCR7) and latched.
An A/D conversion operation end interrupt request (INTAD) can be generated at the same time.

Figure 11-6. Basic Operation of A/D Converter



A/D conversion operations are performed successively until the AM0 bit and AM1 bit is cleared (0) by software. If a write operation is performed on the ADM during an A/D conversion operation, the conversion operation is initialized, and if the AM0 bit and AM1 bit is set (1), conversion will be started from the beginning.

The contents of the ADCR n (n = 0 to 7) are undefined after $\overline{\text{RESET}}$ input.

(2) Input voltage and conversion result

The relationship between the analog input voltage input to an analog input pin (ANI0 to ANI15) and the A/D conversion result (value stored in ADCRn) is shown by the following expression:

$$\text{ADCRn} = \text{INT}\left(\frac{V_{IN}}{AV_{REF}} \times 1024 + 0.5\right)$$

or

$$(\text{ADCRn} - 0.5) \times \frac{AV_{REF}}{1024} \leq V_{IN} < (\text{ADCRn} + 0.5) \times \frac{AV_{REF}}{1024}$$

Remark INT( ) : Function that returns the integer part of the value in ( )

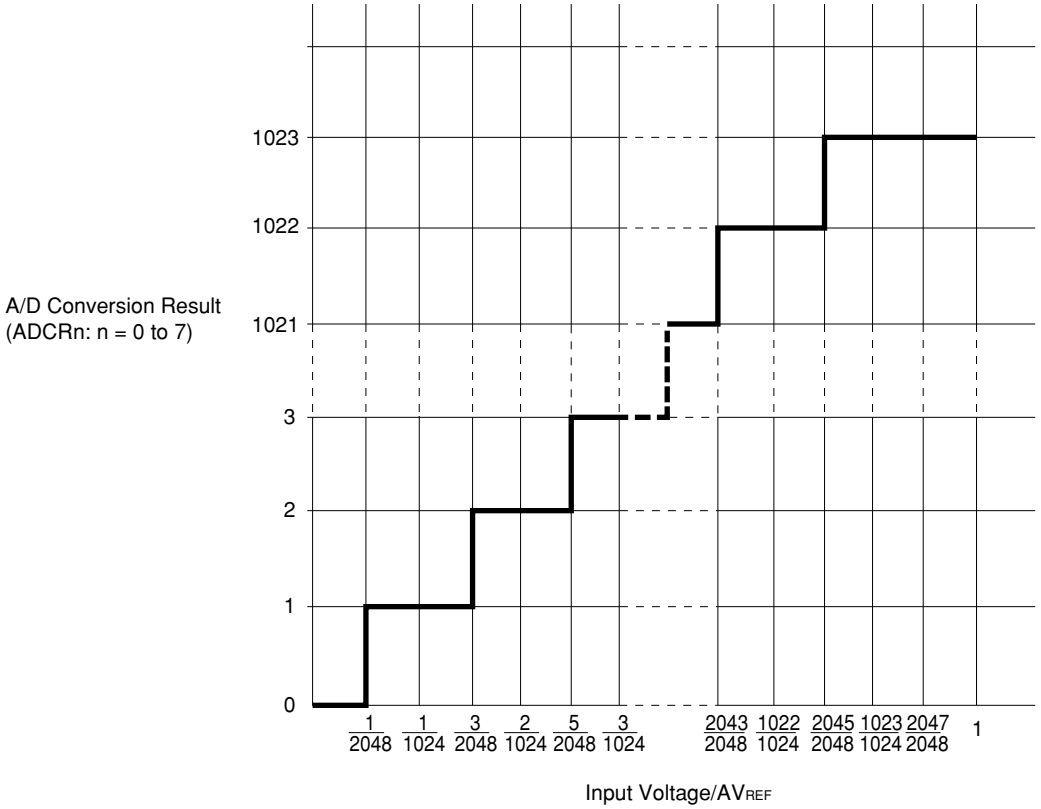
V_{IN} : Analog input voltage

AV_{REF} : AV_{REF} pin voltage

ADCRn : ADCR n (n = 0 to 7) value

Figure 11-7 shows the relationship between the analog input voltage and the A/D conversion result in graphic form.

Figure 11-7. Relationship Between Analog Input Voltage and A/D Conversion Result



(3) A/D conversion time

The A/D conversion time is determined by the system clock frequency (f_{CLK}) and the FR bit of the A/D converter mode register (ADM).

The A/D conversion time includes the entire time required for one A/D conversion operation, and the sampling time is also included in the A/D conversion time.

These values are shown in Table 11-2.

Table 11-2. Time of A/D Conversion

System Clock (f_{CLK}) Range	FR Bit	Conversion Time
f_{CLK} > 12.5 MHz	0	208 clocks
$f_{CLK} \leq$ 12.5 MHz	1	169 clocks

(4) A/D converter operating modes

There are two A/D converter operating modes, scan mode and select mode. These modes are selected according to the setting of bit 5 (AM0) and bit 6 (AM1) of the A/D converter mode register (ADM).

Operation in either mode continues until the ADM is rewritten.

11.4.2 Select mode

In the select mode, one analog input pin is selected by bits 0 to 2 (ANIS0 to ANIS2) of the A/D converter mode select register (ADM), and the specified analog input is converted. The result of the conversion is stored to the A/D conversion result register corresponding to the analog input.

In this mode, the following two modes can be selected depending on how the A/D conversion result is stored.

- 1-buffer mode
- 4-buffer mode

(1) 1-buffer mode

One analog input is converted once, and the result is stored to one A/D conversion result register. Therefore, the analog input and A/D conversion result register correspond on a one-to-one basis (refer to **Table 11-3**).

Each time the conversion has been completed once, an A/D conversion end interrupt request (INTAD) occurs.

Table 11-3. Correspondence between Analog Input and A/D Conversion Result Register (select mode: 1-buffer mode)

Analog Input	A/D Conversion Result Register
ANI0/ANI8	ACDR0
ANI1/ANI9	ADCR1
ANI2/ANI10	ACDR2
ANI3/ANI11	ADCR3
ANI4/ANI12	ACDR4
ANI5/ANI13	ADCR5
ANI6/ANI14	ACDR6
ANI7/ANI15	ADCR7

Figure 11-8. Operating Timing in Select Mode (1-buffer mode) (1/2)

(a) TRG bit ← 0

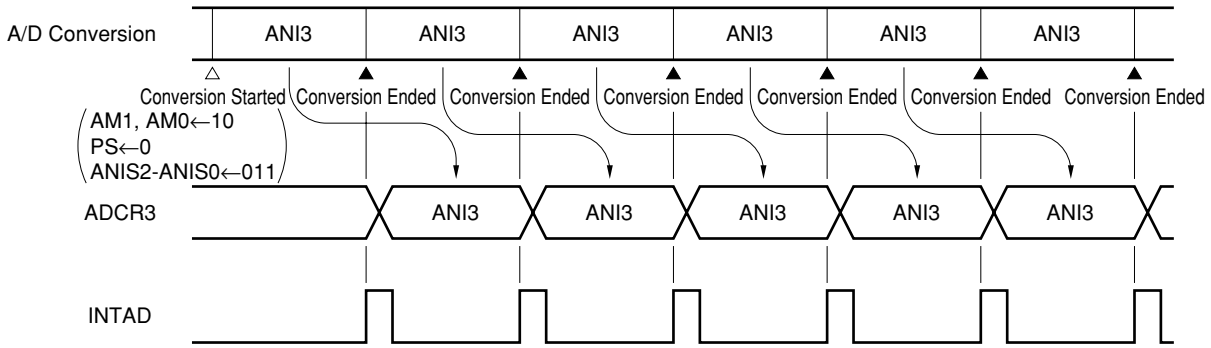
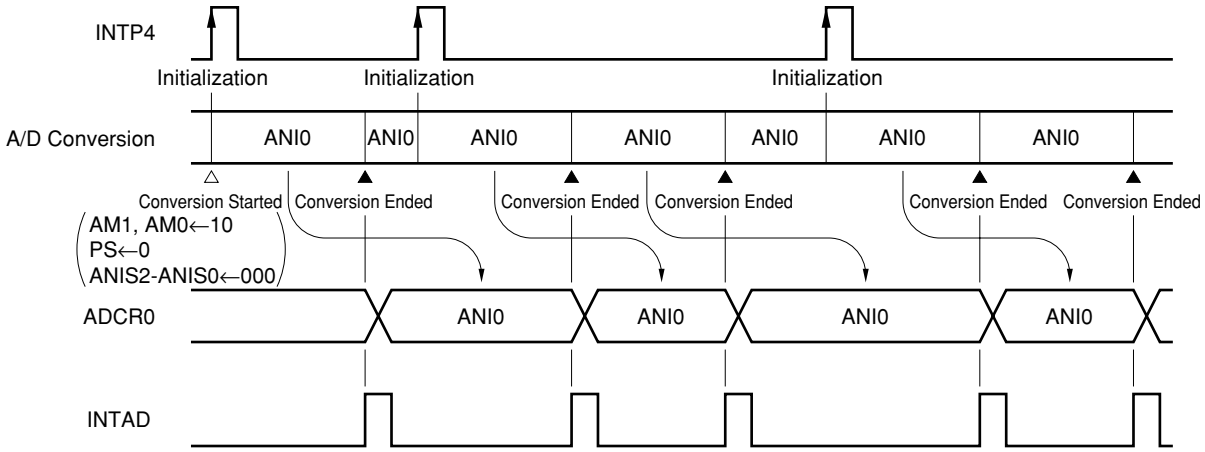


Figure 11-8. Operation Timing in Select Mode (1-buffer mode) (2/2)

(b) TRG bit ← 1



(2) 4-buffer mode

One analog input is converted four times, and the result is stored to four A/D conversion result registers. When one of the analog inputs of ANI0 to ANI3 (ANI8 to ANI11) is selected, the conversion result is stored to A/D conversion result registers ADCR0 to ADCR3. If one of the analog inputs of ANI4 to ANI7 (ANI12 to ANI15) is selected, the conversion result is stored to the A/D conversion result register ADCR4 to ADCR7 (refer to **Table 11-4**).

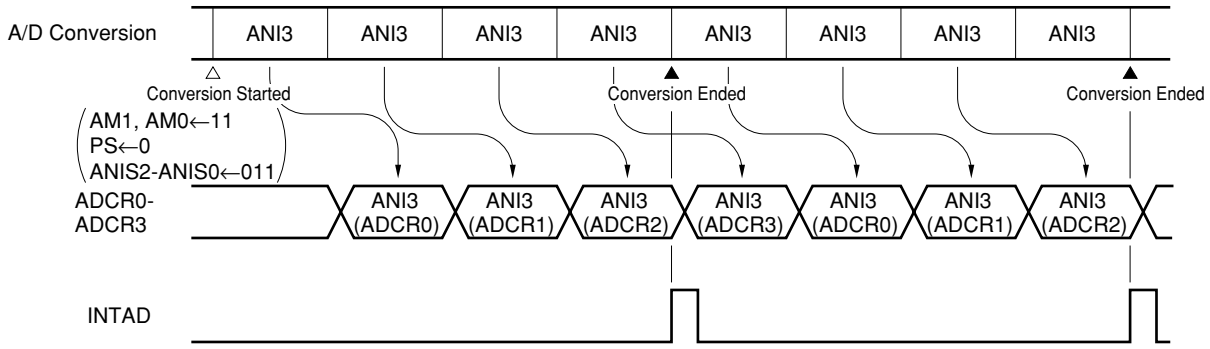
Each time A/D conversion has been completed four times, A/D conversion end interrupt request (INTAD) is generated.

Table 11-4. Correspondence between Analog Input and A/D Conversion Result Register (select mode: 4-buffer mode)

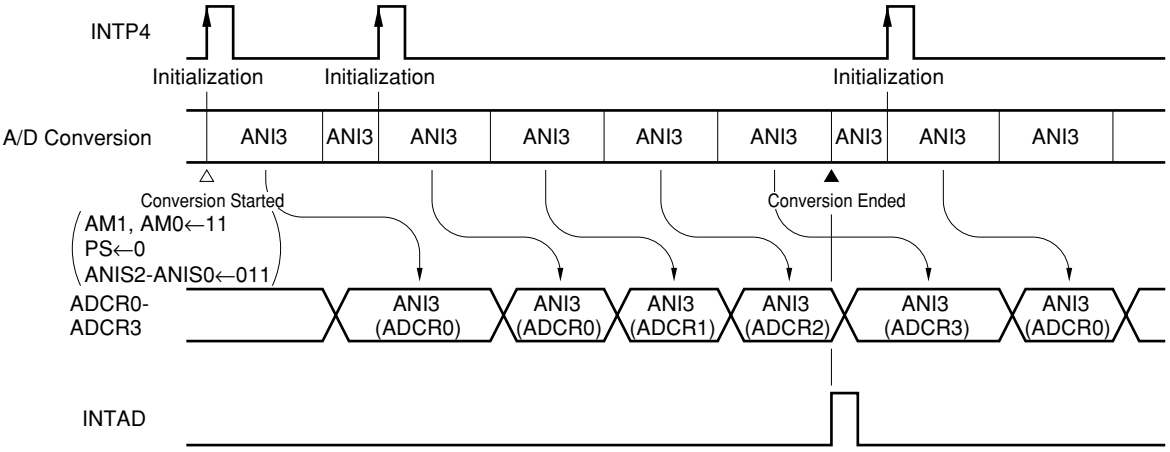
Analog Input	A/D Conversion Result Register
ANI0/ANI8	ADCR0-ADCR3
ANI1/ANI9	
ANI2/ANI10	
ANI3/ANI11	
ANI4/ANI12	ADCR4-ADCR7
ANI5/ANI13	
ANI6/ANI14	
ANI7/ANI15	

Figure 11-9. Operation Timing in Select Mode (4-buffer mode)

(a) TRG bit ← 0



(b) TRG bit ← 1



11.4.3 Scan mode

In this mode, analog input pins specified by the ANIS0 to ANIS2 bits of the A/D converter mode register (ADM) are sequentially selected, starting from ANI0 pin, and A/D conversion is executed. The result of the conversion is stored to the A/D conversion result register that corresponds to an analog input on a one-to-one basis (refer to **Table 11-5**).

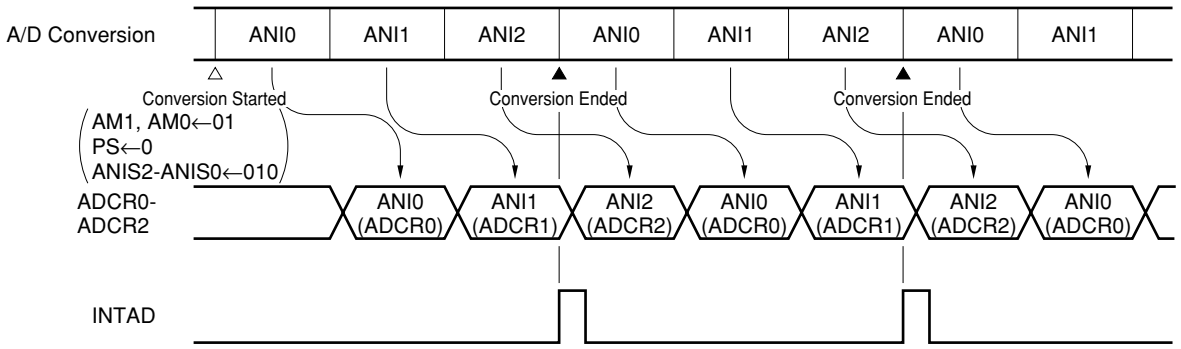
When all the analog inputs have been converted, the A/D conversion end interrupt request (INTAD) is generated.

Table 11-5. Correspondence between Analog Input and A/D Conversion Result Register (scan mode)

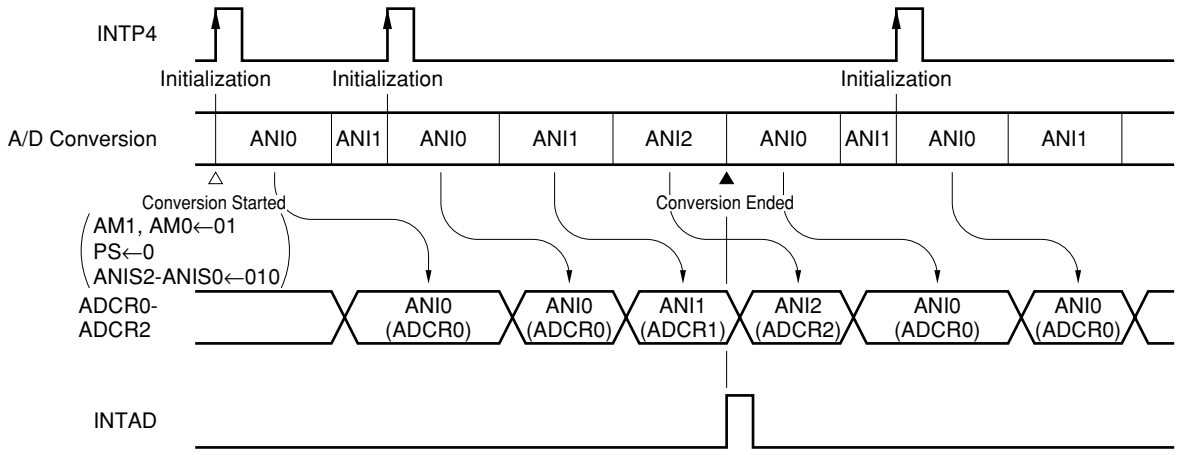
Analog Input	A/D Conversion Result Register
ANI0/ANI8	ADCR0
ANI1/ANI9	ADCR1
ANI2/ANI10	ADCR2
ANI3/ANI11	ADCR3
ANI4/ANI12	ADCR4
ANI5/ANI13	ADCR5
ANI6/ANI14	ADCR6
ANI7/ANI15	ADCR7

Figure 11-10. Operation Timing in Scan Mode

(a) TRG bit ← 0



(b) TRG bit ← 1



11.4.4 A/D conversion operation start by software

An A/D conversion operation start by software is performed by writing a value to the A/D converter mode register (ADM) that sets the TRG bit of the ADM register to 0 and the AM0 bit or AM1 bit to 1.

If a value is written to the ADM during an A/D conversion operation (AM0 bit or AM1 bit = 1) such that the TRG bit is set to 0 and the AM0 bit or AM1 bit to 1 again, the A/D conversion operation being performed at that time is suspended, and A/D conversion is started immediately in accordance with the written value.

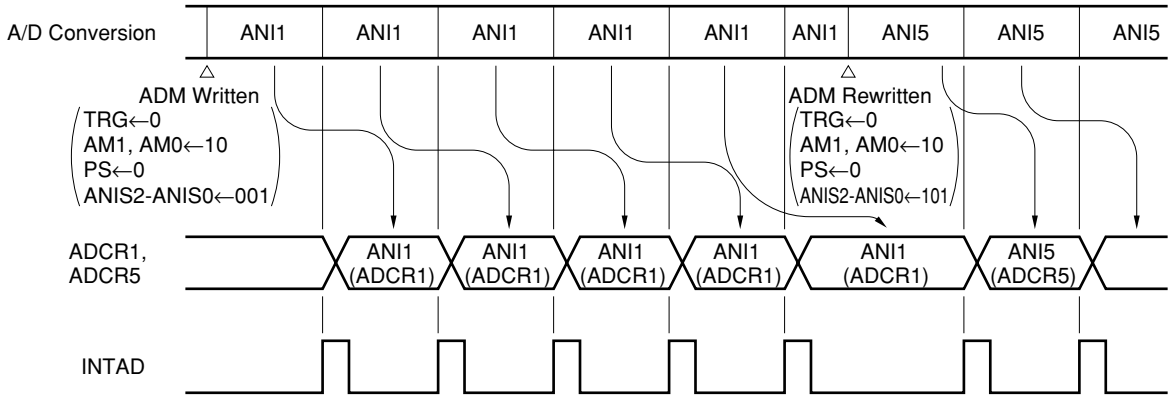
Once A/D conversion operation is started, as soon as one A/D conversion operation ends the next A/D conversion operation is started in accordance with the operating mode set by the ADM, and conversion operations continue repeatedly until an instruction that writes to the ADM is executed.

When A/D conversion operation is started by software (TRG bit = 0), INTP4 pin (P25 pin) input does not affect the A/D conversion operation.

(1) A/D conversion in select mode (1-buffer mode)

A/D conversion of the analog input set by the A/D converter mode register (ADM) is started. When conversion has been completed, the same analog input is converted again. Each time A/D conversion has been completed, the A/D conversion end interrupt request (INTAD) is generated.

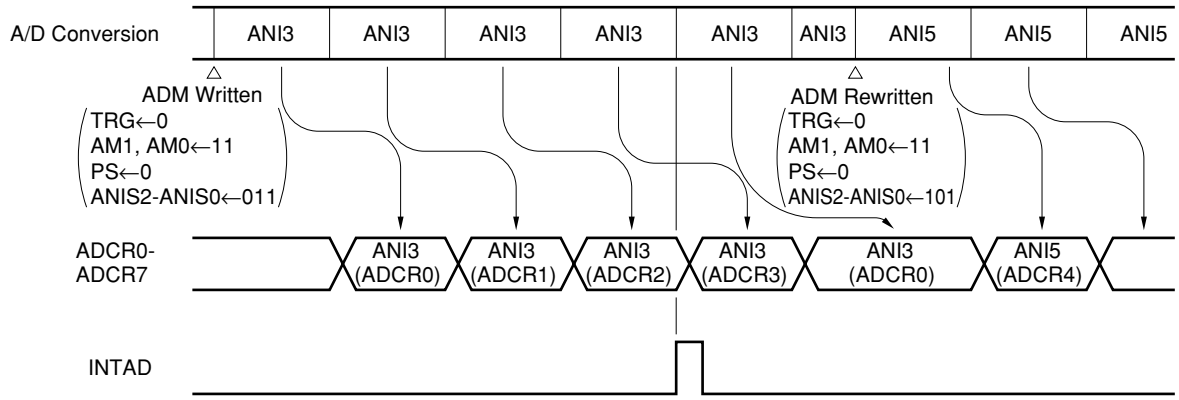
Figure 11-11. A/D Conversion in Select Mode (1-buffer mode) Started by Software



(2) A/D conversion in select mode (4- buffer mode)

The analog input set by the A/D converter mode register (ADM) is converted. One analog input is converted four times. When A/D conversion has been executed four times, the same analog input is converted four times again. Each time conversion has been executed four times, the A/D conversion end interrupt request (INTAD) is generated.

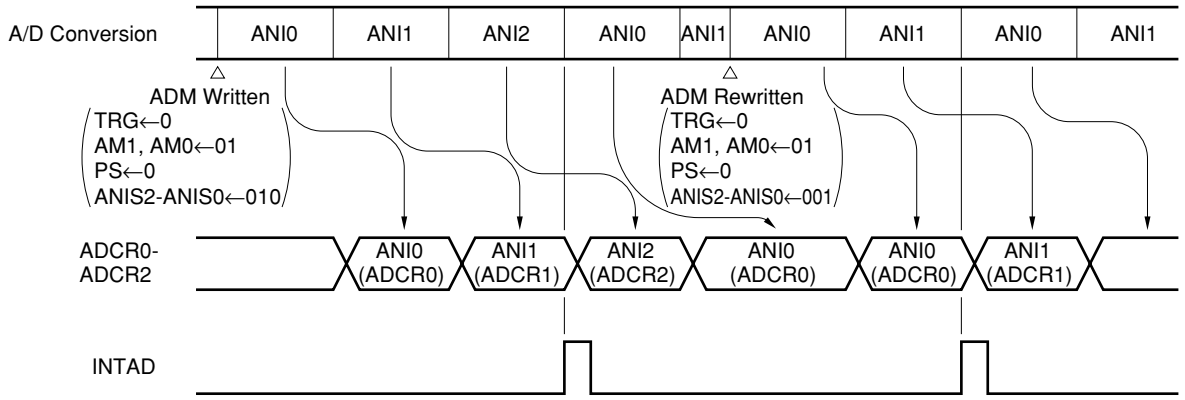
Figure 11-12. A/D Conversion in Select Mode (4-buffer mode) Started by Software



(3) A/D conversion in scan mode

When conversion is started, analog inputs selected by the ANIS0 to ANIS2 bits of the A/D converter mode register (ADM) are sequentially converted, starting from the ANI0 pin. When conversion of all the selected analog inputs has been completed, the same operation (conversion from the ANI0 pin to the specified analog input pin) is repeated again. When a series of A/D conversion from the ANI0 pin to the specified analog input has been completed, the A/D conversion end interrupt request (INTAD) is generated.

Figure 11-13. A/D Conversion in Scan Mode Started by Software



11.4.5 A/D conversion operation start by hardware

An A/D conversion operation start by hardware is made possible by setting both the TRG bit and the AM0 bit or AM1 bit of the A/D converter mode register (ADM) to 1. When the TRG bit and the AM0 bit or AM1 bit of the ADM are both set to 1, external signals are placed in the standby state, and an A/D conversion operation is started when a valid edge is input to the INTP4 pin (P25 pin).

If another valid edge is input to the INTP4 pin after the A/D conversion operation has been started by a valid edge input to the INTP4 pin, the A/D conversion operation being performed at that time is suspended, and A/D conversion is performed from the beginning in accordance with the contents set in the ADM.

If a value is written to the ADM during an A/D conversion operation (AM0 bit or AM1 = 1) such that the TRG bit and AM0 bit or AM1 bit are both set to 1 again, the A/D conversion operation being performed at that time is suspended (the standby state is also suspended), and a state is entered in which the A/D converter waits for input of a valid edge to the INTP4 pin in the A/D conversion operation mode in accordance with the written value, and a conversion operation is started when a valid edge is input.

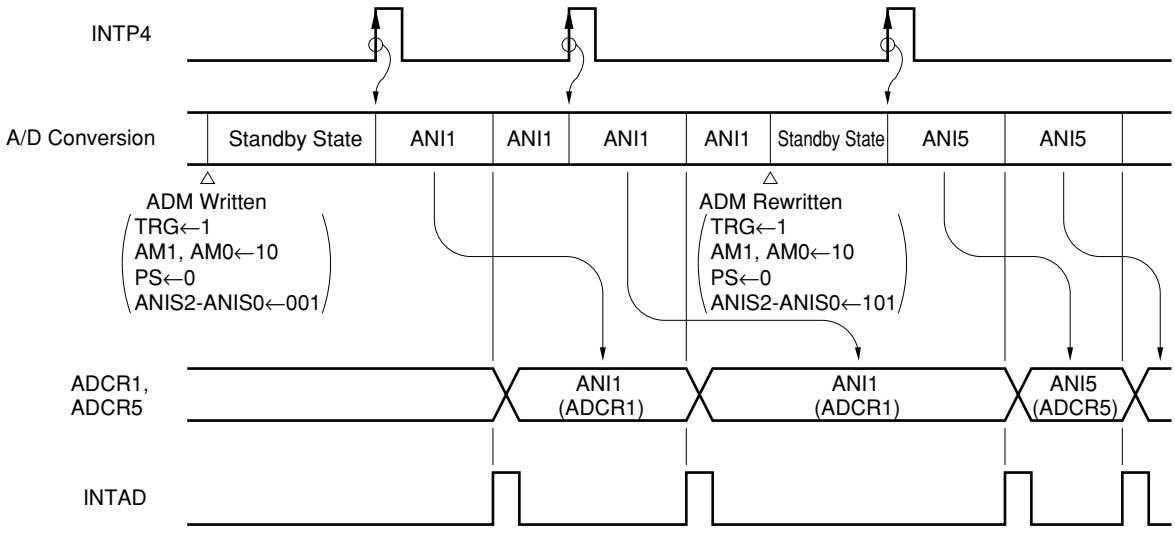
Use of this function allows A/D conversion operations to be synchronized with external signals. Once A/D conversion operation is started, as soon as one A/D conversion operation ends the next A/D conversion operation is started in accordance with the operating mode set by the ADM (the A/D converter does not wait for INTP4 pin input), and conversion operations continue repeatedly until an instruction that writes to the ADM is executed, or a valid edge is input to the INTP4 pin.

(1) A/D conversion in select mode (1-buffer mode)

A/D conversion of the analog input set by the A/D converter mode register (ADM) is started. When conversion has been completed, the same analog input is converted again. Each time A/D conversion has been completed, the A/D conversion end interrupt request (INTAD) is generated.

If the valid edge is input to the INTP4 pin during A/D conversion, the A/D conversion under execution is stopped once, and then conversion is newly started.

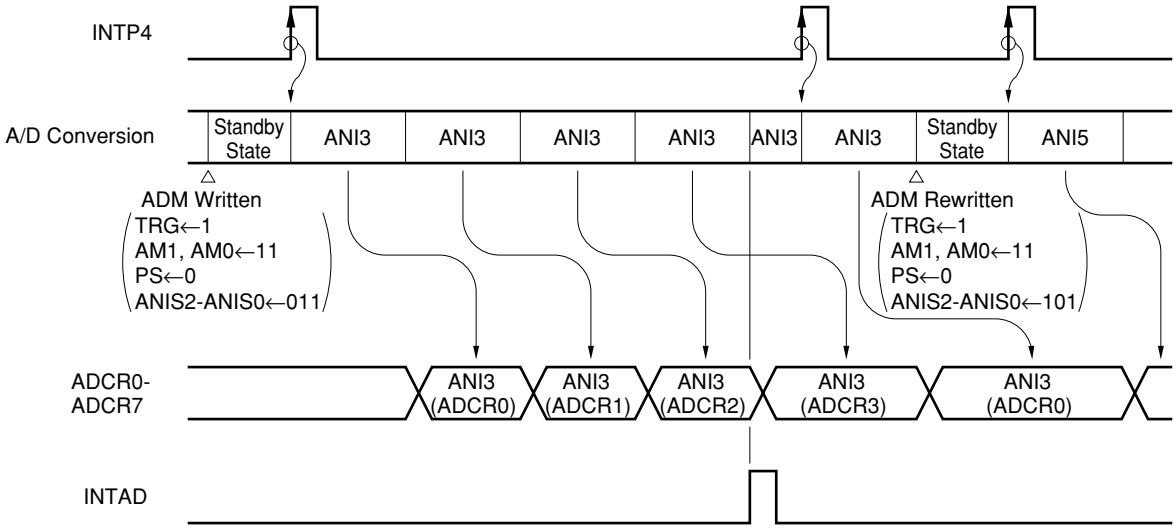
Figure 11-14. A/D Conversion in Select Mode (1-buffer mode) Started by Hardware



(2) A/D conversion in select mode (4-buffer mode)

The analog input set by the A/D converter mode register (ADM) is converted. One analog input is converted four times. When A/D conversion has been executed four times, the same analog input is converted four times again. Each time conversion has been executed four times, the A/D conversion end interrupt request (INTAD) is generated. If the valid edge is input to the INTP4 pin during A/D conversion, the A/D conversion under execution is stopped once, and then conversion is newly started.

Figure 11-15. A/D Conversion in Select Mode (4-buffer mode) Started by Hardware

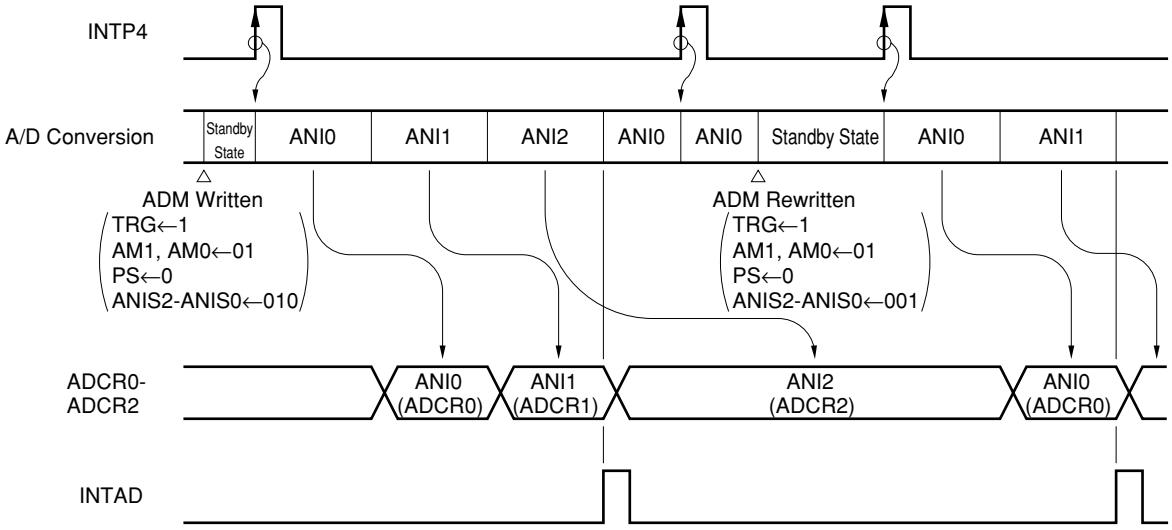


(3) A/D conversion in scan mode

When conversion is started, analog inputs selected by the ANIS0 to ANIS2 bits of the A/D converter mode register are sequentially converted, starting from the ANI0 pin. When conversion of all the selected analog inputs has been completed, the same operation (conversion from the ANI0 pin to the specified analog input pin) is repeated again. When a series of A/D conversion from the ANI0 pin to the specified analog input has been completed, the A/D conversion end interrupt request (INTAD) is generated.

If the valid edge is input to the INTP4 pin during A/D conversion, the A/D conversion under execution is stopped once, and then conversion is newly started.

Figure 11-16. A/D Conversion in Scan Mode Started by Hardware



11.5 External Circuit of A/D Converter

The A/D converter is provided with a sample & hold circuit to stabilize its conversion operation. This sample & hold circuit outputs sampling noise during sampling immediately after an A/D conversion channel has been changed.

To absorb this sampling noise, an external capacitor must be connected. If the impedance of the signal source is high, an error may occur in the conversion result due to the sampling noise. Especially when the scan mode is used, the impedance of the signal source must be kept low because the channel whose signal is to be converted changes one after another.

One way to absorb the sampling noise is to increase the capacitance of the capacitor. However, if the capacitance is increased too much, the sampling noise is accumulated. Therefore, the most effective way is to reduce the resistance component.

11.6 Cautions

(1) Range of voltages applied to analog input pins

The following must be noted concerning A/D converter analog input pins ANI0 to ANI15 (P70 to P77, P80 to P87).

- A voltage outside the range AV_{SS} to AV_{REF} should not be applied to pins subject to A/D conversion during an A/D conversion operation.

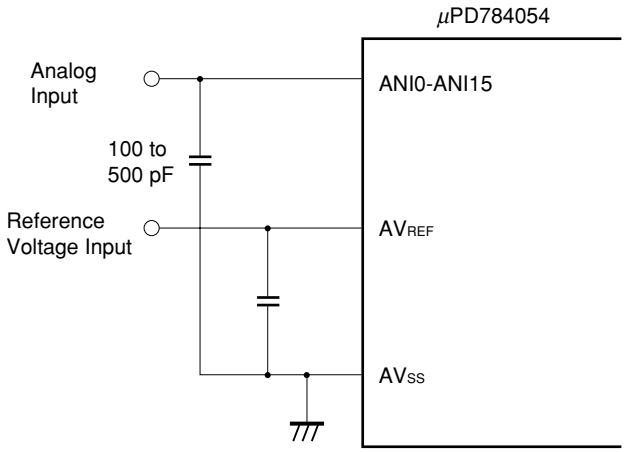
If this restriction is not observed, the μPD784054 may be damaged.

(2) Connecting capacitor to analog input pins

A capacitor should be connected between the analog input pins (ANI0 to ANI15) and AV_{SS} and between the reference voltage input pin (AV_{REF}) and AV_{SS} to prevent misoperation due to noise.

Be sure to connect the capacitor as close to ANI0 through ANI15 and AV_{REF} as possible.

Figure 11-17. Example of Capacitor Connection on A/D Converter Pins



(3) When the STOP mode or IDLE mode is used, the consumption current should be reduced by clearing (0) the AM0 bit and AM1 bit before entering the STOP or IDLE mode. If the AM0 bit and AM1 bit remains set (1), the conversion operation will be stopped by entering the STOP or IDLE mode, but the power supply to the voltage comparator will not be stopped, and therefore the A/D converter consumption current will not be reduced.

(4) Once the A/D converter starts operating, conversion operations are performed repeatedly until the AM0 bit and AM1 bit of the A/D converter mode (ADM) is cleared (0). Therefore, a superfluous interrupt may be generated if ADM setting is performed after interrupt-related registers, etc., are set when A/D converter mode conversion, etc., is performed. The result of this superfluous interrupt is that the conversion result storage address appears to have been shifted when the scan mode is used. Also, when the select mode is used, the first conversion result appears to have been an abnormal value, such as the conversion result for the other channel. It is therefore recommended that A/D converter mode conversion be carried out using the following procedure.

<1> Write to the ADM
<2> Interrupt request flag (ADIF) clearance (0)
<3> Interrupt mask flag setting

Operations <1> to <3> should not be divided by an interrupt or macro service.

Alternatively, the following procedure is recommended.

<1> Stop the A/D conversion operation by clearing (0) the AM0 bit and AM1 bit of the ADM.
<2> Interrupt request flag (ADIF) clearance (0).
<3> Interrupt mask flag setting
<4> Write to the ADM

CHAPTER 12 ASYNCHRONOUS SERIAL INTERFACE/3-WIRE SERIAL I/O

The μPD784054 incorporates two serial interface channels for which asynchronous serial interface (UART) mode or 3-wire serial I/O (IOE) mode can be selected.

The two UART/IOE channels have completely identical functions. In this chapter, therefore, unless stated otherwise, UART/IOE1 will be described as representative of both UART/IOEs. When used as UART2/IOE2, the UART/IOE1 register names, bit names and pin names should be read as their UART2/IOE2 equivalents as shown in Table 12-1.

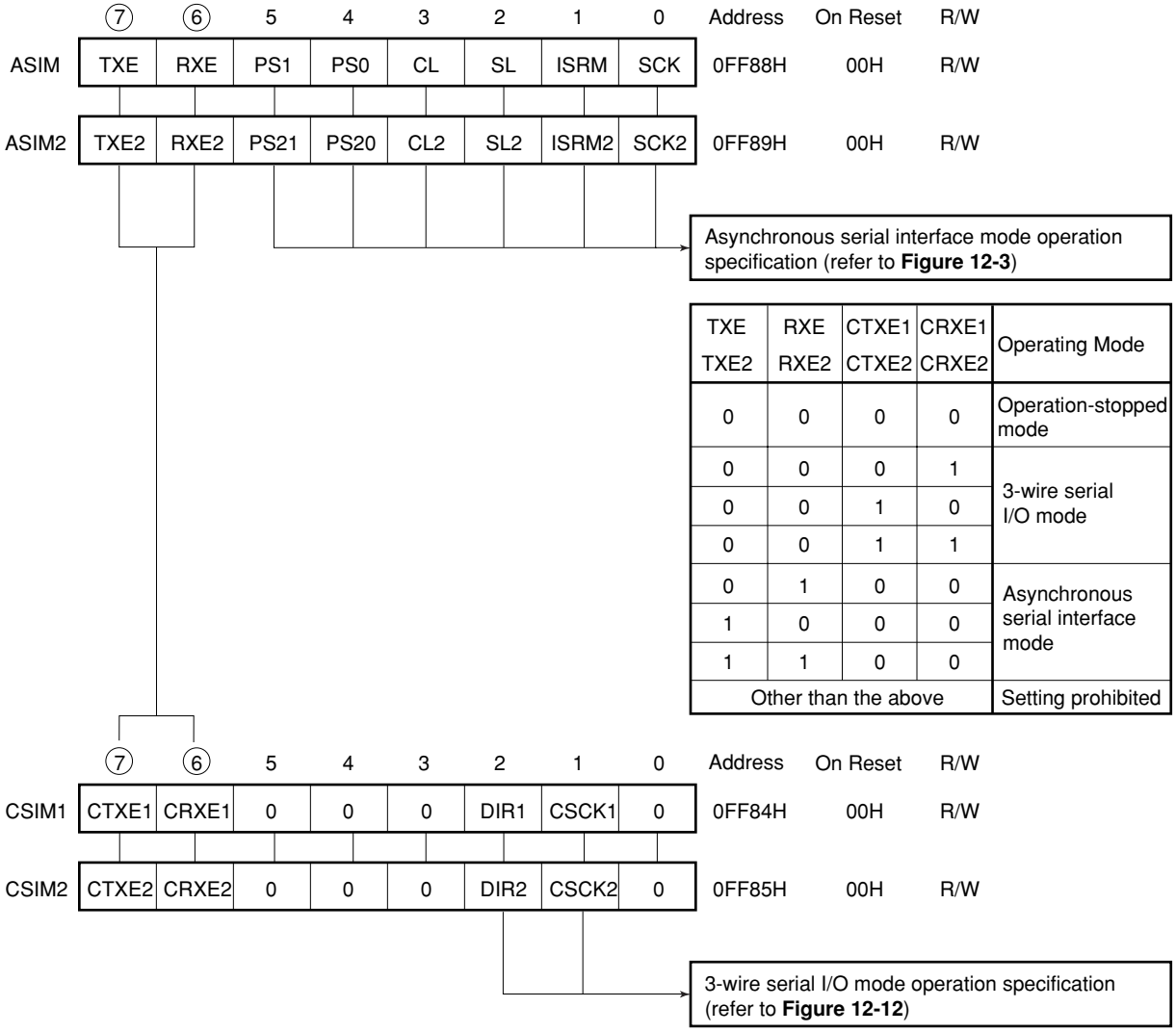
Table 12-1. Differences Between UART/IOE1 and UART2/IOE2 Names

Item	UART/IOE1	UART2/IOE2
Pin names	P32/RxD/SI1, P33/TxD/SO1, P34/ASCK/$\overline{SCK1}$	P35/RxD2/SI2, P36/TxD2/SO2, P37/ASCK2/$\overline{SCK2}$
Asynchronous serial interface mode register	ASIM	ASIM2
Asynchronous serial interface mode register bit names	TXE, RXE, PS1, PS0, CL, SL, ISRM, SCK	TXE2, RXE2, PS21, PS20, CL2, SL2, ISRM2, SCK2
Asynchronous serial interface status register	ASIS	ASIS2
Asynchronous serial interface status register bit names	PE, FE, OVE	PE2, FE2, OVE2
Clocked serial interface mode register	CSIM1	CSIM2
Clocked serial interface mode register bit names	CTXE1, CRXE1, DIR1, CSCK1	CTXE2, CRXE2, DIR2, CSCK2
Baud rate generator control register	BRGC	BRGC2
Baud rate generator control register bit names	TPS0-TPS3, MDL0-MDL3	TPS20-TPS23, MDL20-MDL23
Interrupt request names	INTSR/ITCSI1, INTSER, INTST	INTSR2/INTCSI2, INTSER2, INTST2
Interrupt control registers and bit names used in this chapter	SRIC, CSIIC1, SERIC, STIC, SRIF, CSIIF1, SERIF, STIF	SRIC2, CSIIC2, SERIC2, STIC2, SRIF2, CSIIF2, SERIF2, STIF2

12.1 Switching between Asynchronous Serial Interface Mode and 3-wire Serial I/O Mode

The asynchronous serial interface mode and 3-wire serial I/O mode cannot be used simultaneously. Switching between these modes is performed in accordance with the settings of the asynchronous serial interface mode register (ASIM/ASIM2) and the clocked serial interface mode register (CSIM1/CSIM2) as shown in Figure 12-1.

Figure 12-1. Switching Between Asynchronous Serial Interface Mode and 3-Wire Serial I/O Mode



12.2 Asynchronous Serial Interface Mode

A UART (Universal Asynchronous Receiver Transmitter) mode is incorporated as the asynchronous serial interface. With this method, one byte of data is transmitted following a start bit, and full-duplex operation is possible.

A baud rate generator is incorporated, enabling communication to be performed at any of a wide range of baud rates.

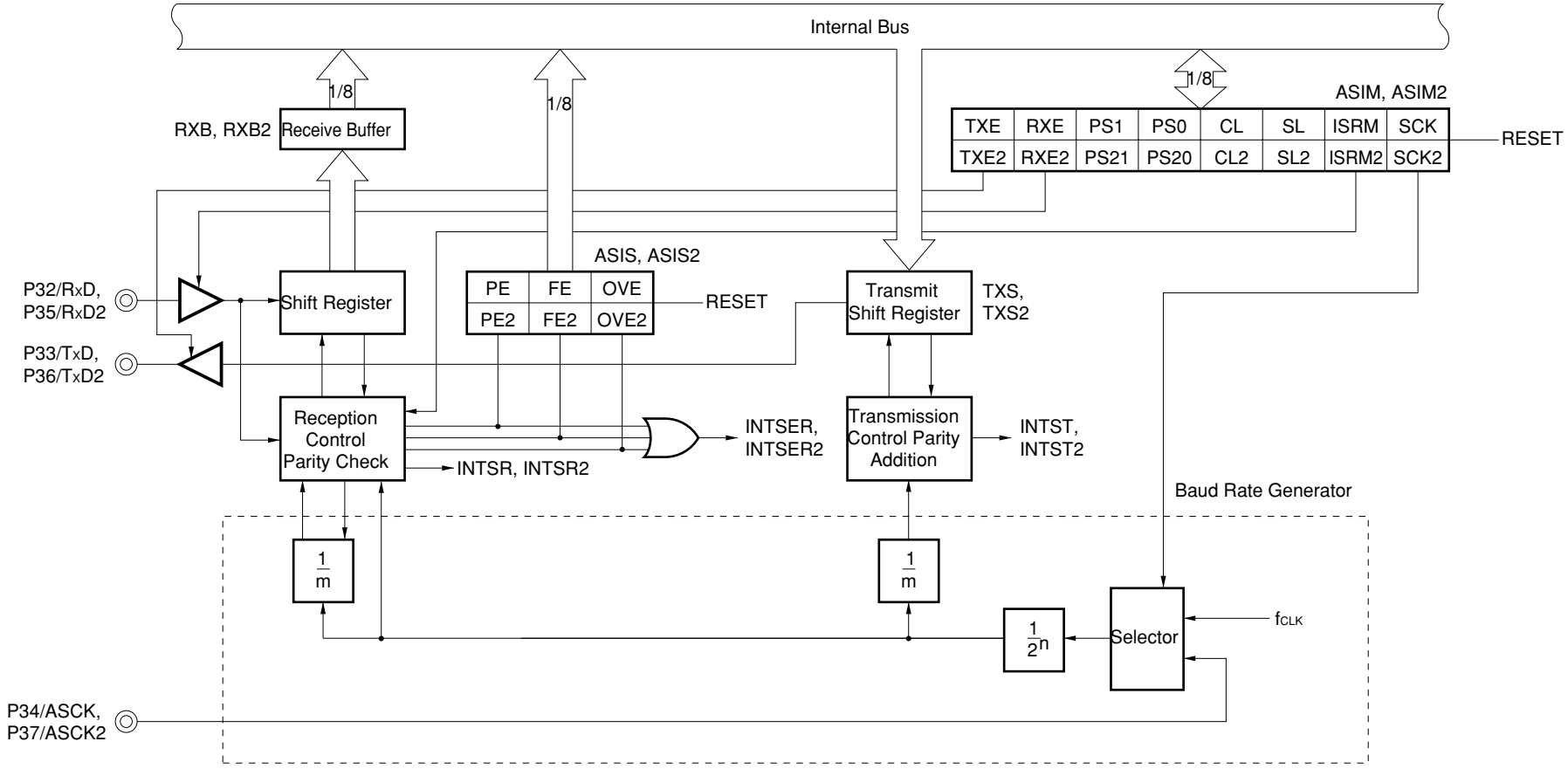
Also, the baud rate can be defined by scaling the clock input to the ASCK pin.

12.2.1 Configuration in asynchronous serial interface mode

The block diagram of the asynchronous serial interface is described in Figure 12-2.

Refer to **12.4 Baud Rate Generator** for details of the baud rate generator.

Figure 12-2. Block Diagram of Asynchronous Serial Interface



Remark m = 16 to 30, n = 0 to 11

(1) Receive buffer (RXB/RXB2)

This is the register that holds the receive data. Each time one byte of data is received, the receive data is transferred from the shift register.

If a 7-bit data length is specified, receive data is transferred to bits 0 to 6 of RXB/RXB2, and the MSB of RXB/RXB2 is always “0”.

RXB/RXB2 can be read only by an 8-bit manipulation instruction. The contents of RXB/RXB2 are undefined after $\overline{\text{RESET}}$ input.

(2) Transmit shift register (TXS/TXS2)

This is the register in which the data to be transmitted is set. Data written to the TXS/TXS2 is transmitted as serial data.

If a 7-bit data length is specified, bits 0 to 6 of the data written in the TXS/TXS2 are treated as transmit data. A transmit operation starts when a write to the TXS/TXS2 is performed. The TXS/TXS2 cannot be written to during a transmit operation.

TXS/TXS2 can be written to only by an 8-bit manipulation instruction. The contents of TXS/TXS2 are undefined after $\overline{\text{RESET}}$ input.

(3) Shift register

This is the shift register that converts the serial data input to the RxD, and RxD2 pin to parallel data. When one byte of data is received, the receive data is transferred to the receive buffer.

The shift register cannot be manipulated directly by the CPU.

(4) Reception control parity check

Receive operations are controlled in accordance with the contents set in the asynchronous serial interface mode register (ASIM/ASIM2). In addition, parity error and other error checks are performed during receive operations, and if an error is detected, a value is set in the asynchronous serial interface status register (ASIS/ASIS2) according to the type of error.

(5) Transmission control parity addition

Transmission operation is controlled by appending a start bit, parity bit, and stop bit to the data written to the transmit shift registers (TXS and TXS2) in accordance with the contents set to the asynchronous serial interface mode registers (ASIM and ASIM2).

(6) Selector

Selects the baud rate clock source.

12.2.2 Asynchronous serial interface control registers

(1) Asynchronous serial interface mode register (ASIM), Asynchronous serial interface mode register 2 (ASIM2)

The ASIM and ASIM2 are 8-bit registers that specify the UART mode operation.

These registers can be read or written to by an 8-bit manipulation instruction or bit manipulation instruction. The format of ASIM and ASIM is shown in Figure 12-3.

These registers are cleared to 00H by $\overline{\text{RESET}}$ input.

Figure 12-3. Formats of Asynchronous Serial Interface Mode Register (ASIM) and Asynchronous Serial Interface Mode Register 2 (ASIM2)

Address: 0FF88H, 0FF89H On reset: 00H R/W

	⑦	⑥	5	4	3	2	1	0
ASIM	TXE	RXE	PS1	PS0	CL	SL	ISRM	SCK
ASIM2	TXE2	RXE2	PS21	PS20	CL2	SL2	ISRM2	SCK2

TXE TXE2	RXE RXE2	Transmission/Reception
0	0	Disables transmission/reception, or sets 3-wire serial I/O mode
0	1	Enables reception
1	0	Enables transmission
1	1	Enables transmission/reception

PS1 PS21	PS0 PS20	Specifies Parity Bit
0	0	No parity
0	1	Transmission: 0 parity appended Reception: Parity error does not occur
1	0	Odd parity
1	1	Even parity

CL CL2	Specifies Character Length of Data
0	7 bits
1	8 bits

SL SL2	Specifies Stop Bit Length (transmission only)
0	1 bit
1	2 bits

ISRM ISRM2	Enables or Disables Occurrence of Reception End Interrupt in Case of Reception Error[Note]
0	Enables
1	Disables

SCK SCK2	Specifies Input Clock To Baud Rate Generator
0	External clock input (ASCK, ASCK2)
1	Internal clock (f_{CLK})

Remark f_{CLK}: internal system clock

Note To disable the reception completion interrupt when a reception error occurs, make sure that wait time equivalent to two pulses of the clock that serves as the reference of the baud rate clock elapse after the reception error occurs until the receive buffers (RXB, RXB2) are read. If the wait time is not inserted, the reception completion interrupt occurs even when it is disabled.
The wait time equivalent to two pulses of the clock that serves as the reference of the baud rate clock can be calculated by the following expression:

$$\text{Wait time} = \frac{2^{n+2}}{f_{CLK}}$$

Remark f_{CLK}: Internal system clock frequency
n : Value of baud rate generator control registers (BRGC, BRGC2) to select tap of 12-bit prescaler (n = 0 to 11)

Caution An asynchronous serial interface mode register (ASIM/ASIM2) rewrite should not be performed during a transmit operation. If an ASIM/ASIM2 register rewrite is performed during a transmit operation, subsequent transmit operations may not be possible (normal operation is restored by $\overline{\text{RESET}}$ input). Software can determine whether transmission is in progress by using a transmission completion interrupt (INTST/INTST2) or the interrupt request flag (STIF/STIF2) set by INTST/INTST2.

(2) Asynchronous serial interface status register (ASIS), Asynchronous serial interface status register 2 (ASIS2)

The ASIS and ASIS2 contain flags that indicate the error contents when a receive error occurs. Flags are set (1) when a receive error occurs, and cleared (0) when data is read from the receive buffer (RXB/RXB2). If the next data is received before RXB/RXB2 is read, the overrun error flag (OVE/OVE2) is set (1), and the other error flags are cleared (0) (if there is an error in the next data, the corresponding error flag is set (1)).
These registers can be read only by an 8-bit manipulation instruction or bit manipulation instruction. The format of ASIS and ASIS2 is shown in Figure 12-4.
These registers are cleared to 00H by $\overline{\text{RESET}}$ input.

Figure 12-4. Formats of Asynchronous Serial Interface Status Register (ASIS) and Asynchronous Serial Interface Status Register 2 (ASIS2)

Address : 0FF8AH, 0FF8BH On reset : 00H R

	7	6	5	4	3	②	①	⓪
ASIS	0	0	0	0	0	PE	FE	OVE
ASIS2	0	0	0	0	0	PE2	FE2	OVE2

PE PE2	Parity Error Flag
0	Parity error does not occur
1	Parity error occurs

FE FE2	Framing Error Flag
0	Framing error does not occur
1	Framing error occurs

OVE OVE2	Overrun Error Flag
0	Reception overrun error does not occur
1	Reception overrun error occurs

Cautions 1. The receive buffer (RXB/RXB2) must be read even if there is a receive error. If RXB/RXB2 is not read, an overrun error will occur when the next data is received, and the receive error state will continue indefinitely.

2. To disable the reception completion interrupt when a reception error occurs, make sure that wait time equivalent to two pulses of the clock that serves as the reference of the baud rate clock elapse after the reception error occurs until the receive buffers (RXB, RXB2) are read. If the wait time is not inserted, the reception completion interrupt occurs even when it is disabled.
The wait time equivalent to two pulses of the clock that serves as the reference of the baud rate clock can be calculated by the following expression:

$$\text{Wait time} = \frac{2^{n+2}}{f_{CLK}}$$

Remark f_{CLK} : Internal system clock frequency
n : Value of baud rate generator control registers (BRGC, BRGC2) to select tap of 12-bit prescaler (n = 0 to 11)

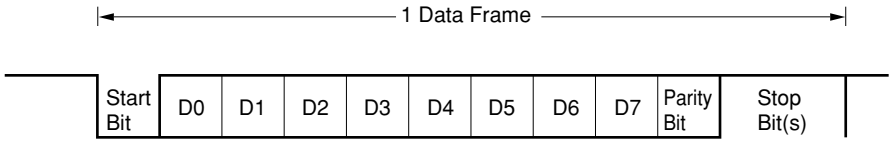
12.2.3 Data format

Serial data transmission/reception is performed in full-duplex asynchronous mode.

The transmit/receive data format is shown in Figure 12-5. One data frame is made up of a start bit, character bits, parity bit, and stop bit(s).

Character bit length specification, parity selection and stop bit length specification for one data frame are performed by means of the asynchronous serial interface mode register (ASIM).

Figure 12-5. Data Format of Asynchronous Serial Interface Transmit/Receive



- Start bit ..... 1 bit
- Character bits ..... 7 bits/8 bits
- Parity bit ..... Even parity/odd parity/0 parity/no parity
- Stop bit(s) ..... 1 bit/2 bits

The serial transfer rate is selected in accordance with the asynchronous serial interface mode register and baud rate generator settings. If a serial data receive error occurs, the nature of the receive error can be determined by reading the asynchronous serial interface status register (ASIS) status.

12.2.4 Parity types and operations

The parity bit is used to detect a bit error in the communication data. Normally, the same kind of parity bit is used on the transmission side and the reception side. With even parity and odd parity, 1 bit (odd number) errors can be detected. With 0 parity and no parity, errors cannot be detected.

- Even parity
  If the number of bits with a value of “1” in the transmit data is odd, the parity bit is set to “1”, and if the number of “1” bits is even, the parity bit is set to “0”. Control is thus performed to make the number of “1” bits in the transmit data plus the parity bit an even number. In reception, the number of “1” bits in the receive data plus the parity bit is counted, and if this number is odd, a parity error is generated.

- Odd parity
  Conversely to the case of even parity, control is performed to make the number of “1“ bits in the transmit data plus the parity bit an odd number.
  In reception, a parity error is generated if the number of “1” bits in the receive data plus the parity bit is even.

- 0 parity
  In transmission, the parity bit is set to “0” irrespective of the receive data.
  In reception, parity bit detection is not performed. Therefore, no parity error is generated irrespective of whether the parity bit is “0” or “1”.

- No parity
  In transmission, a parity bit is not added.
  In reception, reception is performed on the assumption that there is no parity bit. Since there is no parity bit, no parity error is generated.

12.2.5 Transmission

The μPD784054's asynchronous serial interface is set to the transmission enabled state when the TXE bit of the asynchronous serial interface mode register (ASIM) is set (1). A transmit operation is started by writing transmit data to the transmit shift register (TXS) when transmission is enabled. The start bit, parity bit and stop bit(s) are added automatically.

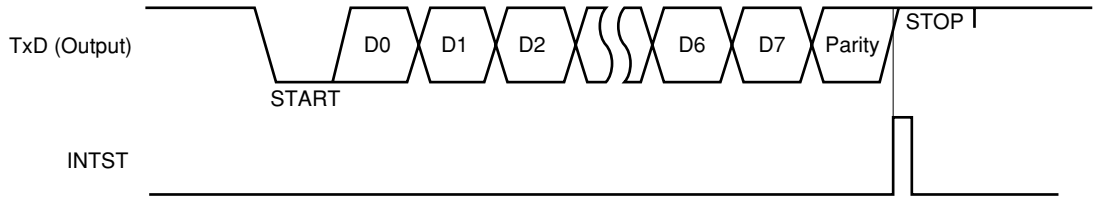
When a transmit operation is started, the data in the TXS is shifted out, and a transmission completion interrupt (INTST) is generated when the TXS is empty.

If no more data is written to the TXS, the transmit operation is discontinued.

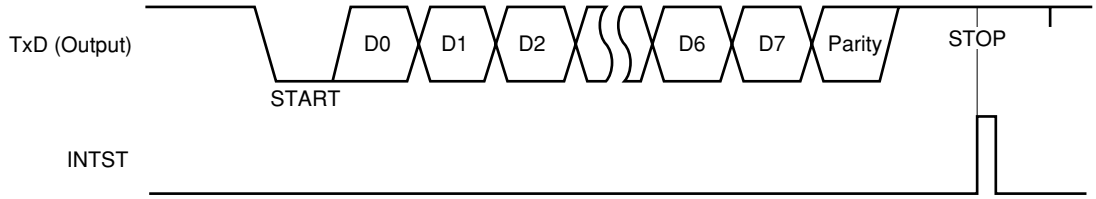
If the TXE bit is cleared (0) during a transmit operation, the transmit operation is discontinued immediately.

Figure 12-6. Interrupt Timing of Asynchronous Serial Interface Transmission Completion

(a) Stop bit length: 1



(b) Stop bit length: 2



Cautions 1. After $\overline{\text{RESET}}$ input the transmit shift register (TXS) is emptied but a transmission completion interrupt is not generated. A transmit operation can be started by writing transmit data to the TXS.

2. An asynchronous serial interface mode register (ASIM) rewrite should not be performed during a transmit operation. If an ASIM rewrite is performed during a transmit operation, subsequent transmit operations may not be possible (normal operation is restored by $\overline{\text{RESET}}$ input). Software can determine whether transmission is in progress by using a transmission completion interrupt (INTST) or the interrupt request flag (STIF) set by INTST.

12.2.6 Reception

When the RXE bit of the asynchronous serial interface mode register (ASIM) is set (1), receive operations are enabled and sampling of the RxD input pin is performed.

RxD input pin sampling is performed using the serial clock (divide-by-m counter input clock) specified by ASIM and band rate generator control register (BRGC).

When the RxD pin input is driven low, the divide-by-m counter starts counting and a data sampling start timing signal is output on the m'th count. If the RxD pin input is low when sampled again by this start timing signal, the input is recognized as a start bit, the divide-by-m counter is initialized and the count is started, and data sampling is performed. When the character data, parity bit and stop bit are detected following the start bit, reception of one data frame ends.

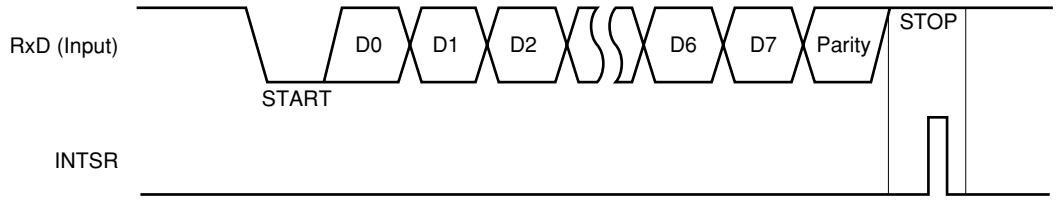
When reception of one data frame ends, the receive data in the shift register is transferred to the receive buffer, RXB, and a reception completion interrupt (INTSR) is generated.

If an error occurs, the receive data in which the error occurred is still transferred to RXB. If bit 1 (ISRM) of the ASIM was cleared (0) when the error occurred, INTSR is generated.

If the ISRM was set (1), INTSR is not generated.

If the RXE bit is cleared (0) during a receive operation, the receive operation is stopped immediately. In this case the contents of RXB and ASIS are not changed, and no INTSR or INTSER interrupt is generated.

Figure 12-7. Interrupt Timing of Asynchronous Serial Interface Reception Completion



Cautions 1. The receive buffer (RXB) must be read even if there is a receive error. If RXB is not read, an overrun error will occur when the next data is received, and the receive error state will continue indefinitely.

2. To disable the reception completion interrupt when a reception error occurs, make sure that wait time equivalent to two pulses of the clock that serves as the reference of the baud rate clock elapse after the reception error occurs until the receive buffers (RXB, RXB2) are read. If the wait time is not inserted, the reception completion interrupt occurs even when it is disabled.
The wait time equivalent to two pulses of the clock that serves as the reference of the baud rate clock can be calculated by the following expression:

$$\text{Wait time} = \frac{2^{n+2}}{f_{CLK}}$$

Remark f_{CLK}: Internal system clock frequency
n : Value of baud rate generator control registers (BRGC, BRGC2) to select tap of 12-bit prescaler (n = 0 to 11)

12.2.7 Receive errors

Three kinds of errors can occur in a receive operation: parity errors, framing errors and overrun errors. As the result of data reception, an error flag is raised in the asynchronous serial interface status register (ASIS) and a receive error interrupt (INTSER) is generated. Receive error causes are shown in Table 12-2.

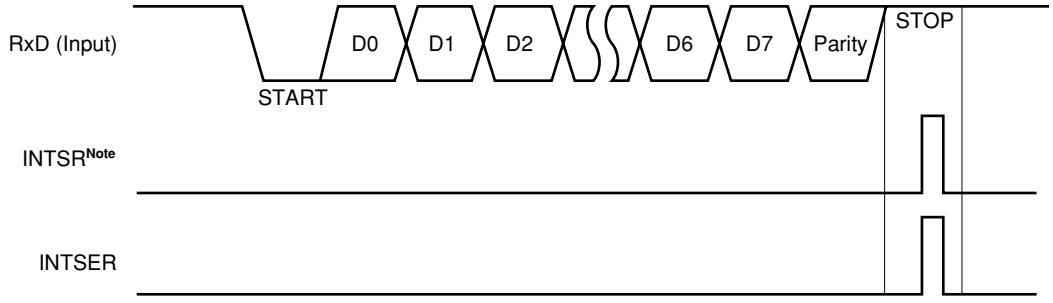
It is possible to detect the occurrence of any of the above errors during reception by reading the contents of the ASIS (refer to **Figures 12-4** and **12-8**).

The contents of the ASIS register are cleared (0) by reading the receive buffer (RXB) or by reception of the next data (if there is an error in the next data, the corresponding error flag is set).

Table 12-2. Causes of Receive Error

Receive Error	Cause
Parity error	Transmit data parity specification and receive data parity do not match
Framing error	Stop bit not detected
Overrun error	Reception of next data completed before data is read from receive buffer

Figure 12-8. Timing of Receive Error



Note If a receive error occurs while the ISRM bit is set (1), INTSR is not generated.

Remark In the μPD784054, a break signal cannot be detected by hardware. As a break signal is a low-level signal of two characters or more, a break signal may be judged to have been input if software detects the occurrence of two consecutive framing errors in which the receive data was 00H. The chance occurrence of two consecutive framing errors can be distinguished from a break signal by having the RxD pin level read by software (confirmation is possible by setting "1" in bit 2 of the port 3 mode register (PM3) and reading port 3 (P3)) and confirming that it is "0".

Cautions 1. The contents of the asynchronous serial interface status register (ASIS) are cleared (0) by reading the receive buffer (RXB) or by reception of the next data. If you want to find the details of an error, therefore, ASIS must be read before reading RXB.

2. The RXB must be read even if there is a receive error. If RXB is not read, an overrun error will occur when the next data is received, and the receive error state will continue indefinitely.

3. To disable the reception completion interrupt when a reception error occurs, make sure that wait time equivalent to two pulses of the clock that serves as the reference of the baud rate clock elapse after the reception error occurs until the receive buffers (RXB, RXB2) are read. If the wait time is not inserted, the reception completion interrupt occurs even when it is disabled.
The wait time equivalent to two pulses of the clock that serves as the reference of the baud rate clock can be calculated by the following expression:

$$\text{Wait time} = \frac{2^{n+2}}{f_{CLK}}$$

Remark f_{CLK}: Internal system clock frequency
n : Value of baud rate generator control registers (BRGC, BRGC2) to select tap of 12-bit prescaler (n = 0 to 11)

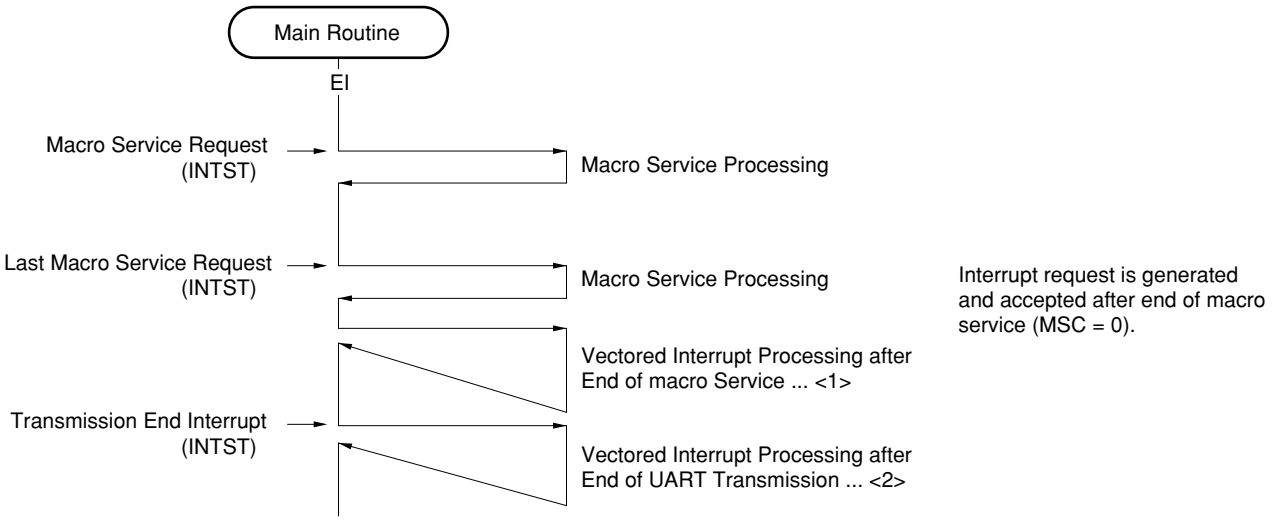
12.2.8 Transmitting/receiving data with macro service

When data is transmitted using a macro service, a vectored interrupt occurs two times. On the other hand, the interrupt occurs only once when data is received using the macro service.

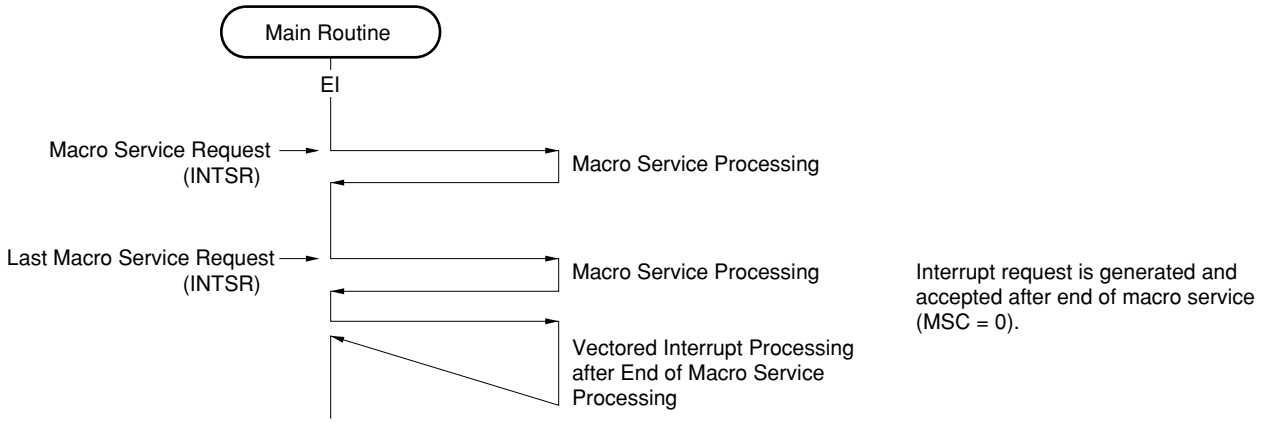
- **Transmitting/receiving data by using macro service**
  Transmission is started by writing data to the transmit shift register (TXS). If this is executed by using a macro service, data is written to TXS and transmitted the specified number of times. The transmission end interrupt (INTST) that occurs after completion of the transmission performs the macro service processing that writes the next data. When the last data has been written to TXS, the macro service is completed (MSC = 0), and a vectored interrupt request is generated (refer to **<1> in Figure 12-9**).
  When data transmission is completed after that (when one frame has been transmitted), INTST is generated again, and the vectored interrupt request is generated again (**<2> in Figure 12-9**).
  To start a macro service by INTST in this way, therefore, a vectored interrupt is generated two times by the same interrupt request (INTST in this case).
  When reception is executed, however, a vectored interrupt request is not generated two times. Because macro service processing that transfers received data to memory is executed by the reception and interrupt (INTSR) that occurs after reception has been completed, a vectored interrupt request is generated only once after the macro service has been completed.

Figure 12-9. Transmission/Reception with Macro Service

(a) Transmission



(b) Reception



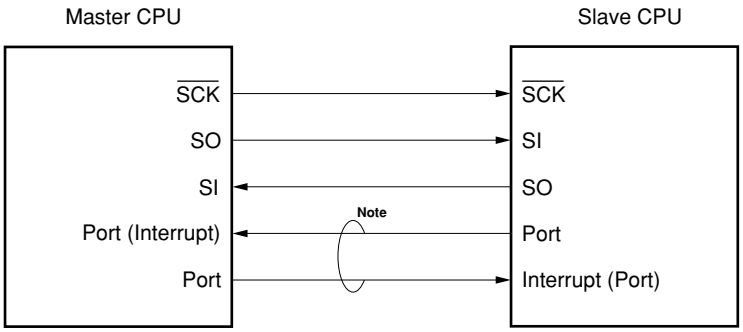
12.3 3-Wire Serial I/O Mode

The 3-wire serial I/O mode is used to communicate with devices that incorporate a conventional clocked serial interface.

Basically, communication is performed using three lines: the serial clock ($\overline{SCK}$), serial data output (SO), and serial data input (SI). Handshaking lines are required when a number of devices are connected.

Figure 12-10. Example of 3-Wire Serial I/O System Configuration

3-wire serial I/O ↔ 3-wire serial I/O

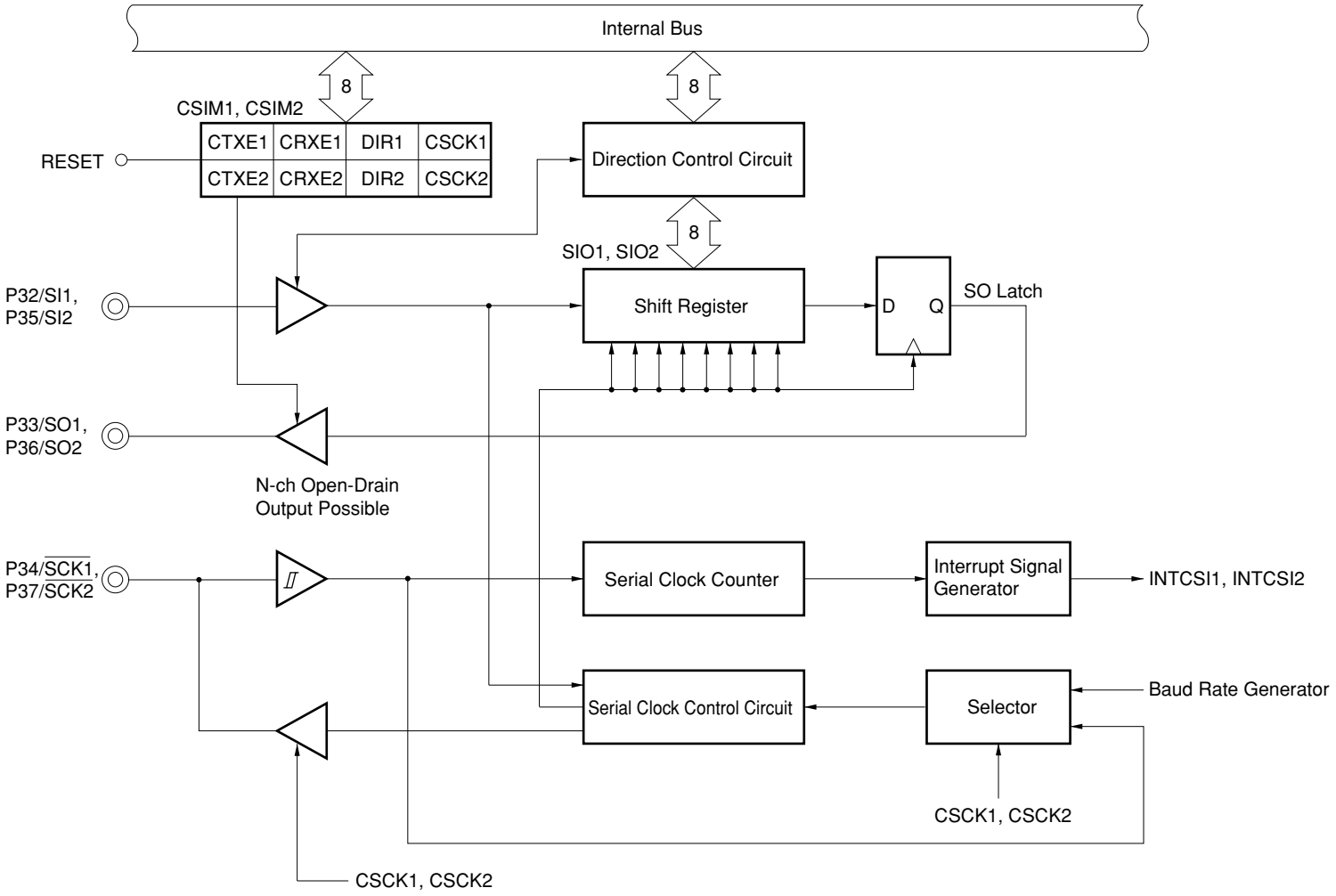


Note Handshaking lines

12.3.1 Configuration in 3-wire serial I/O mode

The block diagram in the 3-wire serial I/O mode is shown in Figure 12-11.

Figure 12-11. Block Diagram of 3-Wire Serial I/O Mode



(1) Shift register (SIO1/SIO2)

The SIO1 and SIO2 convert 8-bit serial data to 8-bit parallel data, and vice versa. SIO1/SIO2 is used for both transmission and reception.

Actual transmit/receive operations are controlled by writing to/reading from SIO1/SIO2.

Reading/writing can be performed by 8-bit manipulation instruction.

The contents of SIO1/SIO2 are undefined after $\overline{\text{RESET}}$ input.

(2) SO latch

The SO latch holds the SO1/SO2 pin output level.

(3) Serial clock selector

Selects the serial clock to be used.

(4) Serial clock counter

Counts the serial clocks output or input in a transmit/receive operation, and checks that 8-bit data transmission/reception has been performed.

(5) Interrupt signal generator

Generates an interrupt request when 8 serial clocks have been counted by the serial clock counter.

(6) Serial clock control circuit

Controls the supply of the serial clock to the shift register, and also controls the clock output to the $\overline{\text{SCK1}}$/$\overline{\text{SCK2}}$ pins when the internal clock is used.

(7) Direction control circuit

Switches between MSB-first and LSB-first modes.

12.3.2 Clocked serial interface mode registers (CSIM1, CSIM2)

The CSIM1 and CSIM2 are 8-bit registers that specify operations in the 3-wire serial I/O mode.

These registers can be read or written to by an 8-bit manipulation instruction or bit manipulation instruction. The CSIM1 and CSIM2 format is shown in Figure 12-12.

These registers are cleared to 00H by $\overline{\text{RESET}}$ input.

Figure 12-12. Formats of Clocked Serial Interface Mode Register 1 (CSIM1) and Clocked Serial Interface Mode Register 2 (CSIM2)

Address : 0FF84H, 0FF85H On reset: 00H R/W

	⑦	⑥	5	4	3	2	1	0
CSIM1	CTXE1	CRXE1	0	0	0	DIR1	CSCK1	0
CSIM2	CTXE2	CRXE2	0	0	0	DIR2	CSCK2	0

(n = 1, 2)

CTXEn	CRXEn	Transmission/Reception
0	0	Disables transmission/reception, or asynchronous serial interface mode
0	1	Enables reception
1	0	Enables transmission
1	1	Enables transmission/reception

DIRn	Specifies Operation Mode (transfer bit sequence)
0	MSB first
1	LSB first

CSCKn	Serial Clock Select Bit	
	Source Clock	$\overline{\text{SCKn}}$ (when CTXEn, CRXEn = 1)
0	External input clock to $\overline{\text{SCKn}}$ pin	Input
1	Baud rate generator output	CMOS output

Caution Even if the DIRn (n = 1, 2) bit is changed after writing to the shift register (SIOn: n = 1, 2), data is output with the setting before change. Therefore, set the DIRn bit before writing to SIOn.

12.3.3 Basic operation timing

In the 3-wire serial I/O mode, data transmission/reception is performed in 8-bit units. Data is transmitted/received bit by bit in MSB-first or LSB-first order in synchronization with the serial clock.

MSB/LSB switching is specified by the DIR1 bit of the clock serial interface mode register (CSIM1).

Transmit data is output in synchronization with the fall of $\overline{SCK1}$, and receive data is sampled on the rise of $\overline{SCK1}$.

An interrupt request (INTCSI1) is generated on the 8th rise of $\overline{SCK1}$.

When the internal clock is used as $\overline{SCK1}$, $\overline{SCK1}$ output is stopped on the 8th rise of $\overline{SCK1}$ and $\overline{SCK1}$ remains high until the next data transmit or receive operation is started.

3-wire serial I/O mode timing is shown in Figure 12-13.

Figure 12-13. Timing of 3-Wire Serial I/O Mode (1/2)

(a) MSB-first

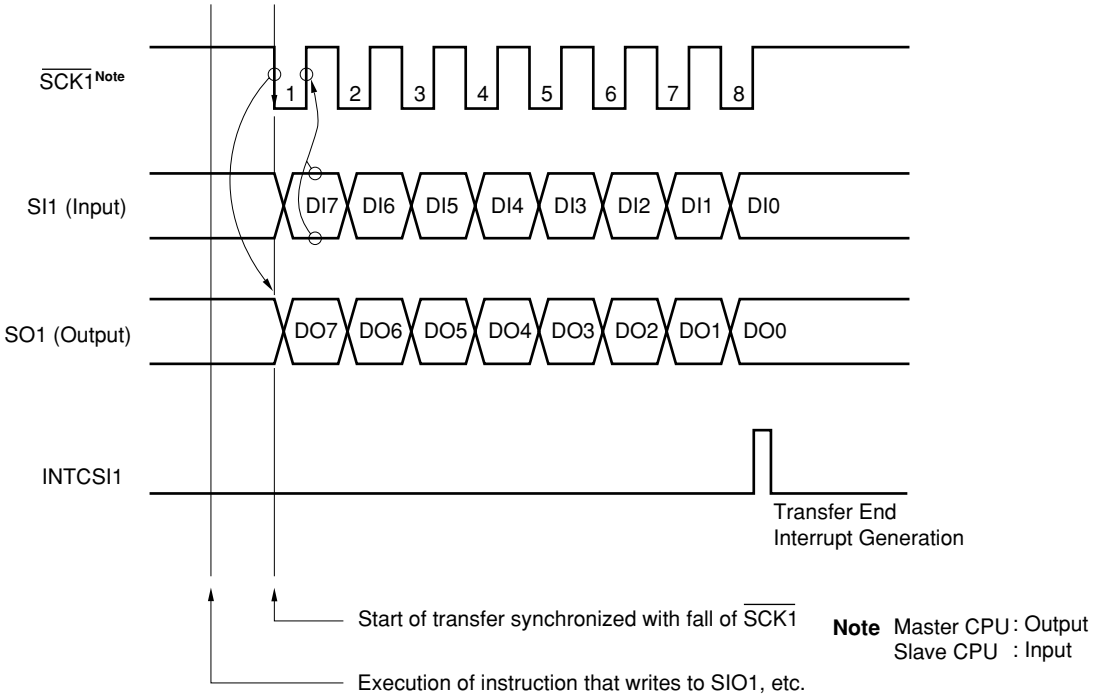
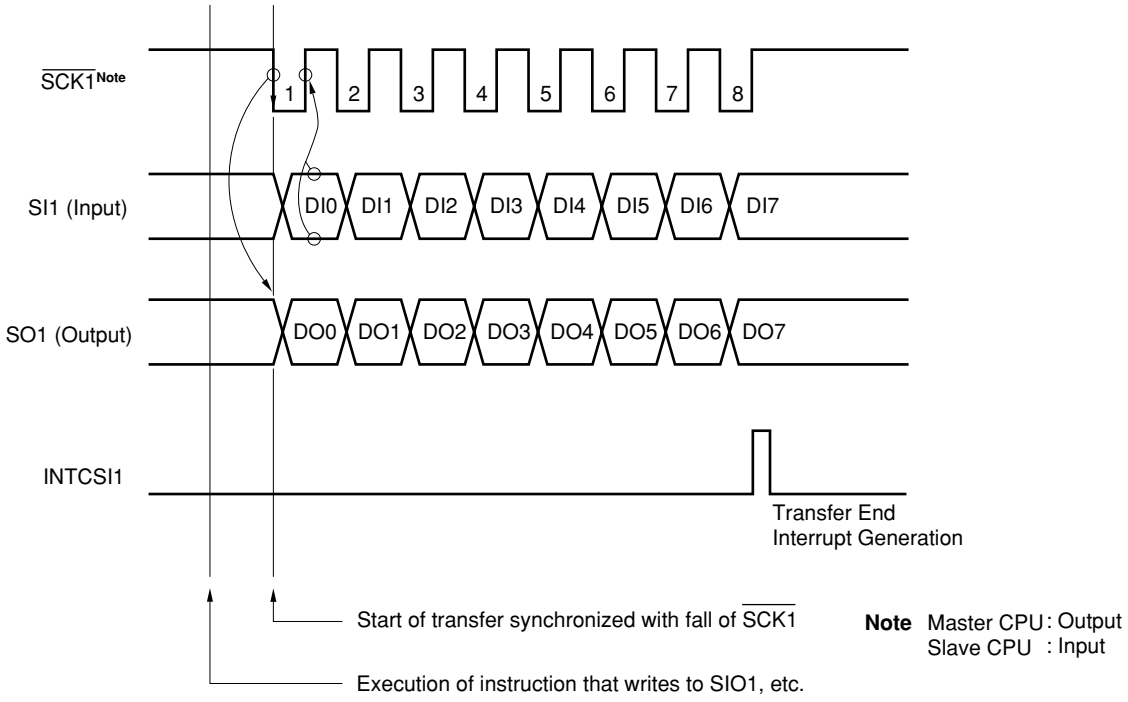


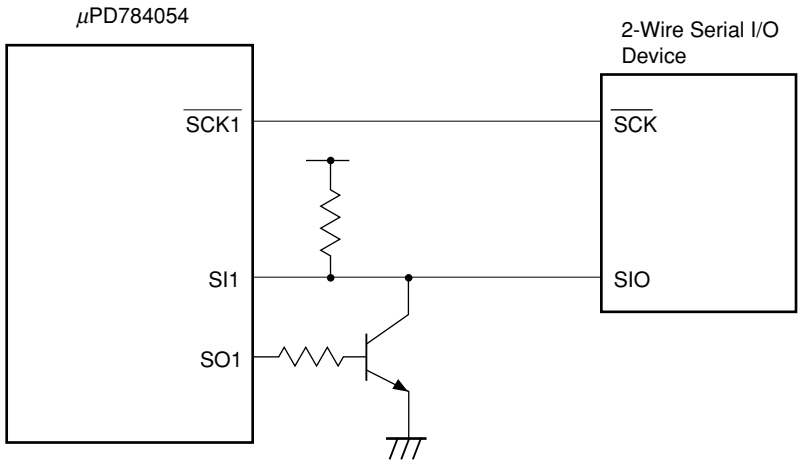
Figure 12-13. Timing of 3-Wire Serial I/O Mode (2/2)

(b) LSB-first



Remark If the μPD784054 is connected to a 2-wire serial I/O device, a buffer should be connected to the SO1 pin as shown in Figure 12-14. In the example shown in Figure 12-14, the output level is inverted by the buffer, and therefore the inverse of the data to be output should be written to SIO1.
In addition, non-connection of the internal pull-up resistor should be specified for the P33/SO1 pin.

Figure 12-14. Example of Connection to 2-Wire Serial I/O



12.3.4 Operation when transmission only is enabled

A transmit operation is performed when the CTXE1 bit of clocked serial interface mode register (CSIM1) is set (1). The transmit operation starts when a write to the shift register (SIO1) is performed while the CTXE1 bit is set (1).

When the CTXE1 bit is cleared (0), the SO1 pin is in the output high level.

(1) When the internal clock is selected as the serial clock

When transmission starts, the serial clock is output from the $\overline{\text{SCK1}}$ pin and data is output in sequence from SIO1 to the SO1 pin in synchronization with the fall of the serial clock, and SI1 pin signals are shifted into SIO1 in synchronization with the rise of the serial clock.

There is a delay of up to one $\overline{\text{SCK1}}$ clock cycle between the start of transmission and the first fall of $\overline{\text{SCK1}}$.

If transmission is disabled during the transmit operation (by clearing (0) the CTXE1 bit), $\overline{\text{SCK1}}$ clock output is stopped and the transmit operation is discontinued on the next rise of $\overline{\text{SCK1}}$. In this case an interrupt request (INTCSI1) is not generated, and the SO1 pin becomes output high level.

(2) When an external clock is selected as the serial clock

When transmission starts, data is output in sequence from SIO1 to the SO1 pin in synchronization with the fall of the serial clock input to the $\overline{\text{SCK1}}$ pin after the start of transmission, and SI1 pin signals are shifted into SIO1 in synchronization with the rise of the $\overline{\text{SCK1}}$ pin input. If transmission has not started, shift operations are not performed and the SO1 pin output level does not change even if the serial clock is input to the $\overline{\text{SCK1}}$ pin.

If transmission is disabled during the transmit operation (by clearing (0) the CTXE1 bit), the transmit operation is discontinued and subsequent $\overline{\text{SCK1}}$ input is ignored. In this case an interrupt request (INTCSI1) is not generated, and the SO1 pin becomes output high level.

12.3.5 Operation when reception only is enabled

A receive operation is performed when the CRXE1 bit of the clocked serial interface mode register (CSIM1) is set (1). The receive operation starts when the CRXE1 changes from “0” to “1”, or when a read from shift register (SIO1) is performed.

(1) When the internal clock is selected as the serial clock

When reception starts, the serial clock is output from the $\overline{\text{SCK1}}$ pin and the SI1 pin data is fetched in sequence into shift register (SIO1) in synchronization with the rise of the serial clock.

There is a delay of up to one $\overline{\text{SCK1}}$ clock cycle between the start of reception and the first fall of $\overline{\text{SCK1}}$.

If reception is disabled during the receive operation (by clearing (0) the CRXE1 bit), $\overline{\text{SCK1}}$ clock output is stopped and the receive operation is discontinued on the next rise of $\overline{\text{SCK1}}$. In this case an interrupt request (INTCSI1) is not generated, and the contents of the SIO1 are undefined.

(2) When an external clock is selected as the serial clock

When reception starts, the SI1 pin data is fetched into shift register (SIO1) in synchronization with the rise of the serial clock input to the $\overline{\text{SCK1}}$ pin after the start of reception. If reception has not started, shift operations are not performed even if the serial clock is input to the $\overline{\text{SCK1}}$ pin.

If reception is disabled during the receive operation (by clearing (0) the CRXE1 bit), the receive operation is discontinued and subsequent $\overline{\text{SCK1}}$ input is ignored. In this case an interrupt request (INTCSI1) is not generated.

12.3.6 Operation when transmission/reception is enabled

When the CTXE1 bit and CRXE1 bit of the clocked serial interface mode register (CSIM1) register are both set (1), a transmit operation and receive operation can be performed simultaneously (transmit/receive operation). The transmit/receive operation is started when the CRXE1 bit is changed from "0" to "1", or by performing a write to shift register (SIO1).

When a transmit/receive operation is started for the first time, the CRXE1 bit always changes from "0" to "1", and there is thus a possibility that the transmit/receive operation will start immediately, and undefined data will be output. The first transmit data should therefore be written to SIO1 beforehand when both transmission and reception are disabled (when the CTXE1 bit and CRXE1 bit are both cleared (0)), before enabling transmission/reception.

When transmission/reception is disabled (CTXE1 = CRXE1 = 0), the SO1 pin is in the output high level.

(1) When the internal clock is selected as the serial clock

When transmission/reception starts, the serial clock is output from the $\overline{\text{SCK1}}$ pin, data is output in sequence from shift register (SIO1) to the (SO1) pin in synchronization with the fall of the serial clock, and SI1 pin data is shifted in order into SIO1 in synchronization with the rise of the serial clock.

There is a delay of up to one $\overline{\text{SCK1}}$ clock cycle between the start of transmission and the first fall of $\overline{\text{SCK1}}$.

If either transmission or reception is disabled during the transmit/receive operation, only the disabled operation is discontinued. If transmission only is disabled, the SO1 pin becomes output high level. If reception only is disabled, the contents of the SIO1 will be undefined.

If transmission and reception are disabled simultaneously, $\overline{\text{SCK1}}$ clock output is stopped and the transmit and receive operations are discontinued on the next rise of $\overline{\text{SCK1}}$. When transmission and reception are disabled simultaneously, the contents of SIO1 are undefined, an interrupt request (INTCSI1) is not generated, and the SO1 pin becomes output high level.

(2) When an external clock is selected as the serial clock

When transmission/reception starts, data is output in sequence from shift register (SIO1) to the SO1 pin in synchronization with the fall of the serial clock input to the $\overline{\text{SCK1}}$ pin after the start of transmission/reception, and SI1 pin data is shifted in order into SIO1 in synchronization with the rise of the serial clock. If transmission/reception has not started, the SIO1 shift operations are not performed and the SO1 pin output level does not change even if the serial clock is input to the $\overline{\text{SCK1}}$ pin.

If either transmission or reception is disabled during the transmit/receive operation, only the disabled operation is discontinued. If transmission only is disabled, the SO1 pin becomes output high level. If reception only is disabled, the contents of the SIO1 will be undefined.

If transmission and reception are disabled simultaneously, the transmit and receive operations are discontinued and subsequent $\overline{\text{SCK1}}$ input is ignored. When transmission and reception are disabled simultaneously, the contents of SIO1 are undefined, an interrupt request (INTCSI1) is not generated, and the SO1 pin becomes output high level.

12.3.7 Corrective action in case of slippage of serial clock and shift operations

When an external clock is selected as the serial clock, there may be slippage between the number of serial clocks and shift operations due to noise, etc. In this case, since the serial clock counter is initialized by disabling both transmit operations and receive operations (by clearing (0) the CTXE1 bit and CRXE1 bit), synchronization of the shift operations and the serial clock can be restored by using the first serial clock input after reception or transmission is next enabled as the first clock.

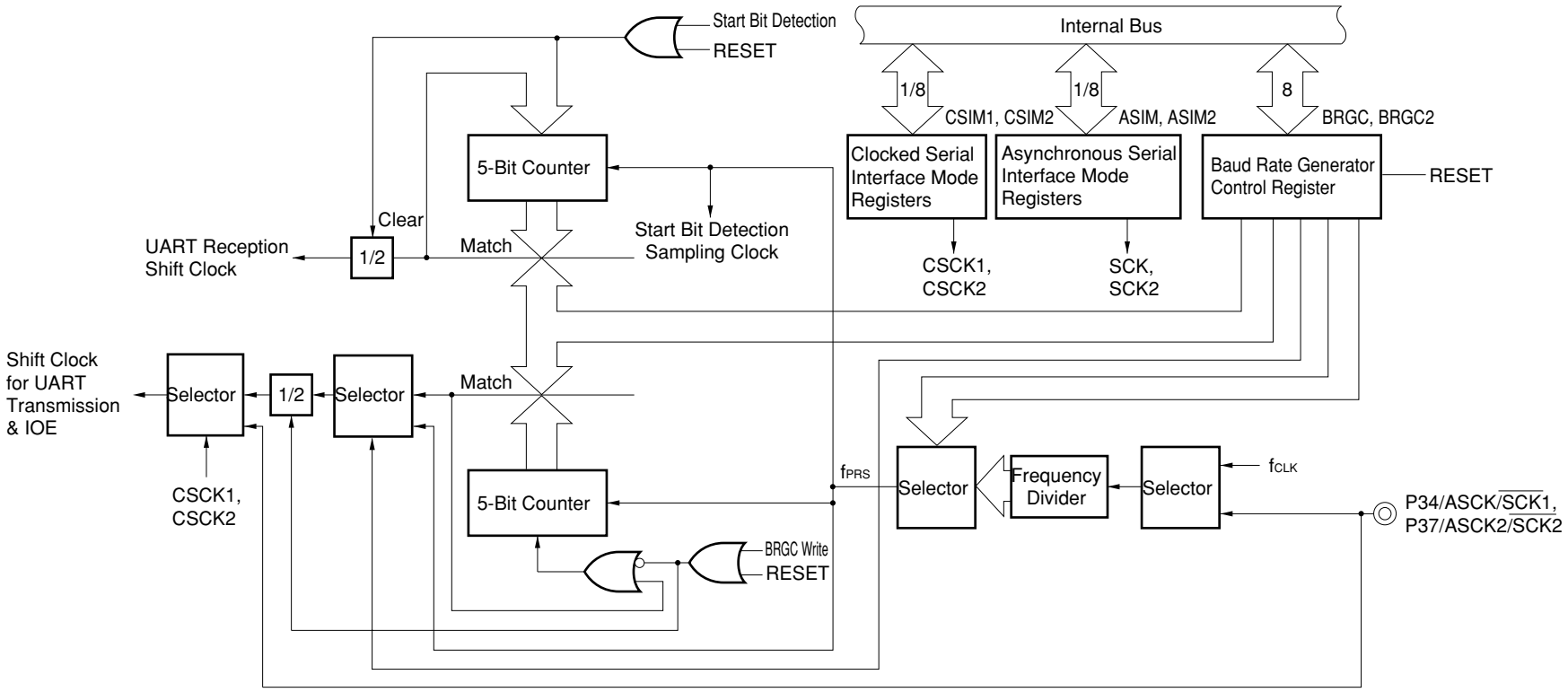
12.4 Baud Rate Generator

The baud rate generator is the circuit that generates the UART/IOE serial clock. Two independent circuits are incorporated, one for each serial interface.

12.4.1 Baud rate generator configuration

The baud rate generator block diagram is shown in Figure 12-15.

Figure 12-15. Block Diagram of Baud Rate Generator



(1) 5-bit counter

Counter that counts the clock (f_{PRS}) by which the output from the frequency divider is selected. Generates a signal with the frequency selected by the low-order 4 bits of the baud rate generator control registers (BRGC/BRGC2).

(2) Frequency divider

Scales the internal clock (f_{CLK}) or, in asynchronous serial interface mode, a clock that is twice the external baud rate input (ASCK/ASCK2), and selects f_{PRS} with the next-stage selector.

(3) Both-edge detection circuit

Detects both edges of the ASCK/ASCK2 pin input signal and generates a signal with a frequency twice that of the ASCK/ASCK2 input clock.

12.4.2 Baud rate generator control register

The BRGC and BRGC2 are 8-bit registers that set the baud rate clock in asynchronous serial interface mode or the shift clock in 3-wire serial I/O mode.

These registers can be read or written to with an 8-bit manipulation instruction. The BRGC and BRGC2 format is shown in Figure 12-16.

$\overline{\text{RESET}}$ input clears the BRGC register to 00H.

Caution When a baud rate generator control register (BRGC, BRGC2) write instruction is executed, the 5-bit counter and 1/2 frequency divider operations are reset. Consequently, if a write to the BRGC and BRGC2 is performed during communication, the generated baud rate clock may be disrupted, preventing normal communication from continuing. The BRGC and BRGC2 should therefore not be written to during communication.

Figure 12-16. Formats of Baud Rate Generator Control Register (BRGC) and Baud Rate Generator Control Register 2 (BRGC2)

Address: 0FF90H, 0FF91H On reset: 00H R/W

	7	6	5	4	3	2	1	0
BRGC	TPS3	TPS2	TPS1	TPS0	MDL3	MDL2	MDL1	MDL0
BRGC2	TPS23	TPS22	TPS21	TPS20	MDL23	MDL22	MDL21	MDL20

TPS3 TPS23	TPS2 TPS22	TPS1 TPS21	TPS0 TPS20	n	Selects Prescaler Output (f_{PRS})
0	0	0	0	0	$f_{CLK}/2$, $f_{ASCK}/2$ Note 1
0	0	0	1	1	$f_{CLK}/4$, $f_{ASCK}/4$
0	0	1	0	2	$f_{CLK}/8$, $f_{ASCK}/8$
0	0	1	1	3	$f_{CLK}/16$, $f_{ASCK}/16$
0	1	0	0	4	$f_{CLK}/32$, $f_{ASCK}/32$
0	1	0	1	5	$f_{CLK}/64$, $f_{ASCK}/64$
0	1	1	0	6	$f_{CLK}/128$, $f_{ASCK}/128$
0	1	1	1	7	$f_{CLK}/256$, $f_{ASCK}/256$
1	0	0	0	8	$f_{CLK}/512$, $f_{ASCK}/512$
1	0	0	1	9	$f_{CLK}/1024$, $f_{ASCK}/1024$
1	0	1	0	10	$f_{CLK}/2048$, $f_{ASCK}/2048$
1	0	1	1	11	$f_{CLK}/4096$, $f_{ASCK}/4096$
Other				Setting prohibited	

MDL3 MDL23	MDL2 MDL22	MDL1 MDL21	MDL0 MDL20	k	Input Clock of Baud Rate Generator Note 2
0	0	0	0	0	$f_{PRS}/16$
0	0	0	1	1	$f_{PRS}/17$
0	0	1	0	2	$f_{PRS}/18$
0	0	1	1	3	$f_{PRS}/19$
0	1	0	0	4	$f_{PRS}/20$
0	1	0	1	5	$f_{PRS}/21$
0	1	1	0	6	$f_{PRS}/22$
0	1	1	1	7	$f_{PRS}/23$
1	0	0	0	8	$f_{PRS}/24$
1	0	0	1	9	$f_{PRS}/25$
1	0	1	0	10	$f_{PRS}/26$
1	0	1	1	11	$f_{PRS}/27$
1	1	0	0	12	$f_{PRS}/28$
1	1	0	1	13	$f_{PRS}/29$
1	1	1	0	14	$f_{PRS}/30$
1	1	1	1	15	f_{PRS} Note 3

Notes 1. This cannot be selected when k = 15 is selected by MDL3 to MDL0 (MDL23 to MDL20).

2. Only $f_{PRS}/16$ can be selected when ASCK (ASCK2) input is used.

3. This can be used only in the 3-wire serial I/O mode.

Remark f_{ASCK} : ASCK (ASCK2) input clock

f_{CLK} : internal system clock

f_{PRS} : Selected clock of prescaler output

12.4.3 Baud rate generator operation

The baud rate generator only operates when UART/IOE transmit/receive operations are enabled. The generated baud rate clock is a signal scaled from the internal clock (f_{CLK}) or a signal scaled from the clock input from the external baud rate input (ASCK) pin.

Caution If a write to the baud rate generator control register (BRGC) is performed during communication, the generated baud rate clock may be disrupted, preventing normal communication from continuing. The BRGC should therefore not be written to during communication.

(1) Baud rate clock generation in UART mode

(a) Using internal clock (f_{CLK})

This function is selected by setting (1) bit 0 (SCK) of the asynchronous serial interface mode register (ASIM). The internal clock (f_{CLK}) is scaled by the frequency divider, this signal (f_{PRS}) is scaled by the 5-bit counter, and the signal further divided by 2 is used as the baud rate. The baud rate is given by the following expression:

$$(\text{Baud rate}) = \frac{f_{CLK}}{(k + 16) \cdot 2^{n+2}}$$

f_{CLK} : Internal system clock frequency
k : Value set in bit MDL3 to bit MDL0 of BRGC (k = 0 to 14)
n : Value set in bit TPS3 to bit TPS0 of BRGC (n = 0 to 11)

(b) Using external baud rate input

This function is selected by clearing (0) bit 0 (SCK) of the asynchronous serial interface mode register (ASIM). When this function is used, bit MDL3 to bit MDL0 of the baud rate generator control register (BRGC) must all be cleared (0) (k= 0).

Set P34 pin (when used with UART2, set P37 pin) in the control mode by using the port 3 mode control register (PMC3).

The ASCK pin input clock is scaled by the frequency divider, and the signal obtained by dividing this signal by 32 (f_{PRS}) (division by 16 and division by 2) is used as the baud rate. The baud rate is given by the following expression:

$$(\text{Baud rate}) = \frac{f_{ASCK}}{2^{n+6}}$$

f_{ASCK}: ASCK pin input clock frequency
n : Value set in bit TPS3 to bit TPS0 of BRGC (n = 0 to 11)

When this function is used, a number of baud rates can be generated by one external input clock.

(3) Serial clock generation in 3-wire serial I/O mode

Selected when the CSCK1 bit of the clocked serial interface mode register 1 (CSIM1) is set (1) and $\overline{SCK1}$ is output.

(a) Normal mode

The internal clock (f_{CLK}) is scaled by the frequency divider, this signal (f_{PRS}) is scaled by the 5-bit counter, and the signal further divided by 2 is used as the serial clock. The serial clock is given by the following expression:

$$(\text{Serial clock}) = \frac{f_{CLK}}{(k + 16) \cdot 2^{n+2}}$$

f_{CLK} : Internal system clock frequency
k : Value set in bit MDL3 to bit MDL0 of BRGC (k = 0 to 14)
n : Value set in bit TPS3 to bit TPS0 of BRGC (n = 0 to 11)

(b) High-speed mode

When this function is used, bit MDL3 to bit MDL0 of the baud rate generator control register (BRGC) are all set (1) (k= 15).

The internal clock (f_{CLK}) is scaled by the frequency divider, and this signal (f_{PRS}) divided by 2 is used as the serial clock. The serial clock is given by the following expression:

$$(\text{Serial clock}) = \frac{f_{CLK}}{2^{n+2}}$$

f_{CLK} : Internal system clock frequency
n : Value set in bit TPS3 to bit TPS0 of BRGC (n = 1 to 11)

12.4.4 Baud rate setting in asynchronous serial interface mode

There are two methods of setting the baud rate, as shown in Table 12-3.

This table shows the range of baud rates that can be generated, the baud rate calculation expression and selection method for each case.

Table 12-3. Methods of Baud Rate Setting

Baud Rate Clock Source		Selection Method	Baud Rate Calculation Expression	Baud Rate Range
Baud rate generator	Internal system clock	SCK in ASIM = 1	$\frac{f_{CLK}}{(k + 16) \cdot 2^{n+2}}$	$\frac{f_{CLK}}{245760} - \frac{f_{CLK}}{64}$
	ASCK input	SCK in ASIM = 0	$\frac{f_{ASCK}}{2^{n+6}}$	$\frac{f_{ASCK}}{131072} - \frac{f_{ASCK}}{64}$ **Note**

f_{CLK} : Internal system clock frequency
k : Value set in bit MDL3 to bit MDL0 of BRGC (k = 0 to 14; refer to **Figure 12-16**)
n : Value set in bit TPS3 to bit TPS0 of BRGC (n = 0 to 11; refer to **Figure 12-16**)
f_{ASCK} : ASCK input clock frequency $(0 - \frac{f_{CLK}}{2})$

Note Including f_{ASCK} input range: $(0 - \frac{f_{CLK}}{128})$

(1) Examples of settings when baud rate generator is used

Examples of baud rate generator control register (BRGC) settings when the baud rate generator is used are shown below.

When the baud rate generator is used, the SCK bit of the asynchronous serial interface mode register (ASIM) should be set (1).

Table 12-4. Examples of BRGC Settings When Baud Rate Generator Is Used

Internal System Clock (f_{CLK})	16.0 MHz		12.5 MHz		10.0 MHz		8.0 MHz	
Baud Rate [bps]	BRGC Value	Baud Rate Error (%)	BRGC Value	Baud Rate Error (%)	BRGC Value	Baud Rate Error (%)	BRGC Value	Baud Rate Error (%)
75	BAH	0.16	B4H	1.73	B0H	1.73	AAH	0.16
110	B2H	1.36	ACH	0.92	A6H	0.88	A2H	1.36
150	AAH	0.16	A4H	1.73	A0H	1.73	9AH	0.16
300	9AH	0.16	94H	1.73	90H	1.73	8AH	0.16
600	8AH	0.16	84H	1.73	80H	1.73	7AH	0.16
1200	7AH	0.16	74H	1.73	70H	1.73	6AH	0.16
2400	6AH	0.16	64H	1.73	60H	1.73	5AH	0.16
4800	5AH	0.16	54H	1.73	50H	1.73	4AH	0.16
9600	4AH	0.16	44H	1.73	40H	1.73	3AH	0.16
19200	3AH	0.16	34H	1.73	30H	1.73	2AH	0.16
31520	30H	0.00	29H	0.00	24H	0.00	20H	0.00
38400	2AH	0.16	24H	1.73	20H	1.73	1AH	0.16
76800	1AH	0.16	14H	1.73	10H	1.73	0AH	0.16

(2) Examples of settings when external baud rate input (ASCK) is used

Table 12-5 shows an example of setting when external baud rate input (ASCK) is used. When using the ASCK input, clear (0) the SCK bit of the asynchronous serial interface mode register (ASIM), and set P34 pin (when used with UART2, set P37 pin) in the control mode by using port 3 mode control register (PMC3).

Table 12-5. Examples of Settings When External Baud Rate Input (ASCK) Is Used

f_{ASCK} (ASCK Input Frequency)	153.6 kHz	4.9152 MHz
Baud Rate [bps]	BRGC Value	BRGC Value
75	50H	A0H
150	40H	90H
300	30H	80H
600	20H	70H
1200	10H	60H
2400	00H	50H
4800	—	40H
9600	—	30H
19200	—	20H
38400	—	10H
76800	—	00H

12.5 Cautions

(1) An asynchronous serial interface mode register (ASIM) rewrite should not be performed during a transmit operation. If an ASIM rewrite is performed during a transmit operation, subsequent transmit operations may not be possible (normal operation is restored by $\overline{\text{RESET}}$ input). Software can determine whether transmission is in progress by using a transmission completion interrupt (INTST) or the interrupt request flag (STIF) set by INTST.

(2) After $\overline{\text{RESET}}$ input the transmit shift register (TXS) is emptied but a transmission completion interrupt is not generated. A transmit operation can be started by writing transmit data to the TXS.

(3) The receive buffer (RXB) must be read even if there is a receive error. If RXB is not read, an overrun error will occur when the next data is received, and the receive error state will continue indefinitely.

(4) To disable the reception completion interrupt when a reception error occurs, make sure that wait time equivalent to two pulses of the clock that serves as the reference of the baud rate clock elapse after the reception error occurs until the receive buffers (RXB, RXB2) are read. If the wait time is not inserted, the reception completion interrupt occurs even when it is disabled.
The wait time equivalent to two pulses of the clock that serves as the reference of the baud rate clock can be calculated by the following expression:

$$\text{Wait time} = \frac{2^{n+2}}{f_{CLK}}$$

Remark f_{CLK}: Internal system clock frequency
n : Value of baud rate generator control registers (BRGC, BRGC2) to select tap of 12-bit prescaler (n = 0 to 11)

(5) The contents of the asynchronous serial interface status register (ASIS) are cleared (0) by reading the receive buffer (RXB) or by reception of the next data. If you want to find the details of an error, therefore, ASIS must be read before reading RXB.

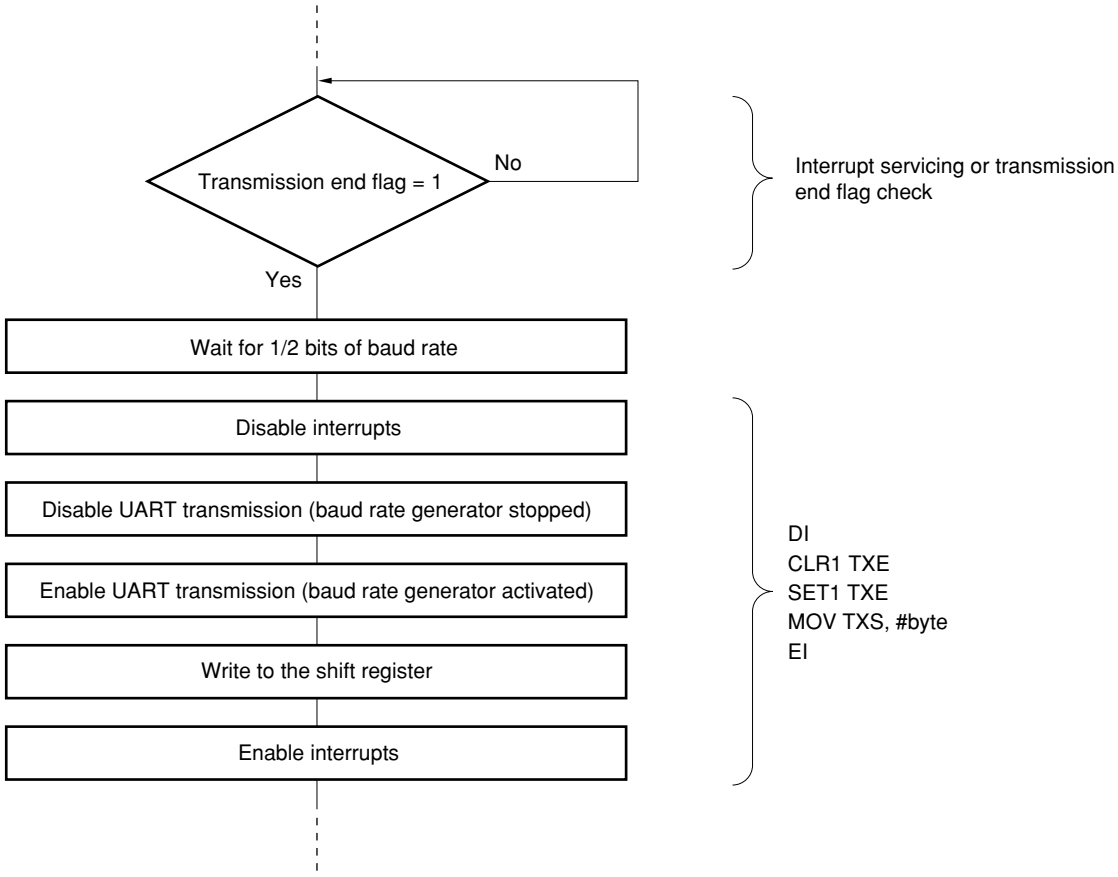
(6) In the 3-wire serial I/O mode, even if the DIRn (n = 1, 2) bit of the clocked serial interface mode register (CSIMn: n = 1, 2) is changed after writing to the shift register (SIOn: n = 1, 2), data is output with the setting before change. Therefore, set the DIRn bit before writing to SIOn.

(7) The baud rate generator control register (BRGC) should not be written to during communication. If a write instruction is executed, the 5-bit counter and 1/2 frequency divider operations will be reset, and the generated baud rate clock may be disrupted, preventing normal communication from continuing.

★ (8) The start bit may not be output correctly if the timing at which the shift register shifts data conflicts with a write to the shift register in asynchronous serial interface mode.

When writing data to the shift register, restart the baud rate generator to prevent the timing at which the UART shift register shifts data matching the timing of a write to the shift register is performed. At this time, disable acknowledgement of interrupt requests, as shown in the preventive program below, so that the UART transmission enable and the write processing to the shift register can be performed successively. At the same time, disable the activation and operation of the macro service during UART transmission. In addition, set the data to be transmitted next to the shift register after the time of 1/2 the bits of the baud rate has elapsed after the transmission end flag was set.

[Flowchart of preventive program]



Note that the relationship between the division ratio of the internal system clock to the oscillation frequency and the baud rate must satisfy the following expressions <1> to <4> when the above preventive program is used.

- When high-speed fetch is selected (the IFCH bit of the memory expansion mode register (MM) is set to 1) and the internal clock is specified as the clock to generate the baud rate clock:

$$(k+15) \times 2^{n+3} > 17 \times a \text{ ..... <1>}$$

- When high-speed fetch is selected (the IFCH bit of the memory expansion mode register (MM) is set to 1) and the clock input from the ASCK pin is specified as the clock to generate the baud rate clock:

$$15 \times 2^{n+2}/f_{ASCK} > 17 \times a/f_{XX} \text{ ..... <2>}$$

- When normal fetch is selected (the IFCH bit of the memory expansion mode register (MM) is set to 0) and the internal clock is specified as the clock to generate the baud rate clock:

$$(k+15) \times 2^{n+3} > \{3 \times (3+b+c)+13\} \times a \text{ ..... <3>}$$

- When normal fetch is selected (the IFCH bit of the memory expansion mode register (MM) is set to 0) and the clock input from the ASCK pin is specified as the clock to generate the baud rate clock:

$$15 \times 2^{n+2}/f_{ASCK} > \{3 \times (3+b+c)+13\} \times a/f_{XX} \text{ ..... <4>}$$

Remark f_{XX}: Oscillation frequency or external clock input frequency
f_{ASCK}: Frequency of clock input from ASCK pin
a: Division ratio of internal clock to oscillation frequency
b: Access wait value to read/write when external memory is accessed
c: Address wait value to address output when external memory is accessed
k: Set value of the MDL3 to MDL0 bits (MDL23 to MDL20) of the BRGC (BRGC2) register
n: Set value of the TPS3 to TPS0 bits (TPS23 to TPS20) of the BRGC (BRGC2) register

<Formats of Baud Rate Generator Control Register (BRGC) and Baud Rate Generator Control Register 2 (BRGC2)>

Address: 0FF90H, 0FF91H On reset: 00H R/W

	7	6	5	4	3	2	1	0
BRGC	TPS3	TPS2	TPS1	TPS0	MDL3	MDL2	MDL1	MDL0
BRGC2	TPS23	TPS22	TPS21	TPS20	MDL23	MDL22	MDL21	MDL20

TPS3 TPS23	TPS2 TPS22	TPS1 TPS21	TPS0 TPS20	n	Selects Prescaler Output (f_{PRS})
0	0	0	0	0	$f_{XX}/4$, $f_{ASCK}/2$[Note 1]
0	0	0	1	1	$f_{XX}/8$, $f_{ASCK}/4$
0	0	1	0	2	$f_{XX}/16$, $f_{ASCK}/8$
0	0	1	1	3	$f_{XX}/32$, $f_{ASCK}/16$
0	1	0	0	4	$f_{XX}/64$, $f_{ASCK}/32$
0	1	0	1	5	$f_{XX}/128$, $f_{ASCK}/64$
0	1	1	0	6	$f_{XX}/256$, $f_{ASCK}/128$
0	1	1	1	7	$f_{XX}/512$, $f_{ASCK}/256$
1	0	0	0	8	$f_{XX}/1024$, $f_{ASCK}/512$
1	0	0	1	9	$f_{XX}/2048$, $f_{ASCK}/1024$
1	0	1	0	10	$f_{XX}/4096$, $f_{ASCK}/2048$
1	0	1	1	11	$f_{XX}/8192$, $f_{ASCK}/4096$
Other				Setting prohibited	

MDL3 MDL23	MDL2 MDL22	MDL1 MDL21	MDL0 MDL20	k	Input Clock of Baud Rate Generator[Note 2]
0	0	0	0	0	$f_{PRS}/16$
0	0	0	1	1	$f_{PRS}/17$
0	0	1	0	2	$f_{PRS}/18$
0	0	1	1	3	$f_{PRS}/19$
0	1	0	0	4	$f_{PRS}/20$
0	1	0	1	5	$f_{PRS}/21$
0	1	1	0	6	$f_{PRS}/22$
0	1	1	1	7	$f_{PRS}/23$
1	0	0	0	8	$f_{PRS}/24$
1	0	0	1	9	$f_{PRS}/25$
1	0	1	0	10	$f_{PRS}/26$
1	0	1	1	11	$f_{PRS}/27$
1	1	0	0	12	$f_{PRS}/28$
1	1	0	1	13	$f_{PRS}/29$
1	1	1	0	14	$f_{PRS}/30$
1	1	1	1	15	f_{PRS}[Note 3]

Notes 1. This cannot be selected when k = 15 is selected by MDL3 through MDL0 (MDL23 through MDL20).

2. Only $f_{PRS}/16$ can be selected when ASCK (ASCK2) input is used.

3. This can be used only in the 3-wire serial I/O mode.

Remark f_{ASCK} : ASCK (ASCK2) input clock

f_{XX} : Oscillation frequency or external clock input frequency

f_{PRS} : Selected clock of prescaler output

[Usage example for expression <3>]

For example, assume that when normal fetch is selected and the internal clock is specified as the clock to generate the baud rate clock, no waits are set to the external memory (b = c = 0), and the highest baud rate is set (k = n = 0). Under these conditions, the result of expression <3> is as follows.

$$a < 5.45$$

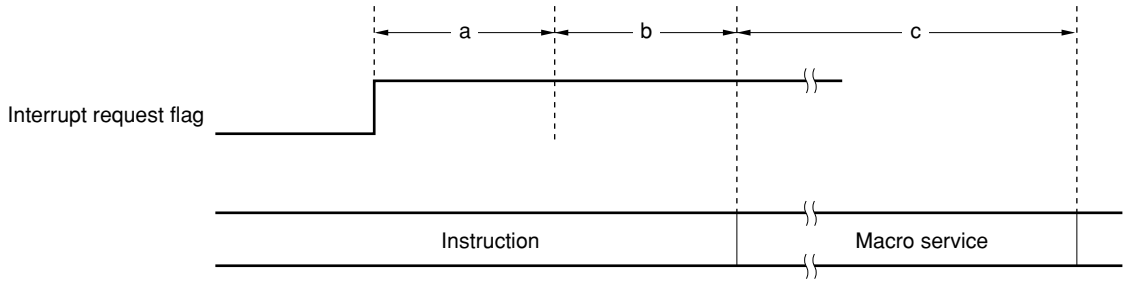
This example shows that 1/2 or 1/4 of the oscillation frequency can be used for the internal system clock, but not for 1/8 or 1/16.

[Cautions on using preventive program]

This bug also occurs if the timing at which the UART shift register shifts data matches a write to the shift register when the UART transmission is performed using a macro service. This bug, however, can be prevented if the following three methods are implemented (these methods eliminate the timing that causes this bug.)

- Activate the macro service immediately after enabling UART transmission (SET1 TXE).
- Set a longer cycle for the baud rate than the time from a macro service request to its termination.
- Make sure that the macro service for UART transmission is not held pending by another macro service. (When UART transmission is performed using a macro service, disable the processing of other macro service requests with a higher priority.)

The execution time from a macro service request to its termination is the sum of a, b, and c in the figure below.



a: Time for judging the interrupt priority after the interrupt request flag is set
It takes 8 system clocks to judge the interrupt priority after the interrupt request flag is set.

b: Time from when the interrupt request flag is set until the instruction being executed is terminated
The macro service is executed when the instruction that was being executed when the interrupt request flag was set is terminated. If the instruction being executed is an instruction to hold the macro service pending temporarily, the macro service is acknowledged when the instruction after instruction is terminated.

[Instruction to hold the macro service pending temporarily]

- EI
- DI
- BRK
- BRKCS
- RETCS
- RETCSB !addr16
- RETI
- RETB
- LOCATION 0H
- LOCATION 0FH
- POP PSW
- POPU post
- MOV PSWL,A
- MOV PSWL,#byte
- MOVG SP,#imm24
- Write and bit manipulation instructions for the interrupt control registers, MK0, MK1L, IMC, ISPR, SNM[Note 1]
- Bit manipulation instruction of PSW[Note 2]

Notes 1. Except for the BT, BF instructions

2. Except for the following instructions

- BT PSWL.bit,$ADDR20
- BF PSWL.bit,$ADDR20
- BT PSWH.bit,$ADDR20
- BF PSWH.bit,$ADDR20
- SET1 CY
- NOT1 CY
- CLR1 CY

c: Time for macro service processing

The macro service processing time when data is transferred to the SFR is shown below.

Macro Service Processing Type		Data Area	
		IRAM	Other
Memory to SFR (1 byte)	Block transfer mode: : BLKTRS	24	–
	Block transfer mode (with memory pointer) : BLKTRS-P	30	32

Unit: Clock = 1/f_{CLK}

Remarks 1. Add the number of waits (number of clocks) at data access to the above value when using the data area as external memory or internal ROM not specified for high-speed fetch (EMEM16, EMEM8).

2. IRAM: Internal high-speed RAM

EMEM16: Memory that is external memory or internal ROM not specified for high-speed fetch, and is set to a 16-bit bus width.

EMEM8: Memory that is external memory or internal ROM not specified for high-speed fetch, and is set to an 8-bit bus width.

CHAPTER 13 EDGE DETECTION FUNCTION

P20 to P27 have an edge detection function that allows a rising edge/falling edge to be set programmably, and the detected edge is sent to internal hardware. The relation between pins P20 to P27 and the use of the detected edge is shown in Table 13-1.

Table 13-1. Pins P20 to P27 and Use of Detected Edge

Pin	Use	Detected Edge Specification Register
P20	NMI, standby circuit control	INTM0
P21	INTP0, CC00 capture signal of timer 0	
P22	INTP1, CC01 capture signal of timer 0	
P23	INTP2, CC02 capture signal of timer 0	
P24	INTP3, CC03 capture signal of timer 0	INTM1
P25	INTP4, conversion start signal of A/D converter	
P26	INTP5	
P27	INTP6	

The edge detection function operates at all times except in STOP mode and IDLE mode (although the edge detection function for pin P20 also operates in STOP mode and IDLE mode).

For pins P21 to P27, the noise elimination time when edge detection is performed can be selected by software.

13.1 Edge Detection Function Control Registers

13.1.1 External interrupt mode registers (INTM0, INTM1)

The INTMn (n = 0, 1) specify the valid edge to be detected on pins P20 to P27. The INTM0 specifies the valid edge for pins P20 to P23, and the INTM1 specifies the valid edge for pins P24 to P27.

The INTMn can be read or written to with an 8-bit manipulation instruction or bit manipulation instruction. The format of INTM0 and INTM1 are shown in Figures 13-1 and 13-2 respectively.

$\overline{\text{RESET}}$ input clears these registers to 00H.

Figure 13-1. Format of External Interrupt Mode Register 0 (INTM0)

Address: 0FF3CH On reset: 00H R/W

	7	6	5	4	3	2	1	0
INTM0	ES21	ES20	ES11	ES10	ES01	ES00	0	ESNMI

ES21	ES20	Specifies Edge To Be Detected of P23 (INTP2, CC02 capture trigger) pin input
0	0	Falling edge
0	1	Rising edge
1	0	Setting prohibited
1	1	Both falling and rising edges

ES11	ES10	Specifies Edge To Be Detected of P22 (INTP1, CC01 capture trigger) Pin Input
0	0	Falling edge
0	1	Rising edge
1	0	Setting prohibited
1	1	Both falling and rising edges

ES01	ES00	Specifies Edge To Be Detected of P21 (INTP0, CC00 capture trigger) Pin Input
0	0	Falling edge
0	1	Rising edge
1	0	Setting prohibited
1	1	Both falling and rising edges

ESNMI	Specifies Edge To Be Detected of P20 (NMI) Pin Input
0	Falling edge
1	Rising edge

Figure 13-2. Format of External Interrupt Mode Register 1 (INTM1)

Address : 0FF3DH On reset : 00H R/W

	7	6	5	4	3	2	1	0
INTM1	ES61	ES60	ES51	ES50	ES41	ES40	ES31	ES30

ES61	ES60	Specifies Edge To Be Detected of P27 (INTP6) Pin Input
0	0	Falling edge
0	1	Rising edge
1	0	Setting prohibited
1	1	Both rising and falling edges

ES51	ES50	Specifies Edge To Be Detected of P26 (INTP5) Pin Input
0	0	Falling edge
0	1	Rising edge
1	0	Setting prohibited
1	1	Both rising and falling edges

ES41	ES40	Specifies Edge To Be Detected of P25 (INTP4, A/D conversion start trigger) Pin Input
0	0	Falling edge
0	1	Rising edge
1	0	Setting prohibited
1	1	Both rising and falling edges

ES31	ES30	Specifies Edge To Be Detected of P24 (INTP3, CC03 capture trigger) Pin Input
0	0	Falling edge
0	1	Rising edge
1	0	Setting prohibited
1	1	Both rising and falling edges

Caution If the valid edge is changed by writing to the external interrupt mode register (INTMn: n = 0, 1), the valid edge is not detected. If the edge is input while the valid edge is changed, whether the input edge is judged as the valid edge or not is undefined.

13.1.2 Interrupt valid edge flag registers (IEF1, IEF2)

IEF1 and IEF2 are flag registers that indicate which of the rising or falling edge is generated when an edge is detected by the INPT0 to INTP6 pin. By checking these flag registers, which of the rising or falling edge is generated can be determined if both the rising and falling edges are specified as the valid edge of an interrupt.

These registers can be read or written by using an 8-bit manipulation instruction or a bit manipulation instruction. Figures 13-3 and 13-4 show the formats of IEF1 and IEF2.

The values of these registers are undefined when $\overline{\text{RESET}}$ is input.

Figure 13-3. Format of Interrupt Valid Edge Flag Register 1 (IEF1)

Address : 0FF3EH On reset : undefined R/W

	⑦	⑥	⑤	④	③	②	①	⓪
IEF1	IEFH3	IEFL3	IEFH2	IEFL2	IEFH1	IEFL1	IEFH0	IEFL0

IEFHn	Rising Edge Flag of INTPn Pin (n = 0 to 3)
0	Rising edge is not generated
1	Rising edge is generated

IEFLn	Falling Edge Flag of INTPn Pin (n = 0 to 3)
0	Falling edge is not generated
1	Falling edge is generated

Figure 13-4. Format of Interrupt Valid Edge Flag Register 2 (IEF2)

Address : 0FF3FH On reset : undefined R/W

	7	6	⑤	④	③	②	①	⓪
IEF2	0	0	IEFH6	IEFL6	IEFH5	IEFL5	IEFH4	IEFL4

IEFHn	Rising Edge Flag of INTPn Pin (n = 4 to 6)
0	Rising edge is not generated
1	Rising edge is generated

IEFLn	Falling Edge Flag of INTPn Pin (n = 4 to 6)
0	Falling edge is not generated
1	Falling edge is generated

Cautions 1. After checking the flag, clear the flag to "0" by software.

2. The interrupt valid edge flag register (IEFn: n = 1, 2) indicates that an edge has been generated, and has nothing to do with specification of a valid edge. For example, if the valid edge of the INTP0 pin is specified to be the rising edge, and if the falling edge is generated, the interrupt request signal is not generated, but the IEFL0 flag is set to "1".

3. If the INTPn (n = 0 to 6) pin is "1" after the reset signal has been deasserted, the rising edge is recognized, and the IEFHn (n = 0 to 6) flag is set to "1". Even when the IEFHn flag is used as a digital port (P21 to P27), it may be set to 1. Be sure to clear (0) the IEFHn flag before checking the edge of an external interrupt.

13.1.3 Noise protection control register (NPC)

NPC is a register that specifies a sampling clock used to reject the digital noise on the P21/INTP0 to P27/INTP6 pins.

This register is read or written by using an 8-bit manipulation instruction or a bit manipulation instruction.

Figure 13-5 shows the format of NPC.

The value of this register is cleared to 00H when $\overline{\text{RESET}}$ is input.

Figure 13-5. Format of Noise Protection Control Register (NPC)

Address : 0FF3BH On reset : undefined R/W

	7	6	5	4	3	2	1	0
NPC	0	NI6	NI5	NI4	NI3	NI2	NI1	NI0

(f_{CLK} = 16 MHz)

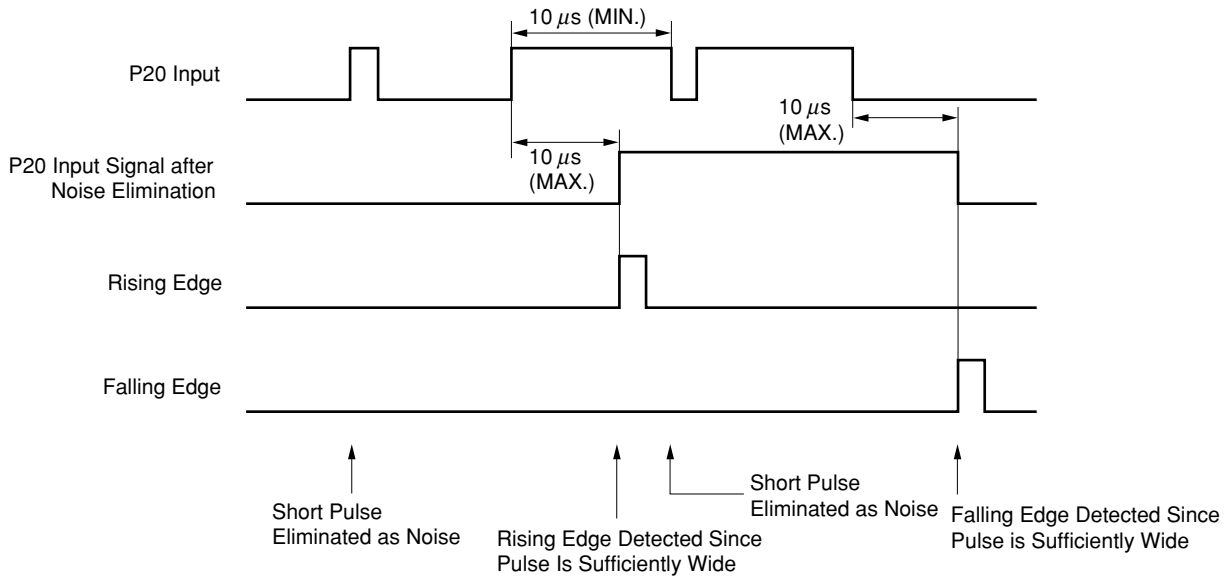
NIn	Specifies Sampling Clock to Reject Noise on INTPn Pin		
	(n = 0 to 6)	Pulse Width Rejected as Noise	Minimum Pulse Width Recognized as Signal
0	f_{CLK}	$3/f_{CLK}$ (0.19 μs)	$4/f_{CLK}$ (0.25 μs)
1	$f_{CLK}/4$	$12/f_{CLK}$ (0.75 μs)	$16/f_{CLK}$ (1.0 μs)

Remark f_{CLK}: internal system clock

13.2 Edge Detection for Pin P20

On pin P20 noise elimination is performed by means of analog delay before edge detection. Therefore, an edge cannot be detected unless the pulse width is a given time (10 μs) or longer.

Figure 13-6. Edge Detection for Pin P20



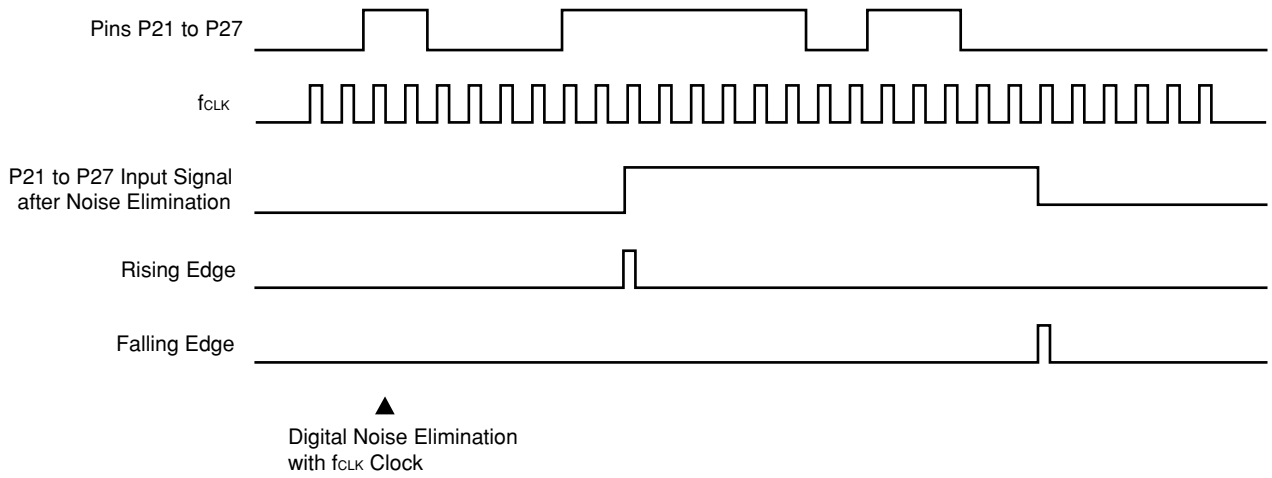
Caution Since analog delay noise elimination is performed on pin P20, an edge is detected up to 10 μs after it is actually input. Also, unlike pins P21 to P27, the delay before an edge is detected is not a specific value, because of differences in the characteristics of various devices.

13.3 Pin Edge Detection for Pins P21 to P27

Edge detection for pins P21 to P27 is performed after digital noise elimination by means of clock sampling. The sampling clock is fixed to f_{CLK}.

In digital noise elimination, input is sampled using the f_{CLK} clock, and if the input level is not the same at least four times in succession (if it is the same only three or fewer times in succession), it is eliminated as noise. Therefore, the level must be maintained for at least 4 f_{CLK} clock cycles (0.25 μs: f_{CLK} = 16 MHz, f_{CLK} = 1/2 f_{XX}, f_{XX} = 32 MHz) in order to be recognized as a valid edge.

Figure 13-7. Edge Detection for Pins P21 to P27



Cautions 1. Since digital noise elimination is performed with the f_{CLK} clock, there is a delay of 4 f_{CLK} clocks between input of an edge to the pin and the point at which the edge is actually detected.

2. If the input pulse width is 4 f_{CLK} clocks, it is uncertain whether a valid edge will be detected. Therefore, to ensure reliable operation, the level should be held for at least 4 clocks.

3. If noise input to a pin is synchronized with the f_{CLK} clock in the μPD784054, it may not be recognized as noise. If there is a possibility of such noise being input, noise should be eliminated by adding a filter to the input pins.

13.4 Cautions

(1) Valid edge detection cannot be performed when the valid edge is changed by a write to the external interrupt mode register (INTMn: n = 0, 1). Also, if an edge is input during a change of the valid edge, that edge may or may not be judged to be a valid edge.

(2) After checking the flag by using the interrupt valid edge flag register (IEFn: n = 1, 2), clear the flag to "0" by software.

(3) The interrupt valid edge flag register (IEFn: n = 1, 2) indicates that an edge has been generated, and has nothing to do with specification of a valid edge. For example, if the valid edge of the INTP0 pin is specified to be the rising edge, and if the falling edge is generated, the interrupt request signal is not generated, but the IEFL0 flag is set to "1".

(4) If the INTPn (n = 0 to 6) pin is "1" after the reset signal has been deasserted, the rising edge is recognized, and the IEFHn (n = 0 to 6) flag of the interrupt valid edge flag register (IEFn: n = 1, 2) is set to "1". Even when the IEFHn flag is used as a digital port (P21 to P27), it may be set (1). Be sure to clear (0) the IEFHn flag before checking the edge of an external interrupt.

(5) Since analog delay noise elimination is performed on pin P20 an edge is detected up to 10 *m*s after it is actually input. Also, unlike pins P21 to P27, the delay before an edge is detected is not a specific value, because of differences in the characteristics of various devices.

(6) Since digital noise elimination is performed on pins P21 to P27 with the f_{CLK} clock, there is a delay of 4 f_{CLK} clocks between input of an edge to the pin and the point at which the edge is actually detected.

(7) If the input pulse width on pins P21 to P27 is 4 f_{CLK} clocks, it is uncertain whether a valid edge will be detected or not. Therefore, to ensure reliable operation, the period of at least 4 clocks and the level must be fixed.

(8) If noise input to pins P21 to P27 is synchronized with the f_{CLK} clock in the μPD784054, it may not be recognized as noise. If there is a possibility of such noise being input, noise should be eliminated by adding a filter to the input pins.

CHAPTER 14 INTERRUPT FUNCTIONS

The μPD784054 is provided with three interrupt request processing modes (refer to **Table 14-1**). These three service modes can be set as required in the program. However interrupt processing by macro service can only be selected for interrupt request sources provided with the macro service processing mode shown in Table 14-2. Context switching cannot be selected for non-maskable interrupts or operand error interrupts.

Multi-processing control using 4 priority levels can easily be performed for maskable vectored interrupts.

Table 14-1. Processing Modes of Interrupt Request

Interrupt Request Processing Mode	Processing Performed	PC & PSW Contents	Processing
Vectored interrupts	Software	Saving to & restoration from stack	Executed by branching to service program at address[Note] specified by vector table
Context switching		Saving to & restoration from fixed area in register bank	Executed by automatic switching to register bank specified by vector table and branching to service program at address[Note] specified by fixed area in register bank
Macro service	Hardware (firmware)	Retained (however, PSW is 0x00H in CPU monitor mode 0)	Execution of pre-set processing 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.

14.1 Interrupt Request Sources

The μPD784054 has the 29 interrupt request sources shown in Table 14-2, with a vector table allocated to each.

Table 14-2. Sources of Interrupt Request (1/2)

Type of Interrupt Request	Default Priority	Interrupt Request Generating Source	Generating Unit	Interrupt Control Register Name	Context Switching	Macro Service	Macro Service Control Word Address	Vector Table Address
Software	None	BRK instruction execution	—	—	Not possible	Not possible	—	3EH
		BRKCS instruction execution	—	—	Possible	Not possible	—	—
Operand error	None	Invalid operand in MOV STBC, #byte instruction or MOV WDM, #byte instruction, and LOCATION instruction	—	—	Not possible	Not possible	—	3CH
Non-maskable	None	NMI (pin input edge detection)	Edge detection	—	Not possible	Not possible	—	2H
		INTWDT (watchdog timer overflow)	Watchdog timer	—	Not possible	Not possible	—	4H

Table 14-2. Sources of Interrupt Request (2/2)

Type of Interrupt Request	Default Priority	Interrupt Request Generating Source	Generating Unit	Interrupt Control Register Name	Context Switching	Macro Service	Macro Service Control Word Address	Vector Table Address
Maskable	0 (highest)	INTOV0 (overflow of timer 0)	Timer 0	OVIC0	Possible	Possible	0FE06H	6H
	1	INTOV1 (overflow of timer 1)	Timer 1	OVIC1			0FE08H	8H
	2	INTOV4 (overflow of timer 4)	Timer 4	OVIC4			0FE0AH	0AH
	3	INTP0 (pin input edge detection)	Edge detection	PIC0			0FE0CH	0CH
		INTCC00 (TM0-CC00 match signal generation)	Timer 0					
	4	INTP1 (pin input edge detection)	Edge detection	P1C1			0FE0EH	0EH
		INTCC01 (TM0-CC01 match signal generation)	Timer 0					
	5	INTP2 (pin input edge detection)	Edge detection	PIC2			0FE10H	10H
		INTCC002 (TM0-CC02 match signal generation)	Timer 0					
	6	INTP3 (pin input edge detection)	Edge detection	PIC3			0FE12H	12H
		INTCC03 (TM0-CC03 match signal generation)	Timer 0					
	7	INTP4 (pin input edge detection)	Edge detection	PIC4			0FE14H	14H
	8	INTP5 (pin input edge detection)	Edge detection	PIC5			0FE16H	16H
	9	INTP6 (pin input edge detection)	Edge detection	PIC6			0FE18H	18H
	10	INTCM10 (TM1-CM10 match signal generation)	Timer 1	CMIC10			0FE1AH	1AH
	11	INTCM11 (TM1-CM11 match signal generation)	Timer 1	CMIC11			0FE1CH	1CH
	12	INTCM40 (TM4-CM40 match signal generation)	Timer 4	CMIC40			0FE26H	26H
	13	INTCM41 (TM4-CM41 match signal generation)	Timer 4	CMIC41			0FE28H	28H
	14	INTSER (UART0 reception error)	Asynchronous serial interface 0	SERIC			0FE2AH	2AH
	15	INTSR (UART0 reception end)		SRIC			0FE2CH	2CH
		INTCSI1 (3-wire serial I/O1 transfer end)	3-wire serial I/O1	CSIIC1				
	16	INTST (UART0 transmission end)	Asynchronous serial interface 0	STIC			0FE2EH	2EH
	17	INTSER2 (UART2 reception error)	Asynchronous serial interface 2	SERIC2			0FE30H	30H
	18	INTSR2 (UART2 reception end)		SRIC2			0FE32H	32H
		INTCSI2 (3-wire serial I/O2 transfer end)	3-wire serial I/O2	CSIIC2				
	19	INTST2 (UART2 transmission end)	Asynchronous serial interface 2	STIC2			0FE34H	34H
	20 (lowest)	INTAD (A/D conversion end)	A/D converter	ADIC			0FE36H	36H

Remarks 1. Th e default priority is a fixed number. This indicates the order of priority when interrupt requests specified as having the same priority are generated simultaneously,

2. The INTSR and INTCSI1 interrupts are generated by the same hardware (they cannot both be used simultaneously). Therefore, although the same hardware is used for the interrupts, two names are provided, for use in each of the two modes. The same applies to INTSR2 and INTCSI2.

14.1.1 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.

14.1.2 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.

14.1.3 Non-maskable interrupts

A non-maskable interrupt is generated by NMI pin input or the watchdog timer.

Non-maskable interrupts are acknowledged unconditionally[Note], even in the interrupt disabled state. They are not subject to interrupt priority control, and are of higher priority that 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

14.1.4 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 interruption, maskable interrupts can be acknowledged by context switching and macro service.

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 14-2. Also, multi-processing 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.

14.2 Interrupt Processing Modes

There are three μPD784054 interrupt processing modes, as follows:

- Vectored interrupt processing
- Macro service
- Context switching

14.2.1 Vectored interrupt processing

When an interrupt is acknowledged, the program counter (PC) and program status word (PSW) are automatically saved to the stack, a branch is made to the address indicated by the data stored in the vector table, and the interrupt processing routine is executed.

14.2.2 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 **14.8 Macro Service Function**).

14.2.3 Context switching

When an interrupt is acknowledged, the prescribed register bank is selected by hardware, a branch is made to a preset 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 **14.4.2 BRKCS instruction software interrupt (software context switching) acknowledgment operation** and **14.7.2 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).

14.3 Interrupt Processing Control Registers

μPD784054 interrupt processing is controlled for each interrupt request by various control registers that perform interrupt processing specification. The interrupt control registers are listed in Table 14-3.

Table 14-3. Control Registers

Register Name	Symbol	Function
Interrupt control registers	OVIC0, OVIC1, OVIC4, PIC0, PIC1, PIC2, PIC3, PIC4, PIC5, PIC6, CMIC10, CMIC11, CMIC40, CMIC41, SERIC, SRIC, CSIIC1, STIC, SERIC2, SRIC2, CSIIC2, STIC2, ADIC	Registers to record generation of interrupt request, control masking, specify vectored interrupt processing or macro service processing, enable or disable context switching function, and specify priority.
Interrupt mask registers	MK0 (MK0L, MK0H) MK1 (MK1L, MK1H)	Control masking of maskable interrupt request. Associated with mask control flag in interrupt control register. Can be accessed in word or 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).
Watchdog timer mode register	WDM	Specifies priorities of interrupt by NMI pin input and overflow of watchdog timer.
Program status word	PSW	Enables or disables accepting maskable interrupt.

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 14-4.

Table 14-4. Interrupt Control Register Flags Corresponding to Interrupt Sources

Default Priority	Interrupt Request Signal		Interrupt Control Registers				
			Interrupt Request Flag	Interrupt Mask Flag	Macro Service Enable Flag	Context Switching Enable Flag	Priority Specification Flag
0 (highest)	INTOV0	OVIC0	OVIF0	OVMK0	OVISM0	OVCSE0	OVPR00 OVPR01
1	INTOV1	OVIC1	OVIF1	OVMK1	OVISM1	OVCSE1	OVPR10 OVPR11
2	INTOV4	OVIC4	OVIF4	OVMK4	OVISM4	OVCSE4	OVPR40 OVPR41
3	INTP0	PIC0	PIF0	PMK0	PISM0	PCSE0	PPR00
	INTCC00						PPR01
4	INTP1	PIC1	PIF1	PMK1	PISM1	PCSE1	PPR10
	INTCC01						PPR11
5	INTP2	PIC2	PIF2	PMK2	PISM2	PCSE2	PPR20
	INTCC02						PPR21
6	INTP3	PIC3	PIF3	PMK3	PISM3	PCSE3	PPR30
	INTCC03						PPR31
7	INTP4	PIC4	PIF4	PMK4	PISM4	PCSE4	PPR40 PPR41
8	INTP5	PIC5	PIF5	PMK5	PISM5	PCSE5	PPR50 PPR51
9	INTP6	PIC6	PIF6	PMK6	PISM6	PCSE6	PPR60 PPR61
10	INTCM10	CMIC10	CMIF10	CMMK10	CMISM10	CMCSE10	CMPR100 CMPR101
11	INTCM11	CMIC11	CMIF11	CMMK11	CMISM11	CMCSE11	CMPR110 CMPR111
12	INTCM40	CMIC40	CMIF40	CMMK40	CMISM40	CMCSE40	CMPR400 CMPR401
13	INTCM41	CMIC41	CMIF41	CMMK41	CMISM41	CMCSE41	CMPR410 CMPR411
14	INTSER	SERIC	SERIF	SERMK	SERISM	SRCSE	SERPR0 SERPR1
15	INTSR	SRIC	SRIF	SRMK	SRISM	SRCSE	SRPR0 SRPR1
	INTCSI1	CSIIC1	CSIIF1	CSIMK1	CSIISM1	CSICSE1	CSIPR10 CSIPR11
16	INTST	STIC	STIF	STMK	STISM	STCSE	STPR0 STPR1
17	INTSER2	SERIC2	SERIF2	SERMK2	SERISM2	SERCSE2	SERPR20 SERPR21
18	INTSR2	SRIC2	SRIF2	SRMK2	SRISM2	SRCSE2	SRPR20 SRPR21
	INTCSI2	CSIIC2	CSIIF2	CSIMK2	CSIISM2	CSICSE2	CSIPR20 CSIPR21
19	INTST2	STIC2	STIF2	STMK2	STISM2	STCSE2	STPR20 STPR21
20 (lowest)	INTAD	ADIC	ADIF	ADMK	ADISM	ADCSE	ADPR0 ADPR1

14.3.1 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 14-1.

(1) Priority specification flags (××PR1, ××PR0)

The priority specification flags specify the priority on an individual interrupt source basis for the 21 maskable interrupts.

Up to 4 priority levels can be specified, and a number of interrupt sources can be specified at the same level. Among maskable interrupt sources, level 0 is the highest priority.

If multiple interrupt requests are generated simultaneously among interrupt source of the same priority level, they are acknowledged in default priority order.

These flags can be manipulated bit-wise by software.

$\overline{\text{RESET}}$ input sets all bits to “1”.

(2) Context switching enable flag (××CSE)

The context switching enable flag specifies that a maskable interrupt request is to be processed by context switching. In context switching, the register bank specified beforehand is selected by hardware, a branch is made to a vector address stored beforehand in the register bank, and at the same time the current contents of the program counter (PC) and program status word (PSW) are saved in the register bank.

Context switching is suitable for real-time processing, since execution of interrupt processing can be started faster than with normal vectored interrupt processing.

This flag can be manipulated bit-wise by software.

$\overline{\text{RESET}}$ input sets all bits to “0”.

(3) Macro service enable flag (××ISM)

The macro service enable flag specifies whether an interrupt request corresponding to that flag is to be handled by vectored interruption or context switching, or by macro service.

When macro service processing is selected, at the end of the macro service the macro service enable flag is automatically cleared (0) by hardware (vectored interrupt processing/context switching processing).

This flag can be manipulated bit-wise by software.

$\overline{\text{RESET}}$ input sets all bits to “0”.

(4) Interrupt mask flag (××MK)

An interrupt mask flag specifies enabling/disabling of vectored interrupt processing and macro service processing for the interrupt request corresponding to that flag.

The interrupt mask flag contents are not changed by the start of interrupt processing, etc., and are the same as the interrupt mask register contents (refer to **14.3.2 Interrupt mask registers (MK0, MK1)**).

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 (××IF)

An interrupt request flag is set (1) by generation of the interrupt request that corresponds to that flag. When the interrupt is acknowledged, the flag is automatically cleared (0) by hardware.

This flag can be manipulated by software.

$\overline{\text{RESET}}$ input sets all bits to “0”.

Figure 14-1. Interrupt Control Registers (××ICn) (1/3)

Address : 0FFE0H-0FFE9H On reset : 43H R/W

	⑦	⑥	⑤	④	3	2	1	0
OVIC0	OVIF0	OVMK0	OVISM0	OVCSE0	0	0	OVPR01	OVPR00
OVIC1	OVIF1	OVMK1	OVISM1	OVCSE1	0	0	OVPR11	OVPR10
OVIC4	OVIF4	OVMK4	OVISM4	OVCSE4	0	0	OVPR41	OVPR40
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
PIC3	PIF3	PMK3	PISM3	PCSE3	0	0	PPR31	PPR30
PIC4	PIF4	PMK4	PISM4	PCSE4	0	0	PPR41	PPR40
PIC5	PIF5	PMK5	PISM5	PCSE5	0	0	PPR51	PPR50
PIC6	PIF6	PMK6	PISM6	PCSE6	0	0	PPR61	PPR60

××IFn	Generation of Interrupt Request
0	No interrupt request (interrupt signal is not generated)
1	Interrupt request (interrupt signal is generated)

××MKn	Enables or Disables Interrupt Processing
0	Enables interrupt processing
1	Disables interrupt processing

××ISMn	Specifies Interrupt Processing Format
0	Vectored interrupt processing/context switching processing
1	Macro service processing

××CSEn	Specifies Context Switching Processing
0	Processed by vectored interrupt
1	Processed by context switching

××PRn1	××PRn0	Specifies Priority of Interrupt Request
0	0	Priority 0 (highest priority)
0	1	Priority 1
1	0	Priority 2
1	1	Priority 3

Figure 14-1. Interrupt Control Registers (××ICn) (2/3)

Address : 0FFEAH-0FFF5H On reset : 43H R/W

	⑦	⑥	⑤	④	3	2	1	0
CMIC10	CMIF10	CMMK10	CMISM10	CMCSE10	0	0	CMPR101	CMPR100
CMIC11	CMIF11	CMMK11	CMISM11	CMCSE11	0	0	CMPR111	CMPR110
CMIC40	CMIF40	CMMK40	CMISM40	CMCSE40	0	0	CMPR401	CMPR400
CMIC41	CMIF41	CMMK41	CMISM41	CMCSE41	0	0	CMPR411	CMPR410
SERIC	SERIF	SERMK	SERISM	SERCSE	0	0	SERPR1	SERPR0
SRIC	SRIF	SRMK	SRISM	SRCSE	0	0	SRPR1	SRPR0
CSIIC1	CSIIF1	CSIMK1	CSIISM1	CSICSE1	0	0	CSIPR11	CSIPR10
STIC	STIF	STMK	STISM	STCSE	0	0	STPR1	STPR0
SERIC2	SERIF2	SERMK2	SERISM2	SERCSE2	0	0	SERPR21	SERPR20

××IFn	Generation of Interrupt Request
0	No interrupt request (interrupt signal is not generated)
1	Interrupt request (interrupt signal is generated)

××MKn	Enables or Disables Interrupt Processing
0	Enables interrupt processing
1	Disables interrupt processing

××ISMn	Specifies Interrupt Processing Format
0	Vectored interrupt processing/context switching processing
1	Macro service processing

××CSEn	Specifies Context Switching Processing
0	Processed by vectored interrupt
1	Processed by context switching

××PRn1	××PRn0	Specifies Priority of Interrupt Request
0	0	Priority 0 (highest priority)
0	1	Priority 1
1	0	Priority 2
1	1	Priority 3

Figure 14-1. Interrupt Control Registers (××ICn) (3/3)

Address : 0FFF6H-0FFF8H On reset : 43H R/W

	⑦	⑥	⑤	④	3	2	1	0
SRIC2	SRIF2	SRMK2	SRISM2	SRCSE2	0	0	SRPR21	SRPR20
CSIIC2	CSIIF2	CSIMK2	CSIISM2	CSICSE2	0	0	CSIPR21	CSIPR20
STIC2	STIF2	STMK2	STISM2	STCSE2	0	0	STPR21	STPR20
ADIC	ADIF	ADMK	ADISM	ADCSE	0	0	ADPR1	ADPR0

××IFn	Generation of Interrupt Request
0	No interrupt request (interrupt signal is not generated)
1	Interrupt request (interrupt signal is generated)

××MKn	Enables or Disables Interrupt Processing
0	Enables interrupt processing
1	Disables interrupt processing

××ISMn	Specifies Interrupt Processing Format
0	Vectored interrupt processing/context switching processing
1	Macro service processing

××CSEn	Specifies Context Switching Processing
0	Processed by vectored interrupt
1	Processed by context switching

××PRn1	××PRn0	Specifies Priority of Interrupt Request
0	0	Priority 0 (highest priority)
0	1	Priority 1
1	0	Priority 2
1	1	Priority 3

14.3.2 Interrupt mask registers (MK0, MK1)

The MK0 and MK1 are composed of interrupt mask flags. MK0 and MK1 are 16-bit registers which can be manipulated as a 16-bit unit. MK0 can be manipulated in 8 bit units using MK0L and MK0H, and similarly MK1 can be manipulated using MK1L and MK1H.

In addition, each bit of the MK0 and MK1L can be manipulated individually with a bit manipulation instruction. Each interrupt mask flag controls enabling/disabling of the corresponding interrupt request.

When an interrupt mask flag is set (1), acknowledgment of the corresponding interrupt request is disabled.

When an interrupt mask flag is cleared (0), the corresponding interrupt request can be acknowledged as a vectored interrupt or macro service request.

Each interrupt mask flag in the MK0 and MK1 is the same flag as the interrupt mask flag in the interrupt control register. The MK0 and MK1 are provided for en bloc control of interrupt masking.

After $\overline{\text{RESET}}$ input, the MK0 and MK1 are set to FFFFH, and all maskable interrupts are disabled.

Figure 14-2. Format of Interrupt Mask Registers (MK0, MK1)

<Byte access>

Address : 0FFACH-0FFAFH On reset : FFH R/W

	⑦	⑥	⑤	④	③	②	①	⓪
MK0L	PMK4	PMK3	PMK2	PMK1	PMK0	OVMK4	OVMK1	OVMK0
MK0H	1	1	1	1	CMMK11	CMMK10	PMK6	PMK5
MK1L	STMK2	SRMK2	SERMK2	STMK	SRMK	SERMK	CMMK41	CMMK40
MK1H	1	1	1	1	1	1	1	ADMK

××MKn	Enables or Disables Interrupt Request
0	Enables interrupt processing
1	Disables interrupt processing

<Word access>

Address : 0FFACH, 0FFAEH On reset : FFFFH R/W

	15	14	13	12	⑪	⑩	⑨	⑧
MK0	1	1	1	1	CMMK11	CMMK10	PMK6	PMK5
	⑦	⑥	⑤	④	③	②	①	⓪
	PMK4	PMK3	PMK2	PMK1	PMK0	OVMK4	OVMK1	OVMK0
	15	14	13	12	11	10	9	⑧
MK1	1	1	1	1	1	1	1	ADMK
	⑦	⑥	⑤	④	③	②	①	⓪
	STMK2	SRMK2	SERMK2	STMK	SRMK	SERMK	CMMK41	CMMK40

××MKn	Enables or Disables Interrupt Request
0	Enables interrupt processing
1	Disables interrupt processing

14.3.3 In-service priority register (ISPR)

The ISPR shows the priority level of the maskable interrupt currently being serviced and the non-maskable interrupt being processed. When a maskable interrupt request is acknowledged, the bit corresponding to the priority of that interrupt request is set (1), and remains set until the service program ends. When a non-maskable interrupt is acknowledged, the bit corresponding to the priority of that non-maskable interrupt is set (1), and remains set until the service program ends.

When an RETI instruction or RETCS instruction is executed, the bit, among those set (1) in the ISPR, that corresponds to the highest-priority interrupt request is automatically cleared (0) by hardware.

The contents of the ISPR are not changed by execution of an RETB or RETCSB instruction.

$\overline{\text{RESET}}$ input clears the ISPR register to 00H.

Figure 14-3. Format of In-Service Priority Register (ISPR)

Address : 0FFA8H On reset : 00H R

	7	6	5	4	3	2	1	0
ISPR	NMIS	WDTS	0	0	ISPR3	ISPR2	ISPR1	ISPR0

NMIS	NMI Processing Status
0	NMI interrupt is not accepted.
1	NMI interrupt is accepted

WDTS	Watchdog Timer Interrupt Processing Status
0	Watchdog timer interrupt is not accepted.
1	Watchdog timer interrupt is accepted.

ISPRn	Priority level (n = 0 to 3)
0	Interrupt of priority level n is not accepted.
1	Interrupt of priority level n is accepted.

Caution The in-service priority register (ISPR) is a read-only register. The microcontroller may malfunction if this register is written.

14.3.4 Interrupt mode control register (IMC)

The IMC contains the PRSL flag. The PRSL flag specifies enabling/disabling of nesting of maskable interrupts for which the lowest priority level (level 3) is specified.

When the IMC is manipulated, the interrupt disabled state (DI state) should be set first to prevent misoperation.

The IMC can be read or written to with an 8-bit manipulation instruction or bit manipulation instruction.

$\overline{\text{RESET}}$ input sets the IMC register to 80H.

Figure 14-4. Format of Interrupt Mode Control Register (IMC)

Address : 0FFAAH On reset : 80H R

	7	6	5	4	3	2	1	0
IMC	PRSL	0	0	0	0	0	0	0

PRSL	Controls Nesting 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.

14.3.5 Watchdog timer mode register (WDM)

The PRC bit of the WDM specifies the priority of NMI pin input non-maskable interrupts and watchdog timer overflow non-maskable interrupts.

The WDM can be written to only by a dedicated instruction. This dedicated instruction, MOV WDM, #byte, has a special code configuration (4 bytes), and a write is not performed unless the 3rd and 4th bytes of the operation code are mutual complements.

If the 3rd and 4th bytes of the operation code are not complements, a write is not performed and an operand error interrupt is generated. In this case, the return address saved in the stack area is the address of the instruction that was the source of the error, and thus the address that was the source of the error can be identified from the return address saved in the stack area.

If recovery from an operand error is simply performed by means of an RETB instruction, an endless loop will result.

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

Other write instructions (MOV WDM, A, AND WDM, #byte, SET1 WDM.7, etc.) are ignored and do not perform any operation. That is, a write is not performed to the WDM, and an interrupt such as an operand error interrupt is not generated.

The WDM can be read at any time by a data transfer instruction.

$\overline{\text{RESET}}$ input clears the WDM register to 00H.

Figure 14-5. Format of Watchdog Timer Mode Register (WDM)

Address : 0FFC2H On reset : 00H R/W

	7	6	5	4	3	2	1	0
WDM	RUN	0	0	PRC	0	WDI2	WDI1	0

RUN	Specifies Operation of Watchdog Timer (refer to **Figure 10-2**).

PRC	Priority of Watchdog Timer Interrupt Request
0	Watchdog timer interrupt request < NMI pin input interrupt request
1	Watchdog timer interrupt request > NMI pin input interrupt request

WDI2	WDI1	
		Specifies count clock of watchdog timer (refer to **Figure 10-2**).

Caution The watchdog timer mode register (WDM) can be written only by using a dedicated instruction (MOV WDM, #byte).

14.3.6 Program status word (PSW)

The PSW is a register that holds the current status regarding instruction execution results and interrupt requests. The IE flag that sets enabling/disabling of maskable interrupts is mapped in the low-order 8 bits of the PSW (PSWL).

PSWL can be read or written to with an 8-bit manipulation instruction, and can also be manipulated with a bit manipulation instruction or dedicated instruction (EI/DI).

When a vectored interrupt is acknowledged or a BRK instruction is executed, PSWL is saved to the stack and the IE flag is cleared (0). PSWL is also saved to the stack by the PUSH PSW instruction, and is restored from the stack by the RETI, RETB and POP PSW instructions.

When context switching or a BRKCS instruction is executed, PSWL is saved to a fixed area in the register bank, and the IE flag is cleared (0). PSWL is restored from the fixed area in the register bank by an RETCSI or RETCSB instruction.

$\overline{\text{RESET}}$ input clears PSWL to 00H.

Figure 14-6. Format of Program Status Word (PSWL)

On reset : 00H

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

S	Used when executing a normal operation
Z	
RSS	
AC	

IE	Enables or Disables Interrupt Accepting
0	Disables interrupt accepting
1	Enables interrupt accepting

P/V	Used when executing a normal operation
CY	

14.4 Software Interrupt Acknowledgment Operations

A software interrupt is acknowledged in response to execution of a BRK or BRKCS instruction. Software interrupts cannot be disabled.

14.4.1 BRK instruction software interrupt acknowledgment operation

When a BRK instruction is executed, the program status word (PSW), program counter (PC) are saved in that order to the stack, the IE flag is cleared (0), the vector table (003EH/003FH) contents are loaded into the low-order 16 bits of the PC, and 0000B into the high-order 4 bits, and a branch is performed (the start of the service program must be in the base area).

The RETB instruction must be used to return from a BRK instruction software interrupt.

Caution The RETI instruction must not be used to return from a BRK instruction software interrupt.

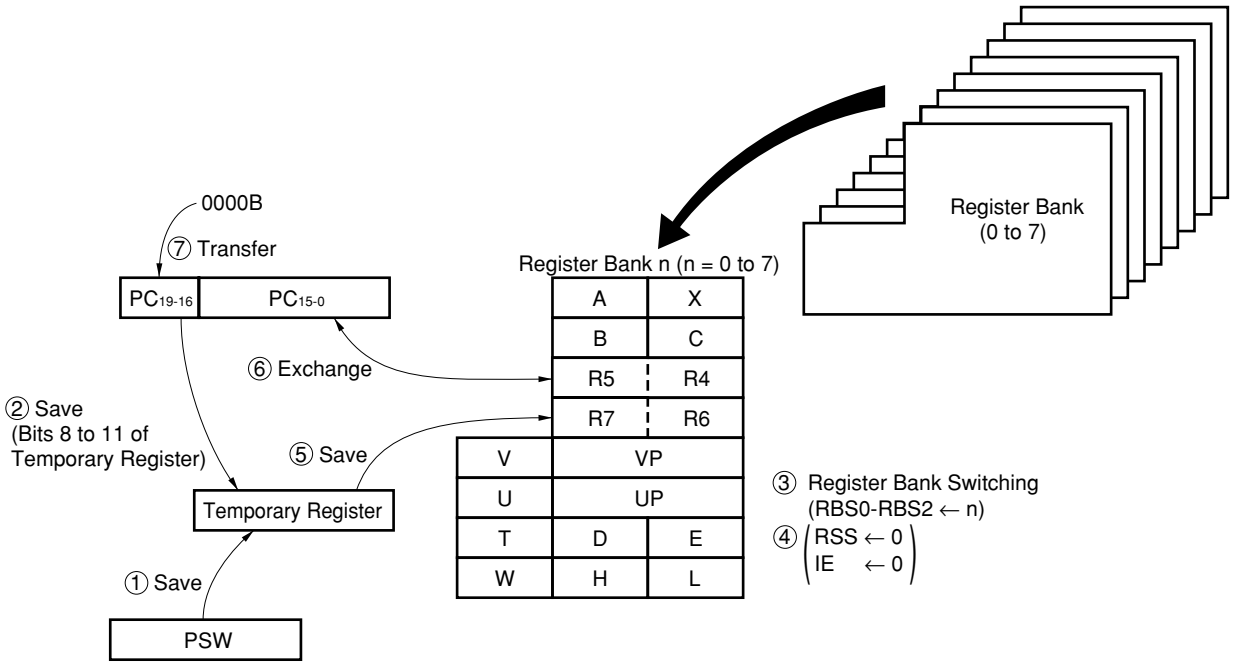
14.4.2 BRKCS instruction software interrupt (software context switching) acknowledgment operation

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.

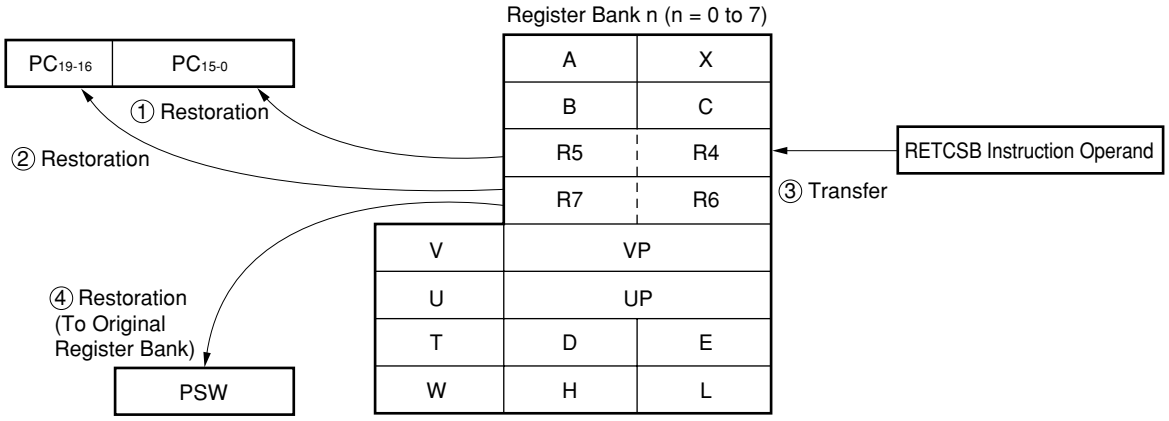
Figure 14-7. Context Switching Operation by Execution of a BRKCS Instruction



The RETCSB instruction is used to return from a software interrupt due to a BRKCS instruction. The RETCSB instruction must specify the start address of the interrupt service program for the next time context switching is performed by a BRKCS instruction. This interrupt service program start address must be in the base area.

Caution The RETCS instruction must not be used to return from a BRKCS instruction software interrupt.

Figure 14-8. Return from BRKCS Instruction Software Interrupt (RETCSB Instruction Operation)



14.5 Operand Error Interrupt Acknowledgment Operation

An operand error interrupt is generated when the data obtained by inverting all the bits of the 3rd byte of the operand of an MOV STBC, #byte instruction or LOCATION instruction or an MOV WDM,#byte instruction does not match the 4th byte of the operand. Operand error interrupts cannot be disabled.

When an operand error interrupt is generated, the program status word (PSW) and the start address of the instruction that caused the error are saved to the stack, the IE flag is cleared (0), the vector table value is loaded into the program counter (PC), and a branch is performed (within the base area only).

As the address saved to the stack is the start address of the instruction in which the error occurred, simply writing an RETB instruction at the end of the operand error interrupt service program will result in generation of another operand error interrupt. You should therefore either process the address in the stack or initialize the program by referring to **14.12 Restoring Interrupt Function To Initial State**.

14.6 Non-Maskable Interrupt Acknowledgment Operation

Non-maskable interrupts are acknowledged even in the interrupt disabled state. Non-maskable interrupts can be acknowledged at all times except during execution of the service program for an identical non-maskable interrupt or a non-maskable interrupt of higher priority.

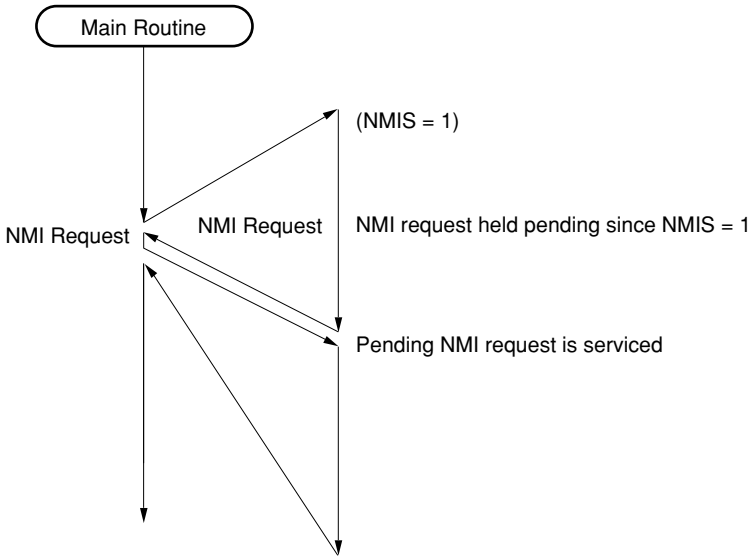
The relative priorities of non-maskable interrupts are set by the PRC bit of the watchdog timer mode register (WDM) (refer to **14.3.5 Watchdog timer mode register (WDM)**).

Except in the cases described in **14.9 When Interrupt Request and Macro Service Are Temporarily Held Pending**, a non-maskable interrupt request is acknowledged immediately. When a non-maskable interrupt request is acknowledged, the program status word (PSW) and program counter (PC) are saved in that order to the stack, the IE flag is cleared (0), the in-service priority register (ISPR) bit corresponding to the acknowledged non-maskable interrupt is set (1), the vector table contents are loaded into the PC, and a branch is performed. The ISPR bit that is set (1) is the NMIS bit in the case of a non-maskable interrupt due to edge input to the NMI pin, and the WDTS bit in the case of watchdog timer overflow.

When the non-maskable interrupt service program is executed, non-maskable interrupt requests of the same priority as the non-maskable interrupt currently being executed and non-maskable interrupts of lower priority than the non-maskable interrupt currently being executed are held pending. A pending non-maskable interrupt is acknowledge after completion of the non-maskable interrupt service program currently being executed (after execution of the RETI instruction). However, even if the same non-maskable interrupt request is generated more than once during execution of the non-maskable interrupt service program, only one non-maskable interrupt is acknowledged after completion of the non-maskable interrupt service program.

Figure 14-9. Operations of Non-Maskable Interrupt Request Acknowledgment (1/2)

(a) When a new NMI request is generated during NMI service program execution



(b) When a watchdog timer interrupt request is generated during NMI service program execution (when the watchdog timer interrupt priority is higher (when PRC in the WDM = 1))

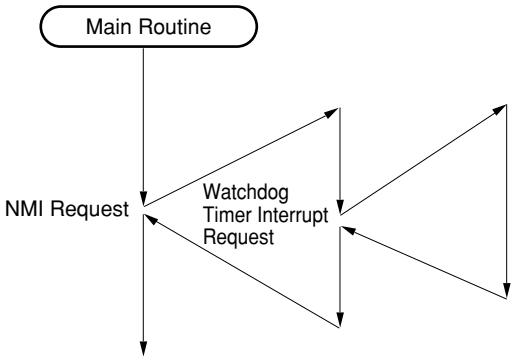
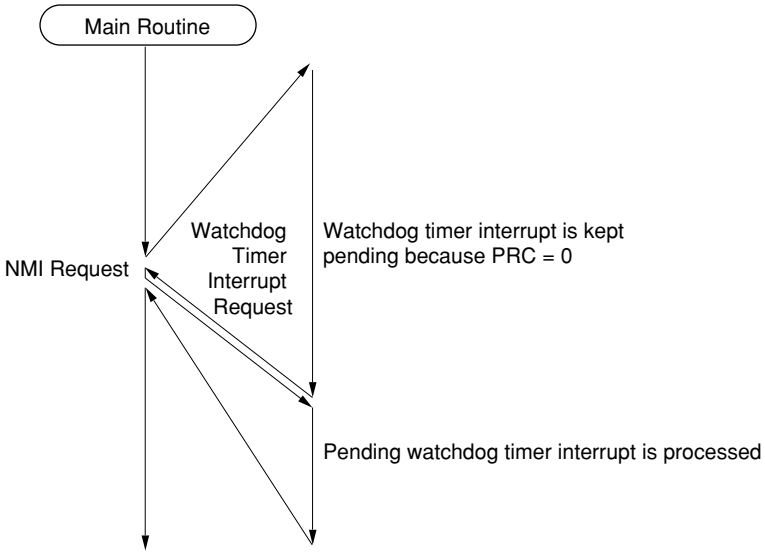
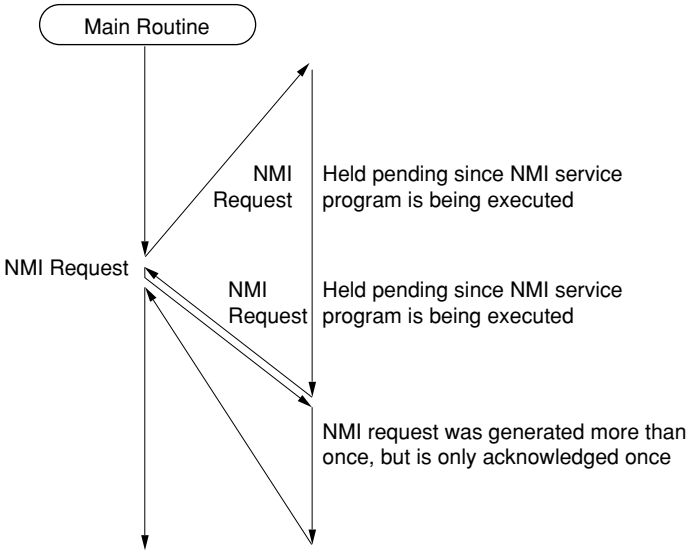


Figure 14-9. Operations of Non-Maskable Interrupt Request Acknowledgment (2/2)

(c) When a watchdog timer interrupt request is generated during NMI service program execution (when the NMI interrupt priority is higher (when PRC in the WDM = 0))



(d) When an NMI request is generated twice during NMI service program execution



Cautions 1. Macro service requests are acknowledged and 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.

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 14.9. 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 processing program is not performed normally and a software upsets occurs. Therefore, the program following $\overline{\text{RESET}}$ release must be as shown below.

```
        CSEG AT 0
        DW   STRT
        CSEG BASE
STRT:
        LOCATION 0FH; or LOCATION 0H
        MOVG SP, #imm24
```

14.7 Maskable Interrupt Acknowledgment Operation

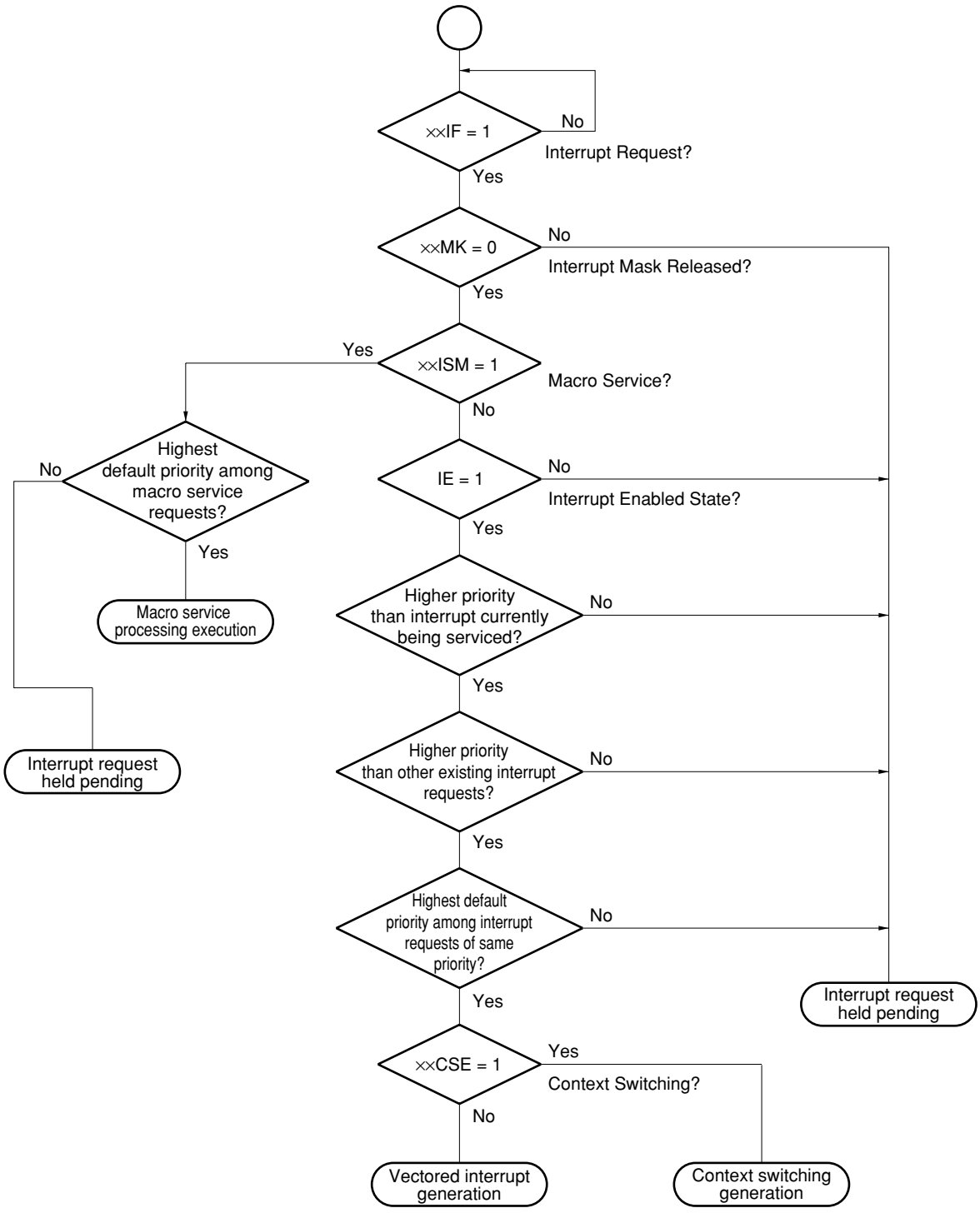
A maskable interrupt can be acknowledged when the interrupt request flag is set (1) and the mask flag for that interrupt is cleared (0). When processing is performed by macro service, the interrupt is acknowledged and processed by macro service immediately. In the case of vectored interruption and context switching, an interrupt is acknowledged in the interrupt enabled state (when the IE flag is set (1)) if the priority of that interrupt is one for which acknowledgment is permitted.

If maskable interrupt requests are generated simultaneously, the interrupt for which the highest priority is specified by the priority specification flag is acknowledged. If the interrupts have the same priority specified, they are acknowledged in accordance with their default priorities.

A pending interrupt is acknowledged when a state in which it can be acknowledged is established.

The interrupt acknowledgment algorithm is shown in Figure 14-10.

Figure 14-10. Algorithm of Interrupt Acknowledgment Processing



14.7.1 Vectored interrupt

When a vectored interrupt maskable interrupt request is acknowledged, the program status word (PSW) and program counter (PC) are saved in that order to the stack, the IE flag is cleared (0) (the interrupt disabled state is set), and the in-service priority register (ISPR) bit corresponding to the priority of the acknowledged interrupt is set (1). Also, data in the vector table predetermined for each interrupt request is loaded into the PC, and a branch is performed. The return from a vectored interrupt is performed by means of the RETI instruction.

Caution When a maskable interrupt is acknowledged by vectored interrupt, the RETI instruction must be used to return from the interrupt. Subsequent interrupt acknowledgment will not be performed normally if a different instruction is used.

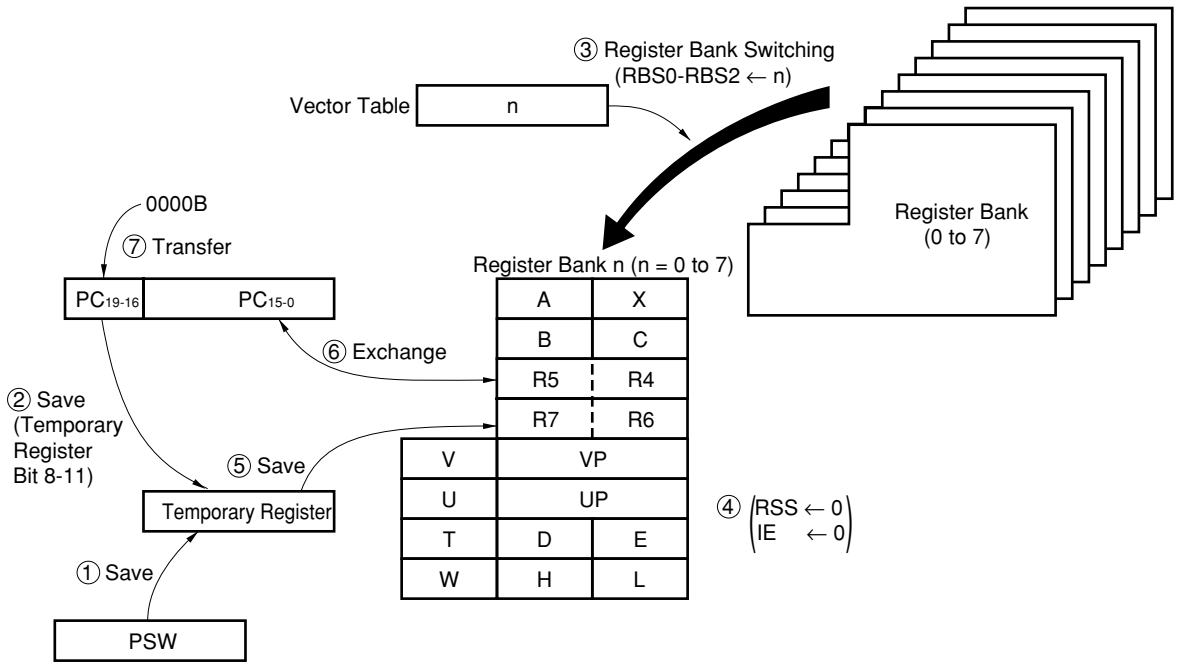
14.7.2 Context switching

Initiation of the context switching function is enabled by setting (1) the context switching enable flag of the interrupt control register.

When an interrupt request for which the context switching function is enabled is acknowledged, the register bank specified by 3 bits of the 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 a branch is made to the interrupt service program.

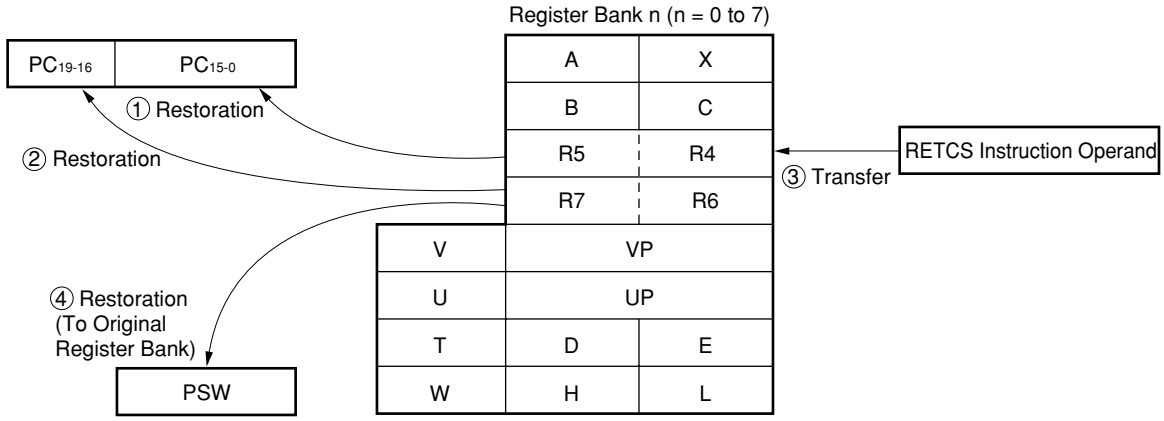
Figure 14-11. Context Switching Operation by Generation of an Interrupt Request



The RETCS instruction is used to return from an interrupt that uses the context switching function. The RETCS instruction must specify the start address of the interrupt service program to be executed when that interrupt is acknowledged next. This interrupt service program start address must be in the base area.

Caution The RETCS instruction must be used to return from an interrupt serviced by context switching. Subsequent interrupt acknowledgment will not be performed normally if a different instruction is used.

Figure 14-12. Return from Interrupt that Uses Context Switching by Means of RETCS Instruction



14.7.3 Maskable interrupt priority levels

The μPD784054 performs multiple interrupt processing in which an interrupt is acknowledged during processing 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 14-2**). 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 interruption is permitted are shown in Table 14-5.

Since the IE flag is cleared (0) automatically when an interrupt is acknowledged, when multiple interruption is used, the IE flag should be set (1) to enable interrupts by executing an EI instruction in the interrupt processing program, etc.

Table 14-5. Multiple Interrupt Processing

Priority of Interrupt Currently Being Acknowledged	ISPR Value	IE Flag in PSW	PRSL in IMC Register	Acknowledgeable Maskable Interrupts
No interrupt being acknowledged	00000000	0	×	• All macro service only
		1	×	• All maskable interrupts
3	00001000	0	×	• All macro service only
		1	0	• All maskable interrupts
		1	1	• All macro service • Maskable interrupts specified as priority 0/1/2
2	0000×100	0	×	• All macro service only
		1	×	• All macro service • Maskable interrupts specified as priority 0/1
1	0000××10	0	×	• All macro service only
		1	×	• All macro service • Maskable interrupts specified as priority 0
0	0000×××1	×	×	• All macro service only
Non-maskable interrupts	1000×××× 0100×××× 1100××××	×	×	• All macro service only

Remark ×: don't care

Figure 14-13. Examples of Processing When Another Interrupt Request Is Generated During Interrupt Processing (1/3)

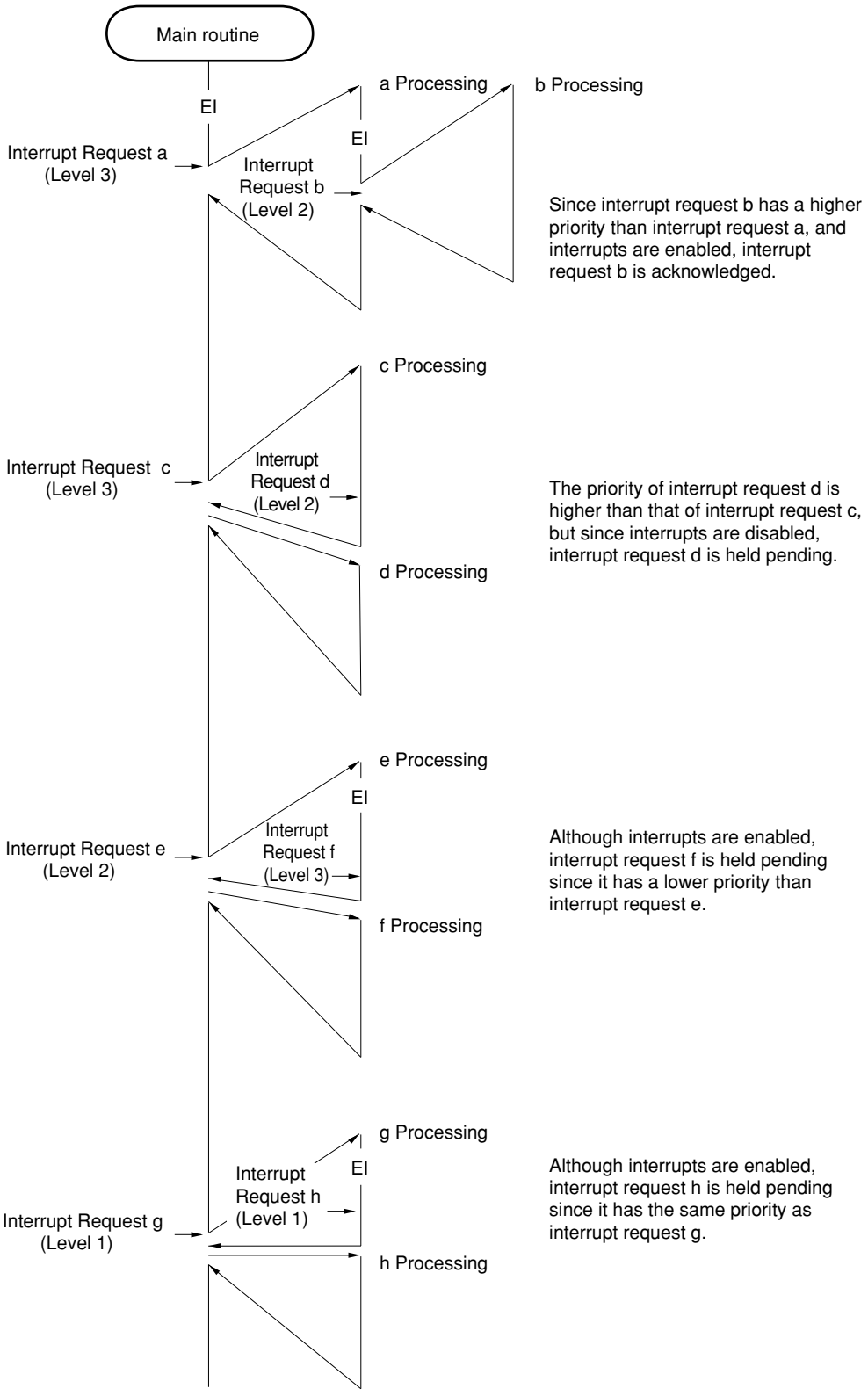
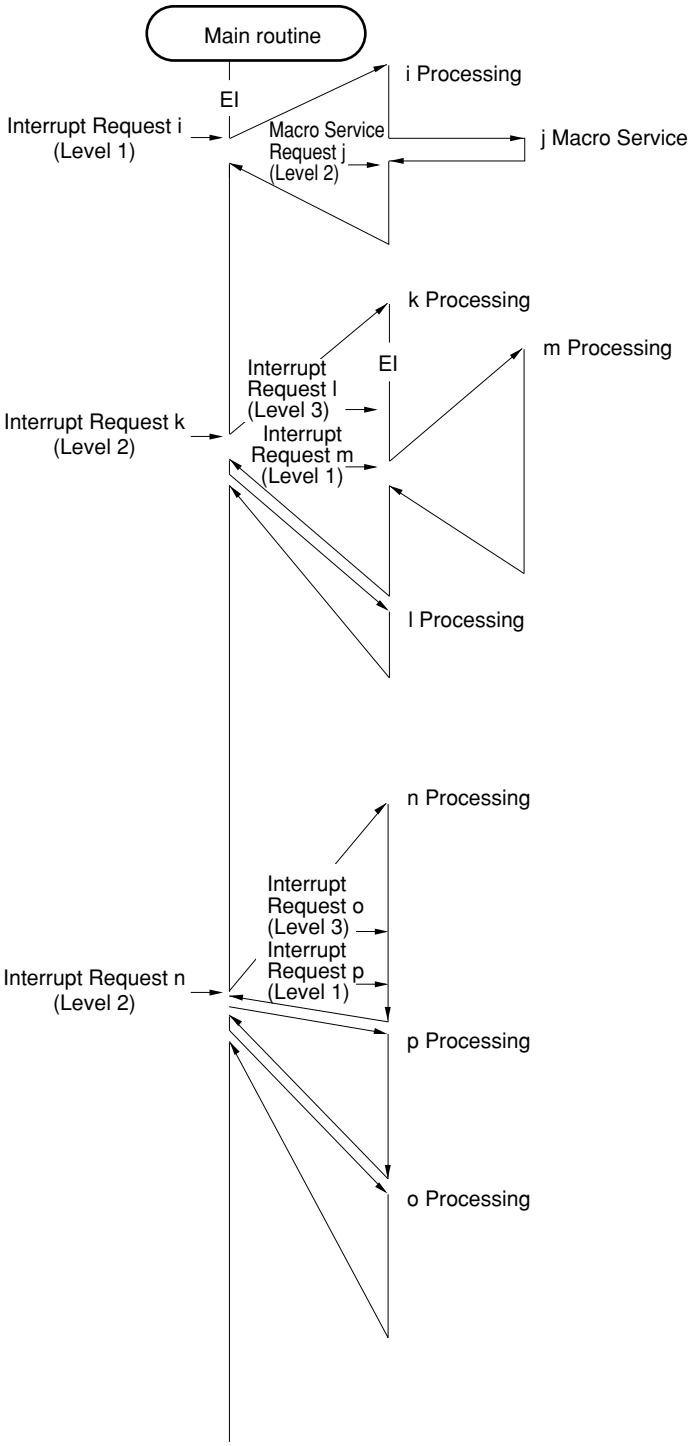


Figure 14-13. Examples of Processing When Another Interrupt Request Is Generated During Interrupt Processing (2/3)

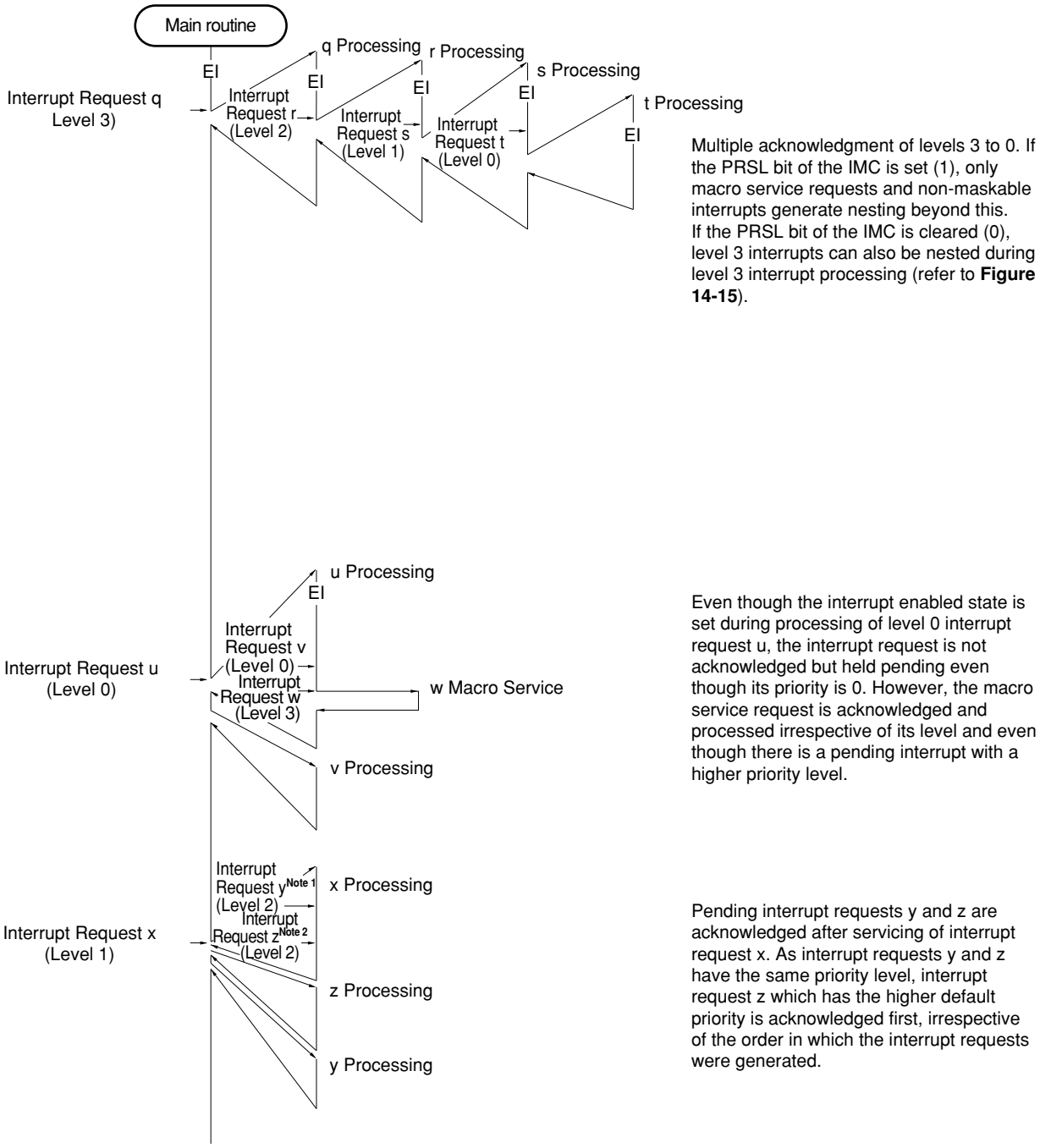


The macro service request is serviced irrespective of interrupt enabling/disabling and priority.

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 processing of interrupt request n performed in the interrupt disabled state, interrupt requests o and p are held pending.
After interrupt request n processing, the pending interrupt requests are acknowledged. Although interrupt request o was generated first, interrupt request p has a higher priority and is therefore acknowledged first.

Figure 14-13. Examples of Processing When Another Interrupt Request Is Generated During Interrupt Processing (3/3)



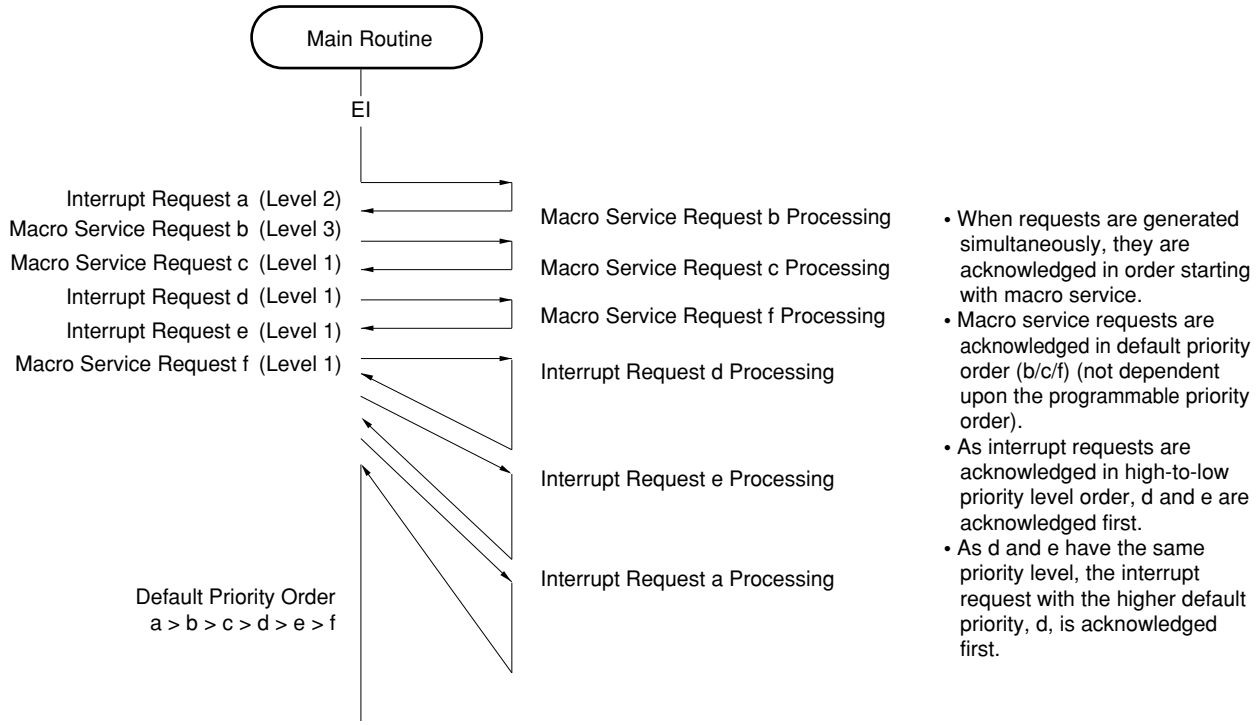
Notes 1. Low default priority

2. High default priority

Remarks 1. “a” to “z” in the figure are arbitrary names used to differentiate between the interrupt requests and macro service requests.

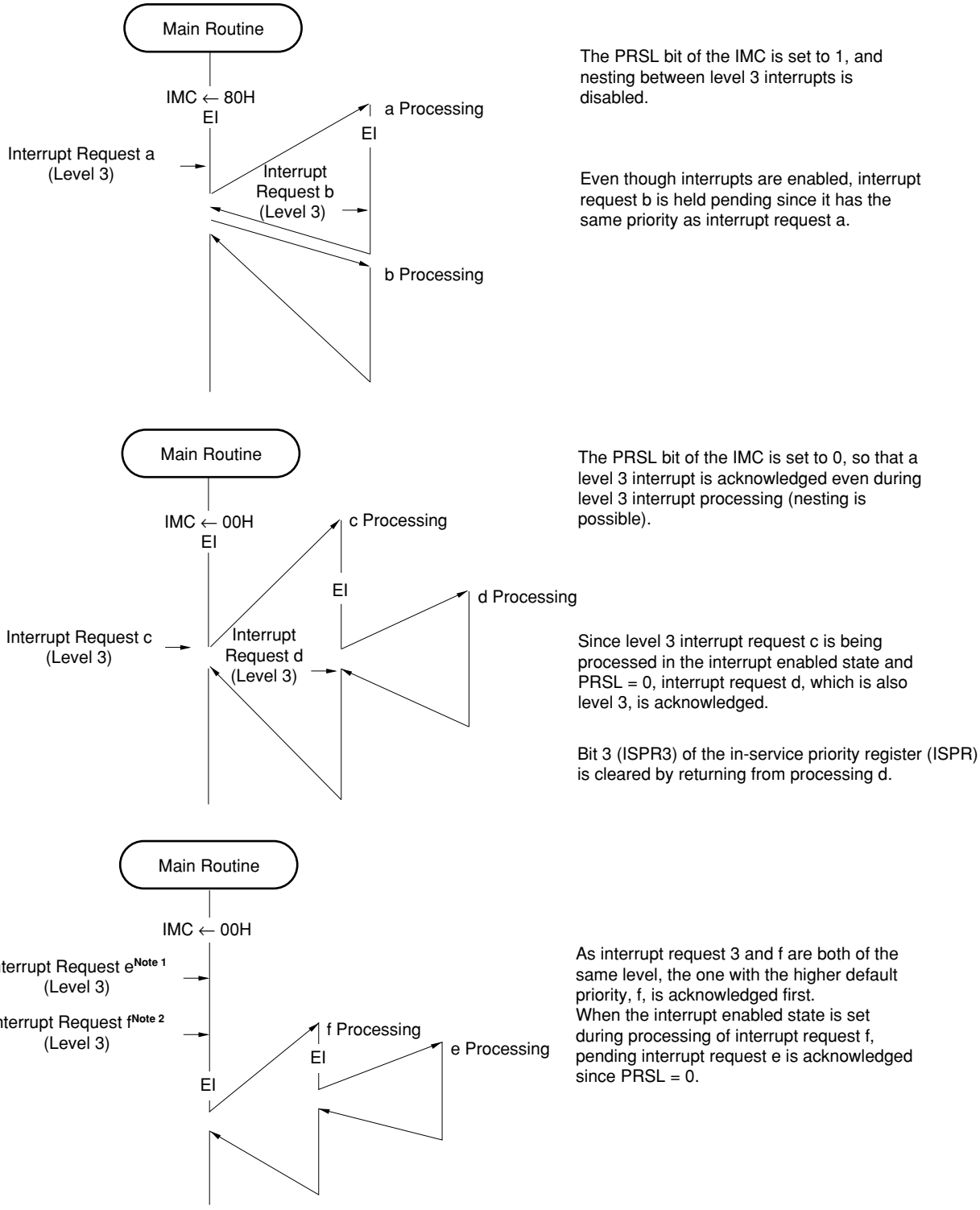
2. High/low default priorities in the figure indicate the relative priority levels of the two interrupt requests.

Figure 14-14. Examples of Processing of Simultaneously Generated Interrupts



Remark "a" to "f" in the figure are arbitrary names used to differentiate between the interrupt requests and macro service requests.

Figure 14-15. Differences in Level 3 Interrupt Acknowledgment According to Setting of Interrupt Mode Control Register (IMC)



Notes 1. Low default priority

2. High default priority

Remarks 1. “a” to “f” in the figure are arbitrary names used to differentiate between the interrupt requests and macro service requests.

2. High/low default priorities in the figure indicate the relative priority levels of the two interrupt requests.

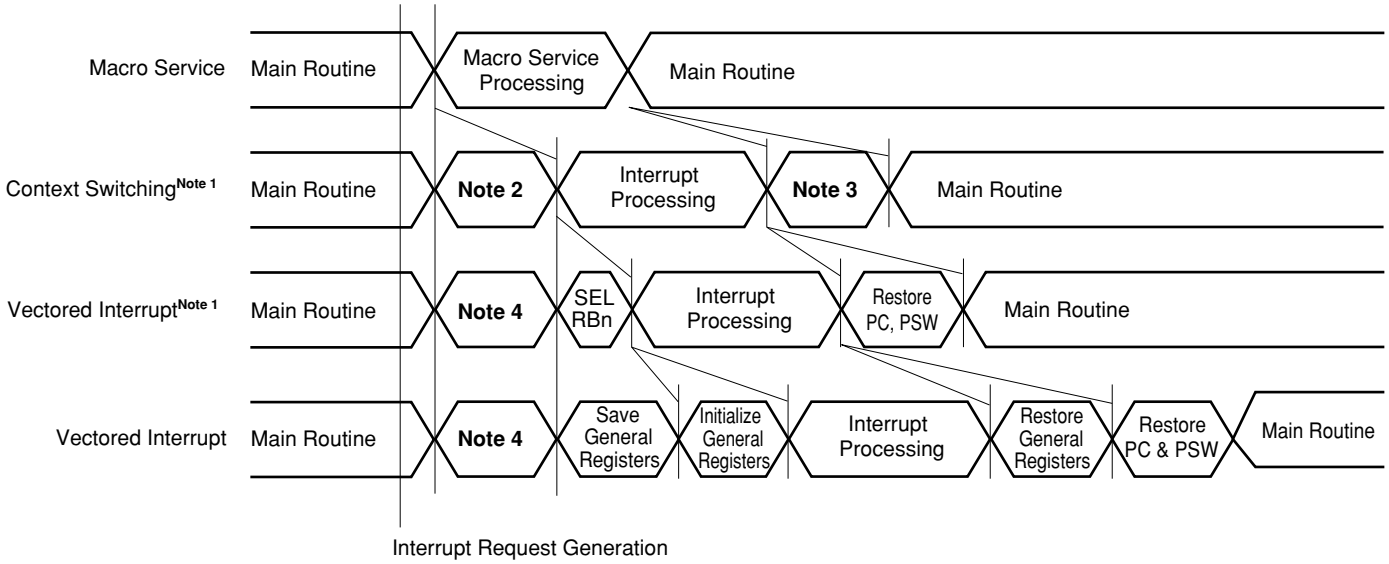
14.8 Macro Service Function

14.8.1 Outline of macro service function

Macro service is one method of processing interrupts. With a normal interrupt, the program counter (PC) and program status word (PSW) are saved, and the start address of the interrupt service program is loaded into the PC, but with macro service, different processing (mainly data transfers) is performed instead of this processing. This enables interrupt requests to be responded to quickly, and moreover, since transfer processing is faster than processing by a program, the processing time can also be reduced.

Also, since a vectored interrupt is generated after processing has been performed the specified number of times, another advantage is that vectored interrupt programs can be simplified.

Figure 14-16. 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

14.8.2 Types of macro service

Macro service can be used with the 21 kinds of interrupt shown in Table 14-6 (18 of which can be used simultaneously). There are seven kinds of operation mode, which can be used to suit the application.

Table 14-6. Interrupts for Which Macro Service Can Be Used

Default Priority	Interrupt Request Generation Source	Generating Unit	Macro Service Control Word Address
0 (highest)	INTOV0 (overflow of timer 0)	Timer 0	0FE06H
1	INTOV1 (overflow of timer 1)	Timer 1	0FE08H
2	INTOV4 (overflow of timer 4)	Timer 4	0FE0AH
3	INTP0 (pin input edge detection)	Edge detection	0FE0CH
	INTCC00 (TM0-CC00 match signal generation)	Timer 0	
4	INTP1 (pin input edge detection)	Edge detection	0FE0EH
	INTCC01 (TM0-CC01 match signal generation)	Timer 0	
5	INTP2 (pin input edge detection)	Edge detection	0FE10H
	INTCC002 (TM0-CC02 match signal generation)	Timer 0	
6	INTP3 (pin input edge detection)	Edge detection	0FE12H
	INTCC03 (TM0-CC03 match signal generation)	Timer 0	
7	INTP4 (pin input edge detection)	Edge detection	0FE14H
8	INTP5 (pin input edge detection)	Edge detection	0FE16H
9	INTP6 (pin input edge detection)	Edge detection	0FE18H
10	INTCM10 (TM1-CM10 match signal generation)	Timer 1	0FE1AH
11	INTCM11 (TM1-CM11 match signal generation)	Timer 1	0FE1CH
12	INTCM40 (TM4-CM40 match signal generation)	Timer 4	0FE26H
13	INTCM41 (TM4-CM41 match signal generation)	Timer 4	0FE28H
14	INTSER (UART0 reception error)	Asynchronous serial interface 0	0FE2AH
15	INTSR (UART0 reception end)		0FE2CH
	INTCSI1 (3-wire serial I/O1 transfer end)	3-wire serial I/O1	
16	INTST (UART0 transmission end)	Asynchronous serial interface 0	0FE2EH
17	INTSER2 (UART2 reception error)	Asynchronous serial interface 2	0FE30H
18	INTSR2 (UART2 reception end)		0FE32H
	INTCSI2 (3-wire serial I/O2 transfer end)	3-wire serial I/O2	
19	INTST2 (UART2 transmission end)	Asynchronous serial interface 2	0FE34H
20 (lowest)	INTAD (A/D conversion end)	A/D converter	0FE36H

Remarks 1. The default priority is a fixed number. This indicates the order of priority when macro service requests are generated simultaneously,

2. The INTSR and INTCSI1 interrupts are generated by the same hardware (they cannot both be used simultaneously). Therefore, although the same hardware is used for the interrupts, two names are provided, for use in each of the two modes. The same applies to INTSR2 and INTCSI2.

The macro service operation is performed in the following seven modes:

(1) Counter mode: EVTCNT

In this mode, each time an interrupt request has been generated, the macro service counter (MSC) is incremented (+1) or decremented (–1). When MSC reaches 00H, a vectored interrupt request is generated.
This mode is used to divide the number of times an interrupt request is generated.

(2) Block transfer mode: BLKTRS

Each time an interrupt request has been generated, 1-byte or 1-word data is transferred between a special function register (SFR) pointed to by the SFR pointer (SFR.PTR) and buffer. When data has been transferred the specified number of times, a vectored interrupt request is generated.
The buffer with which data is to be transferred is limited to the addresses 0FD00H to 0FEFFH[Note] of the main RAM.
This mode is easy to specify and is used for high-speed transfer of a small amount of data.

Note When the LOCATION 0H instruction is executed. FFD00H to FFEFFH when the LOCATION 0FH instruction is executed.

(3) Block transfer mode (with memory pointer): BLKTRS-P

Like the block transfer mode, 1-byte or 1-word data is transferred between an SFR specified by SFR.PTR and buffer each time an interrupt request has been generated, and a vectored interrupt request is generated when data has been transferred the specified number of times.
The buffer with which data is to be transferred is specified by the memory pointer (MEM.PTR) (data can be transferred with the entire 1M-byte memory).
This mode is a general-purpose type of the block transfer mode and is used to transfer a large quantity of data.

(4) Data differential mode: DTADIF

Each time an interrupt request has been generated, the difference between the current value of an SFR specified by SFR.PRT and the “value immediately before” stored in memory is written to the buffer, and the current value is used as the “value immediately before”.
When data has been transferred the specified number of times, a vectored interrupt request is generated.
The buffer with which data is to be transferred is limited to the main RAM of the addresses 0FD00H to 0FEFFH[Note].
This mode is used to measure the cycle of an input pulse, or width of a pulse by using a capture register.

Note When the LOCATION 0H instruction is executed. FFD00H to FFEFFH when the LOCATION 0FH instruction is executed.

(5) Data differential mode (with memory pointer): DTADIF-P

Like the data differential mode, each time an interrupt request has been generated, the difference between the current value of an SFR specified by SFR.PTR and the “value immediately before” stored in memory is written to the buffer, and the current value is used as the “value immediately before”.
When data has been transferred the specified number of times, a vectored interrupt request is generated.
The buffer with which data is to be transferred is specified by the memory pointer (MEM.PTR) (the entire 1M-byte memory can be specified).
This mode is a general-purpose type of the data differential mode, and is used to transfer a large quantity of data.

(6) CPU monitor mode 0: SELF0

Each time an interrupt request has been generated, the internal operation of the CPU is checked. If each block operates normally, a value resulting from subtracting 10 from the initial data is transferred to an SFR specified by SFR.PTR.

This mode is used for self-check of the CPU at initialization.

(7) CPU monitor mode 1: SELF1

Each time an interrupt request has been generated, the internal operation of the CPU is checked. If each block operates normally, a value resulting from subtracting 8 from the initial data is transferred to an SFR specified by SFR.PTR.

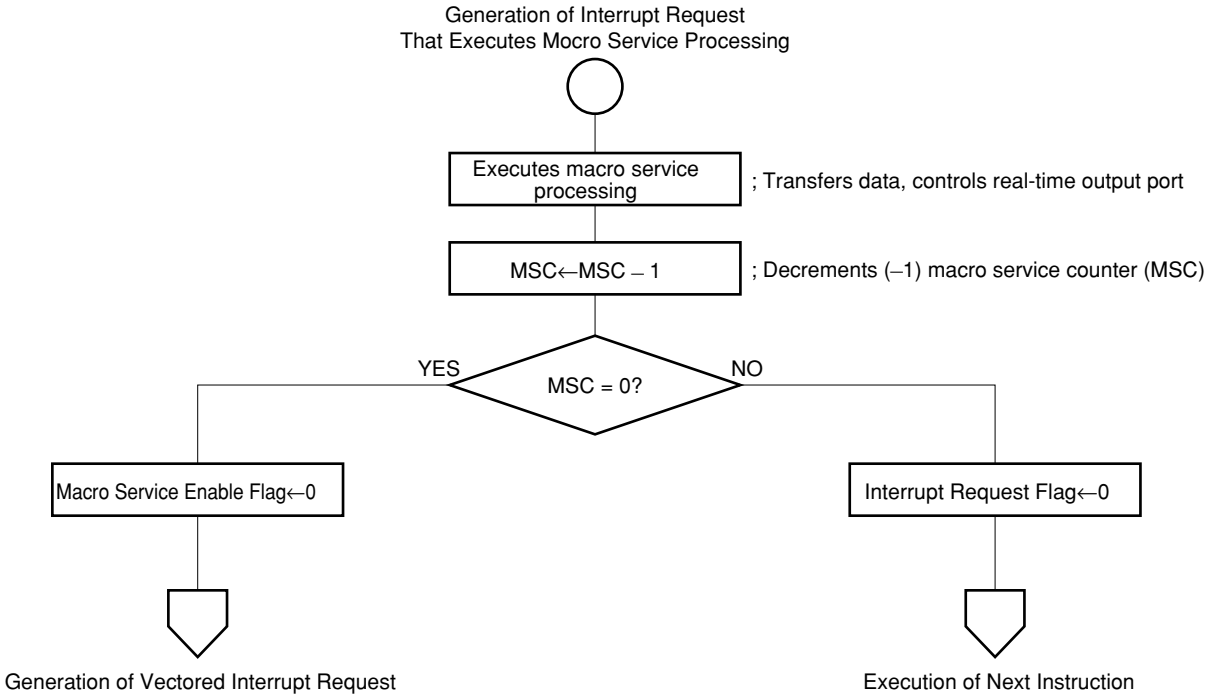
This mode is used for self-check of the CPU during normal operation.

14.8.3 Basic operation of macro service (except CPU monitor modes 0 and 1)

The macro service function is to transfer data between the special function register area and memory space by hardware, using an interrupt request.

When a macro service request is generated, the CPU temporarily stops program execution, and automatically transfers 1/2-byte data between a special function register (SFR) and memory. When data transfer has been completed, an interrupt request flag is reset (0), and the CPU starts program execution again. Data is transferred the number of times set to the macro service counter (MSC) and then a vectored interrupt request is generated.

Figure 14-17. Example of Macro Service Processing Sequence



Unlike other interrupt processing, processing using the macro service function is automatically performed without starting an interrupt processing program. Therefore, operations such as branching to an interrupt service routine, saving/restoring registers, and returning from the interrupt service routine are not performed. This means that the service time of the CPU can be improved and that the number of program steps can be decreased.

When macro service processing is executed, the status before execution of the macro service processing, such as the contents of the general-purpose registers and instruction queue of the CPU, are retained.

The interrupt request that specifies the macro service processing is not affected by the status of the IE flag in the program status word (PSWL). The macro service processing can be executed even in the interrupt disabled status or while an interrupt processing program is executed. It is disabled only when the corresponding bit in the interrupt mask registers (MK0, MK1) is set (1).

If two or more macro service requests are issued at the same time, the sequence in which the macro service requests are processed is determined by the default priority. Until all the macro service requests are processed, instructions are not executed.

The μPD784054 supports macro service processing for all the internal interrupt requests.

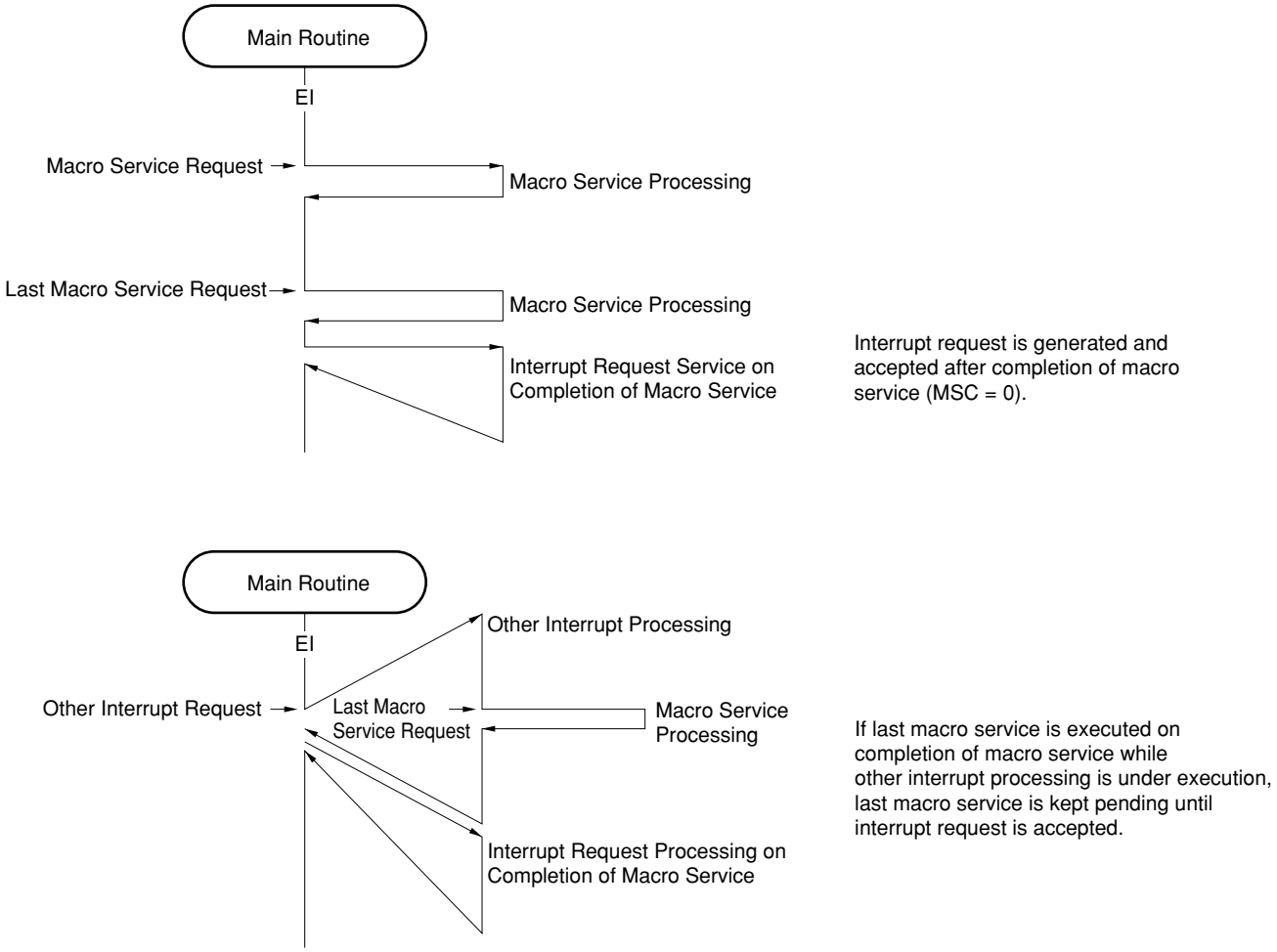
Basically, the macro service processing executes the following two operations:

- Data transfer from memory to special function register (SFR)
- Data transfer from special function register (SFR) to memory

14.8.4 Operation on completion of macro servicing (except CPU monitor modes 0 and 1)

The macro service performs processing the number of times specified during other program is executed. When the processing has been performed the specified number of times (when the macro service counter (MSC) has reached 0), the macro service is completed.

Figure 14-18. Operation on Completion of Macro Service



Caution If data is transmitted with UART by using the macro service, a vectored interrupt request is generated two times (refer to 12.2.8 Transmitting/receiving data with macro service).

14.8.5 Macro service control register

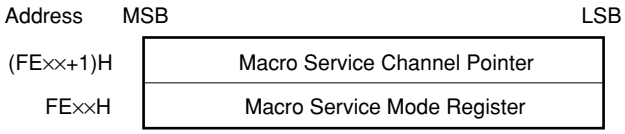
(1) Macro service control word

The macro service control word consists of a macro service mode register that controls the macro service function, and a macro service channel pointer. It is located in the address space from 0FE06H to 0FE37H[Note] in the main RAM area (refer to **Figure 14-20**).

Figure 14-19 shows the basic configuration of the macro service control word.

Note When the LOCATION 0H instruction is executed. FFE06H to FFE37H when the LOCATION 0FH instruction is executed.

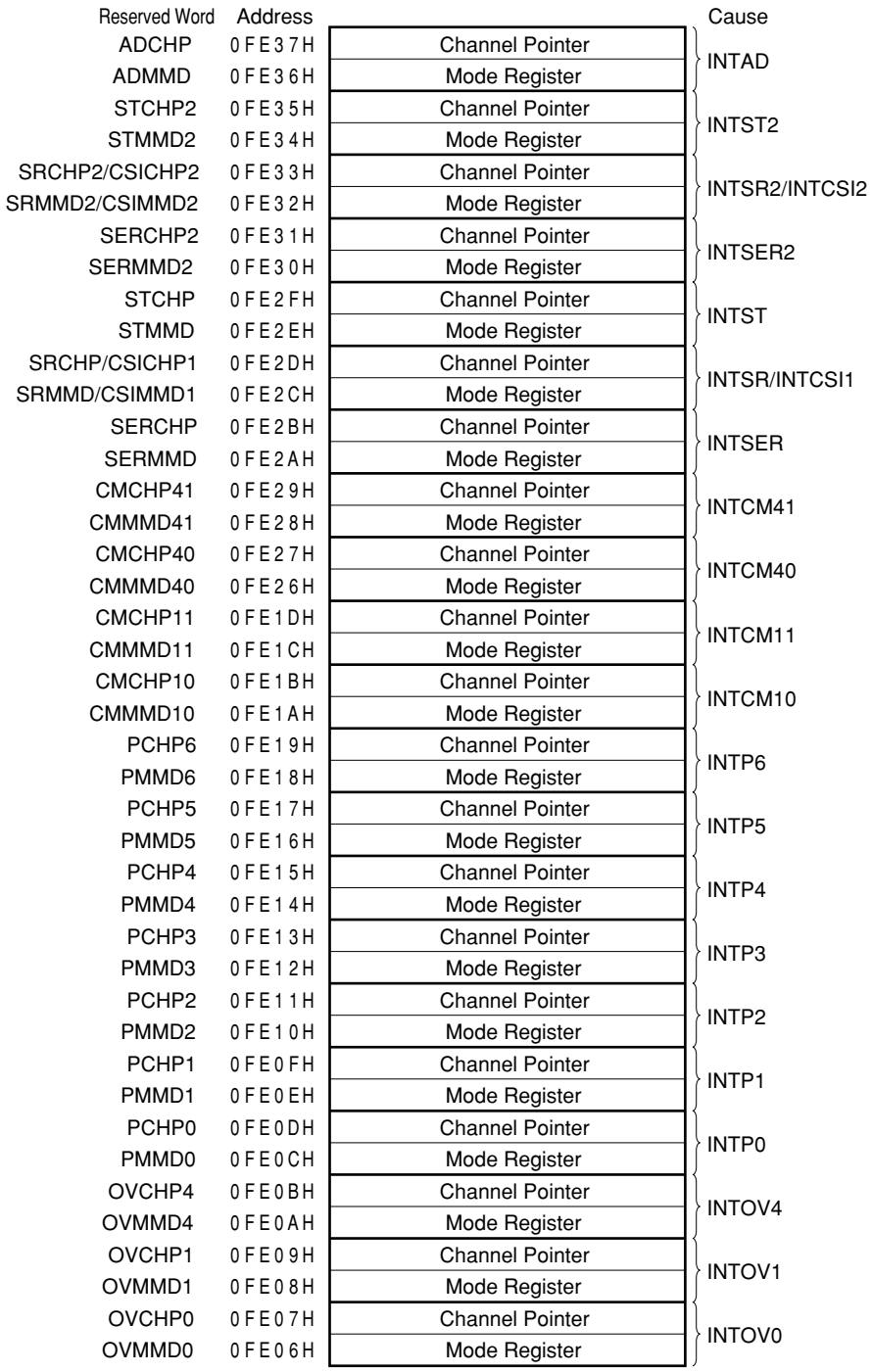
Figure 14-19. Basic Configuration of Macro Service Control Word



The macro service mode register sets a macro service processing mode, and the macro service channel pointer specifies the address of the macro service channel.

To perform macro service processing, a value must be set in advance to the macro service mode register and channel pointer corresponding to the interrupt request that can specify macro service processing.

Figure 14-20. Format of Macro Service Control Word



(2) Macro service mode register

The macro service mode register is an 8-bit register that specifies the operation of the macro service. This register is mapped to the main RAM area as a part of the macro service control word (refer to **Figure 14-19**)

14.8.6 Macro service mode

The operation of the macro service is specified by using the macro service mode register. The macro service mode is specified by the low-order 6 bits of the macro service mode register, and is divided into groups 0 to 2.

- Group 0 ... Type with only control word and without channel
- Group 1 ... Type with both control word and channel
- Group 2 ... Macro service for monitoring CPU

The high-order 2 bits of the macro service mode register of groups 0 and 1 function as a subcommand (refer to **Table 14-7**).

14.8.7 Operation of macro service

The operation of the macro service is performed in the following seven modes:

Table 14-7. Classification of Macro Service Mode

Group	Macro Service Mode Register	Function	
Group 0	CC000001	Counter mode	EVTCNT
Group 1	CC010011	Block transfer mode	BLKTRS
	CC010100	Block transfer mode (with memory pointer)	BLKTRS-P
	10011001	Data differential mode	DTADIF
	10011010	Data differential mode (with memory pointer)	DTADIF-P
Group 2	10101011	CPU monitor mode 0	SELF0
	10001011	CPU monitor mode 1	SELF1

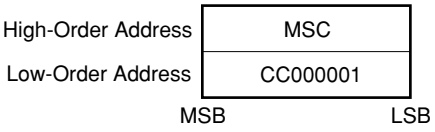
The most significant bit (MSB) C of the macro service mode registers for BLKTRS and BLKTRS-P indicates the length of the data to be handled.

- When C = 0: byte data
- When C = 1: word data

BLKTRS and BLKTRS-P are expressed here in terms of byte buffers. When word data is specified, read byte buffer as word buffer.

(1) Counter mode: EVCNT

[Macro service control word]



[Operation]

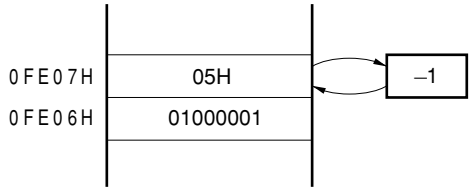
Increments (+1) or decrements (–1) the macro service counter (MSC) each time a macro service has been generated. When the value of MSC has reached 00H (overflow), a vectored interrupt request is generated.

Table 14-8. Specifying Operation of Counter Mode

CC	Operation
00	Increment
01	Decrement
10	Setting prohibited
11	

In this mode, the macro service function serves as a counter that divides the number of times the interrupt request is generated.

Example To divide the number of times the INTOV0 interrupt request has been generated by five by using the macro service

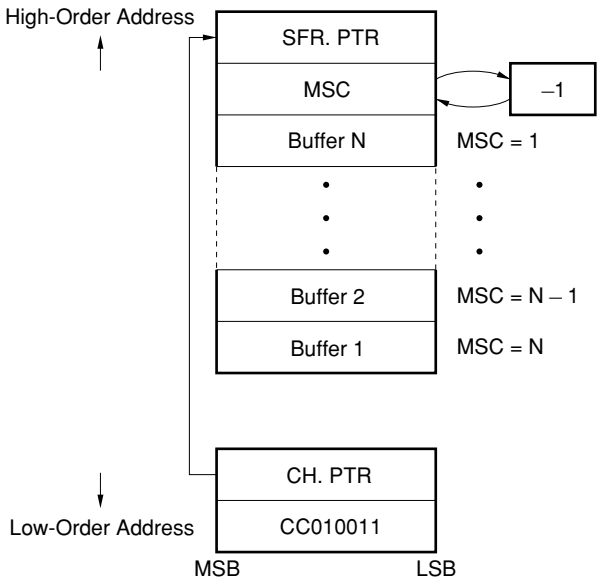


[Usage]

Event counter, measurement of number of times of capture

(2) Block transfer mode: BLKTRS

[Macro service control word]



[Operation]

Specifies an SFR pointer (SFR.PTR) by using a channel pointer (CH.PTR). Addresses a buffer by using CH.PTR and the macro service counter (MSC).

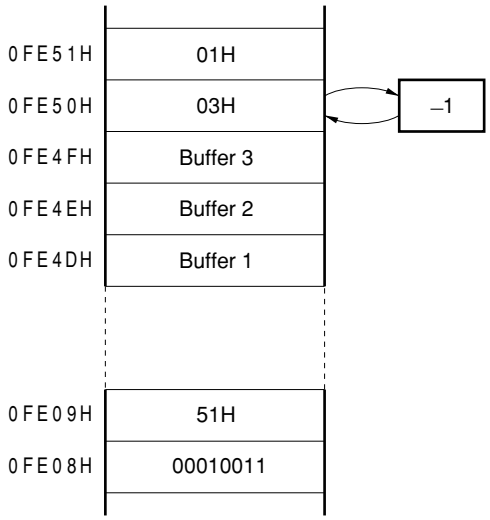
Data is transferred between the SFR specified by SFR.PTR and buffer, starting from buffer 1.

Each time transfer has been completed, MSC is decremented (–1). When MSC has reached 0, a vectored interrupt request is generated.

Table 14-9. Specifying Operation in Block Transfer Mode

CC	Operation	Transfer Data	Buffer Address
00	Buffer ← SFR	Byte	(Contents of CH.PTR) – (Contents of MSC) – 1
01	SFR ← buffer		
10	Buffer ← SFR	Word	(Contents of CH.PTR) – (Contents of MSC × 2) – 1
11	SFR ← buffer		

Example To transfer the contents of port 1 (P1) (0FF01H) to a buffer by using the INTOV1 interrupt request

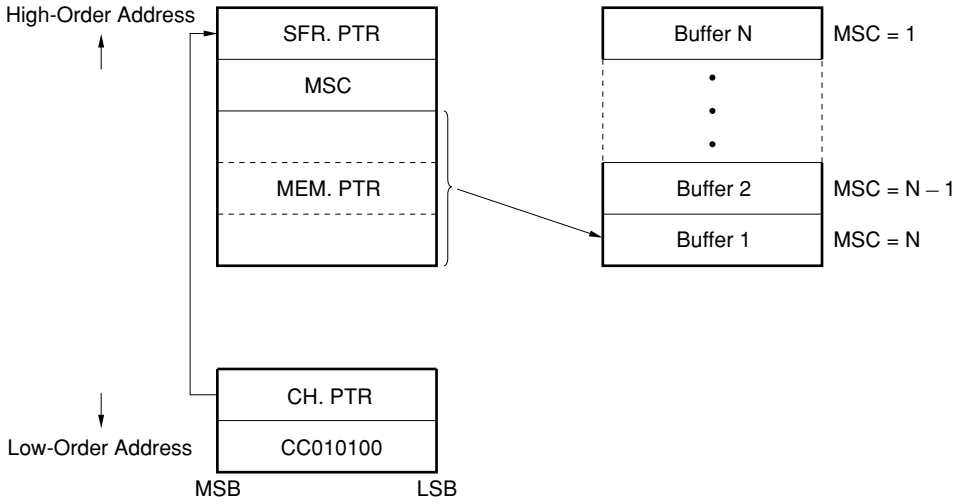


[Usage]

Data transmission/reception with serial interface

(3) Block transfer mode (with memory pointer): BLKTRS-P

[Macro service control word]



[Operation]

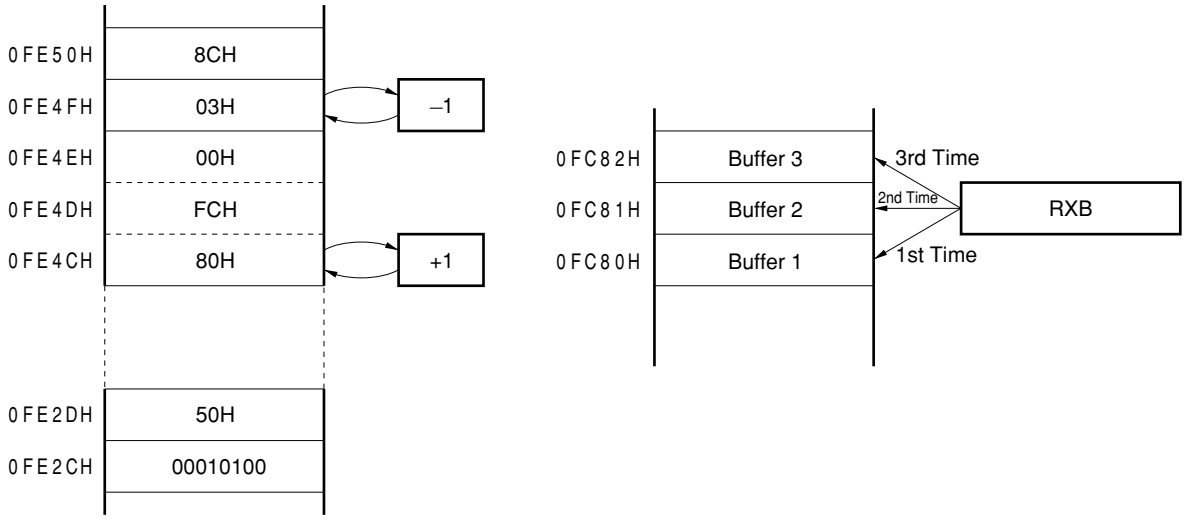
An SFR pointer (SFR.PTR) is specified by a channel pointer (CH.PTR). Data is transferred between an SFR specified by the SFR.PTR and the buffer addressed by the memory pointer (MEM.PTR), starting from buffer 1.

On completion of transferring byte data, the MEM.PTR is incremented (+1). On completion of transferring word data, the MEM.PTR is incremented (+2). Each time transfer has been completed, the macro service counter (MSC) is decremented (–1). When MSC = 0, a vectored interrupt request is generated.

Table 14-10. Specifying Operation in Block Transfer Mode (with memory pointer)

CC	Operation	Transfer Data
00	Buffer ← SFR	Byte
01	SFR ← buffer	
10	Buffer ← SFR	Word
11	SFR ← buffer	

Example To transfer the contents of the serial receive buffer: UART0 (RXB) (0FF8CH) to a buffer by using the INTSR interrupt request

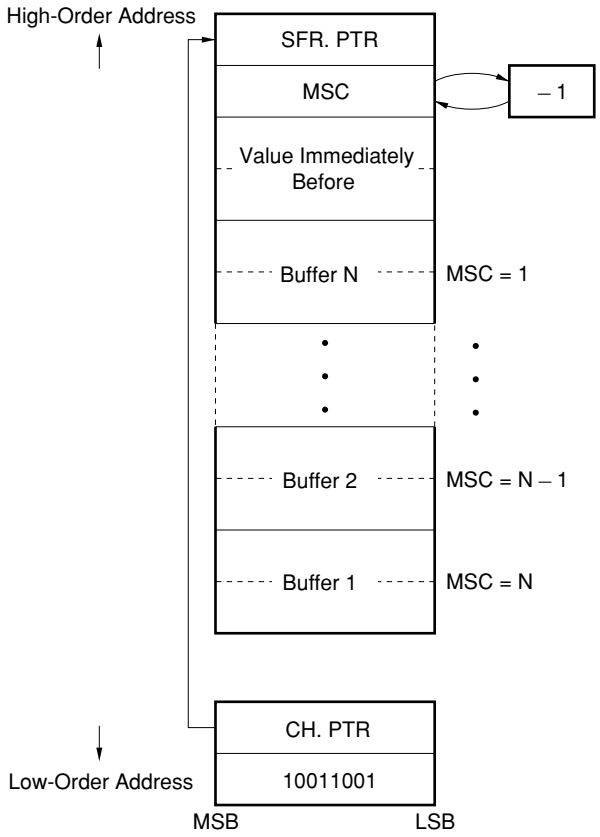


[Usage]

Data transmission/reception with serial interface

(4) Data differential mode: DTADIF

[Macro service control word]



[Operation]

An SFR pointer (SFR.PTR) is specified by a channel pointer (CH.PTR), and a buffer is addressed by the CH.PTR and macro service counter (MSC).

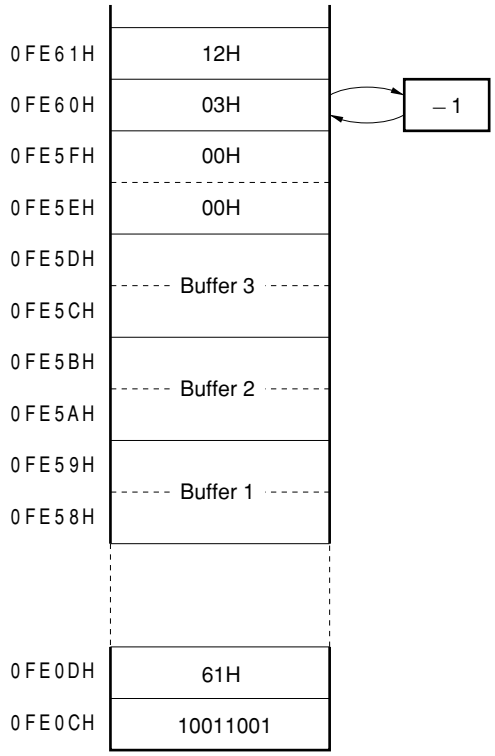
The difference between the current value of the SFR (including capture registers) specified by the SFR.PTR and the "value immediately before" is written to the buffer. This current value of the SFR is used as the "value immediately before". Writing data is started from buffer 1.

Each time data has been written, the MSC is decremented (–1). When MSC = 0, a vectored interrupt request is generated.

The buffer address is determined as follows:

(Buffer address) = (Contents of CH.PTR) – (Contents of MSC × 2) – 3

Example To write the difference between the capture/compare register 00 (CC00) (0FF12H) and the "value immediately before) to the buffer by using the INTP0 input signal as a trigger. The cycle of the INTP0 input signal is measured by using the difference in the vectored interrupt processing routine.



[Usage]

To measure cycles and pulse widths by using a capture register

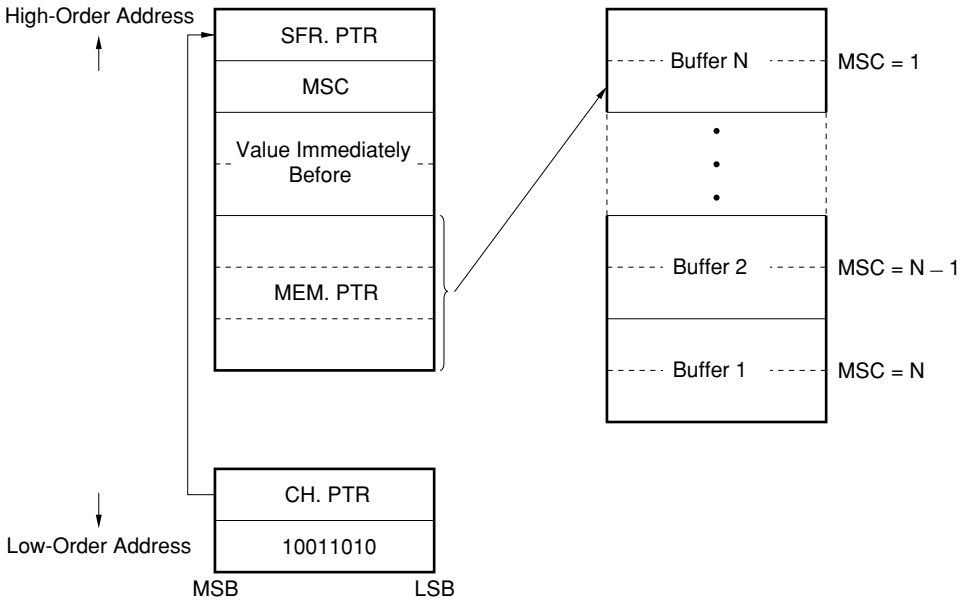
Cautions 1. Do not clear the macro service counter (MSC) to 00H.

2. Initialize the "value immediately before" (with dummy data) in advance.

3. Only a 16-bit SFR can be specified by the SFR pointer (SFR.PTR).

(5) Data differential mode (with memory pointer): DTADIF-P

[Macro service control word]



[Operation]

An SFR pointer (SFR.PTR) is specified by a channel pointer (CH.PTR), and a buffer is addressed by the memory pointer (MEM.PTR) and macro service counter (MSC).

The difference between the current value of the SFR (including capture registers) specified by the SFR.PTR and the "value immediately before" is written to the buffer. This current value of the SFR is used as the "value immediately before". Writing data is started from buffer 1.

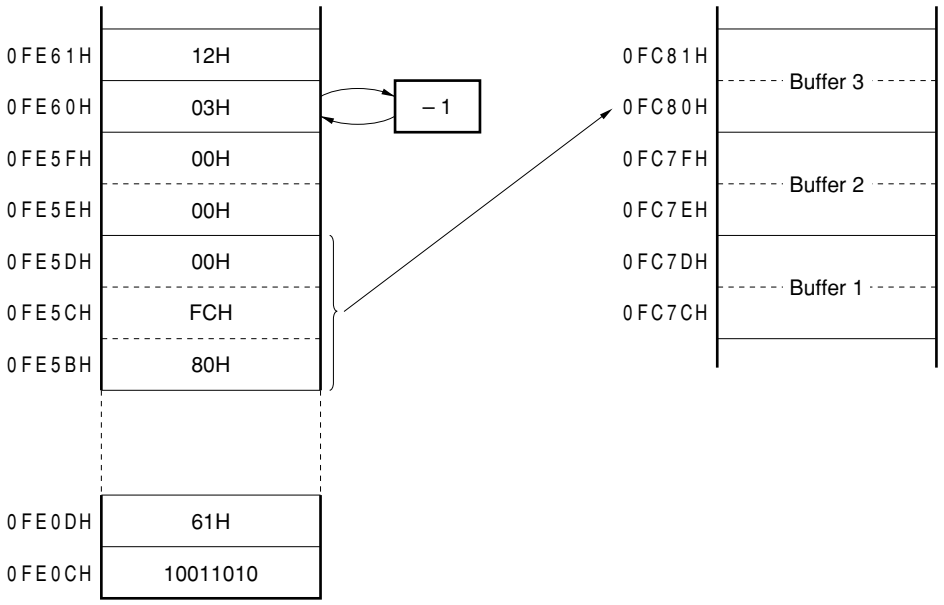
Each time data has been written, the MSC is decremented (–1). When MSC = 0, a vectored interrupt request is generated.

The MEM.PTR is not affected.

The buffer address is determined as follows:

(Buffer address) = (Contents of MEM.PTR) – (Contents of MSC × 2) + 2

Example To write the difference between the capture/compare register 00 (CC00) (0FF12H) and the "value immediately before" to the buffer by using the INTP0 input signal as a trigger. The cycle of the INTP0 input signal is measured by using the difference in the vectored interrupt routine.



[Usage]

To measure cycles and pulse widths by using a capture register

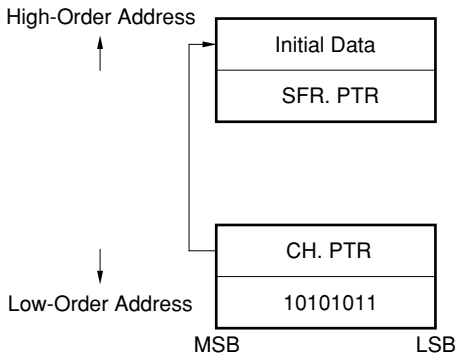
Cautions 1. Do not clear the macro service counter (MSC) to 00H.

2. Initialize the "value immediately before" (with dummy data) in advance.

3. Only a 16-bit SFR can be specified by the SFR pointer (SFR.PTR).

(6) CPU monitor mode 0: SELF0

[Macro service control word]



[Operation]

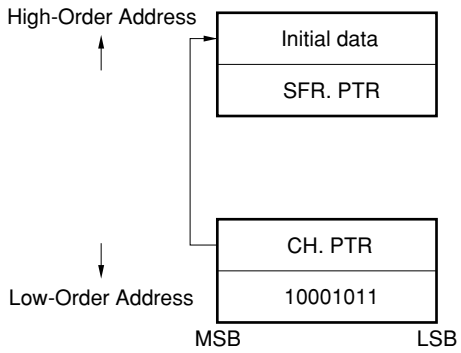
Checks the internal operation of the CPU. The items to be checked are as follows:

- Writing to program status word (PSW)
- Stack pointer (SP)
- Main RAM
- Main RAM addressing
- Compare operation

If the CPU is operating normally, the value resulting from subtracting 10 from the initial data is transferred to an SFR specified by the SFR pointer (SFR.PTR). If an abnormality of the CPU is detected, a value different from that transferred during normal operation is transferred.
After completion of this macro service, the contents of the main RAM and the value of SP are not destroyed, but the value of PSW is set to 0x00H.
Therefore, this macro service must be executed when initialization is performed. After that, use CPU monitor mode 1 to be explained next.

(7) CPU monitor mode 1: SELF1

[Macro service control word]



[Operation]

Checks the internal operation of the CPU. The items to be checked are as follows:

- Stack pointer (SP)
- Main RAM
- Main RAM addressing
- Compare operation

If the CPU is operating normally, the value resulting from subtracting 8 from the initial data is transferred to an SFR specified by the SFR pointer (SFR.PTR). If an abnormality of the CPU is detected, a value different from that transferred during normal operation is transferred.
After completion of this macro service, the contents of the main RAM and the value of SP are not destroyed.

14.9 When Interrupt Request and Macro Service Are Temporarily Held Pending

When the following instructions are executed, interrupt acknowledgment and macro service processing is deferred for 8 system clock cycles. However, software interrupts are not deferred.

EI
DI
BRK
BRKCS
RETCS
RETCSB !addr16
RETI
RETIB
LOCATION 0H or LOCATION 0FH
POP PSW
POPU post
MOV PSWL, A
MOV PSWL, #byte
MOVG SP, #imm24
Write instruction and bit manipulation instruction to an interrupt control register[Note], or the MK0, MK1L, IMC or ISPR register (Excluding BT and BF instructions)
PSW bit manipulation instruction
(Excluding the BT PSWL. bit, $addr20, BF PSWL. bit, $addr20, BT PSWH. bit, $addr20, BF PSWH. bit, $addr20, SET1 CY, NOT1 CY, and CLR1 CY instructions)

Note Interrupt control registers: OVIC0, OVIC1, OVIC4, PIC0-PIC6, CMIC10, CMIC11, CMIC40, CMIC41, SERIC, SRIC, CSIIC1, STIC, SERIC2, SRIC2, CSIIC2, STIC2, ADIC

Cautions 1. When an interrupt related register is polled using a BF instruction, etc., the branch destination of that BR instruction, etc., should not be that instruction. If a program is written in which a branch is made to that instruction itself, all interrupts and macro service requests will be held pending until a condition whereby a branch is not made by that instruction arises.

```
    Bad Example
            ⋮
LOOP :  BF  PIC0.7,  $LOOP      All interrupts and macro service requests are held pending until
                                PIC0.7 is 1.
          ×××                   ← Interrupts and macro service requests are not serviced until
                                after execution of the instruction following the BF instruction.
            ⋮

  Good Example (1)
            ⋮
LOOP :  NOP
        BF  PIC0.7,  $LOOP      ← Interrupts and macro service requests are serviced after execu
                                tion of the NOP instruction, so that interrupts are never held
            ⋮                   pending for a long period.

  Good Example (2)
            ⋮
LOOP :  BT  PIC0.7,  $NEXT      Using a BTCLR instruction instead of a BT instruction has the
                                advantage that the flag is cleared (0) automatically.
        BR  $LOOP               ← Interrupts and macro service requests are serviced after execu-
NEXT :      ⋮                   tion of the BR instruction, so that interrupts are never held
                                pending for a long period.
```

2. For a similar reason, if problems are caused by a long pending period for interrupts and macro service when instructions to which the above applies are used in succession, a time at which interrupts and macro service requests can be acknowledged should be provided by inserting an NOP instruction, etc., in the series of instructions.

14.10 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

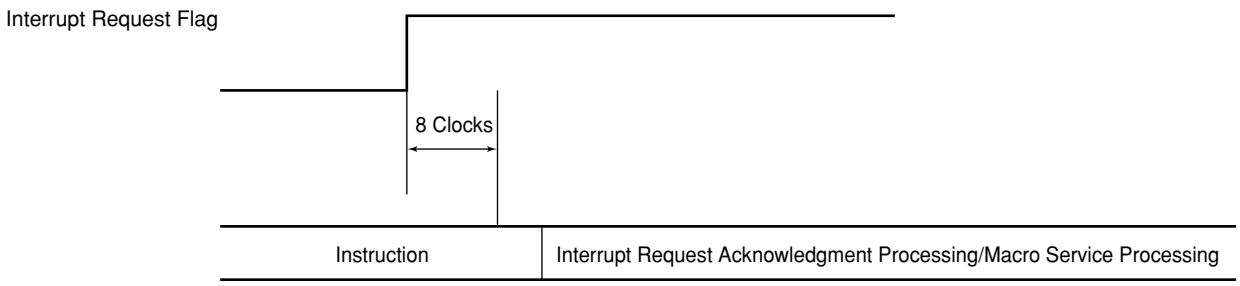
14.11 Interrupt and Macro Service Operation Timing

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 (0.5 μs: f_{CLK} = 16 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 **14.9 When Interrupt Request and Macro Service Are Temporarily Held Pending** for deferred instructions).

Figure 14-21. Interrupt Request Generation and Acknowledgment (Unit: Clocks)



14.11.1 Interrupt acceptance processing time

To accept an interrupt, the time shown in Table 14-11 is required. After this time has elapsed, the interrupt processing routine is executed.

Table 14-11. Interrupt Acceptance Processing Time

(unit: clock)

Interrupt Processing Mode / Branch Detection	Vector table / Stack	Vectored Interrupt								Context Switching
		IROM, EMEM16				EMEM8				
		IRAM	PRAM	EMEM16	EMEM8	IRAM	PRAM	EMEM16	EMEM8	
IROM, PRAM		26	30	30+2n	38+4n	30	34	34+2n	42+4n	22
EMEM16, EMEM8		27	31	31+2n	39+4n	31	35	35+2n	43+4n	23

Remarks 1. IROM : internal ROM (with high-speed fetch specified)
IRAM : internal high-speed RAM
PRAM : peripheral RAM (only when the LOCATION 0H instruction is executed in the case of branch destination)
EMEM16 : external memory and internal ROM not specified for high-speed fetch and set to 16-bit bus width
EMEM8 : external memory and internal ROM not specified for high-speed fetch and set to 8-bit bus width

2. n indicates the number of wait states per byte necessary for writing to the stack.

3. If the vector table is EMEM16 or EMEM8 and if wait states are inserted when reading the vector table, the processing time is extended. Add 2m in the case of vector interrupt with EMEM8 or m in the case of context switching with EMEM16 to the values in the above table. m is the number of wait states per byte necessary for reading the vector table.

4. If the branch destination is EMEM16 or EMEM8, and if wait states are inserted when reading the instruction at the branch destination, add the number of wait states to the value in the above table.

5. If the stack is in PRAM and the value of the stack pointer (SP) is odd, add 8 to the value in the above table. If the value of SP is odd with EMEM16, add 8+2n to the value in the above table.

6. The number of wait states is the total number of address wait and access wait states.

14.11.2 Processing time of macro service

The processing time of the macro service differs depending on the type of the macro service, as shown in Table 14-12.

Table 14-12. Macro Service Processing Time

(unit: clock)

Type of Macro Service				Processing Time	
				IRAM	Other Data Area
Group 0	Counter mode: EVTCNT			18	—
Group 1	Block transfer mode: BLKTRS	Buffer ← SFR	Byte	24	—
			Word	25	—
		SFR ← buffer	Byte	24	—
			Word	25	—
	Block transfer mode (with memory pointer): BLKTRS-P	Buffer ← SFR	Byte	30	32
			Word	31	33
		SFR ← buffer	Byte	30	32
			Word	31	33
	Data differential mode: DTADIF			28	—
	Data differential mode (with memory pointer): DTADIF-P			33	35
Group 2	CPU monitor mode 0: SELF0			—	78
	CPU monitor mode 1: SELF1			—	60

Remarks 1. Add the number of clocks specified for each case in the following cases in the other data areas.

- If data size is word and data is located at an odd address in IROM or PRAM: 8 clocks
- If data size is byte in EMEM16 or EMEM8, or if data size is word in EMEM16 and data is located at an even address: n (n is the number of wait states per byte)
- If data size is word in EMEM8, or if data size is word in EMEM16 and data is located at an odd address: 4 + 2n (n is the number of wait states per byte)

2. Data is output to an SFR in the CPU monitor modes.

3. IRAM : internal high-speed RAM
IROM : internal ROM (with high-speed fetch specified)
PRAM : peripheral RAM
EMEM16 : external memory and internal ROM not specified for high-speed fetch and set to 16-bit bus width
EMEM8 : external memory and internal ROM not specified for high-speed fetch and set to 8-bit bus width

14.12 Restoring Interrupt Function To Initial State

If an inadvertent program loop or system error is detected by means of an operand error interrupt, the watchdog timer, NMI pin input, etc., the entire system must be restored to its initial state. In the μPD784054, 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   MK1, #0FFFFH
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 & HIGHW are assembler operators for calculating low-order
                                16 bits and high-order 16 bits respectively of symbol NEXT
NEXT    :
```

- It is necessary to ensure that a non-maskable interrupt request is not generated via the NMI pin during execution of this program.
- After this, 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).

14.13 Cautions

(1) The in-service priority register (ISPR) is read-only. Writing to this register may result in misoperation.

(2) The watchdog timer mode register (WDM) can only be written to with a dedicated instruction (MOV WDM/#byte).

(3) The RETI instruction must not be used to return from a software interrupt caused by a BRK instruction.

(4) The RETCS instruction must not be used to return from a software interrupt caused by a BRKCS instruction.

(5) 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.

(6) 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.

(7) 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 14.9. 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 (SFR)**), and the CPU becomes deadlocked, or the PC and PSW are written to an unexpected signal is output from a pin, or an address is which RAM is not mounted, with the result that the return from the non-maskable interrupt service program is not performed normally and a software upsets occurs.
Therefore, the program following $\overline{\text{RESET}}$ release must be as follows.

```
        CSEG  AT 0
        DW    STRT
        CSEG  BASE
STRT:
        LOCATION 0FH; or LOCATION 0H
        MOVG  SP, #imm24
```

(8) When a maskable interrupt is acknowledged by vectored interruption, 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.

(9) 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.

(10) If data is transmitted with UART by using the macro service, a vectored interrupt is generated two times (refer to **12.2.8 Transmitting/receiving data with macro service**).

(11) Do not clear the macro service counter (MSC) to 00H in the data differential mode and data differential mode (with memory pointer).

(12) Initialize the “value immediately before” (with dummy data) in advance in the data differential mode and data differential mode (with memory pointer).

(13) Only a 16-bit SFR can be specified by the SFR pointer (SFR.PTR) in the data differential mode and data differential mode (with memory pointer).

(14) When an interrupt related register is polled using a BF instruction, etc., the branch destination of that BR instruction, etc., should not be that instruction. If a program is written in which a branch is made to that instruction itself, all interrupts and macro service requests will be held pending until a condition whereby a branch is not made by that instruction arises.

```
        Bad Example
              ⋮
LOOP:   BF  PIC0.7, $LOOP      All interrupts and macro service requests are held pending until PIC0.7 is
                               1.
          ×××                  ← Interrupts and macro service requests are not processed until after exe-
              ⋮                  cution of the instruction following the BF instruction.

        Good Example (1)
              ⋮
LOOP:   NOP
        BF  PIC0.7, $LOOP      ← Interrupts and macro service requests are serviced after execution of the
                                 NOP instruction, so that interrupts are never held pending for a long period.
              ⋮

        Good Example (2)
              ⋮
LOOP:   BT  PIC0.7, $NEXT      Using a BTCLR instruction instead of a BT instruction has the advantage
                               that the flag is cleared (0) automatically.
        BR  $LOOP              ← Interrupts and macro service requests are serviced after execution of the
                                 BR instruction, so that interrupts are never held pending for a long period.
NEXT:         ⋮
```

(15) For a similar reason to that given in (14), if problems are caused by a long pending period for interrupts and macro service when instructions to which the above applies are used in succession, a time at which interrupts and macro service requests can be acknowledged should be provided by inserting an NOP instruction, etc., in the series of instructions.

CHAPTER 15 LOCAL BUS INTERFACE FUNCTION

The local bus interface function is provided for the connection of external memory (ROM and RAM) and I/Os.

External memory (ROM and RAM) and I/Os are accessed using the $\overline{RD}$, $\overline{LWR}$, $\overline{HWR}$ and ASTB pin signals, with pins AD0 to AD15 used as the multiplexed address/data bus and pins A16 to A19 as the address bus.

The basic bus interface timing is shown in Figures 15-3 to 15-8.

In addition, a wait function that is used to interface with a low-speed memory, and a bus sizing function that can change the external data bus width between 8 bits and 16 bits are also provided.

15.1 Memory Extension Function

With the μPD784054, external memory and I/O extension can be performed by setting the memory extension mode register (MM).

15.1.1 Memory extension mode register (MM)

The MM is an 8-bit register that performs external extension memory control, address wait number specification, and internal fetch cycle control.

The MM register can be read or written to with an 8-bit manipulation instruction or bit manipulation instruction. The MM format is shown in Figure 15-1.

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

Figure 15-1. Format of Memory Expansion Mode Register (MM)

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

	7	6	5	4	3	2	1	0
MM	IFCH	0	AW	0	MM3	MM2	MM1	MM0

IFCH	Fetches Internal ROM
0	Fetches at same speed as external memory. All setting of wait control is valid.
1	High-speed fetch. Specification of wait control is invalid.

AW	Specifies Address Wait
0	Disabled
1	Enabled

MM3	MM2	MM1	MM0	Mode	Port 4 (P40 to P47)	Port 5 (P50 to P57)	Port 6 (P60 to P63)	P90-P93
0	0	0	0	Single-chip mode	Port			
0	0	1	1	256-byte extension mode Note 1	AD0-AD7			P90 : $\overline{RD}$ P91 : $\overline{LWR}$ P92 : $\overline{HWR}$ P93 : ASTB
0	1	0	0	1K-byte extension mode Note 1		AD8, AD9 Note2 Port		
0	1	0	1	4K-byte extension mode Note 1		AD8-AD11 Note2 Port		
0	1	1	0	16K-byte extension mode Note 1		AD8-AD13 Note2 Port		
0	1	1	1	64K-byte extension mode		AD8-AD15		
1	0	0	0	256K-byte extension mode			A16, A17 Port	
1	0	0	1	1M-byte extension mode			A16-A19	
Other				Setting prohibited				

Notes 1. Setting prohibited when external 16-bit bus is specified.

2. Used as an address bus.

15.1.2 Memory map with external memory extension

The memory map when memory extension is used is shown in Figure 15-2. External devices at the same addresses as the internal ROM area, internal RAM area and SFR area (excluding the external SFR area (0FFD0H to 0FFDFH)) cannot be accessed. If an access is made to these addresses, the memory or SFR in the μPD784054 has access priority and no ASTB signal, $\overline{RD}$ signal, $\overline{LWR}$, or $\overline{HWR}$ signal is output (these pins remain at the inactive level). The address bus output level remains at the level output prior to this, and the address/data bus output becomes high-impedance.

Except in 1M-byte extension mode, the address output externally is output with the upper part of the address specified by the program masked.

Example 1:

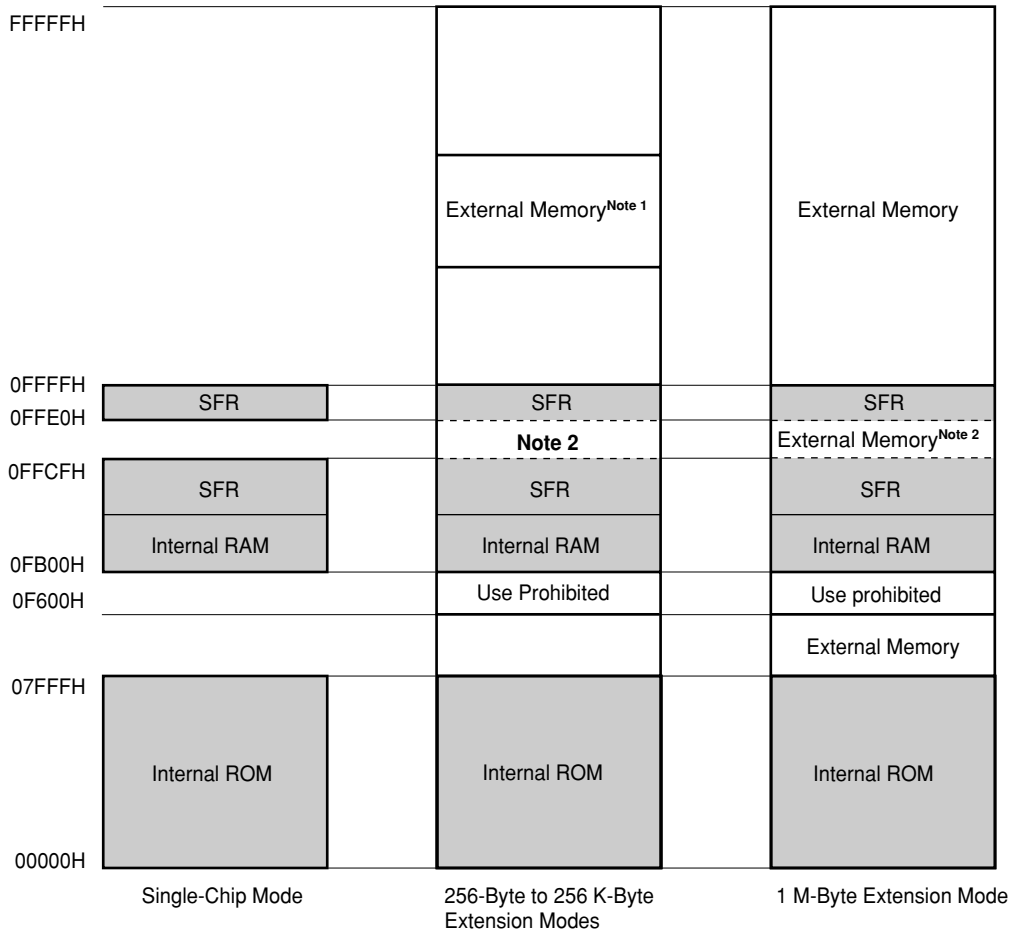
In 256-byte extension mode, when address 54321H is accessed by the program, the output address is 21H.

Example 2:

In 256-byte extension mode, when address 67821H is accessed by the program, the output address is 21H.

Figure 15-2. Memory Map (1/2)

(a) When LOCATION 0H instruction is executed

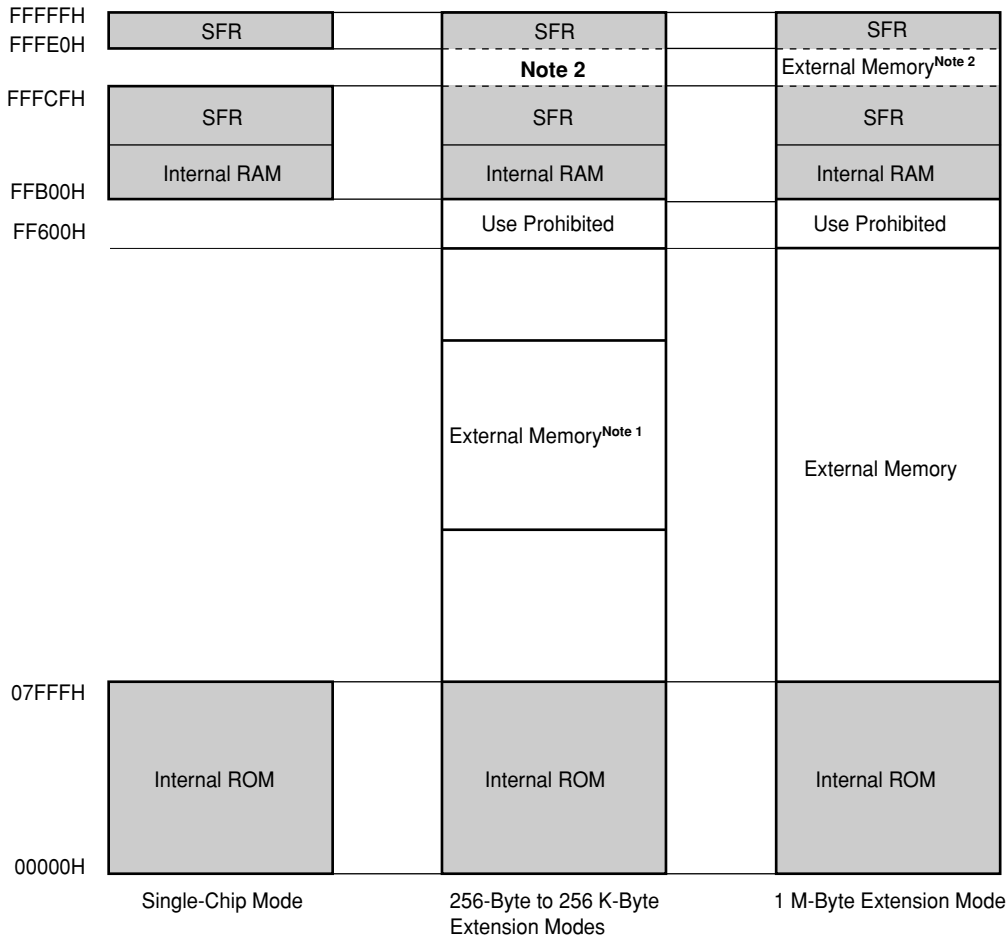


Notes 1. Any extension size area in unshaded part

2. External SFR area

Figure 15-2. Memory Map (2/2)

(b) When LOCATION 0FH instruction is executed



Notes 1. Any extension size area in unshaded part

2. External SFR area

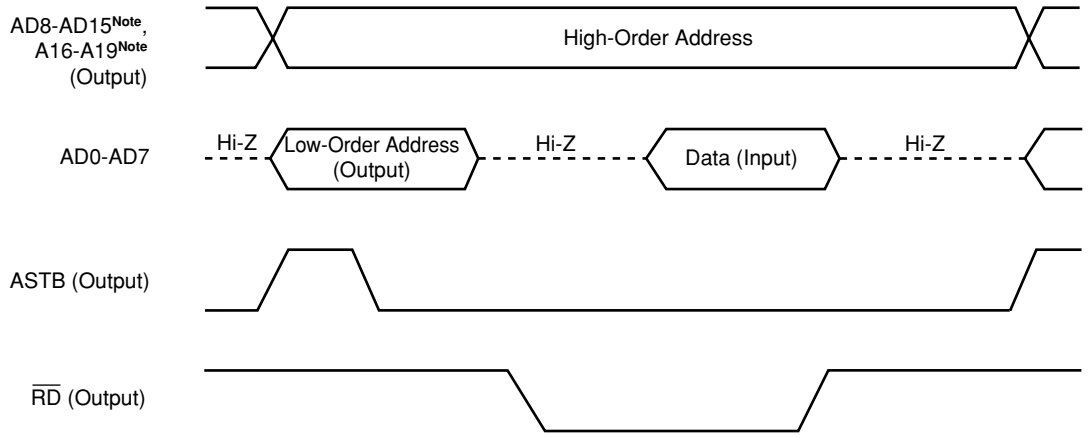
15.1.3 Basic operation of local bus interface

The local bus interface accesses external memory using ASTB, $\overline{RD}$, $\overline{LWR}$, $\overline{HWR}$, an address/data bus (AD0 to AD15) and address bus (A8 to A19). When the local bus interface is used, port 4 and P90 to P93 automatically operate as AD0 to AD7, $\overline{RD}$, $\overline{LWR}$, $\overline{HWR}$, and ASTB. In ports 5 and 6, only the pins that correspond to the extension memory size operate as address bus pins.

An outline of the memory access timing is shown in Figures 15-3 to 15-8.

Figure 15-3. Read Timing (8 Bits)

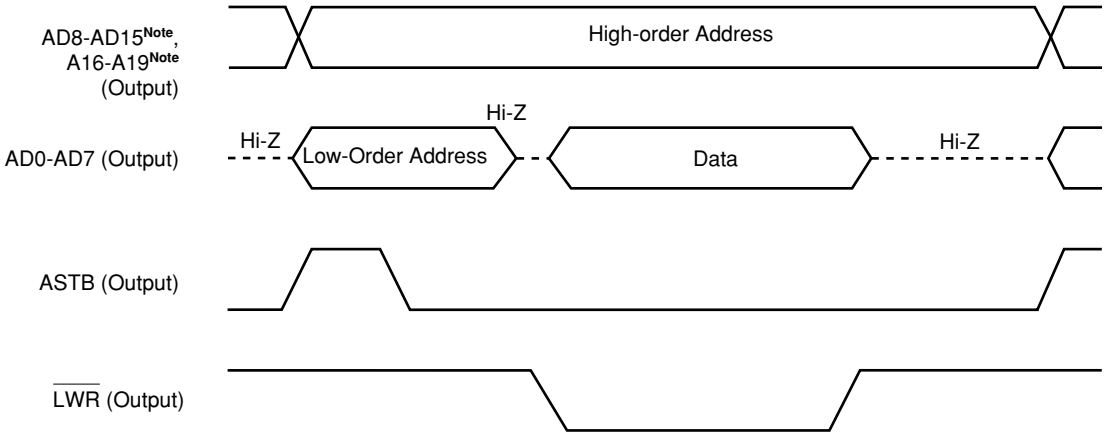
Condition • Bus Size : 8 bits
• Bus Cycle : No Wait



Note The number of address bus pins used depends on the extension mode size.

Figure 15-4. Write Timing (8 Bits)

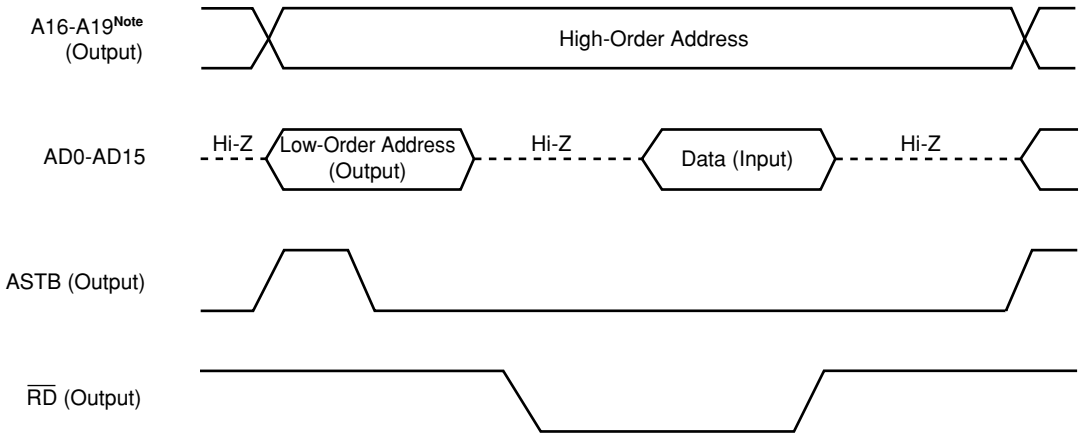
Condition • Bus Size : 8 bits
• Bus Cycle : No Wait



Note The number of address bus pins used depends on the extension mode size.

Figure 15-5. Read Timing (16 Bits, Even Address Access)

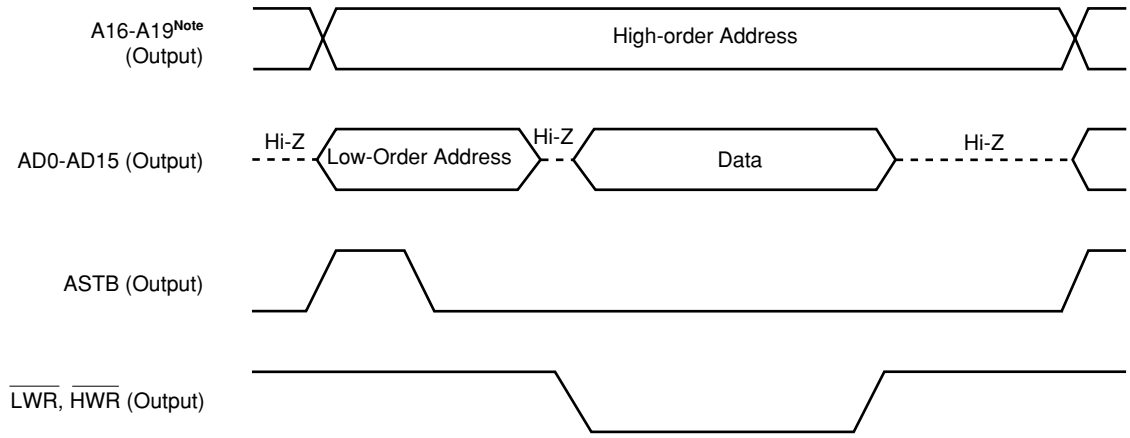
Condition • Bus Size : 16 bits • Low-Order 8-Bit Data : Even Address
• Bus Cycle : No Wait • High-Order 8-Bit Data : Odd Address



Note The number of address bus pins used depends on the extension mode size.

Figure 15-6. Write Timing (16 Bits, Even Address Access)

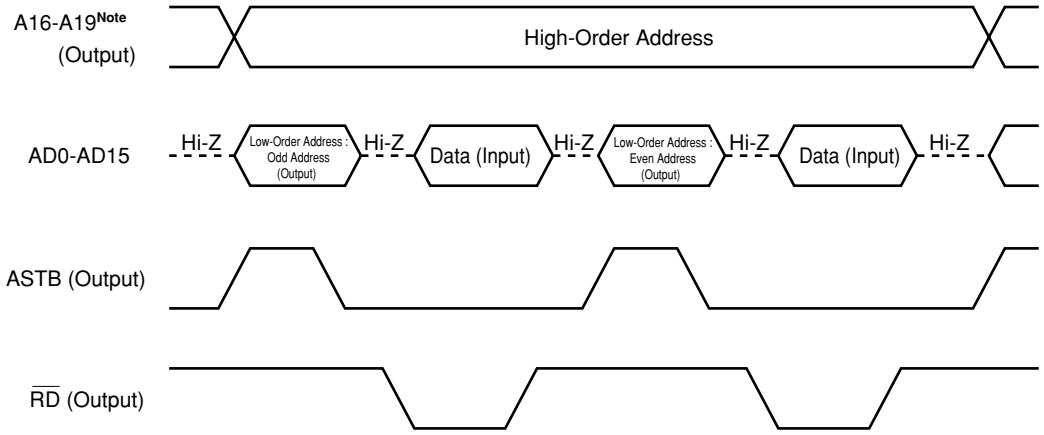
Condition • Bus Size : 16 bits • Low-Order 8-Bit Data : Even Address
• Bus Cycle : No Wait • High-Order 8-Bit Data : Odd Address



Note The number of address bus pins used depends on the extension mode size.

Figure 15-7. Read Timing (16 Bits, Odd Address Access)

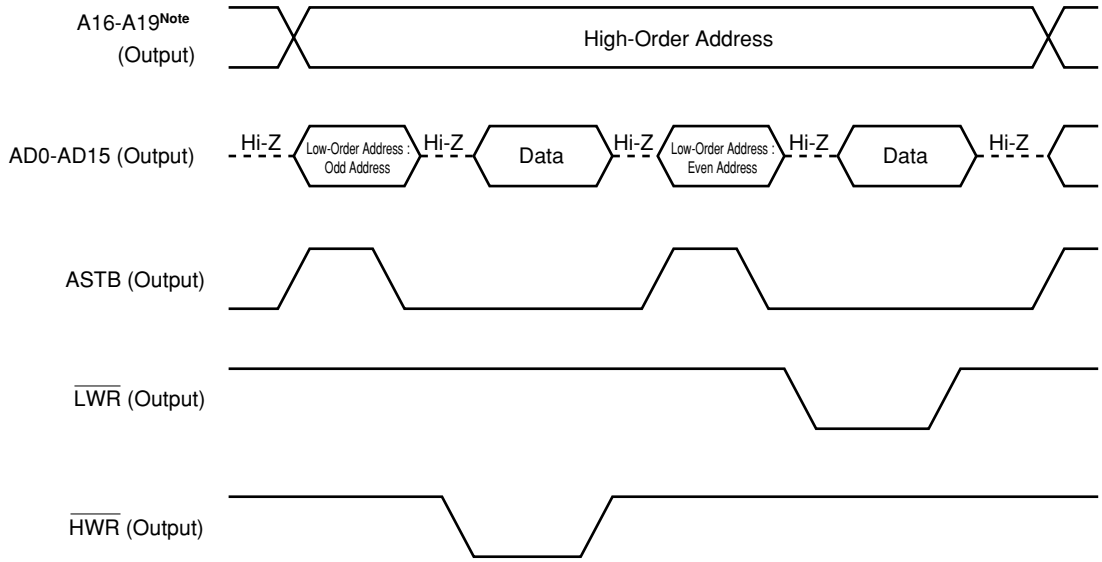
Condition • Bus Size : 16 bits
• Bus Cycle : No Wait
• Low-Order 8-Bit Data : Odd Address
• High-Order 8-Bit Data : Even Address



Note The number of address bus pins used depends on the extension mode size.

Figure 15-8. Write Timing (16 Bits, Odd Address Access)

Condition • Bus Size : 16 bits
• Bus Cycle : No Wait
• Low-Order 8-Bit Data : Odd Address
• High-Order 8-Bit Data : Even Address



Note The number of address bus pins used depends on the extension mode size.

15.2 Wait Function

When a low-speed memory or I/O is connected externally to the μPD784054, waits can be inserted in the external memory access cycle.

There are two kinds of wait cycle, an address wait for securing the address decoding time, and an access wait for securing the access time.

15.2.1 Wait function control registers

(1) Memory extension mode register (MM)

The IFCH bit of the MM performs wait control setting for internal ROM accesses, and the AW bit performs address wait setting.

The MM can be read or written to with an 8-bit manipulation instruction. The MM format is shown in Figure 15-9. When $\overline{\text{RESET}}$ is input, the MM register is set to 20H, the same cycle as for external memory is used for internal ROM accesses, and the address wait function is validated.

Figure 15-9. Format of Memory Extension Mode Register (MM)

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

	7	6	5	4	3	2	1	0
MM	IFCH	0	AW	0	MM3	MM2	MM1	MM0

IFCH	Fetches Internal ROM
0	Fetches at same speed as external memory. All setting of wait control is valid.
1	High-speed fetch. Specification of wait control is invalid.

AW	Specifies Address Wait
0	Disabled
1	Enabled

MM3	MM2	MM1	MM0	Sets memory extension mode (refer to **Figure 15-1**).

(2) Programmable wait control registers (PWC1/PWC2)

The PWC1 and PWC2 specify the number of waits.

PWC1 is an 8-bit register that divides the space from 0 to FFFFH into four, and specifies wait control for each of these four spaces. PWC2 is a 16-bit register that divides the space from 10000H to FFFFH into four, and specifies wait control for each of these four spaces.

The PWC1 can be read or written to with an 8-bit manipulation instruction, and the PWC2 with a 16-bit manipulation instruction. The PWC1 and PWC2 formats are shown in Figures 15-10 and 15-11.

The high-order 8 bits of the PWC2 are fixed at AAH, and therefore ensure that the high-order 8 bits are set to AAH.

When $\overline{\text{RESET}}$ is input, the PWC1 is set to AAH, and the PWC2 to AAAAH, and 2-wait insertion is performed on the entire space.

Figure 15-10. Format of Programmable Wait Control Register 1 (PWC1)

Address : 0FFC7H On reset : AAH R/W

	7	6	5	4	3	2	1	0
PWC1	PW31	PW30	PW21	PW20	PW11	PW10	PW01	PW00

Valid Address	PW31	PW30	Inserted Wait Cycle	Data Access Cycle, Fetch Cycle
00C000H-00FFFFH Note	0	0	0	3
	0	1	1	4
	1	0	2	5
	1	1	Time of low level input to $\overline{\text{WAIT}}$ pin	–

Valid Address	PW21	PW20	Inserted Wait Cycle	Data Access Cycle, Fetch Cycle
008000H-00BFFFH	0	0	0	3
	0	1	1	4
	1	0	2	5
	1	1	Time of low level input to $\overline{\text{WAIT}}$ pin	–

Valid Address	PW11	PW10	Inserted Wait Cycle	Data Access Cycle, Fetch Cycle
004000H-007FFFH	0	0	0	3
	0	1	1	4
	1	0	2	5
	1	1	Time of low level input to $\overline{\text{WAIT}}$ pin	–

Valid Address	PW01	PW00	Inserted Wait Cycle	Data Access Cycle, Fetch Cycle
000000H-003FFFH	0	0	0	3
	0	1	1	4
	1	0	2	5
	1	1	Time of low level input to $\overline{\text{WAIT}}$ pin	–

Note Except the portion overlapping the internal data area.

Cautions 1. The above number of cycles is when no address cycle is appended. If an address cycle is appended, one cycle must be added.

2. No wait cycle is inserted when fetching instructions from the internal ROM or peripheral RAM area at high-speed.

3. Do not insert a wait cycle in the internal ROM area by using the $\overline{\text{WAIT}}$ pin.

Figure 15-11. Format of Programmable Wait Control Register 2 (PWC2)

Address : 0FFC8H On reset : AAAAH R/W

	15	14	13	12	11	10	9	8
PWC2	1	0	1	0	1	0	1	0
	7	6	5	4	3	2	1	0
	PW71	PW70	PW61	PW60	PW51	PW50	PW41	PW40

Valid Address	PW71	PW70	Inserted Wait Cycle	Data Access Cycle, Fetch Cycle
080000H-	0	0	0	3
0FFFFFH Note	0	1	1	4
	1	0	2	5
	1	1	Time of low level input to $\overline{\text{WAIT}}$ pin	—

Valid Address	PW61	PW60	Inserted Wait Cycle	Data Access Cycle, Fetch Cycle
040000H-	0	0	0	3
07FFFFH	0	1	1	4
	1	0	2	5
	1	1	Time of low level input to $\overline{\text{WAIT}}$ pin	—

Valid Address	PW51	PW50	Inserted Wait Cycle	Data Access Cycle, Fetch Cycle
020000H-	0	0	0	3
03FFFFH	0	1	1	4
	1	0	2	5
	1	1	Time of low level input to $\overline{\text{WAIT}}$ pin	—

Valid Address	PW41	PW40	Inserted Wait Cycle	Data Access Cycle, Fetch Cycle
010000H-	0	0	0	3
01FFFFH	0	1	1	4
	1	0	2	5
	1	1	Time of low level input to $\overline{\text{WAIT}}$ pin	—

Note Except the portion overlapping the internal data area.

Cautions 1. The above number of cycles is when no address cycle is appended. If an address cycle is appended, one cycle must be added.

2. No wait cycle is inserted when fetching instructions from the peripheral RAM area.

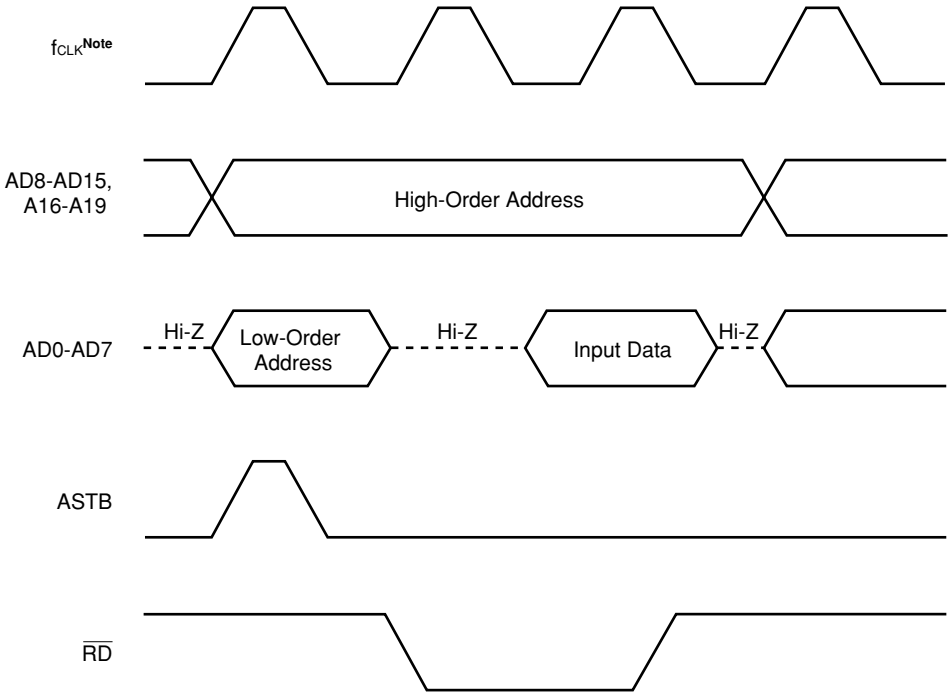
15.2.2 Address waits

Address waits are used to secure the address decoding time. If the AW bit of the memory extension mode register (MM) is set (1), waits are inserted in every memory access**Note**. When an address wait is inserted, the high-level period of the ASTB signal is extended by one system clock cycle (62.5 ns: f_{CLK} = 16 MHz).

Note Except for the internal RAM, internal SFRs, and internal ROM during high-speed fetch.
If it is specified that the internal ROM is accessed in the same cycle as the external ROM, an address wait state is inserted even when the internal ROM is accessed.

Figure 15-12. Read/Write Timing of Address Wait Function (1/3)

(a) Read timing with no address wait insertion

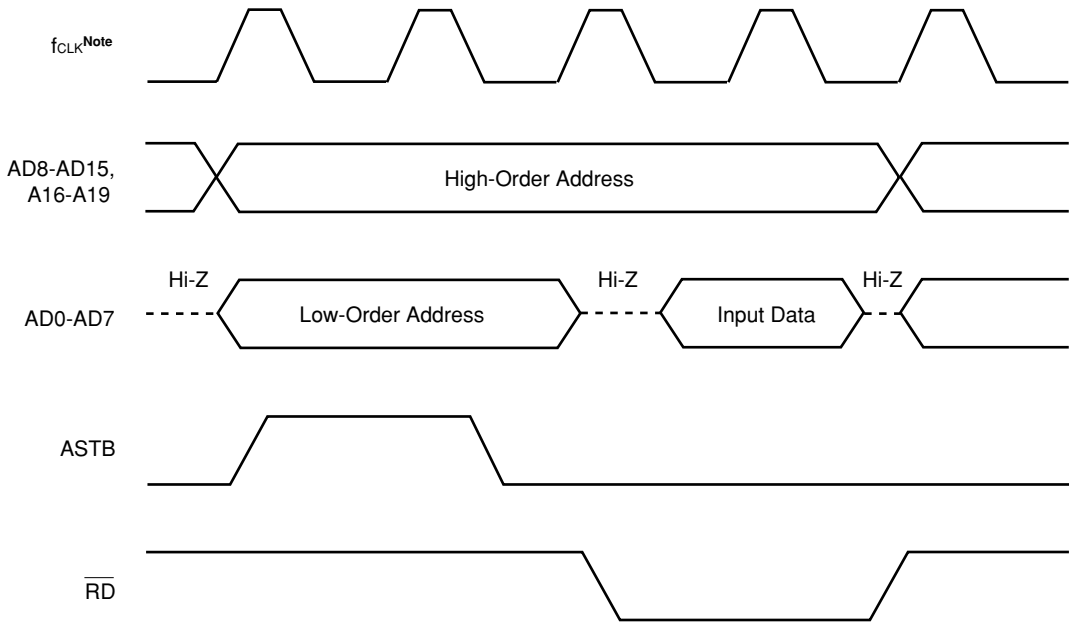


Note f_{CLK}: Internal system clock frequency. This signal is present inside the μPD784054 only.

Remark The above figure is an example of the 8-bit bus.

Figure 15-12. Read/Write Timing of Address Wait Function (2/3)

(b) Read timing with address wait insertion

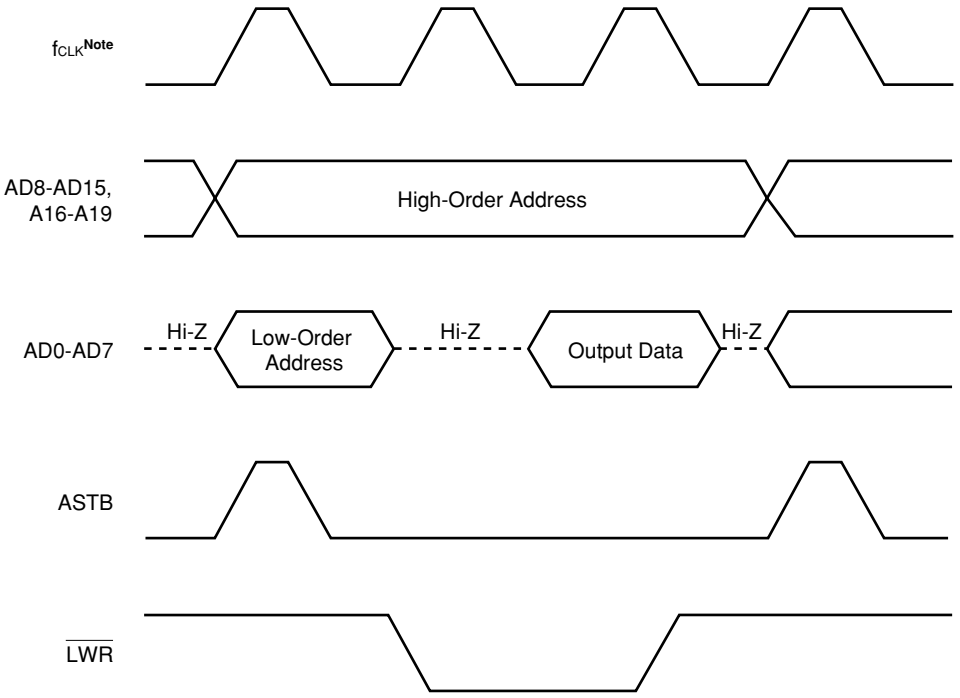


Note f_{CLK}: Internal system clock frequency. This signal is present inside the μPD784054 only.

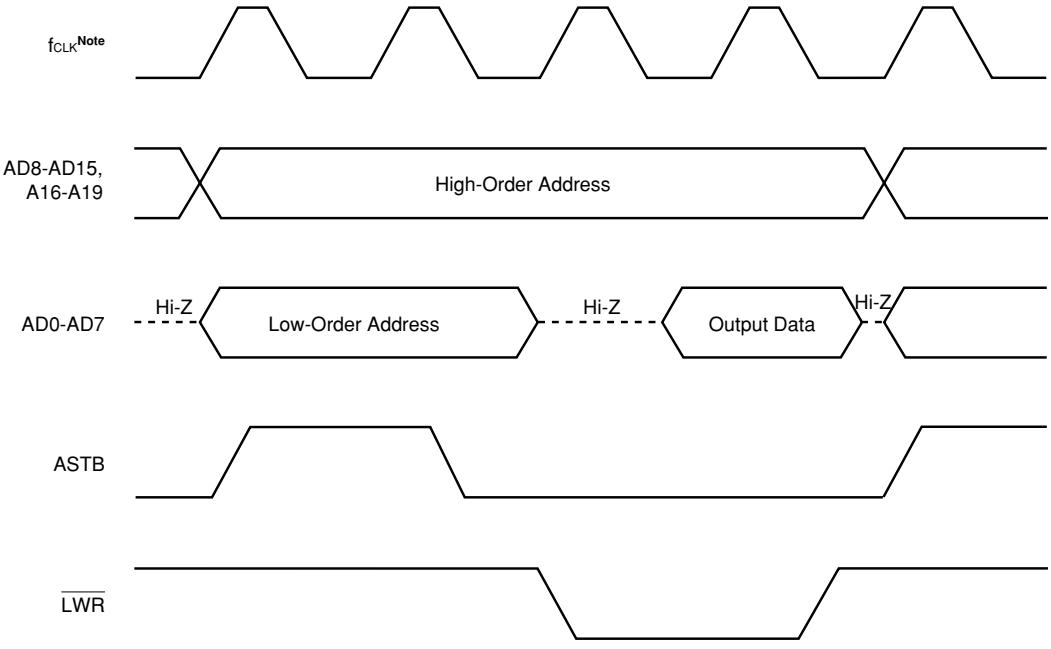
Remark The above figure is an example of the 8-bit bus.

Figure 15-12. Read/Write Timing of Address Wait Function (3/3)

(c) Write timing with no address wait insertion



(d) Write timing with address wait insertion



Note f_{CLK}: Internal system clock frequency. This signal is present inside the μPD784054 only.

Remark The above figure is an example of the 8-bit bus.

15.2.3 Access waits

Access waits are inserted in the $\overline{RD}$, $\overline{LWR}$, or $\overline{HWR}$ signal low-level period, and extend the low-level period by 1/f_{CLK} (62.5 ns: f_{CLK} = 16 MHz) per cycle.

There are two wait insertion methods, using either the programmable wait function that automatically inserts the preset number of cycles, or the external wait function controlled by a wait signal from outside.

For wait cycle insertion control, the 1 M-byte memory space is divided into eight as shown in Figure 15-14, and control is specified for each space by means of the programmable wait control registers (PWC1/PWC2). Waits are not inserted in accesses to internal ROM or internal RAM using high-speed fetches. In accesses to internal SFRs, waits are inserted at the necessary times regardless of this specification.

If access operations are specified as being performed in the same number of cycles as for external ROM, waits are inserted also in internal ROM accesses in accordance with the PWC1 settings.

The P94 pin functions as a $\overline{WAIT}$ input pin when the PMC94 bit of the port 9 mode control register (PMC9) is set (1). The P94 pin operates as a general-purpose I/O port pin when $\overline{RESET}$ is input (refer to **Figure 15-13**).

Bus timing in the case of access wait insertion is shown in Figures 15-15 to 15-17.

Caution Do not insert a wait cycle in the internal ROM area by using the $\overline{WAIT}$ pin.

Figure 15-13. Format of Port 9 Mode Control Register (PMC9)

Address : 0FF49H On reset : 00H R/W

	7	6	5	4	3	2	1	0
PMC9	0	0	0	PMC94	0	0	0	0

PMC94	Specifies Control Mode of Pin P94
0	I/O port mode
1	$\overline{WAIT}$ input mode

Figure 15-14. Wait Control Spaces

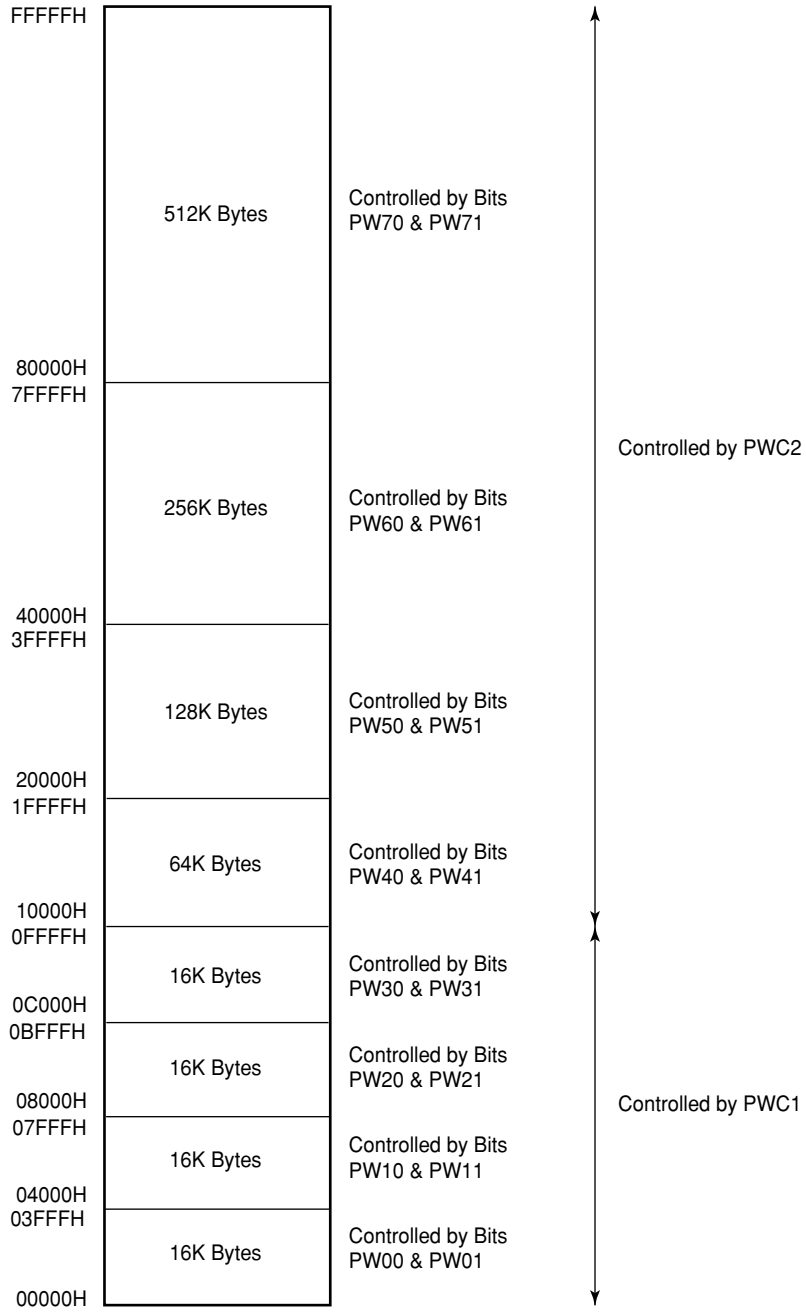
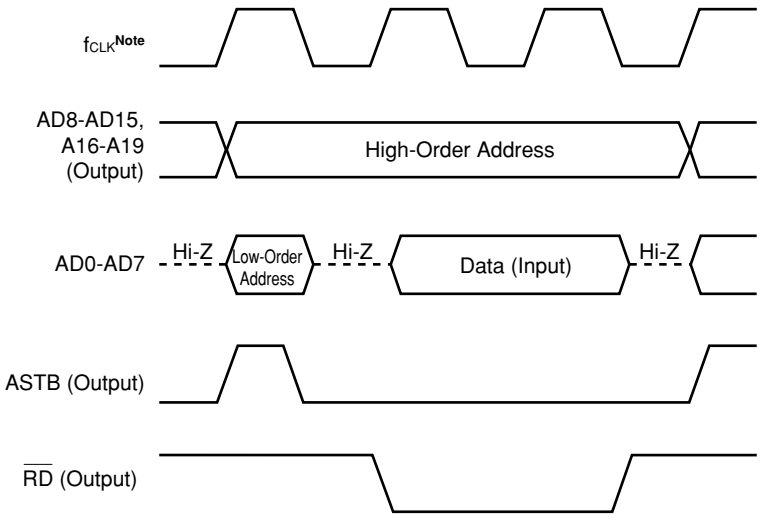
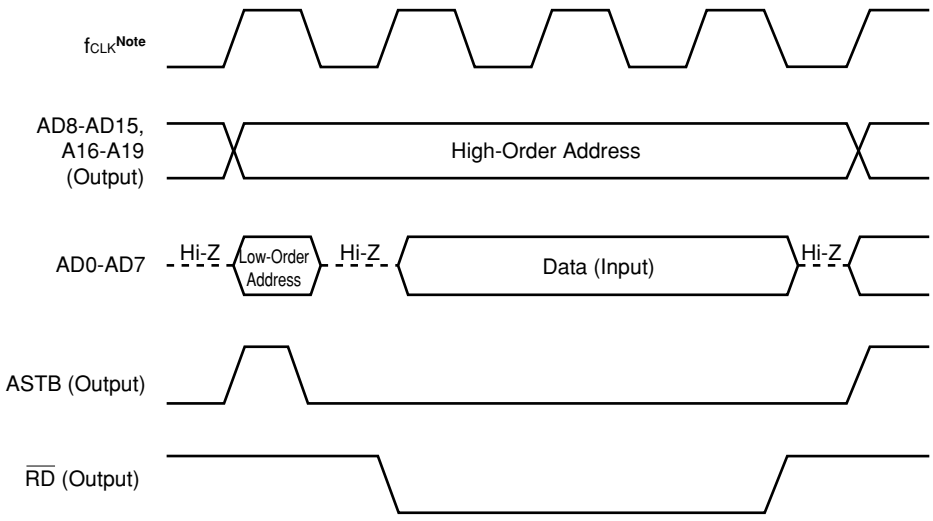


Figure 15-15. Read Timing of Access Wait Function (1/2)

(a) 0 wait cycles set



(b) 1 wait cycle set

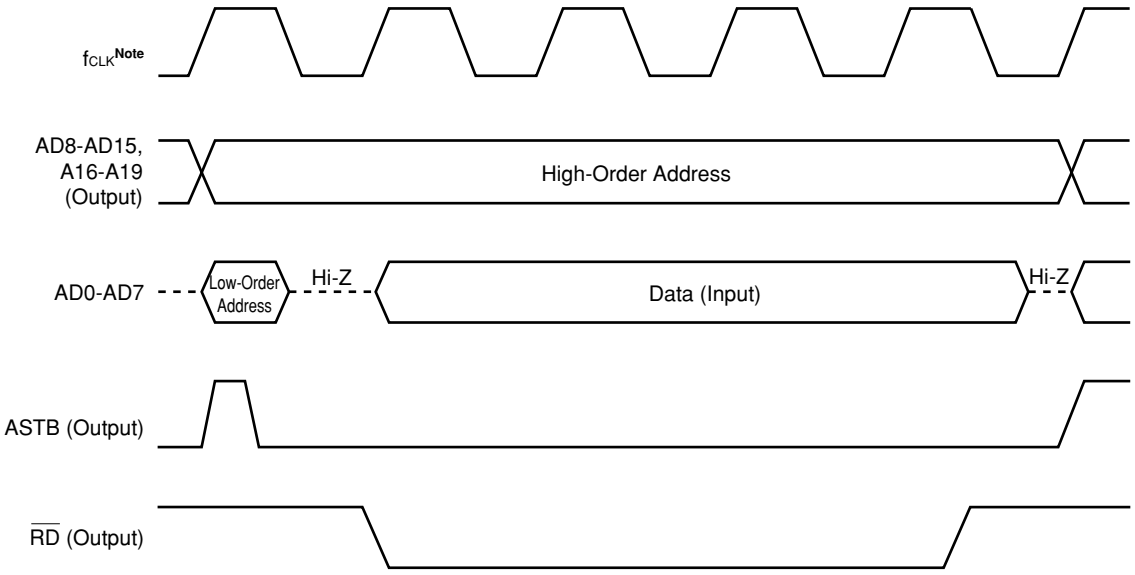


Note f_{CLK}: Internal system clock frequency. This signal is only present inside the μPD784054.

Remark The above figure is an example of the 8-bit bus.

Figure 15-15. Read Timing of Access Wait Function (2/2)

(c) 2 wait cycles set

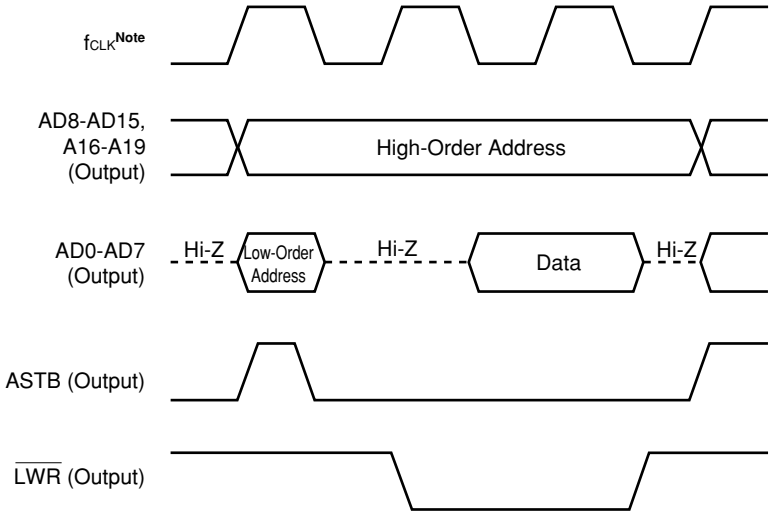


Note f_{CLK}: Internal system clock frequency. This signal is only present inside the μPD784054.

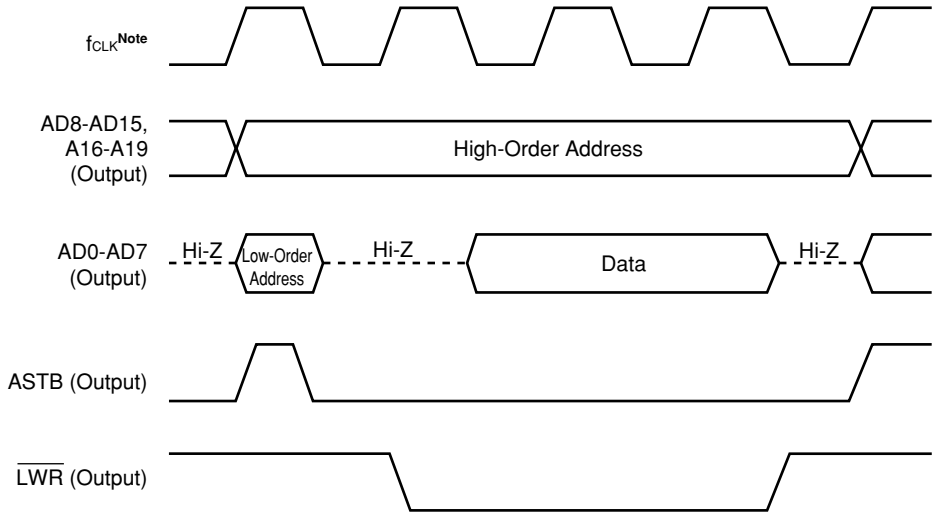
Remark The above figure is an example of the 8-bit bus.

Figure 15-16. Write Timing of Access Wait Function (1/2)

(a) 0 wait cycles set



(b) 1 wait cycle set

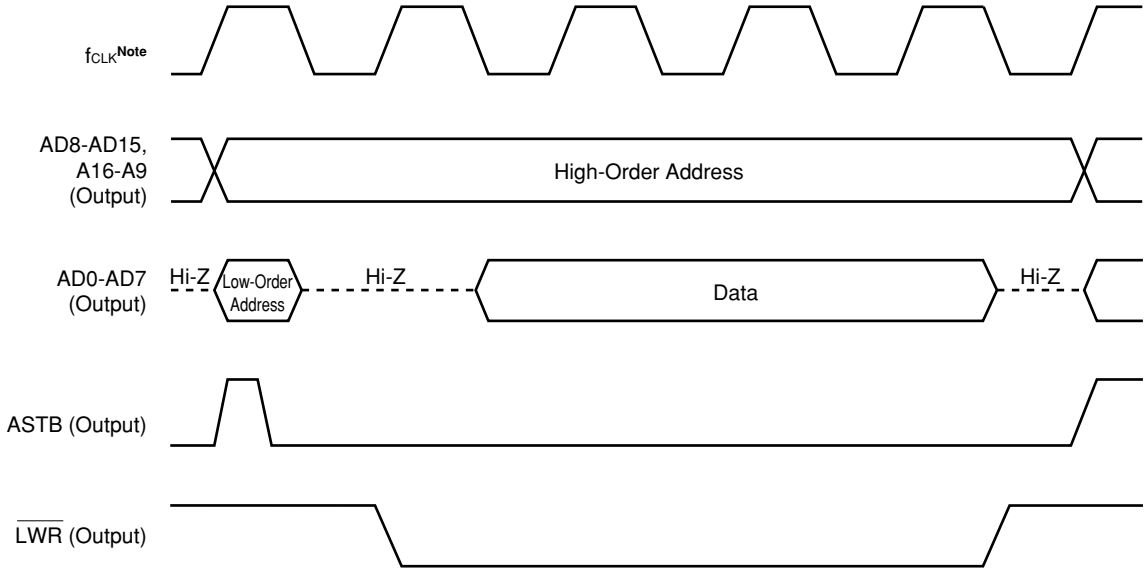


Note fCLK: Internal system clock frequency. This signal is only present inside the μPD784054.

Remark The above figure is an example of the 8-bit bus.

Figure 15-16. Write Timing of Access Wait Function (2/2)

(c) 2 wait cycles set

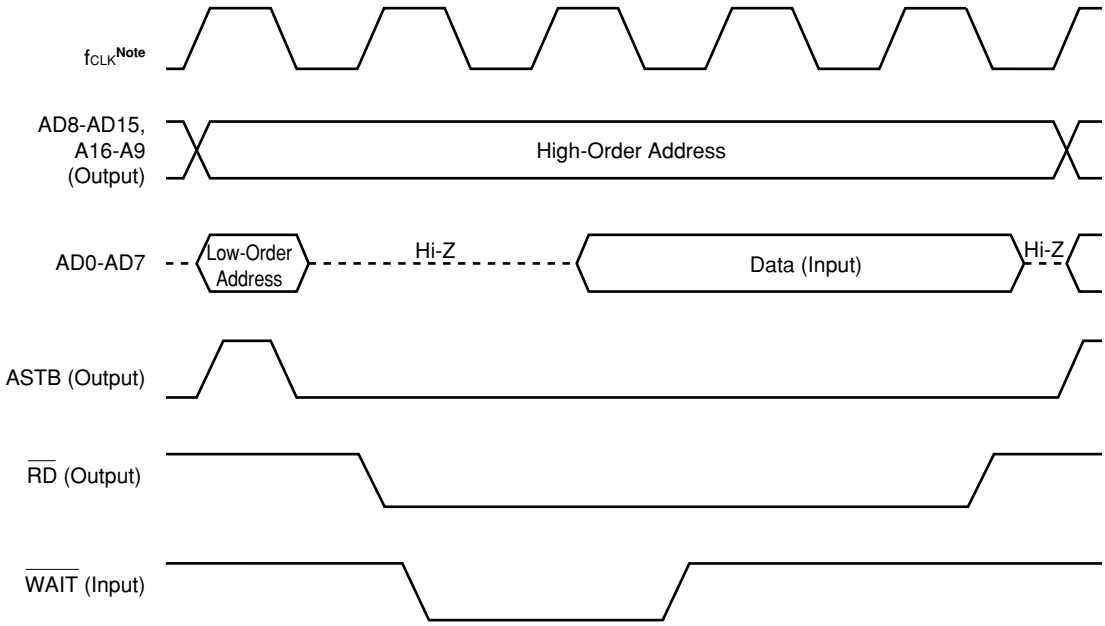


Note f_{CLK}: Internal system clock frequency. This signal is only present inside the μPD784054.

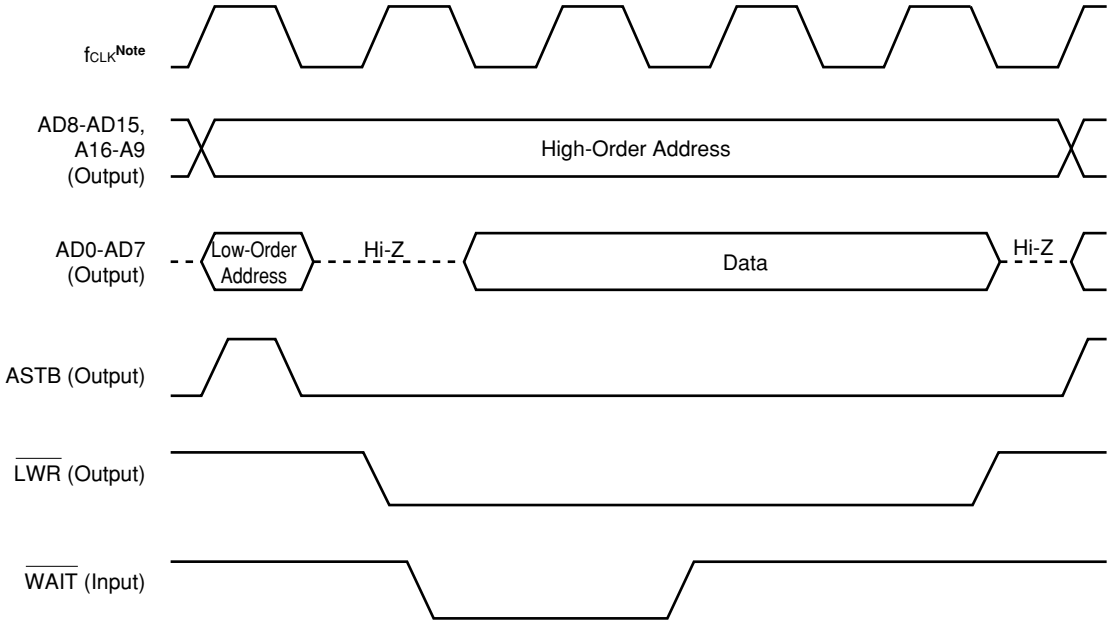
Remark The above figure is an example of the 8-bit bus.

Figure 15-17. Timing with External Wait Signal

(a) Read timing



(b) Write timing



Note f_{CLK}: Internal system clock frequency. This signal is only present inside the μPD784054.

Remark The above figure is an example of the 8-bit bus.

15.3 Bus Sizing Function

The μPD784054 has a bus sizing function that changes the external data bus width between 8 bits and 16 bits when an external device is connected. By using this function, the 1M-byte memory space can be divided by eight, and the external bus width can be specified in each memory space by using the bus width specification register (BW).

15.3.1 Bus width specification register (BW)

BW is a 16-bit register that specifies the bus width when an external device is connected.

This register cannot be accessed in 8-bit units. Be sure to access it by using a 16-bit data manipulation instruction. Figure 15-18 shows the format of BW

The value of BW differs depending on the setting of the BWD pin after $\overline{\text{RESET}}$ is input. When BWD = 0, the value of BW is 0000H; when BWD = 1, it is 00FFH.

Figure 15-18. Format of Bus Width Specification Register (BW)

Address : 0FFCAH On reset : **Note** R/W

	15	14	13	12	11	10	9	8
BW	0	0	0	0	0	0	0	0
	7	6	5	4	3	2	1	0
	BW7	BW6	BW5	BW4	BW3	BW2	BW1	BW0

Valid Address	BW7	Specifies external data bus width
080000H-	0	8-bit bus
0FFFFFH	1	16-bit bus

Valid Address	BW6	Specifies external data bus width
040000H-	0	8-bit bus
07FFFFH	1	16-bit bus

Valid Address	BW5	Specifies external data bus width
020000H-	0	8-bit bus
03FFFFH	1	16-bit bus

Valid Address	BW4	Specifies external data bus width
010000H-	0	8-bit bus
01FFFFH	1	16-bit bus

Valid Address	BW3	Specifies external data bus width
00C000H-	0	8-bit bus
00FFFFH	1	16-bit bus

Valid Address	BW2	Specifies external data bus width
008000H-	0	8-bit bus
00BFFFH	1	16-bit bus

Valid Address	BW1	Specifies external data bus width
004000H-	0	8-bit bus
007FFFH	1	16-bit bus

Valid Address	BW0	Specifies external data bus width
000000H-	0	8-bit bus
003FFFH	1	16-bit bus

Note The value of this register on reset differs depending on the setting of the BWD pin, as follows:
BWD = 0: 0000H
BWD = 1: 00FFH

15.4 Cautions

(1) No wait cycle is inserted when instructions are fetched from the internal ROM or peripheral RAM area at high speeds.

(2) Do not insert a wait cycle in the internal ROM area by using the $\overline{\text{WAIT}}$ pin.

CHAPTER 16 STANDBY FUNCTION

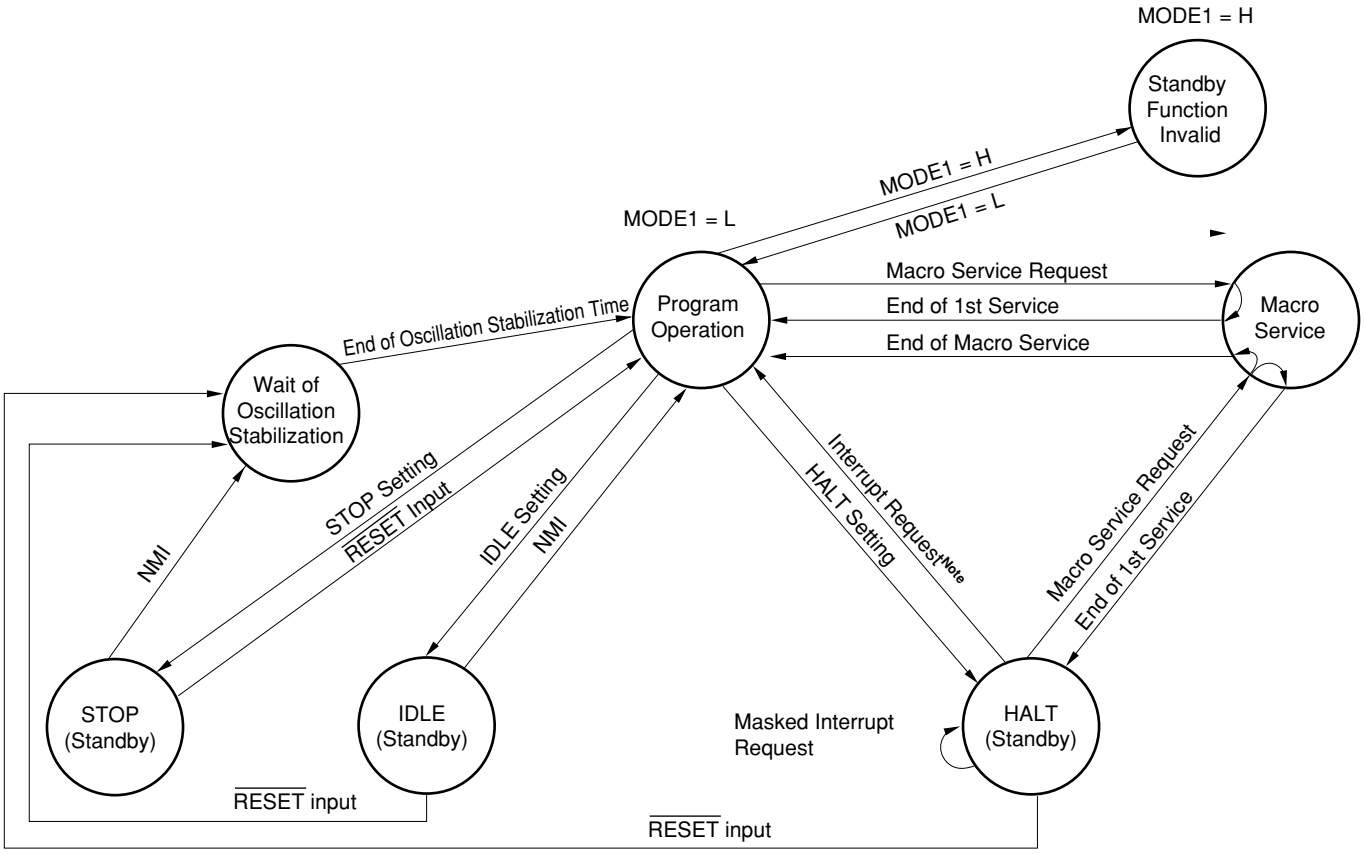
16.1 Configuration and Function

The μPD784054 has a standby function that enables the system power consumption to be reduced. The standby function includes four 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.
- Standby function mode .... In this mode the standby function (HALT/IDLE/STOP mode) can be made invalid by inputting a high level to the MODE 1 pin. It can be used when the standby mode must not be used for some reason in an application.

These modes are set by software. The diagram of the standby mode (STOP/IDLE/HALT mode) transition is shown in Figure 16-1, and the block diagram of the standby function in Figure 16-2.

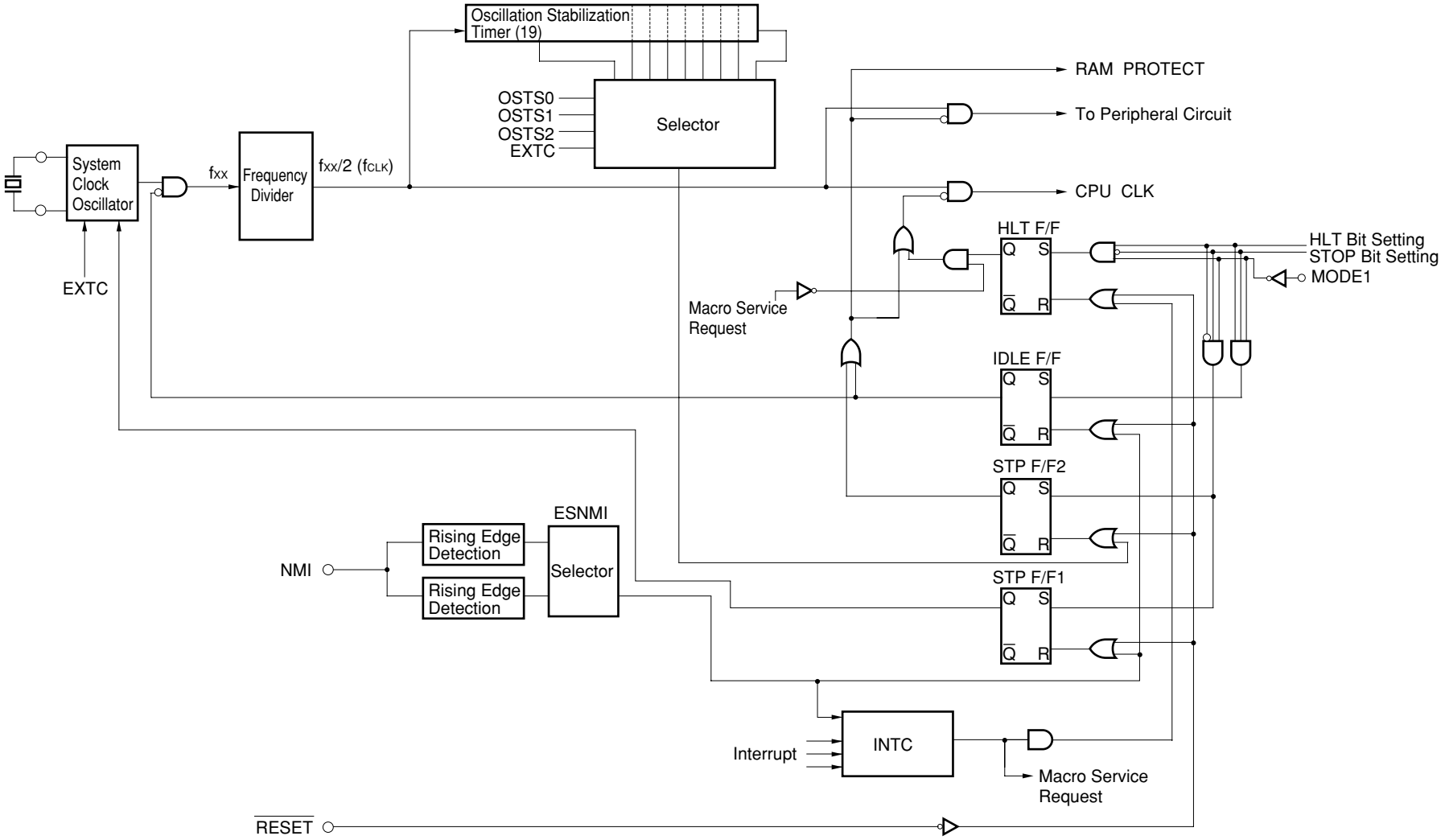
★ **Figure 16-1. Diagram of Standby Mode Transition**



Note Unmasked interrupt request only

Remark Only external input is valid as NMI. The watchdog timer must not be used to release the standby mode (STOP, HALT, or IDLE mode).

Figure 16-2. Block Diagram of Standby Function



16.2 Control Registers

16.2.1 Standby control register (STBC)

The STBC is a register used to control the standby mode.

To prevent entry into the standby mode due to an inadvertent program loop, the STBC register can only be written to with a dedicated instruction. This dedicated instruction, MOV STBC, #byte, has a special code configuration (4 bytes), and a write is only performed if the 3rd and 4th bytes of the operation code are mutual complements.

If the 3rd and 4th bytes of the operation code are not mutual 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 Electronics assembler, RA78K4, only the correct dedicated instruction is generated when MOV STBC, #byte is written), system initialization should be performed by the program.

Other write instructions (MOV STBC, A, AND STBC, #byte, SET1 STBC.7, etc.) are ignored and do not perform any operation. That is, a write is not performed to the STBC, and an interrupt such as an operand error interrupt is not generated.

The STBC can be read at any time by a data transfer instruction.

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

The format of the STBC is shown in Figure 16-3.

Figure 16-3. Standby Control Register (STBC) Format

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

	7	6	5	4	3	2	1	0
STBC	0	0	1	1	0	0	STP	HLT

STP	HLT	Controls CPU Operation Control
0	0	Normal operating mode
0	1	HALT mode
1	0	STOP mode
1	1	IDLE mode

Caution If the STOP mode is used when using external clock input, the EXTC bit of the oscillation stabilization time specification register (OSTS) must be set (1) before setting STOP mode. If the STOP mode is used with the EXTC bit cleared (0) when using external clock input, the μPD784054 may suffer damage or reduced reliability.

When setting the EXTC bit of OSTS to 1, be sure to input a clock in phase reverse to that of the clock input to the X1 pin, to the X2 pin (refer to 4.3.1 Clock oscillator).

16.2.2 Oscillation stabilization time specification register (OSTS)

The OSTS specifies the oscillator operation and the oscillation stabilization time when STOP mode is released. Set the state of the clock oscillator operation to the EXTC bit of the OSTS. STOP mode can be set when external clock input is used only when the EXTC bit is set (1).

Bits OSTS0 to OSTS2 of the OSTS select the oscillation stabilization time when STOP mode is released. In general, an oscillation stabilization time of at least 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.

The OSTS can be read/written only with an 8-bit manipulation instruction.

$\overline{\text{RESET}}$ input clears the OSTS register to 00H.

The format of the OSTS is shown in Figure 16-4.

Figure 16-4. Format of Oscillation Stabilization Time Specification Register (OSTS)

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

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

EXTC	Selects External Clock
0	X2 pin is open when crystal/ceramic oscillation is used or when external clock is used.
1	Input signal in reverse phase to that input to X1 pin to X2 pin when external clock is used.

(f_{CLK} = 16 MHz)

EXTC	OSTS2	OSTS1	OSTS0	Selects Oscillation Stabilization Time
0	0	0	0	$2^{19}/f_{CLK}$ (32.8 ms)
0	0	0	1	$2^{18}/f_{CLK}$ (16.4 ms)
0	0	1	0	$2^{17}/f_{CLK}$ (8.19 ms)
0	0	1	1	$2^{16}/f_{CLK}$ (4.10 ms)
0	1	0	0	$2^{15}/f_{CLK}$ (2.05 ms)
0	1	0	1	$2^{14}/f_{CLK}$ (1.02 ms)
0	1	1	0	$2^{13}/f_{CLK}$ (512 μs)
0	1	1	1	$2^{12}/f_{CLK}$ (256 μs)
1	×	×	×	$2^{8}/f_{CLK}$ (16 μs)

Remark f_{CLK} : internal system clock
× : don't care

Cautions 1. When crystal/ceramic oscillation is used, the EXTC bit of the oscillation stabilization time specification register (OSTS) must be cleared (0) before use. If the EXTC bit is set (1), oscillation will stop.

2. If the STOP mode is used when using external clock input, the EXTC bit must be set (1) before setting STOP mode. If the STOP mode is used with the EXTC bit cleared (0) the μPD784054 may suffer damage or reduced reliability.

When setting the EXTC bit of OSTS to 1, be sure to input a clock in phase reverse to that of the clock input to the X1 pin, to the X2 pin (refer to 4.3.1 Clock oscillator).

16.3 HALT Mode

16.3.1 HALT mode setting and operating states

The HALT mode is selected by setting (1) the HLT bit of the standby control (STBC) register or clearing (0) the STP bit.

The only writes that can be performed on the STBC are 8-bit data writes by means of a dedicated instruction. HALT mode setting is therefore performed by means of the “MOV STBC, #byte” instruction.

Caution If a condition that releases the HALT mode comes into effect when the HALT mode is being set, the HALT mode is not entered, and the next instruction is executed, or a branch to a vectored interrupt service program is performed. Before this branch execution, the instructions after the HALT mode setting may be executed for 6 clocks. After restoring from the interrupt service, to execute an instruction after setting the HALT mode, insert three NOP instructions before the instruction. To be sure to set the HALT mode, take the necessary precautions such as clearing the interrupt request before setting the HALT mode.

Table 16-1. Operating States in HALT Mode

Clock oscillator		Operating
Internal system clock		Operating
CPU		Operation stopped[Note 1]
I/O lines		Retain state prior to HALT mode setting
Peripheral functions		Continue operating
Internal RAM		Retained
Bus lines	AD0 to AD7	High-impedance
	AD8 to AD15	Retained[Note 2]
	A16 to A19	
$\overline{RD}$, $\overline{LWR}$, $\overline{HWR}$ output		High level
ASTB output		Low level

Notes 1. Macro service processing is executed.

2. If the fetch address is in external memory with 16-bit bus width, AD8 to AD15 are made high-impedance after the interrupt processing of the macro service is executed.

16.3.2 HALT mode release

HALT mode can be released by the following three sources.

- Non-maskable interrupt request
- Maskable interrupt request (vectored interrupt/context switching/macro service)
- $\overline{RESET}$ input

Release sources and an outline of operations after release are shown in Table 16-2.

Table 16-2. HALT Mode Release and Operations after Release

Release Source	MK[Note 1]	IE[Note 2]	State on Release	Operation after Release
Non-maskable interrupt request (NMI pin input only. Excluding watchdog timer[Note 5].)	×	×	• Non-maskable interrupt service program not being executed • Low-priority non-maskable interrupt service program being executed	Interrupt request acknowledgment
			• Service program for same request being executed • High-priority non-maskable interrupt service program being executed	Execution of instruction after MOV STBC/ #byte instruction (interrupt request that released HALT mode is held pending[Note 3])
Maskable interrupt request (excluding macro service request)	0	1	• Interrupt service program not being executed • Low-priority maskable interrupt service program being executed • PRSL bit[Note 4] cleared (0) during execution of priority level 3 interrupt service program	Interrupt request acknowledgment
			• Same-priority maskable interrupt service program being executed (If PRSL bit[Note 4] is cleared (0), excluding execution of priority level 3 interrupt service program) • High-priority interrupt service program being executed	Execution of instruction after MOV STBC/ #byte instruction (interrupt request that released HALT mode is held pending[Note 3])
	0	0	—	
	1	×	—	HALT mode maintained
Macro service request	0	×	—	Macro service processing execution End condition not established → HALT mode again End condition established → Same as release by maskable interrupt request
	1	×	—	HALT mode maintained
$\overline{\text{RESET}}$ input	×	×	—	Normal reset operation

Notes 1. Interrupt mask bit in individual interrupt request source

2. Interrupt enable flag in program status word (PSW)

3. Pending interrupt requests are acknowledged when acknowledgment becomes possible.

4. Bit in interrupt mode control register (IMC)

5. The HALT mode cannot be released by the watchdog timer.

(1) Release by non-maskable interrupt

When a non-maskable interrupt is generated, the μPD784046 is released from HALT mode irrespective of whether the interrupt acknowledgment enabled state (EI) or disabled state (DI) is in effect.

When the μPD784054 is released from HALT mode, if the non-maskable interrupt that released HALT mode can be acknowledged, acknowledgment of that non-maskable interrupt is performed and a branch is made to the service program. If the interrupt cannot be acknowledged, the instruction following the instruction that set the HALT mode (the MOV STBC, #byte instruction) is executed, and the non-maskable interrupt that released the HALT mode is acknowledged when acknowledgment becomes possible. Refer to **14.6 Non-Maskable Interrupt Acknowledgment Operation** for details of non-maskable interrupt acknowledgment.

(2) Release by maskable interrupt request

HALT mode release by a maskable interrupt request can only be performed by an interrupt for which the interrupt mask flag is 0.

When HALT mode is released, if an interrupt can be acknowledged when the interrupt request enable flag (IE) is set (1), a branch is made to the interrupt service program. If the interrupt cannot be acknowledged and if the IE flag is cleared (0), execution is resumed from the instruction following the instruction that set the HALT mode. Refer to **14.7 Maskable Interrupt Acknowledgment Operation** for details of interrupt acknowledgment.

With macro service, HALT mode is released temporarily, service is performed once, then HALT mode is restored. When macro service has been performed the specified number of times, HALT mode is released. The operation after release in this case is the same as for release by a maskable interrupt described earlier.

(3) Release by $\overline{\text{RESET}}$ 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.

16.4 STOP Mode

16.4.1 STOP mode setting and operating states

The STOP mode is selected by setting (1) the STP bit of the standby control register (STBC) register or clearing (0) the HLT bit.

The only writes that can be performed on the STBC register are 8-bit data writes by means of a dedicated instruction. STOP mode setting is therefore performed by means of the “MOV STBC, #byte” instruction,

Caution If a condition that releases the HALT mode comes into effect when the STOP mode is being set (refer to 16.3.2 HALT mode release), the STOP mode is not entered, and the next instruction is executed, or a branch to a vectored interrupt service program is performed. Before this branch execution, the instructions after the STOP mode setting may be executed for 6 clocks. After restoring from the interrupt service, to execute an instruction after setting the STOP mode, insert three NOP instructions before the instruction. To be sure to set the STOP mode, take the necessary precautions such as clearing the interrupt request before setting the STOP mode.

Table 16-3. Operating States in STOP Mode

Clock oscillator		Oscillation stopped
Internal system clock		Stopped
CPU		Operation stopped
I/O lines		Retain state prior to STOP mode setting
Peripheral functions		All operation stopped[Note]
Internal RAM		Retained
Bus lines	AD0 to AD15	High-impedance
	A16 to A19	High-impedance
$\overline{RD}$, $\overline{LWR}$, $\overline{HWR}$ output		High-impedance
ASTB output		High-impedance

Note A/D converter operation is stopped, but if the AM0 bit or AM1 bit of the A/D converter mode register (ADM) is set (1), the current consumption does not decrease.

Cautions 1. If the STOP mode is set when the EXTC bit of the oscillation stabilization time specification (OSTS) register is cleared (0), the X1 pin is shorted internally to Vss (GND potential) to suppress clock generator leakage. Therefore, when the STOP mode is used in a system that uses an external clock, the EXTC bit of the OSTS must be set (1). If STOP mode setting is performed in a system to which an external clock is input when the EXTC bit of the OSTS is cleared (0), the μPD784054 may suffer damage or reduced reliability.

When setting the EXTC bit of OSTS to 1, be sure to input a clock in phase reverse to that of the clock input to the X1 pin, to the X2 pin (refer to 4.3.1 Clock oscillator).

2. Stop the A/D converter (by clearing (0) the AM0 and AM1 bits of the A/D converter mode register (ADM)) before setting the STOP mode.

16.4.2 STOP mode release

STOP mode is released by NMI input, INTP4 input, INTP5 input, and $\overline{\text{RESET}}$ input.

Table 16-4. STOP Mode Release and Operations after Release

Release Source	State after Release	Operation after Release
NMI pin input	• Non-maskable interrupt service program not being executed • Low-priority non-maskable interrupt service program being executed	Interrupt request acknowledgment
	• NMI pin input service program being executed • High-priority non-maskable interrupt service program being executed	Execution of instruction after MOV STBC/ #byte instruction (interrupt request that released STOP mode is held pending[Note])
$\overline{\text{RESET}}$ input	—	Normal reset operation

Note Pending interrupt requests are acknowledged when acknowledgment becomes possible.

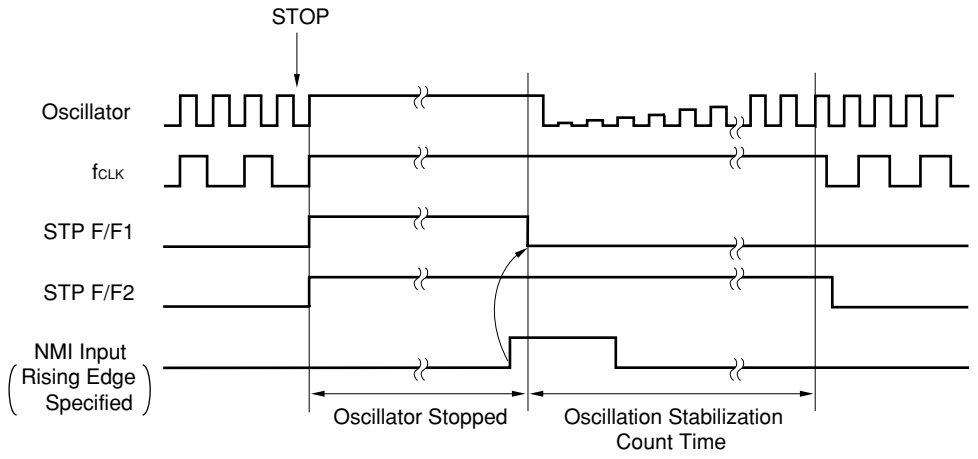
(1) STOP mode release by NMI input

The oscillator resumes oscillation when the valid edge specified by external interrupt mode register 0 (INTM0) is input to the NMI input. STOP mode is released after the oscillation stabilization time specified by the oscillation stabilization time specification register (OSTS) elapses.

When the μPD784054 is released from STOP mode, if a non-maskable interrupt by NMI pin input can be acknowledged, a branch is made to the NMI interrupt service program. If the interrupt cannot be acknowledged (if the STOP mode is set in an NMI interrupt service program, etc.), execution is resumed from the instruction following the instruction that set the STOP mode, and a branch is made to the NMI interrupt service program when acknowledgment becomes possible (by execution of an RETI instruction, etc.).

Refer to **14.6 Non-Maskable Interrupt Acknowledgment Operation** for details of NMI interrupt acknowledgment.

Figure 16-5. STOP Mode Release by NMI Input



(2) STOP mode release by $\overline{\text{RESET}}$ input

When $\overline{\text{RESET}}$ input falls from high to low and the reset state is established, the oscillator resumes oscillation. The oscillation stabilization time should be secured while $\overline{\text{RESET}}$ is active. Thereafter, normal operation is started when $\overline{\text{RESET}}$ rises.

Unlike an ordinary reset operation, data memory retains its contents prior to STOP mode setting.

16.5 IDLE Mode

16.5.1 IDLE mode setting and operating states

The IDLE mode is selected by setting (1) both the STP bit and the HLT bit of the standby control (STBC) register.

The only writes that can be performed on the STBC are 8-bit data writes by means of a dedicated instruction. IDLE mode setting is therefore performed by means of the “MOV STBC, #byte” instruction.

Caution If a condition that releases the HALT mode comes into effect when the IDLE mode is being set (refer to 16.3.2 HALT mode release), the IDLE mode is not entered, and the next instruction is executed, or a branch to a vectored interrupt service program is performed. Before this branch execution, the instructions after the IDLE mode setting may be executed for 6 clocks. After restoring from the interrupt service, to execute an instruction after setting the IDLE mode, insert three NOP instructions before the instruction. To be sure to set the IDLE mode, take the necessary precautions such as clearing the interrupt request before setting the IDLE mode.

Table 16-5. Operating States in IDLE Mode

Clock oscillator		Oscillation continues
Internal system clock		Stopped
CPU		Operation stopped
I/O lines		Retain state prior to IDLE mode setting
Peripheral functions		All operation stopped[Note]
Internal RAM		Retained
Bus lines	AD0 to AD15	High-impedance
	A16 to A19	High-impedance
$\overline{RD}$, $\overline{LWR}$, $\overline{HWR}$ output		High-impedance
ASTB output		High-impedance

Note A/D converter operation is stopped, but if the AM0 bit or AM1 bit of the A/D converter mode register (ADM) is set, the current consumption does not decrease.

Caution Stop the A/D converter (by clearing (0) the AM0 and AM1 bits of the A/D converter mode register (ADM)) before setting the IDLE mode.

16.5.2 IDLE mode release

IDLE mode is released by NMI input, or $\overline{\text{RESET}}$ input.

Table 16-6. IDLE Mode Release and Operations after Release

Release Source	State after Release	Operation after Release
NMI pin input	• Non-maskable interrupt service program not being executed • Low-priority non-maskable interrupt service program being executed	Interrupt request acknowledgment
	• NMI pin input service program being executed • High-priority non-maskable interrupt service program being executed	Execution of instruction after MOV STBC/ #byte instruction (interrupt request that released IDLE mode is held pending[Note])
$\overline{\text{RESET}}$ input	—	Normal reset operation

Note Pending interrupt requests are acknowledged when acknowledgment becomes possible.

(1) IDLE mode release by NMI input

IDLE mode is released when the valid edge specified by external interrupt mode register 0 (INTM0) is input to the NMI input.

When the μPD784054 is released from IDLE mode, if a non-maskable interrupt by NMI pin input can be acknowledged, a branch is made to the NMI interrupt service program. If the interrupt cannot be acknowledged (if the IDLE mode is set in an NMI interrupt service program, etc.), execution is resumed from the instruction following the instruction that set the IDLE mode, and a branch is made to the NMI interrupt service program when acknowledgment becomes possible (by execution of an RETI instruction, etc.).

Refer to **14.6 Non-Maskable Interrupt Acknowledgment Operation** for details of NMI interrupt acknowledgment.

(2) IDLE mode release by $\overline{\text{RESET}}$ input

Normal operation is started when $\overline{\text{RESET}}$ rises after $\overline{\text{RESET}}$ input falls from high to low.

Unlike an ordinary reset operation, data memory retains its contents prior to IDLE mode setting.

Caution When the execution of the IDLE mode instruction contends with the interrupt of release source of the IDLE mode, the STOP mode is released after the STOP mode has been executed, instead of the normal operation where the IDLE mode is released after the IDLE mode has been executed, because of a malfunction of the μPD784054. Therefore, when the IDLE mode is released, the wait operation for the oscillation stabilization time set by the oscillation stabilization time specification register (OSTS) may be executed even though the IDLE mode is set in software (Usually, the μPD784054 does not wait the oscillation stabilization time when the IDLE mode is released.) If there are problems with waiting for the oscillation stabilization time when the IDLE mode is released, set the value of the oscillation stabilization time set by the OSTS as short as possible.

16.6 Check Items When STOP Mode/IDLE Mode Is Used

Check items required to reduce the current consumption when STOP mode/IDLE mode is used are shown below.

(1) Is the output level of each output pin appropriate?

The appropriate output level for each pin varies according to the next-stage circuit. You should select the output level that minimizes the current consumption.

- If high level is output when the input impedance of the next-stage circuit is low, a current will flow from the power supply to the port, resulting in an increased current consumption. This applies when the next-stage circuit is a CMOS IC, etc. When the power supply is off, the input impedance of a CMOS IC is low. In order to suppress the current consumption, or to prevent an adverse effect on the reliability of the CMOS IC, low level should be output. If a high level is output, latchup may result when power is turned on again.

- Depending on the next-stage circuit, inputting low level may increase the current consumption. In this case, high-level or high-impedance output should be used to reduce the current consumption.

- If the next-stage circuit is a CMOS IC, the current consumption of the CMOS IC may increase if the output is made high-impedance when power is supplied to it (the CMOS IC may also be overheated and damaged). In this case you should output an appropriate level, or pull the output high or low with a resistor.

The method of setting the output level depends on the port mode.

- When a port is in control mode, the output level is determined by the status of the 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, its output level can be set easily by changing to port mode.

(2) Is the input pin level appropriate?

The voltage level input to each pin should be in the range between V_{SS} potential and V_{DD} potential. If a voltage outside this range is applied, the current consumption will increase and the reliability of the μPD784054 may be adversely affected.

Also ensure that an intermediate potential is not applied.

(3) Are pull-up resistors necessary?

An unnecessary pull-up resistor will increase the current consumption and cause a latchup of other devices. A mode should be specified in which pull-up resistors are used only for parts that require them.

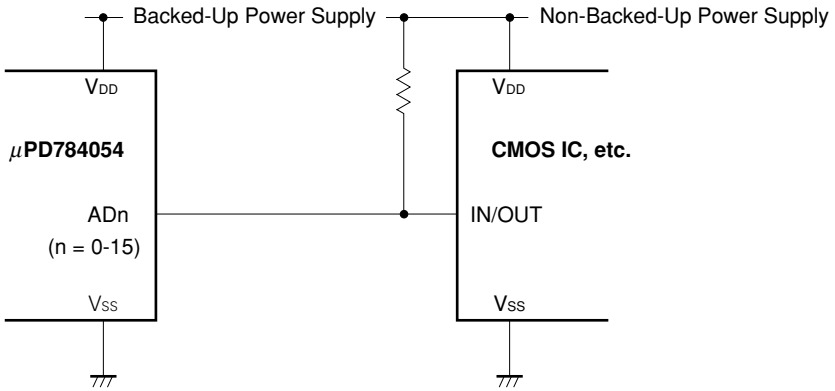
If there is a mixture of parts that do and do not require pull-up resistors, for parts that do, you should connect a pull-up resistor externally and specify a mode in which the on-chip pull-up resistor is not used.

(4) Is processing of the address bus, address/data bus, etc., appropriate?

In STOP mode and IDLE mode, the address bus, address/data bus, $\overline{RD}$ and $\overline{LWR}$, $\overline{HWR}$ pins become high-impedance. Normally, these pins are pulled high with a pull-up resistor. If this pull-up resistor is connected to the backed-up power supply, then if the input impedance of circuitry connected to the non-backed-up power supply is low, a current will flow through the pull-up resistor, and the current consumption will increase. Therefore, the pull-up resistor should be connected to the non-backed-up power supply side as shown in Figure 16-6.

Also, in STOP mode and IDLE mode the ASTB pin also becomes high impedance. Countermeasures should be taken with reference to the points noted in (1).

Figure 16-6. Example of Address/Data Bus Processing



★ **(5) A/D converter**

The current flowing to the AV_{DD}, AV_{REF1} pins can be reduced by clearing (0) the AM0 and AM1 bits of the A/D converter mode register (ADM).

Make sure that the AV_{DD} pin is not at the same potential as the V_{DD} pin. Unless power is supplied to the AV_{DD} pin in the STOP mode, not only does the current consumption increase, but the reliability is also affected.

16.7 Cautions

(1) If a condition that releases the HALT mode comes into effect when the HALT/STOP/IDLE mode (hereafter referred to as standby mode) is being set (refer to **16.3.2 HALT mode release**), the standby mode is not entered, and the next instruction is executed, or a branch to a vectored interrupt service program is performed. Before this branch execution, the instructions after the standby mode setting may be executed for 6 clocks. After restoring from the interrupt service, to execute an instruction after setting the standby mode, insert three NOP instructions before the instruction. To be sure to set the standby mode, take the necessary precautions such as clearing the interrupt request before setting the standby mode.

(2) When crystal/ceramic oscillation is used, the EXTC bit must be cleared (0) before use. If the EXTC bit is set (1), oscillation will stop.

(3) If the STOP mode is set when the EXTC bit of the oscillation stabilization time specification (OSTS) register is cleared (0), the X1 pin is shorted internally to V_{SS} (GND potential) to suppress clock generator leakage. Therefore, when the STOP mode is used in a system that uses an external clock, the EXTC bit of the OSTS must be set (1). If STOP mode setting is performed in a system to which an external clock is input when the EXTC bit of the OSTS is cleared (0), the μPD784054 may suffer damage or reduced reliability.
When setting the EXTC bit of OSTS to 1, be sure to input a clock in phase reverse to that of the clock input to the X1 pin, to the X2 pin (refer to **4.3.1 Clock oscillator**).

(4) Stop the A/D converter (by clearing (0) the AM0 and AM1 bits of the A/D converter mode register (ADM)) before setting the STOP or IDLE mode.

(5) When the execution of the IDLE mode instruction contends with the interrupt of release source of the IDLE mode, the STOP mode is released after the STOP mode has been executed, instead of the normal operation where the IDLE mode is released after the IDLE mode has been executed, because of a malfunction of the μPD784054. Therefore, when the IDLE mode is released, the wait operation for the oscillation stabilization time set by the oscillation stabilization time specification register (OSTS) may be executed even though the IDLE mode is set in software (Usually, the μPD784054 does not wait the oscillation stabilization time when the IDLE mode is released.) If there are problems with waiting for the oscillation stabilization time when the IDLE mode is released, set the value of the oscillation stabilization time set by the OSTS as short as possible.

CHAPTER 17 RESET FUNCTION

17.1 Reset Function

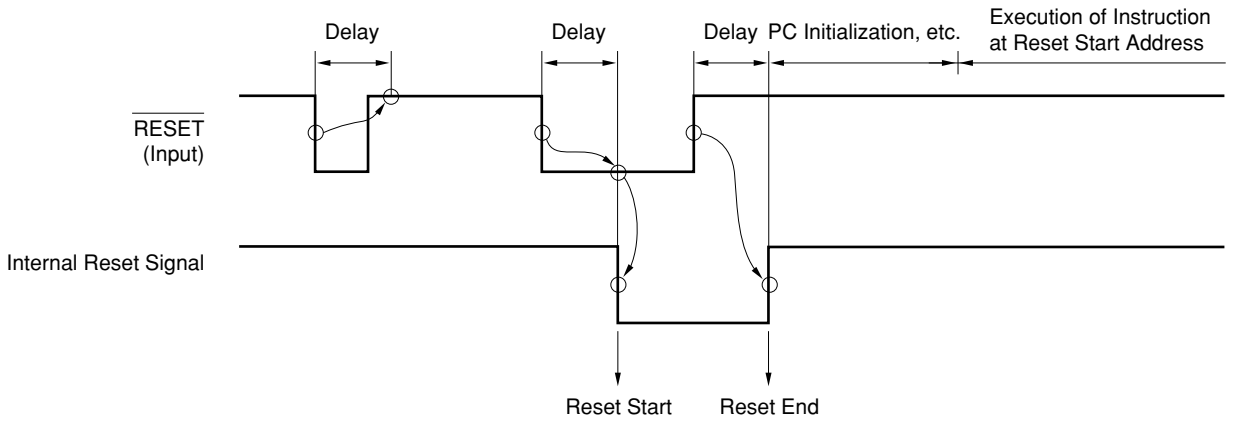
When low level is input to the $\overline{\text{RESET}}$ input pin, a system reset is affected, the various hardware units are set to the states shown in Table 17-2, and all pins except the power supply pins and the X1 and X2 CLKOUT pins are placed in the high-impedance state. Table 17-1 shows the pin statuses on reset and after reset release.

When the $\overline{\text{RESET}}$ input changes from low to high level, the reset state is released, the contents of address 00000H of the reset vector table are set in bits 0 to 7 of the program counter (PC), the contents of address 00001H in bits 8 to 15, and 0000B in bits 16 to 19, a branch is made, and program execution is started at the branch destination address. A reset start can therefore be performed from any address in the base area.

The contents of the various registers should be initialized as required in the program in the base area.

To prevent misoperation due to noise, the $\overline{\text{RESET}}$ input pin incorporates an analog delay noise elimination circuit (refer to **Figure 17-1**).

Figure 17-1. Acknowledgment of Reset Signal



In a reset operation upon powering on and STOP mode release by reset, the $\overline{\text{RESET}}$ signal must be kept active until the oscillation stabilization time has elapsed (approx. 40 ms, depending on the resonator used).

Figure 17-2. Power-On Reset Operation

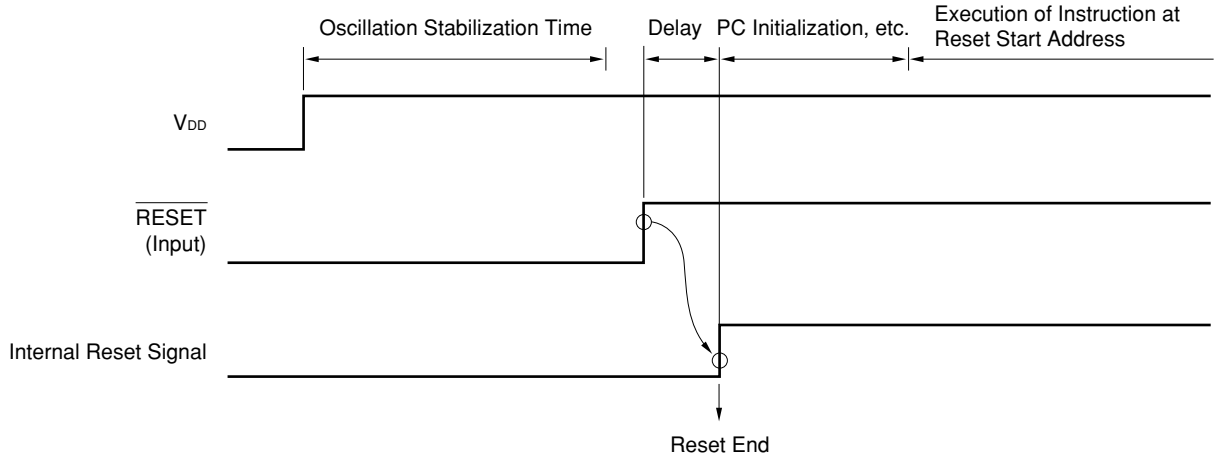


Table 17-1. Pin Status during Reset Input and after Clearing Reset

Pin Name	I/O	During Reset	Immediately after Clearing Reset
P00-P03	I/O	Hi-Z	Hi-Z (input port mode)
P10-P12			
P20	Input		Hi-Z (input port)
P21-P27	I/O		Hi-Z (input port mode)
P30-P37			
P40-P47			
P50-P57			
P60-P63			
P70-P77	Input		Hi-Z (input port)
P80-P87			
P90-P94	I/O		Hi-Z (input port mode)
CLKOUT	Output	Clock output	Clock output

Figure 17-3. Timing on Reset Input

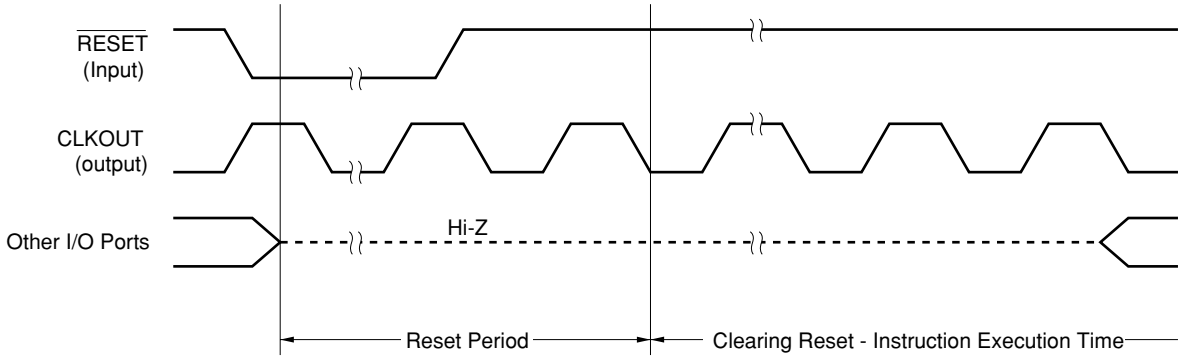


Table 17-2. State of Hardware after Reset (1/2)

Hardware		State after Reset
Program counter (PC)		Contents of reset vector table (0000H, 0001H) are set
Stack pointer (SP)		Undefined[Note]
Program status word (PSW)		02H
On-chip RAM	Data memory	Undefined[Note]
	General-purpose register	
Port	Port 0 to port 9	Undefined (high impedance)
	Mode registers (PM0 to PM6, PM9)	FFH
	Mode control registers (PMC2, PMC3, PMC9)	00H
	Port read control register (PRDC)	
	Pull-up resistor option register (PUOL, PUOH)	
Timer/counter	Timer registers (TM0, TM1, TM4)	0000H
	Capture/compare registers (CC00 to CC03)	Undefined
	Compare registers (CM10, CM11, CM40, CM41)	
	Timer unit mode registers (TUM0)	00H
	Timer mode control registers (TMC, TMC4)	
	Timer output control registers (TOC0, TOC1)	
	Prescaler mode registers (PRM, PRM4)	
	Noise protection control register (NPC)	
	Interrupt valid edge flag registers (IEF1, IEF2)	Undefined
Watchdog timer mode register (WDM)		00H
A/D converter	A/D converter mode register (ADM)	
	A/D conversion result registers (ADCR0 to ADCR7, ADCR0H to ADCR7H)	Undefined
Serial interface	Asynchronous serial interface mode registers (ASIM, ASIM2)	00H
	Asynchronous serial interface status registers (ASIS, ASIS2)	
	Serial receive buffers (RXB, RXB2)	Undefined
	Serial transmit shift registers (TXS, TXS2)	
	Clocked serial interface mode registers (CSIM1, CSIM2)	00H
	Serial shift registers (SIO1, SIO2)	Undefined
	Baud rate generator control registers (BRGC, BRGC2)	00H
External interrupt mode registers (INTM0, INTM1)		

Note If the HALT, STOP, or IDLE mode is released by using the $\overline{\text{RESET}}$ input, the values immediately before each mode has been set are retained.

Table 17-2. State of Hardware after Reset (2/2)

Hardware			State after Reset
Interrupt	Interrupt control registers (OVIC0, OVIC1, OVIC4, PIC0 to PIC6, CMIC10, CMIC11, CMIC40, CMIC41, SERIC, SRIC, CSIIC1, STIC, SERIC2, SRIC2, CSIIC2, STIC2, ADIC)		43H
	Interrupt mask registers	MK0, MK1	FFFFH
		MK0L, MK0H, MK1L, MK1H	FFH
	Interrupt mode control register (IMC)		80H
	In-service priority register (ISPR)		00H
Memory extension mode register (MM)			20H
Programmable wait control register		PWC1	AAH
		PWC2	AAAAH
Bus width specification register (BW)			0000H (BWD = 0) 00FFH (BWD = 1)
Standby control register (STBC)			30H
Oscillation stabilization time specification register (OSTS)			00H
Internal memory size switching register (IMS)			CDH

17.2 Caution

Reset input when powering on must remain at the low level until oscillation stabilizes after the supply voltage has reached the prescribed voltage.

CHAPTER 18 μPD78F4046

18.1 Memory Mapping of μPD78F4046

The μPD78F4046 has 64K bytes of flash memory and 2048 bytes of internal RAM.

The μPD78F4046 has a function to not use part of the internal memory (memory size select function). This function is effected by software.

The memory size is changed by using the internal memory size select register (IMS).

This register can be read or written by using an 8-bit manipulation instruction. Data can be only written to the IMS of the μPD78F4046, however. The IMS of the μPD784054 retains the value at reset even if data is written to it.

Therefore, the value of the IMS at $\overline{\text{RESET}}$ differs depending on the model. In the case of the μPD784054, it is CDH. The value of the IMS of the μPD78F4046 is set to DEH at $\overline{\text{RESET}}$.

Figure 18-1. Format of Internal Memory Size Select Register (IMS)

Address : 0FFFCH On reset : **Note**

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

ROM1	ROM0	Selects Internal ROM Capacity	
		μPD784054	μPD78F4046
0	0	32K bytes	32K bytes
0	1	Invalid	64K bytes
Other		Setting prohibited	

RAM1	RAM0	Selects Peripheral RAM Capacity	
		μPD784054	μPD78F4046
0	1	512 bytes	768 bytes
1	0	Invalid	1.5K bytes
Other		Setting prohibited	

Note The value at reset differs depending on the model.
μPD784054 : CDH
μPD78F4046 : DEH

Cautions 1. Writing to the internal memory size select register (IMS) is valid only with the μPD78F4046. The IMS of the μPD784054 holds the value at $\overline{\text{RESET}}$ even if data is written to it.

2. To develop a program for the μPD784054 using the μPD78F4046, set the value of the IMS to CDH. When the value of the IMS is set to CDH, the peripheral RAM capacity of the μPD78F4046 is 768 bytes, but the peripheral RAM capacity of the μPD784054 is 512 bytes. When using a mask ROM, therefore, exercise care that addresses 0FA00H to 0FAFFH of the peripheral RAM area of the μPD78F4046 are not used (when the LOCATION 0H instruction is executed).

18.2 Programming μPD78F4046

★ The flash memory can be written with the μPD78F4046 mounted on the target system (on-board). Connect a dedicated flash programmer (Flashpro II (part number: FL-PR2)/Flashpro III (part number: FL-PR3, PG-FP3)) to the host machine and target system to write the flash memory.

The flash memory can also be written using the adapter for writing flash memory connected to Flashpro II/Flashpro III.

Remark FL-PR2 and FL-PR3 are products of Naito Densei Machida Mfg. Co., Ltd.

18.2.1 Selecting communication mode

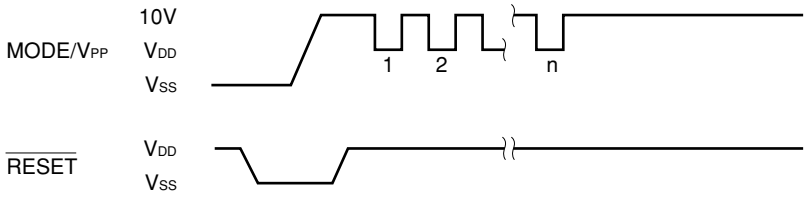
The flash memory is written by using a Flashpro II/Flashpro III and by means of serial communication. Select a communication mode from those listed in Table 18-1. To select a communication mode, the format shown in Figure 18-2 is used. Each communication mode is selected by the number of V_{PP} pulses shown in Table 18-1.

Table 18-1. Communication Modes

Communication Mode	Number of Channels	Pins Used	Number of V_{PP} Pulses
3-wire serial I/O	2	P34/ASCK/$\overline{SCK1}$ P33/TxD/SO1 P32/RxD/SI1	0
		P37/ASCK2/$\overline{SCK2}$ P36/TxD2/SO2 P35/RxD2/SI2	1
UART	2	P33/TxD/SO1 P32/RxD/SI1	8
		P36/TxD2/SO2 P35/RxD2/SI2	9

Caution Be sure to select the communication mode with the number of V_{PP} pulses as shown in Table 18-1.

Figure 18-2. Selecting Format of Communication Mode



18.2.2 Function of flash memory programming

By transmitting/receiving commands and data in the selected communication mode, operations such as writing to the flash memory are performed. Table 18-2 shows the major functions of flash memory programming.

Table 18-2. Major Functions of Flash Memory Programming.

Function	Description
Batch erase	Erases all contents of memory.
Block erase	Erases specified memory block with one block consisting of 16K bytes.
Batch blank check	Checks erased state of entire memory.
Block blank check	Checks erased state of specified block.
Data write	Writes to flash memory based on write start address and number of data written (number of bytes).
Batch verify	Compares all contents of memory with input data.
Block verify	Compares contents of specified memory block with input data.

18.2.3 Connecting Flashpro II/Flashpro III

How the Flashpro II/Flashpro III is connected to the μPD78F4046 differs to the μPD78F4046 depending on the communication mode (3-wire serial I/O or UART). Figures 18-3 and 18-4 show the connections in the respective modes.

Figure 18-3. Connecting Flashpro II/Flashpro III in 3-Wire Serial I/O Mode

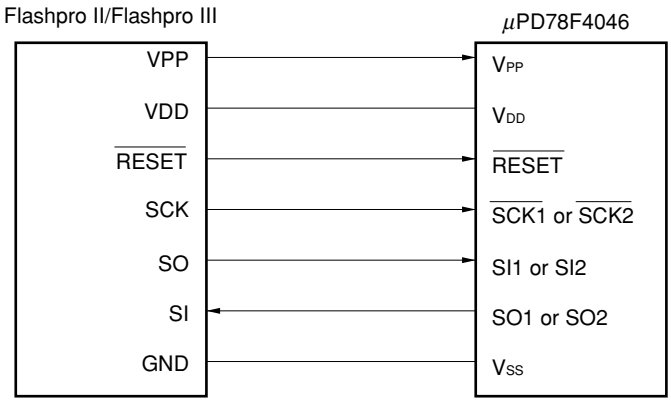
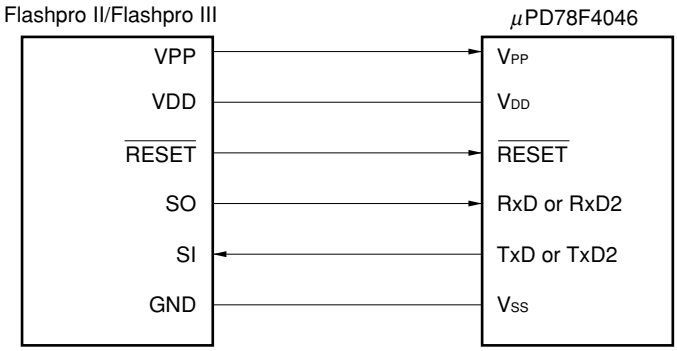


Figure 18-4. Connecting Flashpro II/Flashpro III in UART Mode



★ 18.3 Cautions

(1) Writing to the internal memory size select register (IMS) is valid only with the μPD78F4046. The IMS of the μPD784054 holds the value at $\overline{\text{RESET}}$ even if data is written to it.

(2) To develop a program for the μPD784054 using the μPD78F4046, set the value of the IMS to CDH. When the value of the IMS is set to CDH, the peripheral RAM capacity of the μPD78F4046 is 768 bytes, but the peripheral RAM capacity of the μPD784054 is 512 bytes. When using a mask ROM, therefore, exercise care that addresses 0FA00H to 0FAFFH of the peripheral RAM area of the μPD78F4046 are not used (when the LOCATION 0H instruction is executed).

(3) Number of rewrites
Number of guaranteed rewrites: 10
Perform erasure and writing in area mode or chip mode. Block mode cannot be used to write only a specific block.

(4) Operating ambient temperature
Operating ambient temperature: T_A = –10 to +70°C
However, the temperature during rewrite is T_{PRG} = +10 to +40°C.

(5) Use of pre-writing

Pre-writing is required before erasure.

When Flashpro II (Ver. 2.50 or later) or Flashpro III (PG-FP3 Ver. 3.040 or later) is used, use of the pre-write function can automatically be set by loading the parameter file.

(6) Use of ECC function

Write ECC data to the ECC area in the on-chip flash memory.

Convert the HEX file into a HEX file with ECC using the ECC generator included in the assembler package (PC version Ver.1.20 or later). Then download this HEX file with ECC to Flashpro II or Flashpro III and execute writing.

[How to create ECC data]

<1> Prepare a HEX file created by the object converter in the assembler package.

<2> Convert the HEX file into a HEX file with ECC (program data + ECC data) using the ECC generator (eccgen.exe) included in the assembler package.

Example When converting the file “file.hex” into the HEX file with ECC “file_ec.hex”

```
eccgen file.hex -ofile_ec.hex -a0ffffh, 1000h, 14000h, 14004h
```

(7) How to set and write using Flashpro II or Flashpro III

Perform pre-writing and write to ECC using Flashpro II or Flashpro III.

[How to write]

<1> Download the HEX file with ECC to Flashpro II or Flashpro III.

<2> Set CHIP mode and execute writing using the E.P.V button.

Do not use the Program command; otherwise ECC may not be written.

When Flashpro II Ver. 2.50 or earlier is used, the pre-write and ECC functions must be validated using the following procedure before executing a write.

[How to set using Flashpro II Ver.2.50 or earlier]

<1> Connect the PC and the FL-PR2 and activate the control software (flashpro.exe)
<2> Press the CRTL, SHIFT, GRPH (ALT), and P keys at the same time
<3> Select “Pre-Write set”
<4> Click the OK button.

} Pre-write setting

<5> Select “Setting”
<6> Select “Option”.
<7> Select “ECC code area” in the menu window.
<8> Input “14004” to the “ECC END ADDRESS” field.
<9> Click the OK button.
<10> Click the TYPE button.
<11> Input “14004” to the “ECC ADDRESS” field
<12> Click the OK button.

} ECC write setting

CHAPTER 19 INSTRUCTION OPERATIONS

19.1 Legend

(1) Explanation of operand identifiers (1/2)

Identifier	Explanation
r, r'[Note 1]	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[Note 1]	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'[Note 2]	AX(RP0), BC(RP1), RP2, RP3, VP(RP4), UP(RP5), DE(RP6), HL(RP7)
rp1[Note 2]	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
sfrp	Special function register symbol (register for which 16-bit operation is possible)
post[Note 2]	AX(RP0), BC(RP1), RP2, RP3, VP(RP4), UP(RP5)/PSW, DE(RP6), HL(RP7) Multiple descriptions are permissible. However, UP is only used with PUSH/POP instructions, and PSW with PUSHU/POPU instructions.
mem	[TDE], [WHL], [TDE+], [WHL+], [TDE–], [WHL–], [VVP], [UUP]: Register indirect addressing [TDE+byte], [WHL+byte], [SP+byte], [UUP+byte], [VVP+byte]: Based addressing imm24 [A], imm24 [B], imm24 [DE], imm24 [HL]: Indexed addressing [TDE+A], [TDE+B], [TDE+C], [WHL+A], [WHL+B], [WHL+C], [VVP+DE], [VVP+HL]: Based indexed addressing
mem1	All mem except [WHL+] and [WHL–]
mem2	[TDE], [WHL]
mem3	[AX], [BC], [RP2], [RP3], [VVP], [UUP], [TDE], [WHL]

Notes 1. Setting the RSS bit to 1 enables R4 to R7 to be used as X, A, C and B, but this function should only be used when using a 78K/III series program.

2. Setting the RSS bit to 1 enables RP2 and RP3 to be used as AX and BC, but this function should only be used when using a 78K/III series program.

(1) Explanation of operand identifiers (2/2)

Identifier	Explanation
Note	
saddr, saddr'	FD20H to FF1FH immediate data or label
saddr1	FE00H to FEFFH immediate data or label
saddr2	FD20H to FDFFH, FF00H to FF1FH immediate data or label
saddrp	FD20H to FF1EH immediate data or label (16-bit operation)
saddrp1	FE00H to FEFFH immediate data or label (16-bit operation)
saddrp2	FD20H to FDFFH, FF00H to FF1EH immediate data or label (16-bit operation)
saddrg	FD20H to FEFDH immediate data or label (24-bit operation)
saddrg1	FE00H to FEFDH immediate data or label (24-bit operation)
saddrg2	FD20H to FDFFH immediate data or label (24-bit operation)
addr24	0H to FFFFFFH 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* 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 The addresses shown here apply when 0H is specified by the LOCATION instruction.
When 0FH is specified by the LOCATION instruction, F0000H should be added to the address values shown.

(2) Operand column symbols

Symbol	Explanation
+	Auto-increment
–	Auto-decrement
#	Immediate data
!	16-bit absolute address
!!	24-bit/20-bit absolute address
$	8-bit relative address
$!	16-bit relative address
/	Bit inversion
[ ]	Indirect addressing
[%]	24-bit indirect addressing

(3) Flag column symbols

Symbol	Explanation
(Blank)	No change
0	Cleared to 0
1	Set to 1
×	Set or cleared depending on result
P	P/V flag operates as parity flag
V	P/V flag operates as overflow flag
R	Previously saved value is restored

(4) Operation column symbols

Symbol	Explanation
jdisp8	Signed two's complement data (8 bits) indicating relative address distance between start address of next instruction and branch address
jdisp16	Signed two's complement data (16 bits) indicating relative address distance between start address of next instruction and branch address
PC_{HW}	PC bits 16 to 19
PC_{LW}	PC bits 0 to 15

(5) Number of bytes of instruction that includes mem in operands

mem Mode	Register Indirect Addressing		Based Addressing	Indexed Addressing	Based Indexed Addressing
Number of bytes	1	2**Note**	3	5	2

Note One-byte instruction only when [TDE], [WHL], [TDE+], [TDE-], [WHL+] or [WHL–] is written as mem in an MOV instruction.

(6) Number of bytes of instruction that includes saddr, saddrp, r or rp in operands

For some instructions that include saddr, saddrp, r or rp in their operands, two "Bytes" entries are given, separated by a slash ("/"). The entry that applies is shown in the table below.

Identifier	Left-Hand "Bytes" Figure	Right-Hand "Bytes" Figure
saddr	saddr2	saddr1
saddrp	saddrp2	saddrp1
r	r1	r2
rp	rp1	rp2

(7) Code of instructions that include mem in operands and string instructions

Operands TDE, WHL, VVP and UUP (24-bit registers) can also be written as DE, HL, VP and UP respectively. However, they are still treated as TDE, WHL, VVP and UUP (24-bit registers) when written as DE, HL, VP and UP.

19.2 List of Operations

(1) 8-bit data transfer instruction: MOV

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MOV	r, #byte	2/3	r ← 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	PSW_L ← byte	×	×	×	×	×
	PSWH, #byte	3	PSW_H ← byte					
	PSWL, A	2	PSW_L ← A	×	×	×	×	×
	PSWH, A	2	PSW_H ← A					
	A, PSWL	2	A ← PSW_L					
	A, PSWH	2	A ← PSW_H					
	r3, #byte	3	r3 ← byte					
	A, r3	2	A ← r3					
	r3, A	2	r3 ← A					

(2) 16-bit data transfer instruction: MOVW

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MOVW	rp, #word	3	rp ← word					
	saddrp, #word	4/5	(saddrp) ← word					
	sfrp, #word	4	sfrp ← word					
	!addr16, #word	6	(addr16) ← word					
	!!addr24, #word	7	(addr24) ← word					
	rp, rp'	2	rp ← rp'					
	AX, saddrp2	2	AX ← (saddrp2)					
	rp, saddrp	3	rp ← (saddrp)					
	saddrp2, AX	2	(saddrp2) ← AX					
	saddrp, rp	3	(saddrp) ← rp					
	AX, sfrp	2	AX ← sfrp					
	rp, sfrp	3	rp ← sfrp					
	sfrp, AX	2	sfrp ← AX					
	sfrp, rp	3	sfrp ← rp					
	saddrp, saddrp'	4	(saddrp) ← (saddrp')					
	rp, !addr16	4	rp ← (addr16)					
	!addr16, rp	4	(addr16) ← rp					
	rp, !!addr24	5	rp ← (addr24)					
	!!addr24, rp	5	(addr24) ← rp					
	AX, [saddrp]	3/4	AX ← ((saddrp))					
	AX, [%saddrg]	3/4	AX ← ((saddrg))					
	AX, mem	2-5	AX ← (mem)					
	[saddrp], AX	3/4	((saddrp)) ← AX					
	[%saddrg], AX	3/4	((saddrg)) ← AX					
	mem, AX	2-5	(mem) ← AX					

(3) 24-bit data transfer instruction: MOVG

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MOVG	rg, #imm24	5	rg ← 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	Operands	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	Operands	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 operation instructions: ADD, ADDC, SUB, SUBC, CMP, AND, OR, XOR

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ADD	A, #byte	2	A, CY ← 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	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ADDC	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	Operands	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	Operands	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	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
CMP	A, #byte	2	A – byte	×	×	×	V	×
	r, #byte	3	r – byte	×	×	×	V	×
	saddr, #byte	3/4	(saddr) – byte	×	×	×	V	×
	sfr, #byte	4	sfr – byte	×	×	×	V	×
	r, r'	2/3	r – r'	×	×	×	V	×
	A, saddr2	2	A – (saddr2)	×	×	×	V	×
	r, saddr	3	r – (saddr)	×	×	×	V	×
	saddr, r	3	(saddr) – r	×	×	×	V	×
	r, sfr	3	r – sfr	×	×	×	V	×
	sfr, r	3	sfr – r	×	×	×	V	×
	saddr, saddr'	4	(saddr) – (saddr')	×	×	×	V	×
	A, [saddrp]	3/4	A – ((saddrp))	×	×	×	V	×
	A, [%saddrg]	3/4	A – ((saddrg))	×	×	×	V	×
	[saddrp], A	3/4	((saddrp)) – A	×	×	×	V	×
	[%saddrg], A	3/4	((saddrg)) – A	×	×	×	V	×
	A, !addr16	4	A – (addr16)	×	×	×	V	×
	A, !!addr24	5	A – (addr24)	×	×	×	V	×
	!addr16, A	4	(addr16) – A	×	×	×	V	×
	!!addr24, A	5	(addr24) – A	×	×	×	V	×
	A, mem	2-5	A – (mem)	×	×	×	V	×
	mem, A	2-5	(mem) – A	×	×	×	V	×

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
AND	A, #byte	2	A ← A ∧ byte	×	×		P	
	r, #byte	3	r ← r ∧ byte	×	×		P	
	saddr, #byte	3/4	(saddr) ← (saddr) ∧ byte	×	×		P	
	sfr, #byte	4	sfr ← sfr ∧ byte	×	×		P	
	r, r'	2/3	r ← r ∧ r'	×	×		P	
	A, saddr2	2	A ← A ∧ (saddr2)	×	×		P	
	r, saddr	3	r ← r ∧ (saddr)	×	×		P	
	saddr, r	3	(saddr) ← (saddr) ∧ r	×	×		P	
	r, sfr	3	r ← r ∧ sfr	×	×		P	
	sfr, r	3	sfr ← sfr ∧ r	×	×		P	
	saddr, saddr'	4	(saddr) ← (saddr) ∧ (saddr')	×	×		P	
	A, [saddrp]	3/4	A ← A ∧ ((saddrp))	×	×		P	
	A, [%saddrg]	3/4	A ← A ∧ ((saddrg))	×	×		P	
	[saddrp], A	3/4	((saddrp)) ← ((saddrp)) ∧ A	×	×		P	
	[%saddrg], A	3/4	((saddrg)) ← ((saddrg)) ∧ A	×	×		P	
	A, !addr16	4	A ← A ∧ (addr16)	×	×		P	
	A, !!addr24	5	A ← A ∧ (addr24)	×	×		P	
	!addr16, A	4	(addr16) ← (addr16) ∧ A	×	×		P	
	!!addr24, A	5	(addr24) ← (addr24) ∧ A	×	×		P	
	A, mem	2-5	A ← A ∧ (mem)	×	×		P	
	mem, A	2-5	(mem) ← (mem) ∧ A	×	×		P	

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
OR	A, #byte	2	A ← A ∨ byte	×	×		P	
	r, #byte	3	r ← r ∨ byte	×	×		P	
	saddr, #byte	3/4	(saddr) ← (saddr) ∨ byte	×	×		P	
	sfr, #byte	4	sfr ← sfr ∨ byte	×	×		P	
	r, r'	2/3	r ← r ∨ r'	×	×		P	
	A, saddr2	2	A ← A ∨ (saddr2)	×	×		P	
	r, saddr	3	r ← r ∨ (saddr)	×	×		P	
	saddr, r	3	(saddr) ← (saddr) ∨ r	×	×		P	
	r, sfr	3	r ← r ∨ sfr	×	×		P	
	sfr, r	3	sfr ← sfr ∨ r	×	×		P	
	saddr, saddr'	4	(saddr) ← (saddr) ∨ (saddr')	×	×		P	
	A, [saddrp]	3/4	A ← A ∨ ((saddrp))	×	×		P	
	A, [%saddrg]	3/4	A ← A ∨ ((saddrg))	×	×		P	
	[saddrp], A	3/4	((saddrp)) ← ((saddrp)) ∨ A	×	×		P	
	[%saddrg], A	3/4	((saddrg)) ← ((saddrg)) ∨ A	×	×		P	
	A, !addr16	4	A ← A ∨ (addr16)	×	×		P	
	A, !!addr24	5	A ← A ∨ (addr24)	×	×		P	
	!addr16, A	4	(addr16) ← (addr16) ∨ A	×	×		P	
	!!addr24, A	5	(addr24) ← (addr24) ∨ A	×	×		P	
	A, mem	2-5	A ← A ∨ (mem)	×	×		P	
	mem, A	2-5	(mem) ← (mem) ∨ A	×	×		P	

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
XOR	A, #byte	2	A ← A ∀ byte	×	×		P	
	r, #byte	3	r ← r ∀ byte	×	×		P	
	saddr, #byte	3/4	(saddr) ← (saddr) ∀ byte	×	×		P	
	sfr, #byte	4	sfr ← sfr ∀ byte	×	×		P	
	r, r'	2/3	r ← r ∀ r'	×	×		P	
	A, saddr2	2	A ← A ∀ (saddr2)	×	×		P	
	r, saddr	3	r ← r ∀ (saddr)	×	×		P	
	saddr, r	3	(saddr) ← (saddr) ∀ r	×	×		P	
	r, sfr	3	r ← r ∀ sfr	×	×		P	
	sfr, r	3	sfr ← sfr ∀ r	×	×		P	
	saddr, saddr'	4	(saddr) ← (saddr) ∀ (saddr')	×	×		P	
	A, [saddrp]	3/4	A ← A ∀ ((saddrp))	×	×		P	
	A, [%saddrg]	3/4	A ← A ∀ ((saddrg))	×	×		P	
	[saddrp], A	3/4	((saddrp)) ← ((saddrp)) ∀ A	×	×		P	
	[%saddrg], A	3/4	((saddrg)) ← ((saddrg)) ∀ A	×	×		P	
	A, !addr16	4	A ← A ∀ (addr16)	×	×		P	
	A, !!addr24	5	A ← A ∀ (addr24)	×	×		P	
	!addr16, A	4	(addr16) ← (addr16) ∀ A	×	×		P	
	!!addr24, A	5	(addr24) ← (addr24) ∀ A	×	×		P	
	A, mem	2-5	A ← A ∀ (mem)	×	×		P	
	mem, A	2-5	(mem) ← (mem) ∀ A	×	×		P	

(7) 16-bit operation instructions: ADDW, SUBW, CMPW

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ADDW	AX, #word	3	AX, CY ← 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 operation instructions: ADDG, SUBG

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ADDG	rg, rg'	2	rg, CY ← rg + rg'	×	×	×	V	×
	rg, # imm24	5	rg, CY ← rg + # imm24	×	×	×	V	×
	WHL, saddrg	3	WHL, CY ← WHL + (saddrg)	×	×	×	V	×
SUBG	rg, rg'	2	rg, CY ← rg – rg'	×	×	×	V	×
	rg, # imm24	5	rg, CY ← rg – imm24	×	×	×	V	×
	WHL, saddrg	3	WHL, CY ← WHL – (saddrg)	×	×	×	V	×

(9) Multiplication instructions: MULU, MULUW, MULW, DIVUW, DIVUX

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MULU	r	2/3	AX ← A × r					
MULUW	rp	2	AX (upper half), rp (lower half) ← AX × rp					
MULW	rp	2	AX (upper half), rp (lower half) ← AX × rp					
DIVUW	r	2/3	AX (quotient), r (remainder) ← AX ÷ r**Note 1**					
DIVUX	rp	2	AXDE (quotient), rp (remainder) ← AXDE ÷ rp**Note 2**					

Notes 1. When r = 0, r ← X, AX ← FFFFH

2. When rp = 0, pr ← DE, AXDE ← FFFFFFFFH

(10) Special operation instructions: MACW, MACSW, SACW

Mnemonic	Operands	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)	×	×	×	V	×

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
				×	×	×	V	×

SACW	[TDE +], [WHL +]	4	AX ← |(TDE) – (WHL)| + AX, TDE ← TDE + 2, WHL ← WHL + 2 C ← C – 1 End if(C = 0 or CY = 1)	×	×	×	V	×

(11) Increment/decrement instructions: INC, DEC, INCW, DECW, INCG, DECG

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
INC	r	1/2	r ← r + 1	×	×	×	V	
	saddr	2/3	(saddr) ← (saddr) + 1	×	×	×	V	
DEC	r	1/2	r ← r –1	×	×	×	V	
	saddr	2/3	(saddr) ← (saddr) – 1	×	×	×	V	
INCW	rp	2/1	rp ← rp + 1					
	saddrp	3/4	(saddrp) ← (saddrp) + 1					
DECW	rp	2/1	rp ← rp – 1					
	saddrp	3/4	(saddrp) ← (saddrp) – 1					
INCG	rg	2	rg ← rg + 1					
DECG	rg	2	rg ← rg – 1					

(12) Adjustment instructions: ADJBA, ADJBS, CVTBW

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ADJBA		2	Decimal Adjust Accumulator after Addition	×	×	×	P	×
ADJBS		2	Decimal Adjust Accumulator after Subtract	×	×	×	P	×
CVTBW		1	X ← A, A ← 00H if A_7 = 0 X ← A, A ← FFH if A_7 = 1					

(13) Shift/rotate instructions: ROR, ROL, RORC, ROLC, SHR, SHL, SHRW, SHLW, ROR4, ROL4

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ROR	r, n	2/3	(CY, $r7 \leftarrow r0$, $rm-1 \leftarrow rm$) × n times n = 0 – 7				P	×
ROL	r, n	2/3	(CY, $r0 \leftarrow r7$, $rm+1 \leftarrow rm$) × n times n = 0 – 7				P	×
RORC	r, n	2/3	(CY$\leftarrow r0$, $r7 \leftarrow$ CY, $rm-1 \leftarrow rm$) × n times n = 0 – 7				P	×
ROLC	r, n	2/3	(CY$\leftarrow r7$, $r0 \leftarrow$ CY, $rm+1 \leftarrow rm$) × n times n = 0 – 7				P	×
SHR	r, n	2/3	(CY$\leftarrow r0$, $r7 \leftarrow 0$, $rm-1 \leftarrow rm$) × n times n = 0 – 7	×	×	0	P	×
SHL	r, n	2/3	(CY$\leftarrow r7$, $r0 \leftarrow 0$, $rm+1 \leftarrow rm$) × n times n = 0 – 7	×	×	0	P	×
SHRW	rp, n	2	(CY$\leftarrow rp0$, $rp15 \leftarrow 0$, $rpm-1 \leftarrow$ rpm) × n times n = 0 – 7	×	×	0	P	×
SHLW	rp, n	2	(CY$\leftarrow rp15$, $rp0 \leftarrow 0$, $rpm+1 \leftarrow$ rpm) × n times n = 0 – 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	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MOV1	CY, saddr. bit	3/4	CY ← (saddr. bit)					×
	CY, sfr. bit	3	CY ← sfr. bit					×
	CY, X. bit	2	CY ← X. bit					×
	CY, A. bit	2	CY ← A. bit					×
	CY, PSWL. bit	2	CY ← PSWL. bit					×
	CY, PSWH. bit	2	CY ← PSWH. bit					×
	CY, !addr16. bit	5	CY ← !addr16.bit					×
	CY, !!addr24. bit	2	CY ← !!addr24. bit					×
	CY, mem2. bit	2	CY ← mem2. bit					×
	saddr. bit, CY	3/4	(saddr. bit) ← CY					
	sfr. bit, CY	3	sfr. bit ← CY					
	X. bit, CY	2	X.bit ← CY					
	A. bit, CY	2	A. bit ← CY					
	PSWL. bit, CY	2	PSWL. bit ← CY	×	×	×	×	×
	PSWH. bit, CY	2	PSWH. bit ← CY					
	!addr16. bit, CY	5	!addr16.bit ← CY					
	!!addr24.bit, CY	6	!!addr24.bit ← CY					
	mem2. bit, CY	2	mem2. bit ← CY					

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
AND1	CY, saddr. bit	3/4	CY ← CY ∧ (saddr. bit)					×
	CY, /saddr. bit	3/4	CY ← CY ∧ $\overline{\text{(saddr. bit)}}$					×
	CY, sfr. bit	3	CY ← CY ∧ sfr. bit					×
	CY, /sfr. bit	3	CY ← CY ∧ $\overline{\text{sfr. bit}}$					×
	CY, X. bit	2	CY ← CY ∧ X. bit					×
	CY, /X. bit	2	CY ← CY ∧ $\overline{\text{X. bit}}$					×
	CY, A. bit	2	CY ← CY ∧ A. bit					×
	CY, /A. bit	2	CY ← CY ∧ $\overline{\text{A. bit}}$					×
	CY, PSWL. bit	2	CY ← CY ∧ PSW_{L}. bit					×
	CY, /PSWL. bit	2	CY ← CY ∧ $\overline{\text{PSW}_{\text{L}}\text{. bit}}$					×
	CY, PSWH. bit	2	CY ← CY ∧ PSW_{H}. bit					×
	CY, /PSWH. bit	2	CY ← CY ∧ $\overline{\text{PSW}_{\text{H}}\text{. bit}}$					×
	CY, !addr16. bit	5	CY ← CY ∧ !addr16. bit					×
	CY, /!addr16. bit	5	CY ← CY ∧ $\overline{\text{!addr16. bit}}$					×
	CY, !!addr24. bit	2	CY ← CY ∧ !!addr24. bit					×
	CY, /!!addr24. bit	6	CY ← CY ∧ $\overline{\text{!!addr24. bit}}$					×
	CY, mem2. bit	2	CY ← CY ∧ mem2. bit					×
	CY, /mem2. bit	2	CY ← CY ∧ $\overline{\text{mem2. bit}}$					×
OR1	CY, saddr. bit	3/4	CY ← CY ∨ (saddr. bit)					×
	CY, /saddr. bit	3/4	CY ← CY ∨ $\overline{\text{(saddr. bit)}}$					×
	CY, sfr. bit	3	CY ← CY ∨ sfr. bit					×
	CY, /sfr. bit	3	CY ← CY ∨ $\overline{\text{sfr. bit}}$					×
	CY, X. bit	2	CY ← CY ∨ X. bit					×
	CY, /X. bit	2	CY ← CY ∨ $\overline{\text{X. bit}}$					×
	CY, A. bit	2	CY ← CY ∨ A. bit					×
	CY, /A. bit	2	CY ← CY ∨ $\overline{\text{A. bit}}$					×
	CY, PSWL. bit	2	CY ← CY ∨ PSW_{L}. bit					×
	CY, /PSWL. bit	2	CY ← CY ∨ $\overline{\text{PSW}_{\text{L}}\text{. bit}}$					×
	CY, PSWH. bit	2	CY ← CY ∨ PSW_{H}. bit					×
	CY, /PSWH. bit	2	CY ← CY ∨ $\overline{\text{PSW}_{\text{H}}\text{. bit}}$					×
	CY, !addr16. bit	5	CY ← CY ∨ !addr16. bit					×
	CY, /!addr16. bit	5	CY ← CY ∨ $\overline{\text{!addr16. bit}}$					×
	CY, !!addr24. bit	2	CY ← CY ∨ !!addr24. bit					×
	CY, /!!addr24. bit	6	CY ← CY ∨ $\overline{\text{!!addr24. bit}}$					×
	CY, mem2. bit	2	CY ← CY ∨ mem2. bit					×
	CY, /mem2. bit	2	CY ← CY ∨ $\overline{\text{mem2. bit}}$					×

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
XOR1	CY, saddr. bit	3/4	CY ← CY ∀ (saddr. bit)					×
	CY, sfr. bit	3	CY ← CY ∀ sfr. bit					×
	CY, X. bit	2	CY ← CY ∀ X. bit					×
	CY, A. bit	2	CY ← CY ∀ A. bit					×
	CY, PSWL. bit	2	CY ← CY ∀ PSWL. bit					×
	CY, PSWH. bit	2	CY ← CY ∀ PSWH. bit					×
	CY, !addr16. bit	5	CY ← CY ∀ !addr16. bit					×
	CY, !!addr24. bit	2	CY ← CY ∀ !!addr24. bit					×
	CY, mem2. bit	2	CY ← CY ∀ mem2. bit					×
NOT1	saddr. bit	3/4	(saddr. bit) ← ($\overline{\text{saddr. bit}}$)					
	sfr. bit	3	sfr. bit ← $\overline{\text{sfr. bit}}$					
	X. bit	2	X. bit ← $\overline{\text{X. bit}}$					
	A. bit	2	A. bit ← $\overline{\text{A. bit}}$					
	PSWL. bit	2	PSWL. bit ← $\overline{\text{PSW}_{\text{L}}\text{. bit}}$	×	×	×	×	×
	PSWH. bit	2	PSWH. bit ← $\overline{\text{PSW}_{\text{H}}\text{. bit}}$					
	!addr16. bit	5	!addr16. bit ← $\overline{\text{!addr16. bit}}$					
	!!addr24. bit	2	!!addr24. bit ← $\overline{\text{!!addr24. bit}}$					
	mem2. bit	2	mem2. bit ← $\overline{\text{mem2. bit}}$					
	CY	1	CY ← $\overline{\text{CY}}$					×
SET1	saddr. bit	2/3	(saddr. bit) ← 1					
	sfr. bit	3	sfr. bit ← 1					
	X. bit	2	X. bit ← 1					
	A. bit	2	A. bit ← 1					
	PSWL. bit	2	PSWL. bit ← 1	×	×	×	×	×
	PSWH. bit	2	PSWH. bit ← 1					
	!addr16. bit	5	!addr16. bit ← 1					
	!!addr24. bit	2	!!addr24. bit ← 1					
	mem2. bit	2	mem2. bit ← 1					
	CY	1	CY ← 1					1
CLR1	saddr. bit	2/3	(saddr. bit) ← 0					
	sfr. bit	3	sfr. bit ← 0					
	X. bit	2	X. bit ← 0					
	A. bit	2	A. bit ← 0					
	PSWL. bit	2	PSWL. bit ← 0	×	×	×	×	×
	PSWH. bit	2	PSWH. bit ← 0					
	!addr16. bit	5	!addr16. bit ← 0					
	!!addr24. bit	2	!!addr24. bit ← 0					
	mem2. bit	2	mem2. bit ← 0					
	CY	1	CY ← 0					0

(15) Stack manipulation instructions: PUSH, PUSHU, POP, POPU, MOVG, ADDWG, SUBWG, INCG, DECG

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
PUSH	PSW	1	(SP – 2) ← PSW, SP ← SP – 2					
	sfrp	3	(SP – 2) ← sfrp, SP ← SP – 2					
	sfr	3	(SP – 1) ← sfr, SP ← SP – 1					
	post	2	{(SP – 2) ← post, SP ← SP – 2} × m times[Note]					
	rg	2	(SP – 3) ← rg, SP ← SP – 3					
PUSHU	post	2	{(UUP – 2) ← post, UUP ← UUP – 2} × m times[Note]					
POP	PSW	1	PSW ← (SP), SP ← SP + 2	R	R	R	R	R
	sfrp	3	sfrp ← (SP), SP ← SP + 2					
	sfr	3	sfr ← (SP), SP ← SP + 1					
	post	2	{post ← (SP), SP ← SP + 2} × m times[Note]					
	rg	2	rg ← (SP), SP ← SP + 3					
POPU	post	2	{post ← (UUP), UUP ← UUP + 2} × m times[Note]					
MOVG	SP, # imm24	5	SP ← imm24					
	SP, WHL	2	SP ← WHL					
	WHL, SP	2	WHL ← SP					
ADDWG	SP, #word	4	SP ← SP + word					
SUBWG	SP, #word	4	SP ← SP – word					
INCG	SP	2	SP ← SP + 1					
DECG	SP	2	SP ← SP – 1					

Note m = number of registers specified by “post”

(16) Call/return instructions: CALL, CALLF, CALLT, BRK, BRKCS, RET, RETI, RETB, RETCS, RETCSB

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
CALL	!addr16	3	(SP – 3) ← (PC + 3), SP ← SP – 3, PC_{HW} ← 0, PC_{LW} ← addr16					
	!!addr20	4	(SP – 3) ← (PC + 4), SP ← SP – 3, PC ← addr20					
	rp	2	(SP – 3) ← (PC + 2), SP ← SP – 3, PC_{HW} ← 0, PC_{LW} ← rp					
	rg	2	(SP – 3) ← (PC + 2), SP ← SP – 3, PC ← rg					
	[rp]	2	(SP – 3) ← (PC + 2), SP ← SP – 3, PC_{HW} ← 0, PC_{LW} ← (rp)					
	[rg]	2	(SP – 3) ← (PC + 2), SP ← SP – 3, PC ← (rg)					
	$!addr20	3	(SP – 3) ← (PC + 3), SP ← SP – 3, PC ← PC + 3 + jdisp16					
CALLF	!addr11	2	(SP – 3) ← (PC + 2), SP ← SP – 3, PC_{19-12} ← 0, PC11 ← 1, PC_{10-0} ← addr11					
CALLT	[addr5]	1	(SP – 3) ← (PC + 1), SP ← SP – 3, PC_{HW} ← 0, PC_{LW} ← (addr5)					
BRK		1	(SP – 2) ← PSW, $(SP - 1)_{0-3}$ ← $(PC + 1)_{HW}$, $(SP - 4)_{LW}$ ← $(PC + 1)_{LW}$, SP ← SP – 4, PC_{HW} ← 0, PC_{LW} ← (003EH)					
BRKCS	RBn	2	PC_{LW} ← RP2, RP3 ← PSW, RBS2 – 0 ← n, RSS ← 0, IE ← 0, $RP3_{8-11}$ ← PC_{HW}, PC_{HW} ← 0					
RET		1	PC ← (SP), SP ← SP + 3					
RET1		1	PC_{LW} ← (SP), PC_{HW} ← $(SP + 3)_{0-3}$, PSW ← (SP + 2), SP ← SP + 4 Clears to 0 flag with highest priority of flags of ISPR that are set (1)	R	R	R	R	R
RETB		1	PC_{LW} ← (SP), PC_{HW} ← $(SP + 3)_{0-3}$, PSW ← (SP + 2), SP ← SP + 4	R	R	R	R	R
RETCS	!addr16	3	PSW ← RP3, PC_{LW} ← RP2, RP2 ← addr16, PC_{HW} ← $RP3_{8-11}$ Clears to 0 flag with highest priority of flags of ISPR that are set (1)	R	R	R	R	R
RETCSB	!addr16	4	PSW ← RP3, PC_{LW} ← RP2, RP2 ← addr16, PC_{HW} ← $RP3_{8-11}$	R	R	R	R	R

(17) Unconditional branch instruction: BR

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
BR	!addr16	3	$PC_{HW} \leftarrow 0$, $PC_{LW} \leftarrow$ addr16					
	!!addr20	4	PC ← addr20					
	rp	2	$PC_{HW} \leftarrow 0$, $PC_{LW} \leftarrow$ rp					
	rg	2	PC ← rg					
	[rp]	2	$PC_{HW} \leftarrow 0$, $PC_{LW} \leftarrow$ (rp)					
	[rg]	2	PC ← (rg)					
	$addr20	2	PC ← PC + 2 + jdisp8					
	$!addr20	3	PC ← PC + 3 + 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	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
BNZ	$addr20	2	PC ← PC + 2 + jdisp8 if Z = 0					
BNE								
BZ	$addr20	2	PC ← PC + 2 + jdisp8 if Z = 1					
BE								
BNC	$addr20	2	PC ← PC + 2 + jdisp8 if CY = 0					
BNL								
BC	$addr20	2	PC ← PC + 2 + jdisp8 if CY = 1					
BL								
BNV	$addr20	2	PC ← PC + 2 + jdisp8 if P/V = 0					
BPO								
BV	$addr20	2	PC ← PC + 2 + jdisp8 if P/V = 1					
BPE								
BP	$addr20	2	PC ← PC + 2 + jdisp8 if S = 0					
BN	$addr20	2	PC ← PC + 2 + jdisp8 if S = 1					
BLT	$addr20	3	PC ← PC + 3 + jdisp8 if P/V ∀ S = 1					
BGE	$addr20	3	PC ← PC + 3 + jdisp8 if P/V ∀ S = 0					
BLE	$addr20	3	PC ← PC + 3 + jdisp8 if (P/V ∀ S) ∨ Z = 1					
BGT	$addr20	3	PC ← PC + 3 + jdisp8 if (P/V ∀ S) ∨ Z = 0					
BNH	$addr20	3	PC ← PC + 3 + jdisp8 if Z ∨ CY = 1					
BH	$addr20	3	PC ← PC + 3 + jdisp8 if Z ∨ CY = 0					
BF	saddr. bit, $addr20	4/5	PC ← PC + 4[Note] + jdisp8 if (saddr. bit) = 0					
	sfr. bit, $addr20	4	PC ← PC + 4 + jdisp8 if sfr. bit = 0					
	X. bit, $addr20	3	PC ← PC + 3 + jdisp8 if X. bit = 0					
	A. bit, $addr20	3	PC ← PC + 3 + jdisp8 if A. bit = 0					
	PSWL. bit, $addr20	3	PC ← PC + 3 + jdisp8 if PSW_L. bit = 0					
	PSWH. bit, $addr20	3	PC ← PC + 3 + jdisp8 if PSW_H. bit = 0					
	!addr16. bit, $addr20	6	PC ← PC + 3 + jdisp8 if !addr16. bit = 0					
	!!addr24. bit, $addr20	3	PC ← PC + 3 + jdisp8 if !!addr24. bit = 0					
	mem2. bit, $addr20	3	PC ← PC + 3 + jdisp8 if mem2. bit = 0					

Note When the number of bytes is 4. When 5, the operation is: PC ← PC + 5 + jdisp8.

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
BT	saddr. bit, $addr20	3/4	PC ← PC + 3[Note 1] + jdisp8 if (saddr. bit) = 1					
	sfr. bit, $addr20	4	PC ← PC + 4 + jdisp8 if sfr. bit = 1					
	X. bit, $addr20	3	PC ← PC + 3 + jdisp8 if X. bit = 1					
	A. bit, $addr20	3	PC ← PC + 3 + jdisp8 if A. bit = 1					
	PSWL. bit, $addr20	3	PC ← PC + 3 + jdisp8 if PSW_L. bit = 1					
	PSWH. bit, $addr20	3	PC ← PC + 3 + jdisp8 if PSW_H. bit = 1					
	!addr16. bit, $addr20	6	PC ← PC + 3 + jdisp8 if !addr16. bit = 1					
	!!addr24. bit, $addr20	3	PC ← PC + 3 + jdisp8 if !!addr24. bit = 1					
	mem2. bit, $addr20	3	PC ← PC + 3 + jdisp8 if mem2. bit = 1					
BTCLR	saddr. bit, $addr20	4/5	{PC ← PC + 4[Note 2] + jdisp8, (saddr. bit) ← 0} if (saddr. bit) = 1					
	sfr. bit, $addr20	4	{PC ← PC + 4 + jdisp8, sfr. bit ← 0} if sfr. bit = 1					
	X. bit, $addr20	3	{PC ← PC + 3 + jdisp8, X. bit ← 0} if X. bit = 1					
	A. bit, $addr20	3	{PC ← PC + 3 + jdisp8, A. bit ← 0} if A. bit = 1					
	PSWL. bit, $addr20	3	{PC ← PC + 3 + jdisp8, PSW_L. bit ← 0} if PSW_L. bit = 1	×	×	×	×	×
	PSWH. bit, $addr20	3	{PC ← PC + 3 + jdisp8, PSW_H. bit ← 0} if PSW_H. bit = 1					
	!addr16. bit, $addr20	6	{PC ← PC + 3 + jdisp8, !addr16. bit ← 0} if !addr16. bit = 1					
	!!addr24. bit, $addr20	3	{PC ← PC + 3 + jdisp8, !!addr24. bit ← 0} if !!addr24. bit = 1					
	mem2. bit, $addr20	3	{PC ← PC + 3 + jdisp8, mem2. bit ← 0} if mem2. bit = 1					

Notes 1. When the number of bytes is 3. When 4, the operation is: PC ← PC + 4 + jdisp8.

2. When the number of bytes is 4. When 5, the operation is: PC ← PC + 5 + jdisp8.

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
BFSET	saddr. bit, $addr20	4/5	{PC ← PC + 4[Note 2] + jdisp8, (saddr. bit) ← 1} if (saddr. bit) = 0					
	sfr. bit, $addr20	4	{PC ← PC + 4 + jdisp8, sfr. bit ← 1} if sfr. bit = 0					
	X. bit, $addr20	3	{PC ← PC + 3 + jdisp8, X. bit ← 1} if X. bit = 0					
	A. bit, $addr20	3	{PC ← PC + 3 + jdisp8, A. bit ← 1} if A. bit = 0					
	PSWL. bit, $addr20	3	{PC ← PC + 3 + jdisp8, PSW_L. bit ← 1} if PSW_L. bit = 0	×	×	×	×	×
	PSWH. bit, $addr20	3	{PC ← PC + 3 + jdisp8, PSW_H. bit ← 1} if PSW_H. bit = 0					
	!addr16. bit, $addr20	6	{PC ← PC + 3 + jdisp8, !addr16. bit ← 1} if !addr16. bit = 0					
	!!addr24. bit, $addr20	3	{PC ← PC + 3 + jdisp8, !!addr24. bit ← 1} if !!addr24. bit = 0					
	mem2. bit, $addr20	3	{PC ← PC + 3 + jdisp8, mem2. bit ← 1} if mem2. bit = 0					
DBNZ	B, $addr20	2	B ← B – 1, PC ← PC + 2 + jdisp8 if B ≠ 0					
	C, $addr20	2	C ← C – 1, PC ← PC + 2 + jdisp8 if C ≠ 0					
	$addr, $addr20	3/4	(saddr) ← (saddr) – 1, PC ← PC + 3[Note 1] = jdisp8 if (saddr) ≠ 0					

Notes 1. When the number of bytes is 3. When 4, the operation is: PC ← PC + 4 + jdisp8.

2. When the number of bytes is 4. When 5, the operation is: PC ← PC + 5 + jdisp8.

(19) CPU control instructions: MOV, LOCATION, SEL, SWRS, NOP, EI, DI

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MOV	STBC, #byte	4	STBC ← byte					
	WDM, #byte	4	WDM ← byte					
LOCATION	locaddr	4	SFR, internal data area location address upper word specification					
SEL	RBn	2	RSS ← 0, RBS2 – 0 ← n					
	RBn, ALT	2	RSS ← 1, RBS2 – 0 ← n					
SWRS		2	RSS ← $\overline{RSS}$					
NOP		1	No Operaton					
EI		1	IE ← 1 (Enable interrupt)					
DI		1	IE ← 0 (Disable interrupt)					

(20) Special instructions: CHKL, CHKLA

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
CHKL	sfr	3	(Pin level) ∀ (output latch)	×	×		P	
CHKLA	sfr	3	A ← (pin level) ∀ (output latch)	×	×		P	

(21) String instructions: MOVTBLW, MOVM, XCHM, MOVBK, XCHBK, CMPME, CMPMNE, CMPMC, CMPMNC, CMPBKE, CMPBKNE, CMPBKC, CMPBKNC

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MOVTBLW	!addr8, byte	4	(addr8 + 2) ← (addr8), byte ← byte – 1, addr8 ← addr8 – 2 End if byte = 0					
MOVW	[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					
XCHM	[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					
MOVBK	[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	×	×	×	V	×
	[TDE –], A	2	(TDE) – A, TDE ← TDE – 1, C ← C – 1 End if C = 0 or Z = 0	×	×	×	V	×
CMPMNE	[TDE +], A	2	(TDE) – A, TDE ← TDE + 1, C ← C – 1 End if C = 0 or Z = 1	×	×	×	V	×
	[TDE –], A	2	(TDE) – A, TDE ← TDE – 1, C ← C – 1 End if C = 0 or Z = 1	×	×	×	V	×
CMPMC	[TDE +], A	2	(TDE) – A, TDE ← TDE + 1, C ← C – 1 End if C = 0 or CY = 0	×	×	×	V	×
	[TDE –], A	2	(TDE) – A, TDE ← TDE – 1, C ← C – 1 End if C = 0 or CY = 0	×	×	×	V	×
CMPMNC	[TDE +], A	2	(TDE) – A, TDE ← TDE + 1, C ← C – 1 End if C = 0 or CY = 1	×	×	×	V	×
	[TDE –], A	2	(TDE) – A, TDE ← TDE – 1, C ← C – 1 End if C = 0 or CY = 1	×	×	×	V	×
CMPBKE	[TDE +], [WHL +]	2	(TDE) ← (WHL), TDE ← TDE + 1, WHL ← WHL + 1, C ← C – 1 End if C = 0 or Z = 0	×	×	×	V	×
	[TDE –], [WHL –]	2	(TDE) ← (WHL), TDE ← TDE – 1, WHL ← WHL – 1, C ← C – 1 End if C = 0 or Z = 0	×	×	×	V	×

Mnemonic	Operands	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
CMPBKNE	[TDE +], [WHL +]	2	(TDE) – (WHL), TDE ← TDE + 1, WHL ← WHL + 1, C ← C – 1 End if C = 0 or Z = 1	×	×	×	V	×
	[TDE –], [WHL –]	2	(TDE) – (WHL), TDE ← TDE – 1, WHL ← WHL – 1, C ← C – 1 End if C = 0 or Z = 1	×	×	×	V	×
CMPBKC	[TDE +], [WHL +]	2	(TDE) – (WHL), TDE ← TDE + 1, WHL ← WHL + 1, C ← C – 1 End if C = 0 or CY = 0	×	×	×	V	×
	[TDE –], [WHL –]	2	(TDE) – (WHL), TDE ← TDE – 1, WHL ← WHL – 1, C ← C – 1 End if C = 0 or CY = 0	×	×	×	V	×
CMPBKNC	[TDE +], [WHL +]	2	(TDE) – (WHL), TDE ← TDE + 1, WHL ← WHL + 1, C ← C – 1 End if C = 0 or CY = 1	×	×	×	V	×
	[TDE –], [WHL –]	2	(TDE) – (WHL), TDE ← TDE – 1, WHL ← WHL – 1, C ← C – 1 End if C = 0 or CY = 1	×	×	×	V	×

19.3 Instructions Listed by Type of Addressing

(1) 8-bit instructions (combinations expressed by writing A for r are shown in parentheses)

MOV, XCH, ADD, ADDC, SUB, SUBC, AND OR XOR, CMP, MULU, DIVUW, INC, DEC, ROR, ROL, RORC, ROLC, SHR, SHL, ROR4, ROL4, DBNZ, PUSH, POP, MOVM, XCHM, CMPME, CMPMNE, CMPMNC, CMPMC, MOVBK, XCHBK, CMPBKE, CMPBKNE, CMPBKNC, CMPBKC, CHKL, CHKLA

Table 19-1. List of Instructions by 8-Bit Addressing (1/2)

2nd Operand / 1st Operand	# byte	A	r r'	saddr saddr'	sfr
A	(MOV) ADD **Note 1**	(MOV) (XCH) (ADD) **Note 1**	MOV XCH (ADD) **Note 1**	(MOV) **Note 6** (XCH) **Note 6** (ADD) **Notes 1, 6**	MOV (XCH) (ADD) **Note 1**
r	MOV ADD **Note 1**	(MOV) (XCH) (ADD) **Note 1**	MOV XCH ADD **Note 1**	MOV XCH ADD **Note 1**	MOV XCH ADD **Note 1**
saddr	MOV ADD **Note 1**	(MOV) **Note 6** (ADD) **Note 1**	MOV ADD **Note 1**	MOV XCH ADD **Note 1**	
sfr	MOV ADD **Note 1**	MOV (ADD) **Note 1**	MOV ADD **Note 1**		
!addr16 !!addr24	MOV	MOV ADD **Note 1**	MOV		
mem [saddrp] [%saddrg]		MOV ADD **Note 1**			
mem3					
r3 PSWL PSWH	MOV	MOV			
B, C					
STBC, WDM	MOV				
[TDE +] [TDE –]		(MOV) (ADD) **Note 1** MOVM **Note 4**			

(See the following page for the explanation of **Note**.)

Table 19-1. List of Instructions by 8-Bit Addressing (2/2)

2nd Operand / 1st Operand	!addr16 !!addr24	mem [saddrp] [%saddrg]	r3 PSWL PSWH	[WHL +] [WHL –]	n	None Note 2
A	(MOV) (XCH) ADD Note 1	MOV XCH ADD Note 1	MOV	(MOV) (XCH) (ADD) Note 1		
r	MOV XCH				ROR Note 3	MULU DIVUW INC DEC
saddr						INC DEC DBNZ
sfr						PUSH POP CHKL CHKLA
!addr16 !!addr24						
mem [saddrp] [%saddrg]						
mem3						ROR4 ROL4
r3 PSWL PSWH						
B, C						DBNZ
STBC, WDM						
[TDE +] [TDE –]				MOVBK Note 5		

Notes 1. ADDC, SUB, SUBC, AND, OR, XOR and CMP are the same as ADD.

2. There is no 2nd operand, or the 2nd operand is not an operand address.

3. ROL, RORC, ROLC, SHR and SHL are the same as ROR.

4. XCHM, CMPME, CMPMNE, CMPMNC and CMPMC are the same as MOVM.

5. XCHBK, CMPBKE, CMPBKNE, CMPBKNC and CMPBKC are the same as MOVBK.

6. If saddr is saddr2 in this combination, there is a short code length instruction.

(2) 16-bit instructions (combinations expressed by writing AX for rp are shown in parentheses)

MOVM, XCHW, ADDW, SUBW, CMPW, MULUW, MULW, DIVUX, INCW, DECW, SHRW, SHLW, PUSH, POP, ADDWG, SUBWG, PUSHU, POPU, MOVTBLW, MACW, MACSW, SACW

Table 19-2. List of Instructions by 16-Bit Addressing (1/2)

2nd Operand / 1st Operand	# word	AX	rp rp'	saddrp saddrp'	sfrp
AX	(MOVW) ADDW **Note 1**	(MOVW) (XCHW) (ADD) **Note 1**	(MOVW) (XCHW) (ADDW) **Note 1**	(MOVW) **Note 3** (XCHW) **Note 3** (ADDW) **Notes 1,3**	MOVW (XCHW) (ADDW) **Note 1**
rp	MOVW ADDW **Note 1**	(MOVW) (XCHW) (ADDW) **Note 1**	MOVW XCHW ADDW **Note 1**	MOVW XCHW ADDW **Note 1**	MOVW XCHW ADDW **Note 1**
saddrp	MOVW ADDW **Note 1**	(MOVW) **Note 3** (ADDW) **Note 1**	MOVW ADDW **Note 1**	MOVW XCHW ADDW **Note 1**	
sfrp	MOVW ADDW **Note 1**	MOVW (ADDW) **Note 1**	MOVW ADDW **Note 1**		
!addr16 !!addr24	MOVW	(MOVW)	MOVW		
mem [saddrp] [%saddrg]		MOVW			
PSW					
SP	ADDWG SUBWG				
post					
[TDE +]		(MOVW)			
byte					

(See the following page for the explanation of **Note**.)

Table 19-2. List of Instructions by 16-Bit Addressing (2/2)

2nd Operand / 1st Operand	!!addr16 !!addr24	mem [saddrp] [%saddrg]	[WHL +]	byte	n	None Note 2
AX	(MOVW) XCHW	MOVW XCHW	(MOVW) (XCHW)			
rp	MOVW				SHRW SHLW	MULW Note 4 INCW DECW
saddrp						INCW DECW
sfrp						PUSH POP
!addr16 !!addr24				MOVTBLW		
mem [saddrp] [%saddrg]						
PSW						PUSH POP
SP						
post						PUSH POP PUSHU POPU
[TDE +]			SACW			
byte						MACW MACSW

Notes 1. SUBW and CMPW are the same as ADDW.

2. There is no 2nd operand, or the 2nd operand is not an operand address.

3. If saddrp is saddrp2 in this combination, there is a short code length instruction.

4. MULUW and DIVUX are the same as MULW.

(3) 24-bit instructions (combinations expressed by writing WHL for rg are shown in parentheses)

MOVG, ADDG, SUBG, INCG, DECG, PUSH, POP

Table 19-3. List of Instructions by 24-Bit Addressing

2nd Operand / 1st Operand	# imm24	WHL	rg rg'	saddrg	!!addr24	mem1	[%saddrg]	SP	None*
WHL	(MOVG) (ADDG) (SUBG)	(MOVG) (ADDG) (SUBG)	(MOVG) (ADDG) (SUBG)	(MOVG) ADDG SUBG	(MOVG)	MOVG	MOVG	MOVG	
rg	MOVG ADDG SUBG	(MOVG) (ADDG) (SUBG)	MOVG ADDG SUBG	MOVG	MOVG				INCG DECG PUSH POP
saddrg		(MOVG)	MOVG						
!!addr24		(MOVG)	MOVG						
mem1		MOVG							
[%saddrg)		MOVG							
SP	MOVG	MOVG							INCG DECG

* There is no 2nd operand, or the 2nd operand is not an operand address.

(4) Bit manipulation instructions

MOV1, AND1, OR1, XOR1, SET1, CLR1, NOT1, BT, BF, BTCLR, BFSET

Table 19-4. List of Instructions by Bit Manipulation Instruction Addressing

2nd Operand / 1st Operand	CY	saddr. bit sfr. bit A.bit X. bit PSWL. bit PSWH. bit mem2. bit !addr16. bit !!addr24. bit	/saddr.bit /sfr. bit /A. bit /X. bit /PSWL. bit /PSWH. bit /mem2. bit /!addr16. bit /!!addr24. bit	None*
CY		MOV1 AND1 OR1 XOR1	AND1 OR1	NOT SET1 CLR1
saddr. bit sfr. bit A. bit X. bit PSWL. bit PSWH. bit mem2. bit !addr16. bit !!addr24. bit	MOV1			NOT1 SET1 CLR1 BF BT BTCLR BFSET

* There is no 2nd operand, or the 2nd operand is not an operand address.

(5) Call/return instructions / branch instructions

CALL, CALLF, CALLT, BRK, RET, RETI, RETB, RETCS, RETCSB, BRKCS, BR, BNZ, BNE, BZ, BE, BNC, BNL, BC, BL, BNV, BPO, BV, BPE, BP, BN, BLT, BGE, BLE, BGT, BNH, BH, BF, BT, BTCLR, BFSET, DBNZ

Table 19-5. List of Instructions by Call/Return Instruction/Branch Instruction Addressing

Instruction Address Operand	$addr20	$!addr20	!addr16	!!addr20	rp	rg	[rp]	[rg]	!addr11	[addr5]	RBn	None
Basic instructions	BC* BR	CALL BR	CALL BR RETCS RETCSB	CALL BR	CALL BR	CALL BR	CALL BR	CALL BR	CALLF	CALLT	BRKCS	BRK RET RETI RETB
Compound instructions	BF BT BTCLR BFSET DBNZ											

* BNZ, BNE, BZ, BE, BNC, BNL, BL, BNV, BPO, BV, BPE, BP, BN, BLT, BGE, BLE, BGT, BNH, and BH are the same as BC.

(6) Other instructions

ADJBA, ADJBS, CVTBW, LOCATION, SEL, NOT, EI, DI, SWRS

Refer to **CHAPTER 24 TIMING CHARTS** for the timing charts.

Absolute Maximum Ratings (T_A = 25°C)

Parameter	Symbol	Conditions		Ratings	Unit
Supply voltage	V_{DD}			–0.5 to +7.0	V
	AV_{DD}			–0.5 to V_{DD} + 0.5	V
	AV_{SS}			–0.5 to +0.5	V
Input voltage	V_I	**Note 1**		–0.5 to V_{DD} + 0.5 ≤ 7.0	V
Output voltage	V_O			–0.5 to V_{DD} + 0.5	V
Output current, low	I_{OL}	All output pins		15	mA
		Total of all output pins		150	mA
Output current, high	I_{OH}	All output pins		–10	mA
		Total of all output pins		–100	mA
Analog input voltage	V_{IAN}	**Note 2**	$AV_{DD} > V_{DD}$	–0.5 to V_{DD} + 0.5	V
			$V_{DD} \geq AV_{DD}$	–0.5 to AV_{DD} + 0.5	
A/D converter reference input voltage	AV_{REF}		$AV_{DD} > V_{DD}$	–0.5 to V_{DD} + 0.5	V
			$V_{DD} \geq AV_{DD}$	–0.5 to AV_{DD} + 0.5	
Operating ambient temperature	T_A			–10 to +70	°C
Storage temperature	T_{stg}			–65 to +150	°C

Notes 1. Pins other than the pins in **Note 2**.

2. Pins P70/ANI0 to P77/ANI7, P80/ANI8 to P87/ANI15

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 the absolute maximum ratings are not exceeded.

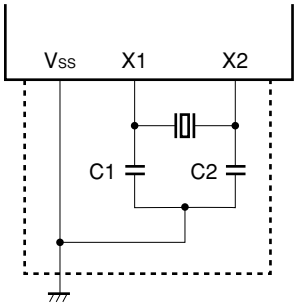
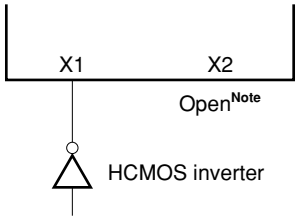
Recommended Operating Conditions

Oscillation Frequency	T_A	V_{DD}
8 MHz ≤ f_{XX} ≤ 32 MHz	–10 to +70°C	4.5 to 5.5 V

Capacitance (T_A = 25°C, V_{SS} = V_{DD} = 0 V)

Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Input capacitance	C_I	f = 1 MHz			10	pF
Output capacitance	C_O	Unmeasured pins returned to 0 V.			10	pF
I/O capacitance	C_{IO}				10	pF

Oscillator Characteristics (T_A = –10 to +70°C, V_{DD} = 4.5 to 5.5 V, V_{SS} = 0 V)

Resonator	Recommended Circuit	Item	MIN.	MAX.	Unit
Ceramic resonator or crystal resonator		Oscillation frequency (f_{XX})	8	32	MHz
External clock		X1 input frequency (f_X)	8	32	MHz
		X1 input rise, fall time	0	5	ns
		X1 input high-, low-level width	20	105	ns

Note When the EXTC bit of the oscillation stabilization time specification register (OSTS) = 0. Input the reverse phase clock of the pin X1 to the pin X2 when the EXTC bit = 1.

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

- **Keep the wiring length as short as possible.**
- **Do not cross the wiring with any 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.**

Remark For the resonator selection and oscillator constant, customers are required to either evaluate the oscillation themselves or apply to the resonator manufacturer for evaluation.

DC Characteristics (T_A = –10 to +70°C, V_{DD} = 4.5 to 5.5 V, V_{SS} = 0 V)

Parameter	Symbol	Conditions		MIN.	TYP.	MAX.	Unit
Input voltage, low	V_{IL}			0		0.8	V
Input voltage, high	V_{IH1}	**Note 1**		2.2		V_{DD}	V
	V_{IH2}	**Note 2**		0.8V_{DD}		V_{DD}	
Output voltage, low	V_{OL}	I_{OL} = 2.0 mA				0.45	V
Output voltage, high	V_{OH}	I_{OH} = –400 μA		V_{DD} – 1.0			V
Input leakage current	I_{LI}	**Note 3**	0 V ≤ V_I ≤ V_{DD}			±10	μA
Analog pin input leakage current	I_{LIAN}	**Note 4**	0 V ≤ V_I ≤ AV_{DD}			±1	μA
Output leakage current	I_{LO}	0 V ≤ V_O ≤ V_{DD}				±10	μA
V_{DD} supply current	I_{DD1}	Operating mode (f_{XX} = 32 MHz)			50	80	mA
	I_{DD2}	HALT mode (f_{XX} = 32 MHz)			30	60	mA
	I_{DD3}	IDLE mode (f_{XX} = 32 MHz)			10	20	mA
Data retention voltage	V_{DDDR}	STOP mode		2.5			V
Data retention current	I_{DDDR}	STOP mode	V_{DDDR} = 2.5 V		2	15	μA
			V_{DDDR} = 5 V ±10%		15	50	μA
Pull-up resistor	R_L			15	40	80	kΩ

Notes 1. Pins other than pins in the **Note 2**

2. P20/NMI, P21/INTP0/TO00, P22/INTP1/TO01, P23/INTP2/TO02, P24/INTP3/TO03, P25/INTP4, P26/INTP5/TI2, P27/INTP6/TI3, P34/ASCK/$\overline{\text{SCK1}}$, P37/ASCK2/$\overline{\text{SCK2}}$, X1, X2, $\overline{\text{RESET}}$

3. Input and I/O pins (except X1 and X2, and P70/ANI0 to P77/ANI7, P80/ANI8 to P87/ANI15 used as analog inputs)

4. Pins P70/ANI0 to P77/ANI7, P80/ANI8 to P87/ANI15 (pins used as analog input, only during the non-sampling operation)

AC Characteristics (T_A = –10 to +70°C, V_{DD} = 4.5 to 5.5 V, V_{SS} = 0 V)

(1) Read/write operation

Parameter	Symbol	Expression	MIN.	MAX.	Unit
System clock cycle time	t_{CYK}		62.5	250	ns
Address setup time (to ASTB↓)	t_{SAST}	(0.5 + a) T – 20	11.2		ns
Address hold time (from ASTB↓)	t_{HSTA}	0.5T – 20	11.2		ns
ASTB high-level width	t_{WSTH}	(0.5 + a) T – 17	14.2		ns
$\overline{RD}$↓ delay time from address	t_{DAR}	(1 + a) T – 15	47.5		ns
Address float time from $\overline{RD}$↓	t_{FRA}			0	ns
Data input time from address	t_{DAID}	(2.5 + a + n) T – 56		100.2	ns
Data input time from $\overline{RD}$↓	t_{DRID}	(1.5 + n) T – 48		45.7	ns
$\overline{RD}$↓ delay time from ASTB↓	t_{DSTR}	0.5T – 16	15.3		ns
Data hold time (to $\overline{RD}$↑)	t_{HRID}		0		ns
Address active time from $\overline{RD}$↑	t_{DRA}	0.5T – 14	17.2		ns
$\overline{RD}$ low-level width	t_{WRL}	(1.5 + n) T – 30	63.7		ns
$\overline{LWR}$, $\overline{HWR}$↓ delay time from address	t_{DAW}	(1 + a) T – 15	47.5		ns
Data output time from $\overline{LWR}$, $\overline{HWR}$↓	t_{DWOD}			15	ns
$\overline{LWR}$, $\overline{HWR}$↓ delay time from ASTB↓	t_{DSTW}	0.5T – 16	15.3		ns
Data setup time (to $\overline{LWR}$, $\overline{HWR}$↑)	t_{SODW}	(1.5 + n) T – 25	68.7		ns
Data hold time (from $\overline{LWR}$, $\overline{HWR}$↑)	t_{HWOD}	0.5T – 14	17.2		ns
ASTB↑ delay time from $\overline{LWR}$, $\overline{HWR}$↑	t_{DWST}	1.5T – 15	78.8		ns
$\overline{LWR}$, $\overline{HWR}$ low-level width	t_{WWL}	(1.5 + n) T – 36	57.7		ns
$\overline{WAIT}$↓ input time from address	t_{DAWT}	(2 + a) T – 50		75	ns
$\overline{WAIT}$↓ input time from ASTB↓	t_{DSTWT}	1.5T – 40		53.7	ns
$\overline{WAIT}$ hold time from ASTB↓	t_{HSTWT}	(1.5 + n) T + 5	98.8		ns
$\overline{WAIT}$↑ delay time from ASTB↓	t_{DSTWTH}	(2.5 + n) T – 40		116.2[Note]	ns
$\overline{WAIT}$↓ input time from $\overline{RD}$↓	t_{DRWT}	T – 40		22.5	ns
$\overline{WAIT}$ hold time from $\overline{RD}$↓	t_{HRWT}	(1 + n) T + 5	67.5		ns
$\overline{WAIT}$↑ delay time from $\overline{RD}$↓	t_{DRWTH}	(1 + n) T – 40		85[Note]	ns
$\overline{WAIT}$↓ input time from $\overline{LWR}$, $\overline{HWR}$↓	t_{DWWT}	T – 40		22.5	ns
$\overline{WAIT}$ hold time from $\overline{LWR}$, $\overline{HWR}$↓	t_{HWWT}	(1 + n) T + 5	67.5		ns
$\overline{WAIT}$↑ delay time from $\overline{LWR}$, $\overline{HWR}$↓	t_{DWWTH}	(1 + n) T – 40		85[Note]	ns

Note Specification when an external wait is inserted

Remarks 1. T = t_{CYK} = 1/f_{CLK} (f_{CLK} is internal system clock frequency)

2. a = 1 when an address wait is inserted, otherwise, 0.

3. n indicates the number of the wait cycles by specifying the external wait pins ($\overline{WAIT}$) or programmable wait control registers 1, 2 (PWC1, PWC2). ($n \geq 0$. $n \geq 1$ for t_{DSTWTH}, t_{DRWTH}, t_{DWWTH}).

4. Calculate values in the expression column with the system clock cycle time to be used because these values depend on the system clock cycle time (t_{CYK} = T). The values in the above expression column are calculated based on T = 62.5 ns.

(2) Serial Operation (T_A = –10 to +70°C, V_{DD} = 4.5 to 5.5 V, V_{SS} = 0 V)

Parameter	Symbol	Conditions		MIN.	MAX.	Unit
Serial clock cycle time	t_{CYSK}	$\overline{SCK1}$, $\overline{SCK2}$ output	BRG	T_{SFT}		ns
		$\overline{SCK1}$, $\overline{SCK2}$ input	External clock	500		ns
Serial clock low-level width	t_{WSKL}	$\overline{SCK1}$, $\overline{SCK2}$ output	BRG	$0.5T_{SFT}$–40		ns
		$\overline{SCK1}$, $\overline{SCK2}$ input	External clock	210		ns
Serial clock high-level width	t_{WSKH}	$\overline{SCK1}$, $\overline{SCK2}$ output	BRG	$0.5T_{SFT}$–40		ns
		$\overline{SCK1}$, $\overline{SCK2}$ input	External clock	210		ns
SI1, SI2 setup time (to $\overline{SCK1}$, $\overline{SCK2}\uparrow$)	t_{SSSK}			80		ns
SI1, SI2 hold time (from $\overline{SCK1}$, $\overline{SCK2}\uparrow$)	t_{HSSK}			80		ns
SO1, SO2 output delay time from $\overline{SCK1}$, $\overline{SCK2}\downarrow$	t_{DSBSK}	R = 1 kΩ, C = 100 pF		0	150	ns

Remarks 1. T_{SFT} is a value set in software. The minimum value is $t_{CYK} \times 8$.

2. $t_{CYK} = 1/f_{CLK}$ (f_{CLK} is internal system clock frequency)

(3) Other Operations (T_A = –10 to +70°C, V_{DD} = 4.5 to 5.5 V, V_{SS} = 0 V)

Parameter	Symbol	Conditions	MIN.	MAX.	Unit
NMI high-, low-level width	t_{WNIH}, t_{WNIL}		10		μs
INTP0 to INTP6 high-, low-level width	t_{WITH}, t_{WITL}		4		t_{CYSMP}
$\overline{RESET}$ high-, low-level width	t_{WRSH}, t_{WRSL}		10		μs

Remarks 1. t_{CYSMP} is a sampling clock set in the noise protection control register (NPC) in software.

When NIn = 0, $t_{CYSMP} = t_{CYK}$

When NIn = 1, $t_{CYSMP} = t_{CYK} \times 4$

2. $t_{CYK} = 1/f_{CLK}$ (f_{CLK} is internal system clock frequency)

3. NIn: Bit n of NPC (n = 0 to 6)

A/D Converter Characteristics (T_A = –10 to +70 °C, V_{DD} = 4.5 to 5.5 V, V_{SS} = AV_{SS} = 0 V, V_{DD} – 0.5 V ≤ AV_{DD} ≤ V_{DD})

Parameter	Symbol	Conditions		MIN.	TYP.	MAX.	Unit
Resolution				10			bit
Total error[Note 1]		4.5 V ≤ AV_{REF} ≤ AV_{DD}				±0.5	%FSR[Note 2]
		3.4 V ≤ AV_{REF} < 4.5 V				±0.7	%FSR[Note 2]
Quantization error						±1/2	LSB
Conversion time	t_{CONV}	80 ns ≤ t_{CYK} ≤ 250 ns		169			t_{CYK}
		62.5 ns ≤ t_{CYK} < 80 ns		208			t_{CYK}
Sampling time	t_{SAMP}	80 ns ≤ t_{CYK} ≤ 250 ns		20			t_{CYK}
		62.5 ns ≤ t_{CYK} < 80 ns		24			t_{CYK}
Zero-scale error[Note 1]		4.5 V ≤ AV_{REF} ≤ AV_{DD}			±1.5	±3.5	LSB
		3.4 V ≤ AV_{REF} < 4.5 V			±1.5	±4.5	LSB
Full-scale error[Note 1]		4.5 V ≤ AV_{REF} ≤ AV_{DD}			±1.5	±3.5	LSB
		3.4 V ≤ AV_{REF} < 4.5 V			±1.5	±4.5	LSB
Integral linearity error[Note 1]		4.5 V ≤ AV_{REF} ≤ AV_{DD}			±1.5	±2.5	LSB
		3.4 V ≤ AV_{REF} < 4.5 V			±1.5	±4.5	LSB
Analog input voltage	V_{IAN}			–0.3		AV_{REF}+0.3	V
A/D converter reference input voltage	AV_{REF}			3.4		AV_{DD}	V
AV_{REF} current	AI_{REF}				1.0	3.0	mA
AV_{DD} supply current	AI_{DD}				2.0	6.0	mA
A/D converter data retention current	AI_{DDDR}	STOP mode	AV_{DDDR} = 2.5 V		2	10	μA
			AV_{DDDR} = 5 V ±10%		10	50	μA

Notes 1. The quantization error is excluded.

2. Indicated as a ratio (%FSR) to the full-scale value.

Remark t_{CYK} = 1/f_{CLK} (f_{CLK} is internal system clock frequency).

Refer to **CHAPTER 24 TIMING CHARTS** for the timing charts.

Absolute Maximum Ratings (T_A = 25°C)

Parameter	Symbol	Conditions		Ratings	Unit
Supply voltage	V_{DD}			–0.5 to +7.0	V
	AV_{DD}			–0.5 to V_{DD} + 0.5	V
	AV_{SS}			–0.5 to +0.5	V
Input voltage	V_I	**Note 1**		–0.5 to V_{DD} + 0.5 ≤ 7.0	V
Output voltage	V_O			–0.5 to V_{DD} + 0.5	V
Output current, low	I_{OL}	All output pins		15	mA
		Total of all output pins		150	mA
Output current, high	I_{OH}	All output pins		–10	mA
		Total of all output pins		–100	mA
Analog input voltage	V_{IAN}	**Note 2**	$AV_{DD} > V_{DD}$	–0.5 to V_{DD} + 0.5	V
			$V_{DD} \geq AV_{DD}$	–0.5 to AV_{DD} + 0.5	
A/D converter reference input voltage	AV_{REF}		$AV_{DD} > V_{DD}$	–0.5 to V_{DD} + 0.5	V
			$V_{DD} \geq AV_{DD}$	–0.5 to AV_{DD} + 0.5	
Operating temperature	T_A			–40 to +85	°C
Storage temperature	T_{stg}			–65 to +150	°C

Notes 1. Pins other than the pins in **Note 2**.

2. Pins P70/ANI0 to P77/ANI7, P80/ANI8 to P87/ANI15

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 the absolute maximum ratings are not exceeded.

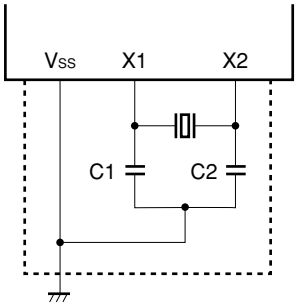
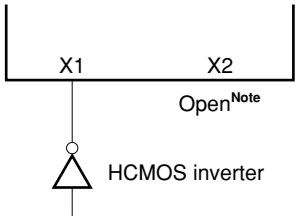
Recommended Operating Conditions

Oscillation Frequency	T_A	V_{DD}
8 MHz ≤ f_{XX} ≤ 25 MHz	–40 to +85°C	4.5 to 5.5 V

Capacitance (T_A = 25°C, V_{SS} = V_{DD} = 0 V)

Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Input capacitance	C_I	f = 1 MHz			10	pF
Output capacitance	C_O	Unmeasured pins returned to 0 V.			10	pF
I/O capacitance	C_{IO}				10	pF

Oscillator Characteristics (T_A = –40 to +85°C, V_{DD} = 4.5 to 5.5 V, V_{SS} = 0 V)

Resonator	Recommended Circuit	Item	MIN.	MAX.	Unit
Ceramic resonator or crystal resonator		Oscillation frequency (f_{XX})	8	25	MHz
External clock		X1 input frequency (f_X)	8	25	MHz
		X1 input rise, fall time	0	5	ns
		X1 input high-, low-level width	20	105	ns

Note When the EXTC bit of the oscillation stabilization time specification register (OSTS) = 0. Input the reverse phase clock of the pin X1 to the pin X2 when the EXTC bit = 1.

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

- **Keep the wiring length as short as possible.**
- **Do not cross the wiring with any 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.**

Remark For the resonator selection and oscillator constant, customers are required to either evaluate the oscillation themselves or apply to the resonator manufacturer for evaluation.

DC Characteristics (T_A = –40 to +85°C, V_{DD} = 4.5 to 5.5 V, V_{SS} = 0 V)

Parameter	Symbol	Conditions		MIN.	TYP.	MAX.	Unit
Input voltage, low	V_{IL}			0		0.8	V
Input voltage, high	V_{IH1}	**Note 1**		2.2		V_{DD}	V
	V_{IH2}	**Note 2**		0.8V_{DD}		V_{DD}	
Output voltage, low	V_{OL}	I_{OL} = 2.0 mA				0.45	V
Output voltage, high	V_{OH}	I_{OH} = –400 μA		V_{DD} – 1.0			V
Input leakage current	I_{LI}	**Note 3**	0 V ≤ V_I ≤ V_{DD}			±10	μA
Analog pin input leakage current	I_{LIAN}	**Note 4**	0 V ≤ V_I ≤ AV_{DD}			±1	μA
Output leakage current	I_{LO}	0 V ≤ V_O ≤ V_{DD}				±10	μA
V_{DD} supply current	I_{DD1}	Operating mode (f_{XX} = 25 MHz)			40	70	mA
	I_{DD2}	HALT mode (f_{XX} = 25 MHz)			25	50	mA
	I_{DD3}	IDLE mode (f_{XX} = 25 MHz)			10	20	mA
Data retention voltage	V_{DDDR}	STOP mode		2.5			V
Data retention current	I_{DDDR}	STOP mode	V_{DDDR} = 2.5 V		2	15	μA
			V_{DDDR} = 5 V ±10 %		15	50	μA
Pull-up resistor	R_L			15	40	80	kΩ

Notes 1. Pins other than pins in **Note 2**.

2. P20/NMI, P21/INTP0/TO00, P22/INTP1/TO01, P23/INTP2/TO02, P24/INTP3/TO03, P25/INTP4, P26/INTP5/TI2, P27/INTP6/TI3, P34/ASCK/$\overline{\text{SCK1}}$, P37/ASCK2/$\overline{\text{SCK2}}$, X1, X2, $\overline{\text{RESET}}$

3. Input and I/O pins (except X1 and X2, and P70/ANI0 to P77/ANI7, P80/ANI8 to P87/ANI15 used as analog inputs)

4. Pins P70/ANI0 to P77/ANI7, P80/ANI8 to P87/ANI15 (pins used as analog input, only during the non-sampling operation)

AC Characteristics (T_A = –40 to +85°C, V_{DD} = 4.5 to 5.5 V, V_{SS} = 0 V)

(1) Read/write operation

Parameter	Symbol	Expression	MIN.	MAX.	Unit
System clock cycle time	t_{CYK}		80	250	ns
Address setup time (to ASTB↓)	t_{SAST}	(0.5 + a) T – 20	20		ns
Address hold time (from ASTB↓)	t_{HSTA}	0.5T – 20	20		ns
ASTB high-level width	t_{WSTH}	(0.5 + a) T – 17	23		ns
$\overline{RD}$↓ delay time from address	t_{DAR}	(1 + a) T – 15	65		ns
Address float time from $\overline{RD}$↓	t_{FRA}			0	ns
Data input time from address	t_{DAID}	(2.5 + a + n) T – 56		144	ns
Data input time from $\overline{RD}$↓	t_{DRID}	(1.5 + n) T – 48		72	ns
$\overline{RD}$↓ delay time from ASTB↓	t_{DSTR}	0.5T – 16	24		ns
Data hold time (to $\overline{RD}$↑)	t_{HRID}		0		ns
Address active time from $\overline{RD}$↑	t_{DRA}	0.5T – 14	26		ns
$\overline{RD}$ low-level width	t_{WRL}	(1.5 + n) T – 30	90		ns
$\overline{LWR}$, $\overline{HWR}$↓ delay time from address	t_{DAW}	(1 + a) T – 15	65		ns
Data output time from $\overline{LWR}$, $\overline{HWR}$↓	t_{DWOD}			15	ns
$\overline{LWR}$, $\overline{HWR}$↓ delay time from ASTB↓	t_{DSTW}	0.5T – 16	24		ns
Data setup time (to $\overline{LWR}$, $\overline{HWR}$↑)	t_{SODW}	(1.5 + n) T – 25	95		ns
Data hold time (from $\overline{LWR}$, $\overline{HWR}$↑)	t_{HWOD}	0.5T – 14	26		ns
ASTB↑ delay time from $\overline{LWR}$, $\overline{HWR}$↑	t_{DWST}	1.5T – 15	105		ns
$\overline{LWR}$, $\overline{HWR}$ low-level width	t_{WWL}	(1.5 + n) T – 36	84		ns
$\overline{WAIT}$↓ input time from address	t_{DAWT}	(2 + a) T – 50		110	ns
$\overline{WAIT}$↓ input time from ASTB↓	t_{DSTWT}	1.5T – 40		80	ns
$\overline{WAIT}$ hold time from ASTB↓	t_{HSTWT}	(1.5 + n) T + 5	125		ns
$\overline{WAIT}$↑ delay time from ASTB↓	t_{DSTWTH}	(1.5 + n) T – 40		160[Note]	ns
$\overline{WAIT}$↓ input time from $\overline{RD}$↓	t_{DRWT}	T – 40		40	ns
$\overline{WAIT}$ hold time from $\overline{RD}$↓	t_{HRWT}	(1 + n) T + 5	85		ns
$\overline{WAIT}$↑ delay time from $\overline{RD}$↓	t_{DRWTH}	(1 + n) T – 40		120[Note]	ns
$\overline{WAIT}$↓ input time from $\overline{LWR}$, $\overline{HWR}$↓	t_{DWWT}	T – 40		40	ns
$\overline{WAIT}$ hold time from $\overline{LWR}$, $\overline{HWR}$↓	t_{HWWT}	(1 + n) T + 5	85		ns
$\overline{WAIT}$↑ delay time from $\overline{LWR}$, $\overline{HWR}$↓	t_{DWWTH}	(1 + n) T – 40		120[Note]	ns

Note Specification when an external wait is inserted

Remarks 1. T = t_{CYK} = 1/f_{CLK} (f_{CLK} is internal system clock frequency)

2. a = 1 when an address wait is inserted, otherwise, 0.

3. n indicates the number of the wait cycles by specifying the external wait pins ($\overline{WAIT}$) or programmable wait control registers 1, 2 (PWC1, PWC2). ($n \geq 0$. $n \geq 1$ for t_{DSTWTH}, t_{DRWTH}, t_{DWWTH}).

4. Calculate values in the expression column with the system clock cycle time to be used because these values depend on the system clock cycle time (t_{CYK} = T). The values in the above expression column are calculated based on T = 80 ns.

(2) Serial Operation (T_A = –40 to +85°C, V_{DD} = 4.5 to 5.5 V, V_{SS} = 0 V)

Parameter	Symbol	Conditions		MIN.	MAX.	Unit
Serial clock cycle time	t_{CYSK}	$\overline{SCK1}$, $\overline{SCK2}$ output	BRG	T_{SFT}		ns
		$\overline{SCK1}$, $\overline{SCK2}$ input	External clock	640		ns
Serial clock low-level width	t_{WSKL}	$\overline{SCK1}$, $\overline{SCK2}$ output	BRG	$0.5T_{SFT}$–40		ns
		$\overline{SCK1}$, $\overline{SCK2}$ input	External clock	280		ns
Serial clock high-level width	t_{WSKH}	$\overline{SCK1}$, $\overline{SCK2}$ output	BRG	$0.5T_{SFT}$–40		ns
		$\overline{SCK1}$, $\overline{SCK2}$ input	External clock	280		ns
SI1, SI2 setup time (to $\overline{SCK1}$, $\overline{SCK2}\uparrow$)	t_{SSSK}			80		ns
SI1, SI2 hold time (from $\overline{SCK1}$, $\overline{SCK2}\uparrow$)	t_{HSSK}			80		ns
SO1, SO2 output delay time from $\overline{SCK1}$, $\overline{SCK2}\downarrow$	t_{DSBSK}	R = 1 kΩ, C = 100 pF		0	150	ns

Remarks 1. T_{SFT} is a value set in software. The minimum value is $t_{CYK} \times 8$.

2. $t_{CYK} = 1/f_{CLK}$ (f_{CLK} is internal system clock frequency)

(3) Other Operations (T_A = –40 to +85°C, V_{DD} = 4.5 to 5.5 V, V_{SS} = 0 V)

Parameter	Symbol	Conditions	MIN.	MAX.	Unit
NMI high, low-level width	t_{WNIH}, t_{WNIL}		10		μs
INTP0-INTP6 high, low-level width	t_{WITH}, t_{WITL}		4		t_{CYSMP}
$\overline{RESET}$ high, low-level width	t_{WRSH}, t_{WRSL}		10		μs

Remarks 1. t_{CYSMP} is a sampling clock set in the noise protection control register (NPC) in software.

When NIn = 0, $t_{CYSMP} = t_{CYK}$

When NIn = 1, $t_{CYSMP} = t_{CYK} \times 4$

2. $t_{CYK} = 1/f_{CLK}$ (f_{CLK} is internal system clock frequency)

3. NIn: Bit n of NPC (n = 0-6)

A/D Converter Characteristics (T_A = –40 to +85°C, V_{DD} = 4.5 to 5.5 V, V_{SS} = AV_{SS} = 0 V, V_{DD} – 0.5 V ≤ AV_{DD} ≤ V_{DD})

Parameter	Symbol	Conditions		MIN.	TYP.	MAX.	Unit
Resolution				10			bit
Total error[Note 1]		4.5 V ≤ AV_{REF} ≤ AV_{DD}				±0.5	%FSR[Note 2]
		3.4 V ≤ AV_{REF} < 4.5 V				±0.7	%FSR[Note 2]
Quantization error						±1/2	LSB
Conversion time	t_{CONV}	80 ns ≤ t_{CYK} ≤ 250 ns		169			t_{CYK}
Sampling time	t_{SAMP}	80 ns ≤ t_{CYK} ≤ 250 ns		20			t_{CYK}
Zero-scale error[Note 1]		4.5 V ≤ AV_{REF} ≤ AV_{DD}			±1.5	±3.5	LSB
		3.4 V ≤ AV_{REF} < 4.5 V			±1.5	±4.5	LSB
Full-scale error[Note 1]		4.5 V ≤ AV_{REF} ≤ AV_{DD}			±1.5	±3.5	LSB
		3.4 V ≤ AV_{REF} < 4.5 V			±1.5	±4.5	LSB
Integral linearity error[Note 1]		4.5 V ≤ AV_{REF} ≤ AV_{DD}			±1.5	±2.5	LSB
		3.4 V ≤ AV_{REF} < 4.5 V			±1.5	±4.5	LSB
Analog input voltage	V_{IAN}			–0.3		AV_{REF}+0.3	V
A/D converter reference input voltage	AV_{REF}			3.4		AV_{DD}	V
AV_{REF} current	AI_{REF}				1.0	3.0	mA
AV_{DD} supply current	AI_{DD}				2.0	6.0	mA
A/D converter data retention current	AI_{DDDR}	STOP mode	AV_{DDDR} = 2.5 V		2	10	μA
			AV_{DDDR} = 5 V ±10%		10	50	μA

Notes 1. The quantization error is excluded.

2. Indicated as a ratio (%FSR) to the full-scale value.

Remark t_{CYK} = 1/f_{CLK} (f_{CLK} is internal system clock frequency).

Refer to **CHAPTER 24 TIMING CHARTS** for the timing charts.

Absolute Maximum Ratings (T_A = 25°C)

Parameter	Symbol	Conditions		Ratings	Unit
Supply voltage	V_{DD}			–0.5 to +7.0	V
	AV_{DD}			–0.5 to V_{DD} + 0.5	V
	AV_{SS}			–0.5 to +0.5	V
Input voltage	V_I	**Note 1**		–0.5 to V_{DD} + 0.5 ≤ 7.0	V
Output voltage	V_O			–0.5 to V_{DD} + 0.5	V
Output current, low	I_{OL}	All output pins		15	mA
		Total of all output pins		150	mA
Output current, high	I_{OH}	All output pins		–10	mA
		Total of all output pins		–100	mA
Analog input voltage	V_{IAN}	**Note 2**	$AV_{DD} > V_{DD}$	–0.5 to V_{DD} + 0.5	V
			$V_{DD} \geq AV_{DD}$	–0.5 to AV_{DD} + 0.5	
A/D converter reference input voltage	AV_{REF}		$AV_{DD} > V_{DD}$	–0.5 to V_{DD} + 0.5	V
			$V_{DD} \geq AV_{DD}$	–0.5 to AV_{DD} + 0.5	
Operating temperature	T_A			–40 to +110	°C
Storage temperature	T_{stg}			–65 to +150	°C

Notes 1. Pins other than the pins in **Note 2**.

2. Pins P70/ANI0 to P77/ANI7, P80/ANI8 to P87/ANI15

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 the absolute maximum ratings are not exceeded.

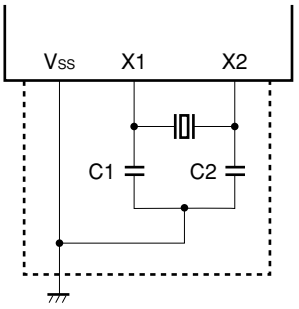
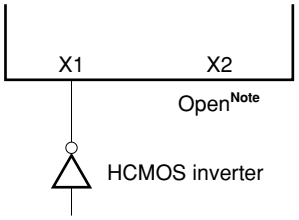
Recommended Operating Conditions

Oscillation Frequency	T_A	V_{DD}
8 MHz ≤ f_{XX} ≤ 20 MHz	–40 to +110°C	4.5 to 5.5 V

Capacitance (T_A = 25°C, V_{SS} = V_{DD} = 0 V)

Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Input capacitance	C_I	f = 1 MHz			10	pF
Output capacitance	C_O	Unmeasured pins returned to 0 V.			10	pF
I/O capacitance	C_{IO}				10	pF

Oscillator Characteristics (T_A = –40 to +110°C, V_{DD} = 4.5 to 5.5 V, V_{SS} = 0 V)

Resonator	Recommended Circuit	Item	MIN.	MAX.	Unit
Ceramic resonator or crystal resonator		Oscillation frequency (f_{XX})	8	20	MHz
External clock		X1 input frequency (f_X)	8	20	MHz
		X1 input rise, fall time	0	5	ns
		X1 input high-, low-level width	20	105	ns

Note When the EXTC bit of the oscillation stabilization time specification register (OSTS) = 0. Input the reverse phase clock of the pin X1 to the pin X2 when the EXTC bit = 1.

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

- **Keep the wiring length as short as possible.**
- **Do not cross the wiring with any 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.**

Remark For the resonator selection and oscillator constant, customers are required to either evaluate the oscillation themselves or apply to the resonator manufacturer for evaluation.

DC Characteristics (T_A = –40 to +110°C, V_{DD} = 4.5 to 5.5 V, V_{SS} = 0 V)

Parameter	Symbol	Conditions		MIN.	TYP.	MAX.	Unit
Input voltage, low	V_{IL}			0		0.8	V
Input voltage, high	V_{IH1}	**Note 1**		2.2		V_{DD}	V
	V_{IH2}	**Note 2**		0.8V_{DD}		V_{DD}	
Output voltage, low	V_{OL}	I_{OL} = 2.0 mA				0.45	V
Output voltage, high	V_{OH}	I_{OH} = –400 μA		V_{DD} – 1.0			V
Input leakage current	I_{LI}	**Note 3**	0 V ≤ V_I ≤ V_{DD}			±10	μA
Analog pin input leakage current	I_{LIAN}	**Note 4**	0 V ≤ V_I ≤ AV_{DD}			±2	μA
Output leakage current	I_{LO}	0 V ≤ V_O ≤ V_{DD}				±10	μA
V_{DD} supply current	I_{DD1}	Operating mode (f_{XX} = 20 MHz)			30	60	mA
	I_{DD2}	HALT mode (f_{XX} = 20 MHz)			15	30	mA
	I_{DD3}	IDLE mode (f_{XX} = 20 MHz)			10	20	mA
Data retention voltage	V_{DDDR}	STOP mode		2.5			V
Data retention current	I_{DDDR}	STOP mode	V_{DDDR} = 2.5 V		2	100	μA
			V_{DDDR} = 5 V ±10 %		15	1000	μA
Pull-up resistor	R_L			15	40	80	kΩ

Notes 1. Pins other than pins in **Note 2**.

2. P20/NMI, P21/INTP0/TO00, P22/INTP1/TO01, P23/INTP2/TO02, P24/INTP3/TO03, P25/INTP4, P26/INTP5/TI2, P27/INTP6/TI3, P34/ASCK/$\overline{\text{SCK1}}$, P37/ASCK2/$\overline{\text{SCK2}}$, X1, X2, $\overline{\text{RESET}}$

3. Input and I/O pins (except X1 and X2, and P70/ANI0 to P77/ANI7, P80/ANI8 to P87/ANI15 used as analog inputs)

4. Pins P70/ANI0 to P77/ANI7, P80/ANI8 to P87/ANI15 (pins used as analog input, only during the non-sampling operation)

AC Characteristics (T_A = –40 to +110°C, V_{DD} = 4.5 to 5.5 V, V_{SS} = 0 V)

(1) Read/write operation

Parameter	Symbol	Expression	MIN.	MAX.	Unit
System clock cycle time	t_{CYK}		100	250	ns
Address setup time (to ASTB↓)	t_{SAST}	(0.5 + a) T – 20	30		ns
Address hold time (from ASTB↓)	t_{HSTA}	0.5T – 20	30		ns
ASTB high-level width	t_{WSTH}	(0.5 + a) T – 17	33		ns
$\overline{RD}$↓ delay time from address	t_{DAR}	(1 + a) T – 15	85		ns
Address float time from $\overline{RD}$↓	t_{FRA}			0	ns
Data input time from address	t_{DAID}	(2.5 + a + n) T – 56		194	ns
Data input time from $\overline{RD}$↓	t_{DRID}	(1.5 + n) T – 53		97	ns
$\overline{RD}$↓ delay time from ASTB↓	t_{DSTR}	0.5T – 16	34		ns
Data hold time (to $\overline{RD}$↑)	t_{HRID}		0		ns
Address active time from $\overline{RD}$↑	t_{DRA}	0.5T – 14	36		ns
$\overline{RD}$ low-level width	t_{WRL}	(1.5 + n) T – 30	120		ns
$\overline{LWR}$, $\overline{HWR}$↓ delay time from address	t_{DAW}	(1 + a) T – 15	85		ns
Data output time from $\overline{LWR}$, $\overline{HWR}$↓	t_{DWOD}			15	ns
$\overline{LWR}$, $\overline{HWR}$↓ delay time from ASTB↓	t_{DSTW}	0.5T – 16	34		ns
Data setup time (to $\overline{LWR}$, $\overline{HWR}$↑)	t_{SODW}	(1.5 + n) T – 25	125		ns
Data hold time (from $\overline{LWR}$, $\overline{HWR}$↑)	t_{HWOD}	0.5T – 14	36		ns
ASTB↑ delay time from $\overline{LWR}$, $\overline{HWR}$↑	t_{DWST}	1.5T – 15	135		ns
$\overline{LWR}$, $\overline{HWR}$ low-level width	t_{WWL}	(1.5 + n) T – 36	114		ns
$\overline{WAIT}$↓ input time from address	t_{DAWT}	(2 + a) T – 50		150	ns
$\overline{WAIT}$↓ input time from ASTB↓	t_{DSTWT}	1.5T – 40		110	ns
$\overline{WAIT}$ hold time from ASTB↓	t_{HSTWT}	(1.5 + n) T + 5	155		ns
$\overline{WAIT}$↑ delay time from ASTB↓	t_{DSTWTH}	(1.5 + n) T – 40		210[Note]	ns
$\overline{WAIT}$↓ input time from $\overline{RD}$↓	t_{DRWT}	T – 40		60	ns
$\overline{WAIT}$ hold time $\overline{RD}$↓	t_{HRWT}	(1 + n) T + 5	105		ns
$\overline{WAIT}$↑ delay time from $\overline{RD}$↓	t_{DRWTH}	(1 + n) T – 40		160[Note]	ns
$\overline{WAIT}$↓ input time from $\overline{LWR}$, $\overline{HWR}$↓	t_{DWWT}	T – 40		60	ns
$\overline{WAIT}$ hold time from $\overline{LWR}$, $\overline{HWR}$↓	t_{HWWT}	(1 + n) T + 5	105		ns
$\overline{WAIT}$↑ delay time from $\overline{LWR}$, $\overline{HWR}$↓	t_{DWWTH}	(1 + n) T – 40		160[Note]	ns

Note Specification when an external wait is inserted

Remarks 1. T = t_{CYK} = 1/f_{CLK} (f_{CLK} is internal system clock frequency)

2. a = 1 when an address wait is inserted, otherwise, 0.

3. n indicates the number of the wait cycles by specifying the external wait pins ($\overline{WAIT}$) or programmable wait control registers 1, 2 (PWC1, PWC2). (n ≥ 0. n ≥ 1 for t_{DSTWTH}, t_{DRWTH}, t_{DWWTH}).

4. Calculate values in the expression column with the system clock cycle time to be used because these values depend on the system clock cycle time (t_{CYK} = T). The values in the above expression column are calculated based on T = 100 ns.

(2) Serial Operation (T_A = –40 to +110°C, V_{DD} = 4.5 to 5.5 V, V_{SS} = 0 V)

Parameter	Symbol	Conditions		MIN.	MAX.	Unit
Serial clock cycle time	t_{CYSK}	$\overline{SCK1}$, $\overline{SCK2}$ output	BRG	T_{SFT}		ns
		$\overline{SCK1}$, $\overline{SCK2}$ input	External clock	800		ns
Serial clock low-level width	t_{WSKL}	$\overline{SCK1}$, $\overline{SCK2}$ output	BRG	$0.5T_{SFT}$–40		ns
		$\overline{SCK1}$, $\overline{SCK2}$ input	External clock	360		ns
Serial clock high-level width	t_{WSKH}	$\overline{SCK1}$, $\overline{SCK2}$ output	BRG	$0.5T_{SFT}$–40		ns
		$\overline{SCK1}$, $\overline{SCK2}$ input	External clock	360		ns
SI1, SI2 setup time (to $\overline{SCK1}$, $\overline{SCK2}\uparrow$)	t_{SSSK}			80		ns
SI1, SI2 hold time (from $\overline{SCK1}$, $\overline{SCK2}\uparrow$)	t_{HSSK}			80		ns
SO1, SO2 output delay time from $\overline{SCK1}$, $\overline{SCK2}\downarrow$	t_{DSBSK}	R = 1 kΩ, C = 100 pF		0	150	ns

Remarks 1. T_{SFT} is a value set in software. The minimum value is $t_{CYK} \times 8$.

2. $t_{CYK} = 1/f_{CLK}$ (f_{CLK} is internal system clock frequency)

(3) Other Operations (T_A = –40 to +110°C, V_{DD} = 4.5 to 5.5 V, V_{SS} = 0 V)

Parameter	Symbol	Conditions	MIN.	MAX.	Unit
NMI high, low-level width	t_{WNIH}, t_{WNIL}		10		μs
INTP0-INTP6 high, low-level width	t_{WITH}, t_{WITL}		4		t_{CYSMP}
$\overline{RESET}$ high, low-level width	t_{WRSH}, t_{WRSL}		10		μs

Remarks 1. t_{CYSMP} is a sampling clock set in the noise protection control register (NPC) in software.

When NIn = 0, $t_{CYSMP} = t_{CYK}$

When NIn = 1, $t_{CYSMP} = t_{CYK} \times 4$

2. $t_{CYK} = 1/f_{CLK}$ (f_{CLK} is internal system clock frequency)

3. NIn: Bit n of NPC (n = 0-6)

A/D Converter Characteristics (T_A = –40 to +110°C, V_{DD} = 4.5 to 5.5 V, V_{SS} = AV_{SS} = 0 V, V_{DD} – 0.5 V ≤ AV_{DD} ≤ V_{DD})

Parameter	Symbol	Conditions		MIN.	TYP.	MAX.	Unit
Resolution				10			bit
Total error[Note 1]		4.5 V ≤ AV_{REF} ≤ AV_{DD}				±0.5	%FSR[Note 2]
		3.4 V ≤ AV_{REF} < 4.5 V				±0.7	%FSR[Note 2]
Quantization error						±1/2	LSB
Conversion time	t_{CONV}			169			t_{CYK}
Sampling time	t_{SAMP}			20			t_{CYK}
Zero-scale error[Note 1]		4.5 V ≤ AV_{REF} ≤ AV_{DD}			±1.5	±3.5	LSB
		3.4 V ≤ AV_{REF} < 4.5 V			±1.5	±4.5	LSB
Full-scale error[Note 1]		4.5 V ≤ AV_{REF} ≤ AV_{DD}			±1.5	±3.5	LSB
		3.4 V ≤ AV_{REF} < 4.5 V			±1.5	±4.5	LSB
Integral linearity error[Note 1]		4.5 V ≤ AV_{REF} ≤ AV_{DD}			±1.5	±2.5	LSB
		3.4 V ≤ AV_{REF} < 4.5 V			±1.5	±4.5	LSB
Analog input voltage	V_{IAN}			–0.3		AV_{REF}+0.3	V
A/D converter reference input voltage	AV_{REF}			3.4		AV_{DD}	V
AV_{REF} current	AI_{REF}				3.0	4.0	mA
AV_{DD} supply current	AI_{DD}				2.0	6.0	mA
A/D converter data retention current	AI_{DDDR}	STOP mode	AV_{DDDR} = 2.5 V		2	100	μA
			AV_{DDDR} = 5 V ±10%		10	1000	μA

Notes 1. The quantization error is excluded.

2. Indicated as a ratio (%FSR) to the full-scale value.

Remark t_{CYK} = 1/f_{CLK} (f_{CLK} is internal system clock frequency).

Refer to **CHAPTER 24 TIMING CHARTS** for the timing charts.

Absolute Maximum Ratings (T_A = 25°C)

Parameter	Symbol	Conditions		Ratings	Unit
Supply voltage	V_{DD}			–0.5 to +7.0	V
	AV_{DD}			–0.5 to V_{DD} + 0.5	V
	AV_{SS}			–0.5 to +0.5	V
Input voltage	V_I	**Note 1**		–0.5 to V_{DD} + 0.5 ≤ 7.0	V
Output voltage	V_O			–0.5 to V_{DD} + 0.5	V
Output current, low	I_{OL}	All output pins		15	mA
		Total of all output pins		150	mA
Output current, low	I_{OH}	All output pins		–10	mA
		Total of all output pins		–100	mA
Analog input voltage	V_{IAN}	**Note 2**	$AV_{DD} > V_{DD}$	–0.5 to V_{DD} + 0.5	V
			$V_{DD} \geq AV_{DD}$	–0.5 to AV_{DD} + 0.5	
A/D converter reference input voltage	AV_{REF}		$AV_{DD} > V_{DD}$	–0.5 to V_{DD} + 0.5	V
			$V_{DD} \geq AV_{DD}$	–0.5 to AV_{DD} + 0.5	
Operating temperature	T_A			–40 to +125	°C
Storage temperature	T_{stg}			–65 to +150	°C

Notes 1. Pins other than the pins in **Note 2**.

2. Pins P70/ANI0 to P77/ANI7, P80/ANI8 to P87/ANI15

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 the absolute maximum ratings are not exceeded.

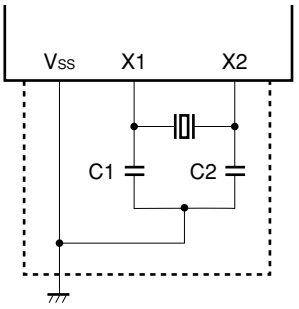
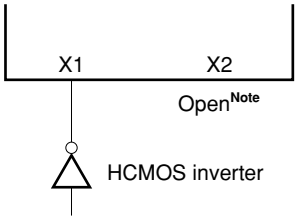
Recommended Operating Conditions

Oscillation Frequency	T_A	V_{DD}
8 MHz ≤ f_{XX} ≤ 20 MHz	–40 to +125°C	4.5 to 5.5 V

Capacitance (T_A = 25°C, V_{SS} = V_{DD} = 0 V)

Parameter	Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Input capacitance	C_I	f = 1 MHz			10	pF
Output capacitance	C_O	Unmeasured pins returned to 0 V.			10	pF
I/O capacitance	C_{IO}				10	pF

Oscillator Characteristics (T_A = –40 to +125°C, V_{DD} = 4.5 to 5.5 V, V_{SS} = 0 V)

Resonator	Recommended Circuit	Item	MIN.	MAX.	Unit
Ceramic resonator or crystal resonator		Oscillation frequency (f_{XX})	8	20	MHz
External clock		X1 input frequency (f_X)	8	20	MHz
		X1 input rise, fall time	0	5	ns
		X1 input high-, low-level width	20	105	ns

Note When the EXTC bit of the oscillation stabilization time specification register (OSTS) = 0. Input the reverse phase clock of the pin X1 to the pin X2 when the EXTC bit = 1.

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

- **Keep the wiring length as short as possible.**
- **Do not cross the wiring with any 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.**

Remark For the resonator selection and oscillator constant, customers are required to either evaluate the oscillation themselves or apply to the resonator manufacturer for evaluation.

DC Characteristics (T_A = –40 to +125°C, V_{DD} = 4.5 to 5.5 V, V_{SS} = 0 V)

Parameter	Symbol	Conditions		MIN.	TYP.	MAX.	Unit
Input voltage, low	V_{IL}			0		0.8	V
Input voltage, high	V_{IH1}	**Note 1**		2.2		V_{DD}	V
	V_{IH2}	**Note 2**		0.8V_{DD}		V_{DD}	
Output voltage, low	V_{OL}	I_{OL} = 2.0 mA				0.45	V
Output voltage, high	V_{OH}	I_{OH} = –400 μA		V_{DD} – 1.0			V
Input leakage current	I_{LI}	**Note 3**	0 V ≤ V_I ≤ V_{DD}			±10	μA
Analog pin input leakage current	I_{LIAN}	**Note 4**	0 V ≤ V_I ≤ AV_{DD}			±2	μA
Output leakage current	I_{LO}	0 V ≤ V_O ≤ V_{DD}				±10	μA
V_{DD} supply current	I_{DD1}	Operating mode (f_{XX} = 20 MHz)			30	60	mA
	I_{DD2}	HALT mode (f_{XX} = 20 MHz)			15	30	mA
	I_{DD3}	IDLE mode (f_{XX} = 20 MHz)			10	20	mA
Data retention voltage	V_{DDDR}	STOP mode		2.5			V
Data retention current	I_{DDDR}	STOP mode	V_{DDDR} = 2.5 V		2	100	μA
			V_{DDDR} = 5 V ±10 %		15	1000	μA
Pull-up resistor	R_L			15	40	80	kΩ

Notes 1. Pins other than pins in **Note 2**.

2. P20/NMI, P21/INTP0/TO00, P22/INTP1/TO01, P23/INTP2/TO02, P24/INTP3/TO03, P25/INTP4, P26/INTP5/TI2, P27/INTP6/TI3, P34/ASCK/$\overline{\text{SCK1}}$, P37/ASCK2/$\overline{\text{SCK2}}$, X1, X2, $\overline{\text{RESET}}$

3. Input and I/O pins (except X1 and X2, and P70/ANI0 to P77/ANI7, P80/ANI8 to P87/ANI15 used as analog inputs)

4. Pins P70/ANI0 to P77/ANI7, P80/ANI8 to P87/ANI15 (pins used as analog input, only during the non-sampling operation)

AC Characteristics (T_A = –40 to +125°C, V_{DD} = 4.5 to 5.5 V, V_{SS} = 0 V)

(1) Read/write operation

Parameter	Symbol	Expression	MIN.	MAX.	Unit
System clock cycle time	t_{CYK}		100	250	ns
Address setup time (to ASTB↓)	t_{SAST}	(0.5 + a) T – 20	30		ns
Address hold time (from ASTB↓)	t_{HSTA}	0.5T – 20	30		ns
ASTB high-level width	t_{WSTH}	(0.5 + a) T – 17	33		ns
$\overline{RD}$↓ delay time from address	t_{DAR}	(1 + a) T – 15	85		ns
Address float time from $\overline{RD}$↓	t_{FRA}			0	ns
Data input time from address	t_{DAID}	(2.5 + a + n) T – 56		194	ns
Data input time from $\overline{RD}$↓	t_{DRID}	(1.5 + n) T – 53		97	ns
$\overline{RD}$↓ delay time from ASTB↓	t_{DSTR}	0.5T – 16	34		ns
Data hold time (to $\overline{RD}$↑)	t_{HRID}		0		ns
Address active time from $\overline{RD}$↑	t_{DRA}	0.5T – 14	36		ns
$\overline{RD}$ low-level width	t_{WRL}	(1.5 + n) T – 30	120		ns
$\overline{LWR}$, $\overline{HWR}$↓ delay time from address	t_{DAW}	(1 + a) T – 15	85		ns
Data output time from $\overline{LWR}$, $\overline{HWR}$↓	t_{DWOD}			15	ns
$\overline{LWR}$, $\overline{HWR}$↓ delay time from ASTB↓	t_{DSTW}	0.5T – 16	34		ns
Data setup time (to $\overline{LWR}$, $\overline{HWR}$↑)	t_{SODW}	(1.5 + n) T – 25	125		ns
Data hold time (from $\overline{LWR}$, $\overline{HWR}$↑)	t_{HWOD}	0.5T – 14	36		ns
ASTB↑ delay time from $\overline{LWR}$, $\overline{HWR}$↑	t_{DWST}	1.5T – 15	135		ns
$\overline{LWR}$, $\overline{HWR}$ low-level width	t_{WWL}	(1.5 + n) T – 36	114		ns
$\overline{WAIT}$↓ input time from address	t_{DAWT}	(2 + a) T – 50		150	ns
$\overline{WAIT}$↓ input time from ASTB↓	t_{DSTWT}	1.5T – 40		110	ns
$\overline{WAIT}$ hold time from ASTB↓	t_{HSTWT}	(1.5 + n) T + 5	155		ns
$\overline{WAIT}$↑ delay time from ASTB↓	t_{DSTWTH}	(1.5 + n) T – 40		210[Note]	ns
$\overline{WAIT}$↓ input time from $\overline{RD}$↓	t_{DRWT}	T – 40		60	ns
$\overline{WAIT}$ hold time from $\overline{RD}$↓	t_{HRWT}	(1 + n) T + 5	105		ns
$\overline{WAIT}$↑ delay time from $\overline{RD}$↓	t_{DRWTH}	(1 + n) T – 40		160[Note]	ns
$\overline{WAIT}$↓ input time $\overline{LWR}$, $\overline{HWR}$↓	t_{DWWT}	T – 40		60	ns
$\overline{WAIT}$ hold time $\overline{LWR}$, $\overline{HWR}$↓	t_{HWWT}	(1 + n) T + 5	105		ns
$\overline{WAIT}$↑ delay time from $\overline{LWR}$, $\overline{HWR}$↓	t_{DWWTH}	(1 + n) T – 40		160[Note]	ns

Note Specification when an external wait is inserted

Remarks 1. T = t_{CYK} = 1/f_{CLK} (f_{CLK} is internal system clock frequency)

2. a = 1 when an address wait is inserted, otherwise, 0.

3. n indicates the number of the wait cycles by specifying the external wait pins ($\overline{WAIT}$) or programmable wait control registers 1, 2 (PWC1, PWC2). (n ≥ 0. n ≥ 1 for t_{DSTWTH}, t_{DRWTH}, t_{DWWTH}).

4. Calculate values in the expression column with the system clock cycle time to be used because these values depend on the system clock cycle time (t_{CYK} = T). The values in the above expression column are calculated based on T = 100 ns.

(2) Serial Operation (T_A = –40 to +125°C, V_{DD} = 4.5 to 5.5 V, V_{SS} = 0 V)

Parameter	Symbol	Conditions		MIN.	MAX.	Unit
Serial clock cycle time	t_{CYSK}	$\overline{SCK1}$, $\overline{SCK2}$ output	BRG	T_{SFT}		ns
		$\overline{SCK1}$, $\overline{SCK2}$ input	External clock	800		ns
Serial clock low-level width	t_{WSKL}	$\overline{SCK1}$, $\overline{SCK2}$ output	BRG	$0.5T_{SFT}$–40		ns
		$\overline{SCK1}$, $\overline{SCK2}$ input	External clock	360		ns
Serial clock high-level width	t_{WSKH}	$\overline{SCK1}$, $\overline{SCK2}$ output	BRG	$0.5T_{SFT}$–40		ns
		$\overline{SCK1}$, $\overline{SCK2}$ input	External clock	360		ns
SI1, SI2 setup time (to $\overline{SCK1}$, $\overline{SCK2}\uparrow$)	t_{SSSK}			80		ns
SI1, SI2 hold time (from $\overline{SCK1}$, $\overline{SCK2}\uparrow$)	t_{HSSK}			80		ns
SO1, SO2 output delay time from $\overline{SCK1}$, $\overline{SCK2}\downarrow$	t_{DSBSK}	R = 1 kΩ, C = 100 pF		0	150	ns

Remarks 1. T_{SFT} is a value set in software. The minimum value is $t_{CYK} \times 8$.

2. $t_{CYK} = 1/f_{CLK}$ (f_{CLK} is internal system clock frequency)

(3) Other Operations (T_A = –40 to +125°C, V_{DD} = 4.5 to 5.5 V, V_{SS} = 0 V)

Parameter	Symbol	Conditions	MIN.	MAX.	Unit
NMI high, low-level width	t_{WNIH}, t_{WNIL}		10		μs
INTP0-INTP6 high, low-level width	t_{WITH}, t_{WITL}		4		t_{CYSMP}
$\overline{RESET}$ high, low-level width	t_{WRSH}, t_{WRSL}		10		μs

Remarks 1. t_{CYSMP} is a sampling clock set in the noise protection control register (NPC) in software.

When NIn = 0, $t_{CYSMP} = t_{CYK}$

When NIn = 1, $t_{CYSMP} = t_{CYK} \times 4$

2. $t_{CYK} = 1/f_{CLK}$ (f_{CLK} is internal system clock frequency)

3. NIn: Bit n of NPC (n = 0-6)

A/D Converter Characteristics (T_A = –40 to +125°C, V_{DD} = 4.5 to 5.5 V, V_{SS} = AV_{SS} = 0 V, V_{DD} – 0.5 V ≤ AV_{DD} ≤ V_{DD})

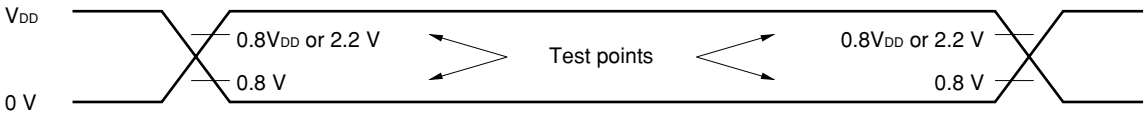
Parameter	Symbol	Conditions		MIN.	TYP.	MAX.	Unit
Resolution				10			bit
Total error[Note 1]		4.5 V ≤ AV_{REF} ≤ AV_{DD}				±0.5	%FSR[Note 2]
		3.4 V ≤ AV_{REF} < 4.5 V				±0.7	%FSR[Note 2]
Quantization error						±1/2	LSB
Conversion time	t_{CONV}			169			t_{CYK}
Sampling time	t_{SAMP}			20			t_{CYK}
Zero-scale error[Note 1]		4.5 V ≤ AV_{REF} ≤ AV_{DD}			±1.5	±3.5	LSB
		3.4 V ≤ AV_{REF} < 4.5 V			±1.5	±4.5	LSB
Full-scale error[Note 1]		4.5 V ≤ AV_{REF} ≤ AV_{DD}			±1.5	±3.5	LSB
		3.4 V ≤ AV_{REF} < 4.5 V			±1.5	±4.5	LSB
Integral linearity error[Note 1]		4.5 V ≤ AV_{REF} ≤ AV_{DD}			±1.5	±2.5	LSB
		3.4 V ≤ AV_{REF} < 4.5 V			±1.5	±4.5	LSB
Analog input voltage	V_{IAN}			–0.3		AV_{REF}+0.3	V
A/D converter reference input voltage	AV_{REF}			3.4		AV_{DD}	V
AV_{REF} current	AI_{REF}				3.0	4.0	mA
AV_{DD} supply current	AI_{DD}				2.0	6.0	mA
A/D converter data retention current	AI_{DDDR}	STOP mode	AV_{DDDR} = 2.5 V		2	100	μA
			AV_{DDDR} = 5 V ±10%		10	1000	μA

Notes 1. The quantization error is excluded.

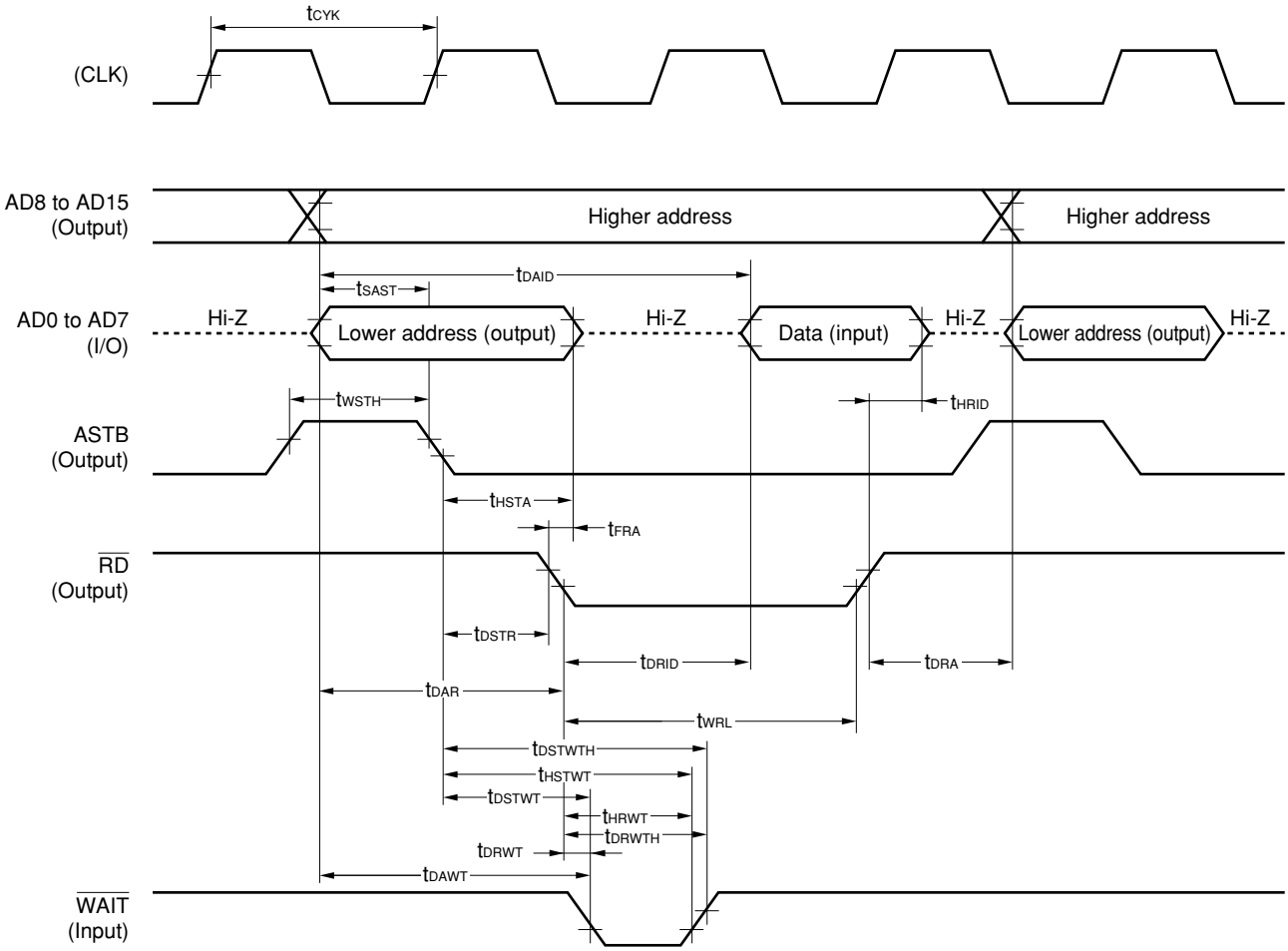
2. Indicated as a ratio (%FSR) to the full-scale value.

Remark t_{CYK} = 1/f_{CLK} (f_{CLK} is internal system clock frequency).

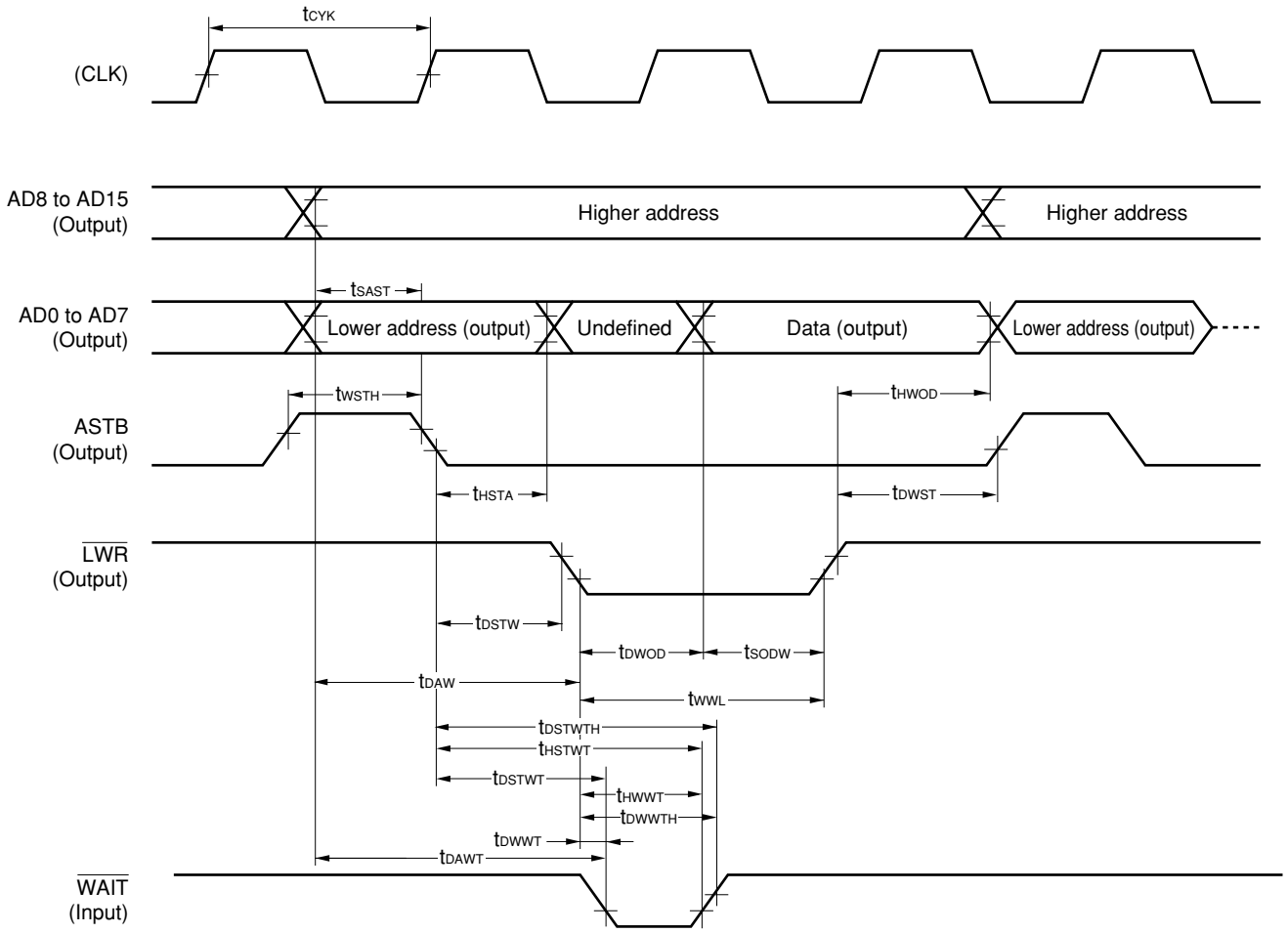
AC Timing Test Points



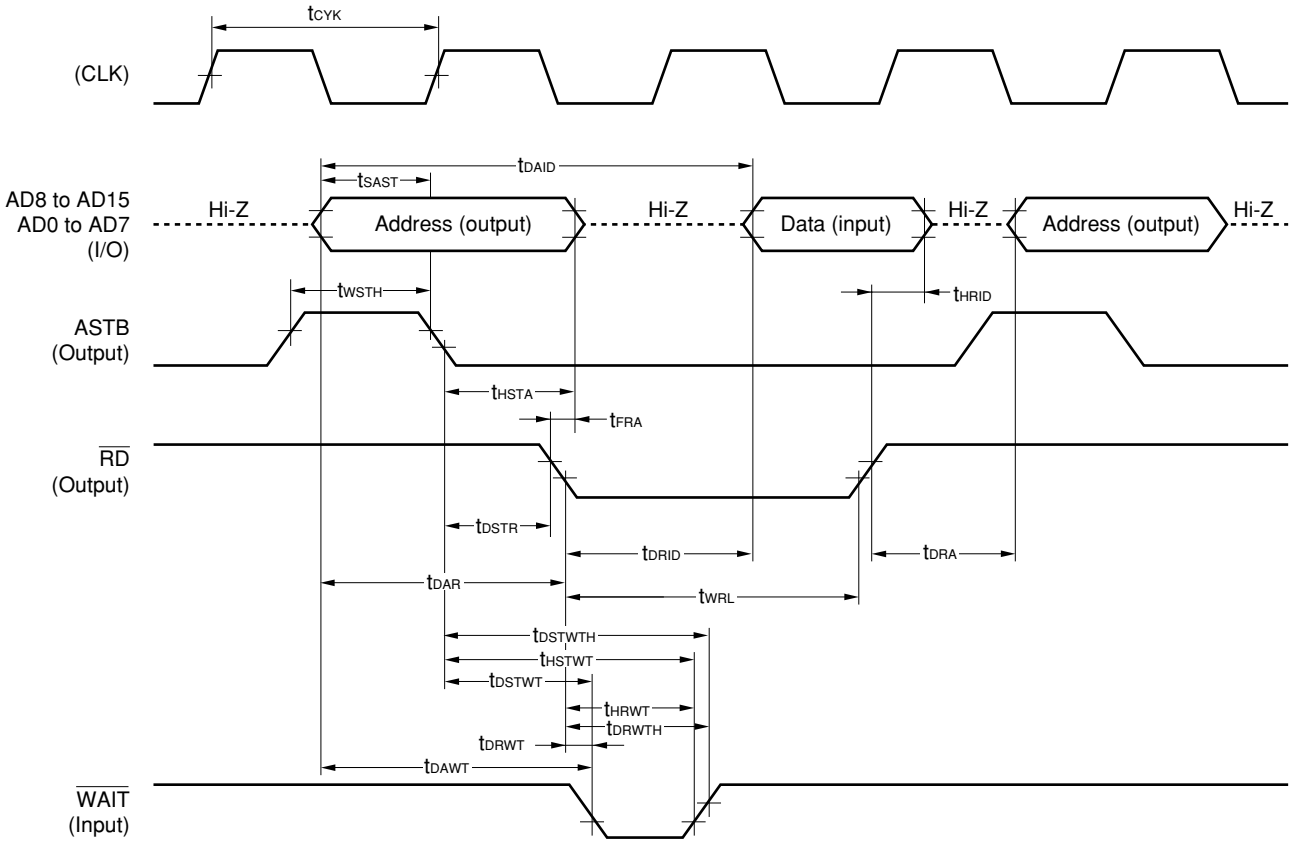
Read Operation (8 bits)



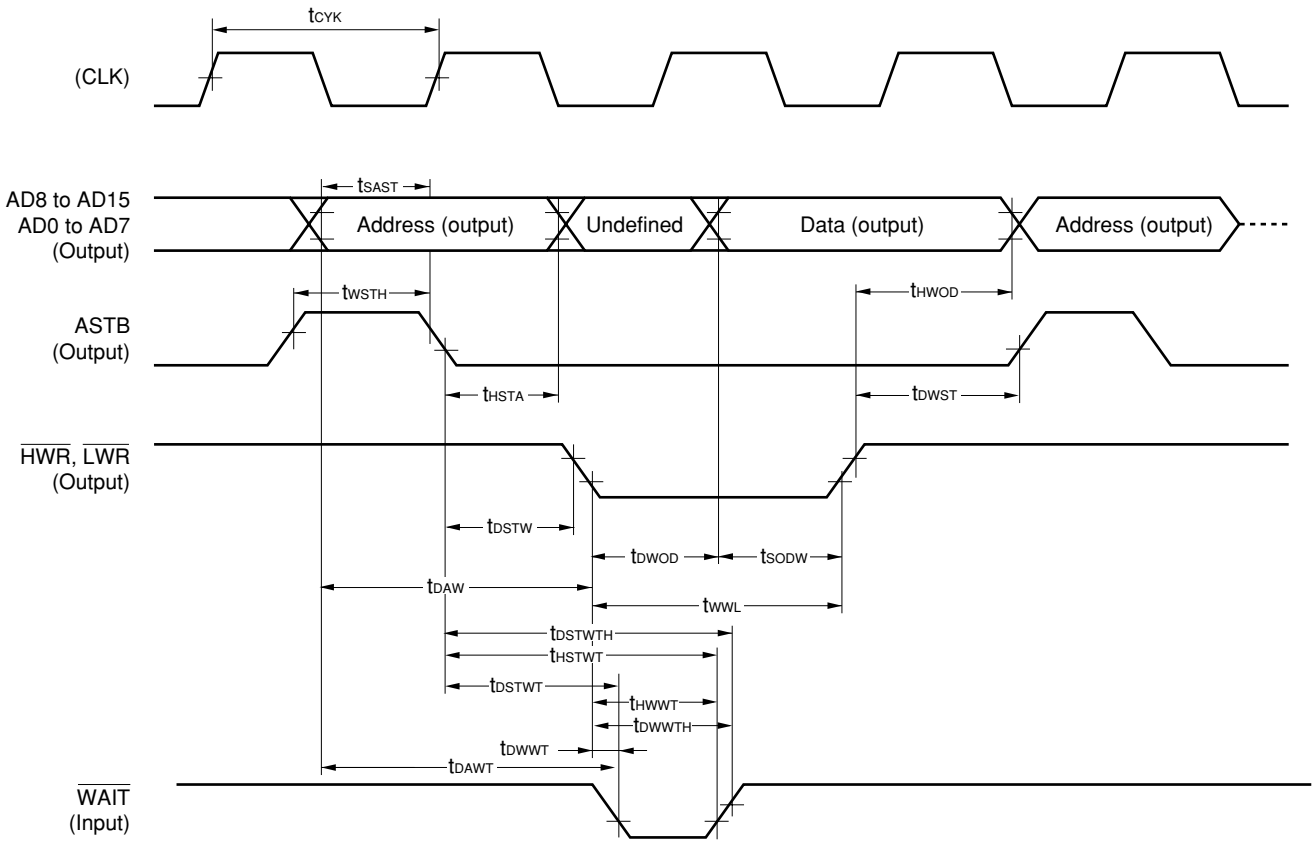
Write Operation (8 bits)



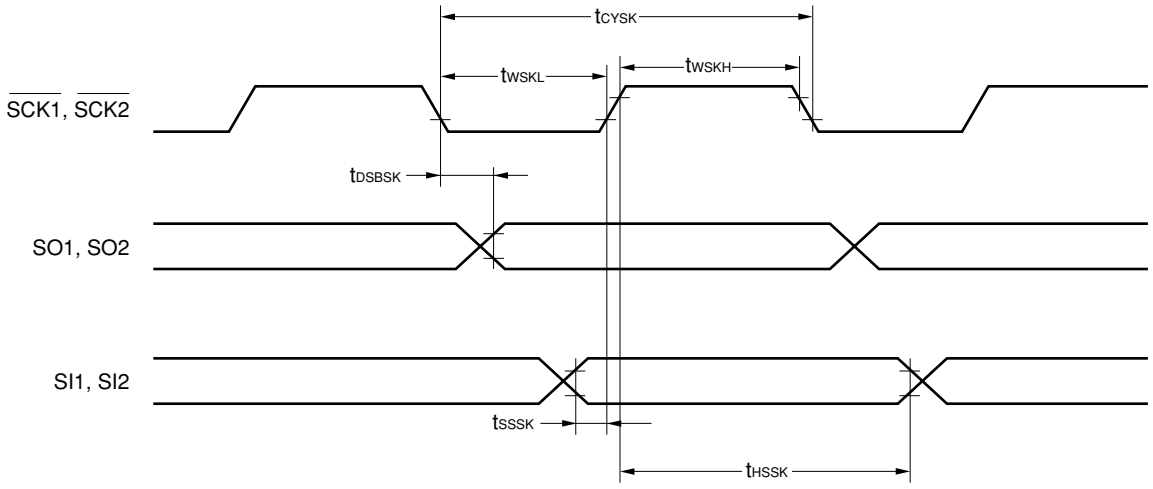
Read Operation (16 bits)



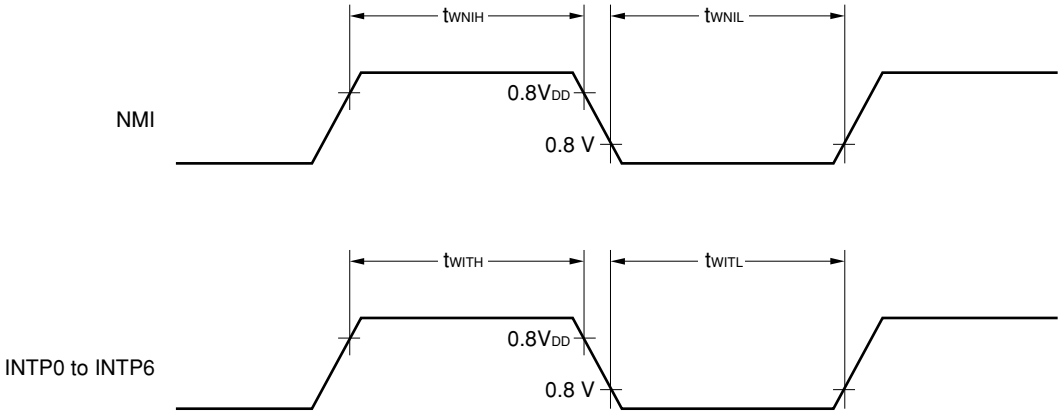
Write Operation (16 bits)



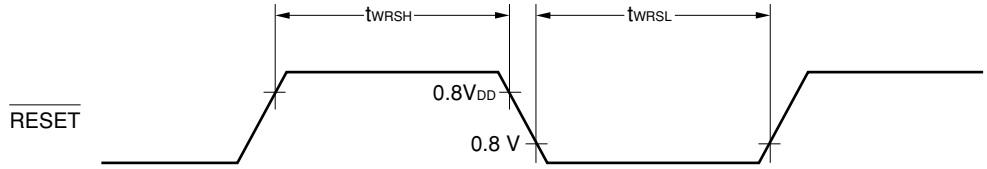
Serial Operation



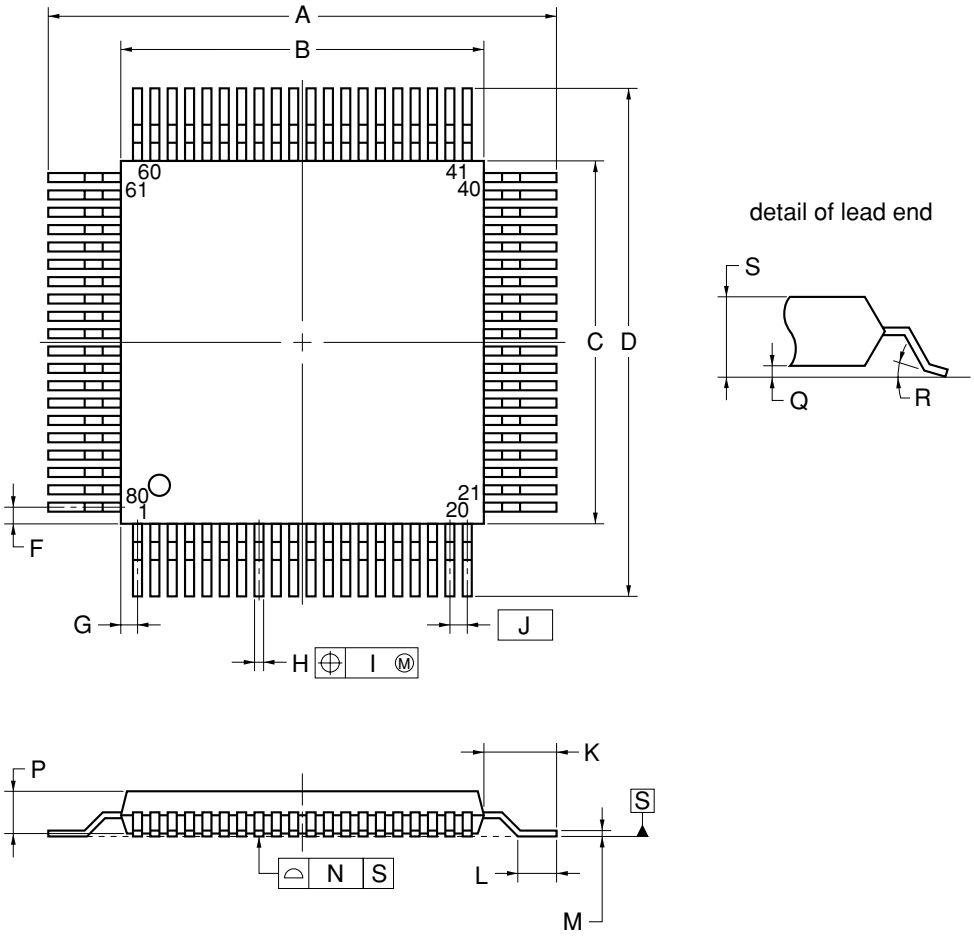
Interrupt Input Timing



Reset Input Timing



80-PIN PLASTIC QFP (14x14)



NOTE

Each lead centerline is located within 0.13 mm of its true position (T.P.) at maximum material condition.

ITEM	MILLIMETERS
A	17.2±0.4
B	14.0±0.2
C	14.0±0.2
D	17.2±0.4
F	0.825
G	0.825
H	0.30±0.10
I	0.13
J	0.65 (T.P.)
K	1.6±0.2
L	0.8±0.2
M	$0.15^{+0.10}_{-0.05}$
N	0.10
P	2.7±0.1
Q	0.1±0.1
R	5°±5°
S	3.0 MAX.

S80GC-65-3B9-6

CHAPTER 26 RECOMMENDED SOLDERING CONDITIONS

The μPD784054 should be soldered and mounted under the following recommended conditions.

For soldering methods and conditions other than those recommended below, contact an NEC Electronics sales representative.

For technical information, see the following website.

Semiconductor Device Mount Manual (http://www.necel.com/pkg/en/mount/index.html)

Table 26-1. Surface Mounting Type Soldering Conditions

μPD784054GC-×××-3B9: 80-pin plastic QFP (14 x 14)
μPD784054GC(A)-×××-3B9: 80-pin plastic QFP (14 x 14)
μPD784054GC(A1)-×××-3B9: 80-pin plastic QFP (14 x 14)
μPD784054GC(A2)-×××-3B9: 80-pin plastic QFP (14 x 14)

Soldering Method	Soldering Conditions	Recommended Condition Symbol
Infrared reflow	Package peak temperature: 235°C, Time: 30 sec. Max. (at 210°C or higher), Count: three times or less	IR35-00-3
VPS	Package peak temperature: 215°C, Time: 40 sec. Max. (at 200°C or higher), Count: three times or less	VP15-00-3
Wave soldering	Solder bath temperature: 260°C Max., Time 10 sec. Max., Count: once, Preheating temperature: 120°C Max. (package surface temperature)	WS60-00-1
Partial heating	Pin temperature: 350°C Max., Time: 3 sec. Max. (per pin row)	–

Caution Do not use different soldering methods together (except for partial heating).

★ **(2) μPD784054GC-×××-3B9-A: 80-pin plastic QFP (14 x 14)**

Soldering Method	Soldering Conditions	Recommended Condition Symbol
Infrared reflow	Package peak temperature: 260°C, Time: 60 seconds max. (at 220°C or higher), Count: Three times or less, Exposure limit: 7 days[Note] (after that, prebake at 125°C for 20 to 72 hours)	IR60-207-3
Wave soldering	For details, contact an NEC Electronics sales representative.	–
Partial heating	Pin temperature: 350°C max., Time: 3 seconds max. (per pin row)	–

Note After opening the dry pack, store it at 25°C or less and 65% RH or less for the allowable storage period.

Caution Do not use different soldering methods together (except for partial heating).

Remark Products that have the part numbers suffixed by “-A” are lead-free products.

CHAPTER 27 CAUTIONS ON USING DEVELOPMENT TOOLS

When developing a program by using in-circuit emulator IE-784000-R, note the following point.

(1) Setting of standby control register (STBC)

- Include an instruction that sets the standby control register (STBC) to 00H following the LOCATION instruction and initialization of the stack pointer (SP) in the program after reset is cleared.

  Program example:

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

  Reason: The internal system clock of the μPD784054 is fixed to fxx/2. However, the internal system clock of the in-circuit emulator is set to fxx/16 after reset has been cleared. Therefore, the setting of the STBC must be changed as described above.

Even if the instruction that sets the STBC to 00H is executed, the operation is not affected because the STBC of the real chip is fixed to 30H. For the same reason, the value of the STBC of the real chip is always 30H when it is read. The value of the STBC on the in-circuit emulator, however, is changed to 00H when the above setting is made. Therefore, note that the value of the STBC of the in-circuit emulator and that of the real chip differ when they are read.

(2) Output of CLKOUT pin

The CLKOUT pin of the real chip always outputs the oscillation frequency (fxx). However, in the case of the in-circuit emulator IE-784000-R, the internal system clock (fxx/2 or fxx/16**Note**) is output. Note that fxx is not output from the in-circuit emulator.

Note The CLKOUT pin output of the in-circuit emulator is set to fxx/16 after the reset has been released, and set to fxx/2 when 00H is set to the standby control register (STBC).

APPENDIX A DEVELOPMENT TOOLS

The following development tools are available for the development of systems that employ the μPD784046 Subseries.

Figure A-1 shows the development tool configuration.

★ • **Support for PC98-NX series**

Unless otherwise specified, products supported by IBM PC/AT™ compatibles can be used for PC98-NX series computers. When using PC98-NX series computers, refer to the description for IBM PC/AT compatibles.

★ • **Windows**

Unless otherwise specified, “Windows” means the following OSs.

- Windows 3.1
- Windows 95
- Windows 98
- Windows 2000
- Windows NT™ Ver. 4.0

Figure A-1. Development Tool Configuration (1/2)

(1) When using the in-circuit emulator IE-78K4-NS

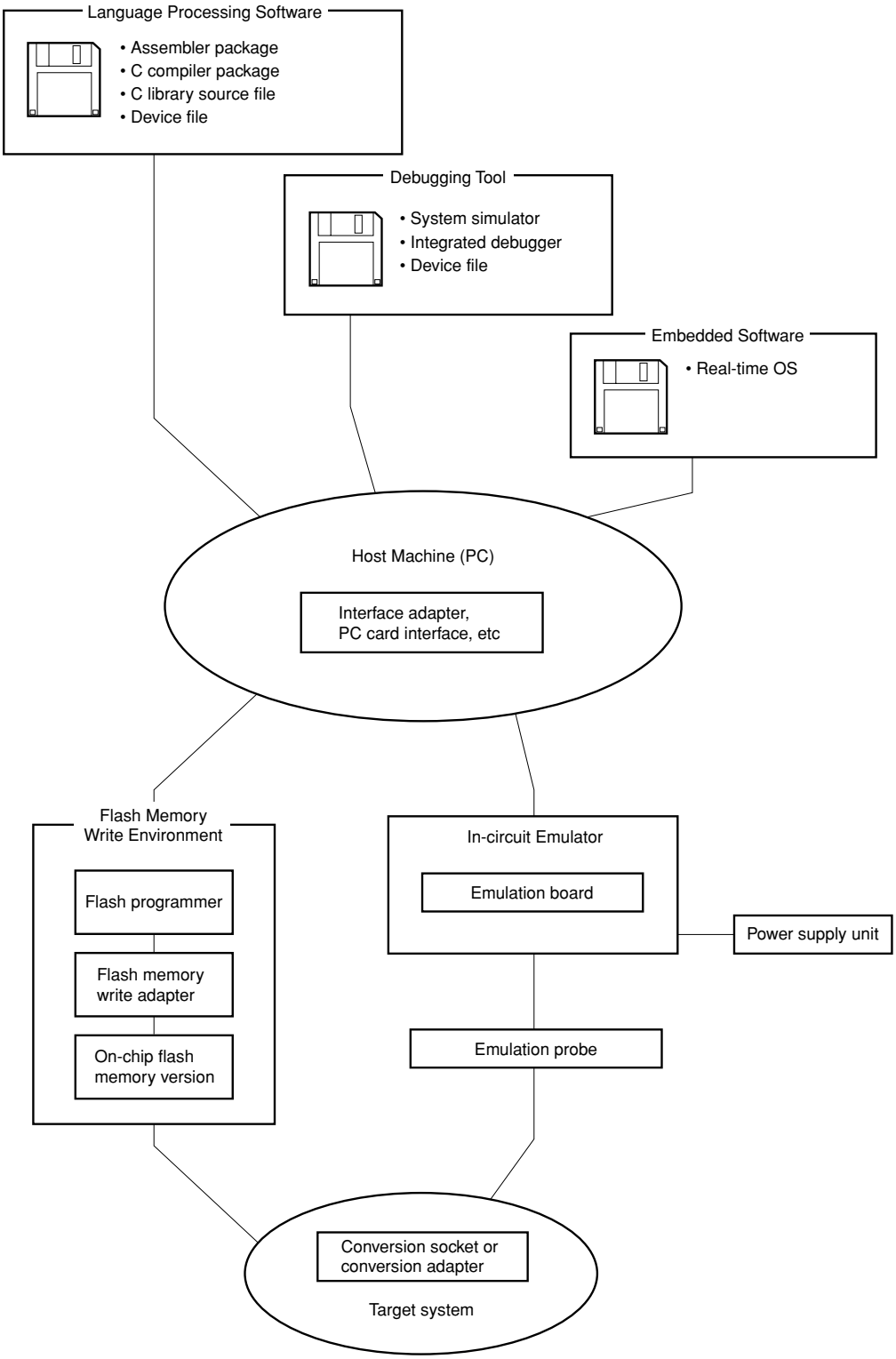
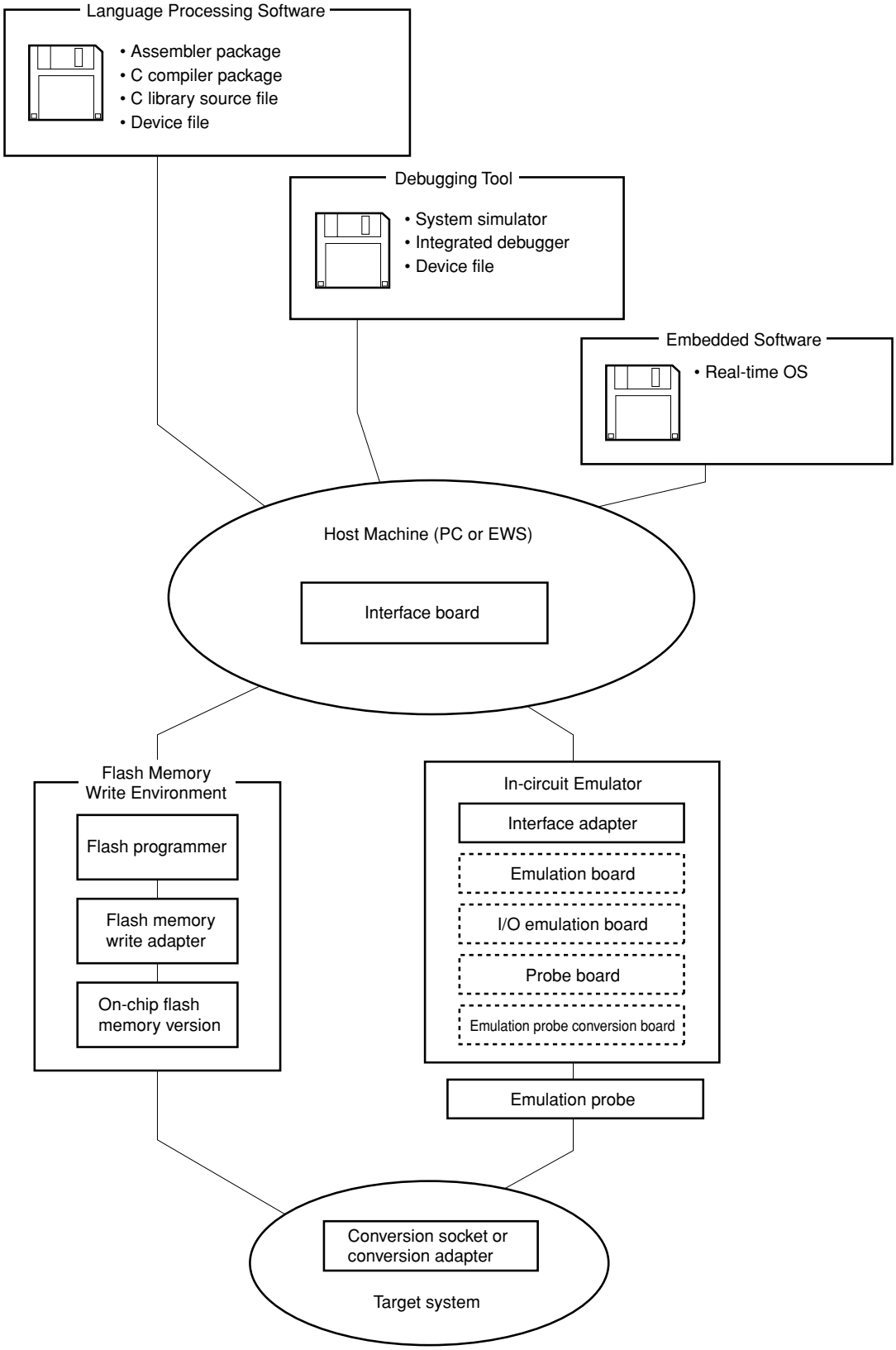


Figure A-1. Development Tool Configuration (2/2)

(2) When using the in-circuit emulator IE-784000-R



Remark Items in broken line boxes differ according to the development environment. Refer to **A.3.1 Hardware**.

A.1 Language Processing Software

★

SP78K4 78K/IV Series Software package	Development tools (software) common to the 78K/IV Series are combined in this package.
	Part number: μS××××SP78K4
RA78K4 Assembler package	This assembler converts programs written in mnemonics into an object codes executable with a microcomputer. Further, this assembler is provided with functions capable of automatically creating symbol tables and branch instruction optimization. This assembler should be used in combination with an optical device file (DF784046). <Precaution when using RA78K4 in PC environment> 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) on Windows.
	Part number: μS××××RA78K4
CC78K4 C compiler package	This compiler converts programs written in C language into object codes executable with a microcomputer. This compiler should be used in combination with an optical assembler package and device file. <Precaution when using CC78K4 in PC environment> 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) on Windows.
	Part number: μS××××CC78K4
DF784046[Note] Device file	This file contains information peculiar to the device. This device file should be used in combination with an optional tool (RA78K4, CC78K4, SM78K4, ID78K4-NS, and ID78K4). Corresponding OS and host machine differ depending on the tool to be used with.
	Part number: μS××××DF784046
CC78K4-L C library source file	This is a source file of functions configuring the object library included in the C compiler package (CC78K4). This file is required to match the object library included in C compiler package to the customer's specifications. The operating environment does not depend on the OS because this is a source file.
	Part number: μS××××CC78K4-L

Note The DF784046 can be used commonly for all the RA78K4, CC78K4, SM78K4, ID78K4-NS, and ID78K4.

★ **Remark** The ××××part number differs depending on the host machine and operating system used.

μS××××SP78K4

××××	Host Machine	OS	Supply Medium
AB17	PC-9800 series, IBM PC/AT compatibles	Japanese Windows	CD-ROM
BB17		English Windows	

μS××××RA78K4
μS××××CC78K4

××××	Host Machine	OS	Supply Medium
AB13	PC-9800 series, IBM PC/AT compatibles	Japanese Windows	3.5-inch 2HD FD
BB13		English Windows	
AB17		Japanese Windows	CD-ROM
BB17		English Windows	
3P17	HP9000 series 700™	HP-UX™ (Rel. 10.10)	
3K17	SPARCstation™	SunOS™ (Rel. 4.1.4), Solaris™ (Rel. 2.5.1)	

μS××××DF784046
μS××××CC78K4-L

××××	Host Machine	OS	Supply Medium
AB13	PC-9800 series, IBM PC/AT compatibles	Japanese Windows	3.5-inch 2HD FD
BB13		English Windows	
3P16	HP9000 series 700	HP-UX (Rel. 10.10)	DAT
3K13	SPARCstation	SunOS (Rel. 4.1.4),	3.5-inch 2HD FD
3K15		Solaris (Rel. 2.5.1)	1/4-inch CGMT

★ A.2 Flash Memory Writing Tools

Flashpro II (part number: FL-PR2) Flashpro III (part number: FL-PR3, PG-FP3) Flash programmer	Flash programmer dedicated to microcontrollers with on-chip flash memory.
FA-80GC Flash memory writing adapter	Flash memory writing adapter used connected to the Flashpro II/Flashpro III. • FA-80GC : 80-pin plastic QFP (GC-3B9 type)

Remark Flashpro II, Flashpro III, and FA-80GC are products of Naito Densei Machida Mfg. Co., Ltd.
Phone: +81-45-475-4191 Naito Densei Machida Mfg. Co., Ltd.

A.3 Debugging Tools

A.3.1 Hardware (1/2)

(1) When using the in-circuit emulator IE-78K4-NS

	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 corresponds to 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 V to 200 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 (C bus supported).
	IE-70000-CD-IF PC card interface	This is PC card and interface cable required when using the PC-9800 Series notebook-type computer as the IE-78K4-NS host machine.
	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 (ISA bus supported).
★	IE-70000-PCI-IF-A Interface adapter	Interface adapter required when using a PC that incorporates PCI bus as the host machine for the IE-78K4-NS
	IE-784046-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-80GC-TQ NP-H80GC-TQ Emulation probe	This probe is used to connect the in-circuit emulator to the target system and is designed for 80-pin plastic QFP (GC-3B9 type).
	TGC-080SBP Conversion socket (Refer to **Figure A-3**)	This conversion socket connects the NP-80GC-TQ or NP-H80GC-TQ to the target system board designed to mount a 80-pin plastic QFP (GC-3B9 type).

Remarks 1. NP-80GC-TQ and NP-H80GC-TQ are products made by Naito Densei Machida Mfg.Co., Ltd.
For further information, contact Naito Densei Machida Mfg. Co., Ltd. (TEL: +81-45-475-4191)

2. TGC-080SBP is a product made by Tokyo Eletech Corporation.
For further information, contact Daimaru Kogyo, Ltd.
Tokyo Electronics Department (TEL: +81-3-3820-7112)
Osaka Electronics Department (TEL: +81-6-6244-6672)

3. The TGC-080SBP is sold individually.

A.3.1 Hardware (2/2)

★ **(2) When using the in-circuit emulator IE-784000-R**

IE-784000-R In-circuit emulator	The IE-784000-R is an in-circuit emulator common to the 78K/IV Series, and is used in combination with IE-784000-R-EM and IE-784046-R-EM1, which are sold separately. This in-circuit emulator debugs the connected host machine. An integrated debugger (ID78K4) and device file (sold separately) are required to enable debugging in C language and structured assembly language at the source program level. More efficient debugging and program verification is possible with functions such as C0 coverage. Connect to a host machine via Ethernet™ or a dedicated bus. An interface adapter (sold separately) is required for connection.
IE-70000-98-IF-C Interface adapter	Interface adapter required when a PC-9800 series (except notebook type PC) is used as the host machine for the IE-784000-R (C bus supported).
IE-70000-PC-IF-C Interface adapter	Interface adapter required when using an IBM PC/AT compatible as the host machine (ISA bus supported).
IE-78000-R-SV3 Interface adapter	Interface adapter and cable required when an EWS is used as the host machine for the IE-784000-R. Connect to a board inside the IE-784000-R. Note that 10Base-5 is supported as the Ethernet. A commercial conversion adapter is required for other systems.
IE-784000-R-EM	Emulation board common to 78K/IV Series
IE-784046-R-EM1 Emulation board	Board to emulate peripheral hardware specific to device
IE-78K4-R-EX2 Emulation probe conversion board	Conversion board for 80-pin packages required when using the IE-784046-R-EM1 on IE-784000-R
EP-78230GC-R Emulation probe	Probe to connect the in-circuit emulator and the target system. For 80-pin plastic QFP (GC-3B9 type).
EV-9200GC-80 Conversion socket (Refer to **Figures A-4** and **A-5**)	Conversion socket to connect the EP-78230GC-R and a target system board on which an 80-pin plastic QFP (GC-3B9 type) can be mounted

Remark EV-9200GC-80 is sold in five units.

A.3.2 Software

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 the optional device file (DF784046).
	Part number: μS××××SM78K4
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. 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 windows 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 (DF784046).
ID78K4 Integrated debugger (supporting in-circuit emulator IE-784000-R)	
	Part number: μS××××ID78K4-NS, μS××××ID78K4

★ **Remark** ×××× in the part number differs depending on the host machine and OS used.

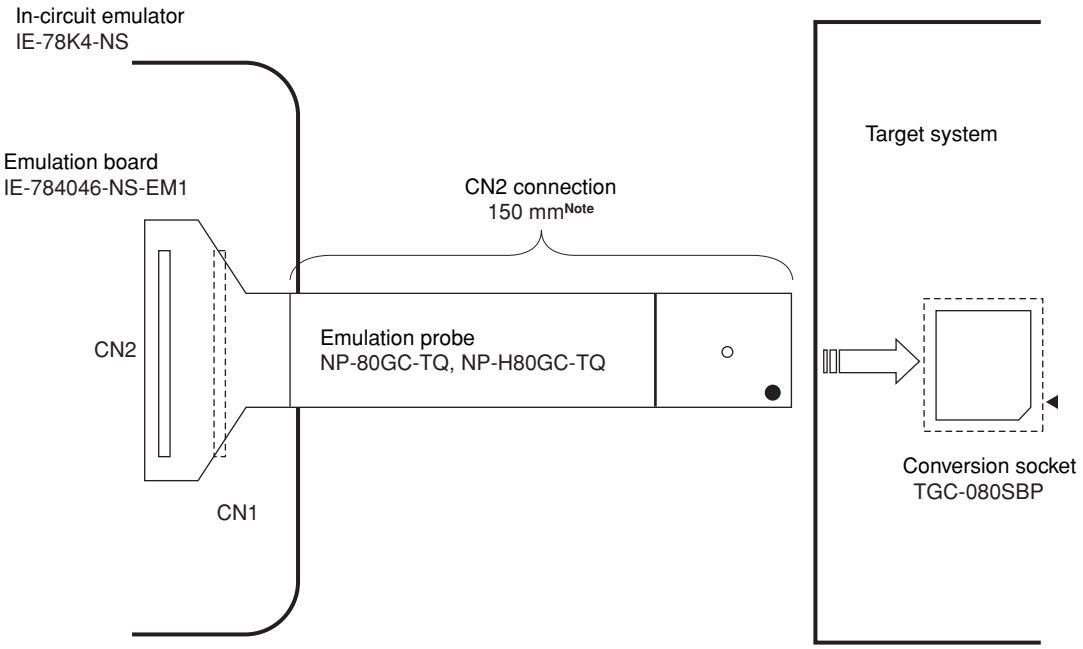
μS××××SM78K4
μS××××ID78K4-NS
μS××××ID78K4

××××	Host Machine	OS	Supply Medium
AB13	IBM PC/AT compatible	Japanese Windows	3.5-inch 2HC FD
BB13		English Windows	
AB17		Japanese Windows	CD-ROM
BB17		English Windows	

★ A.4 Cautions on Designing Target System

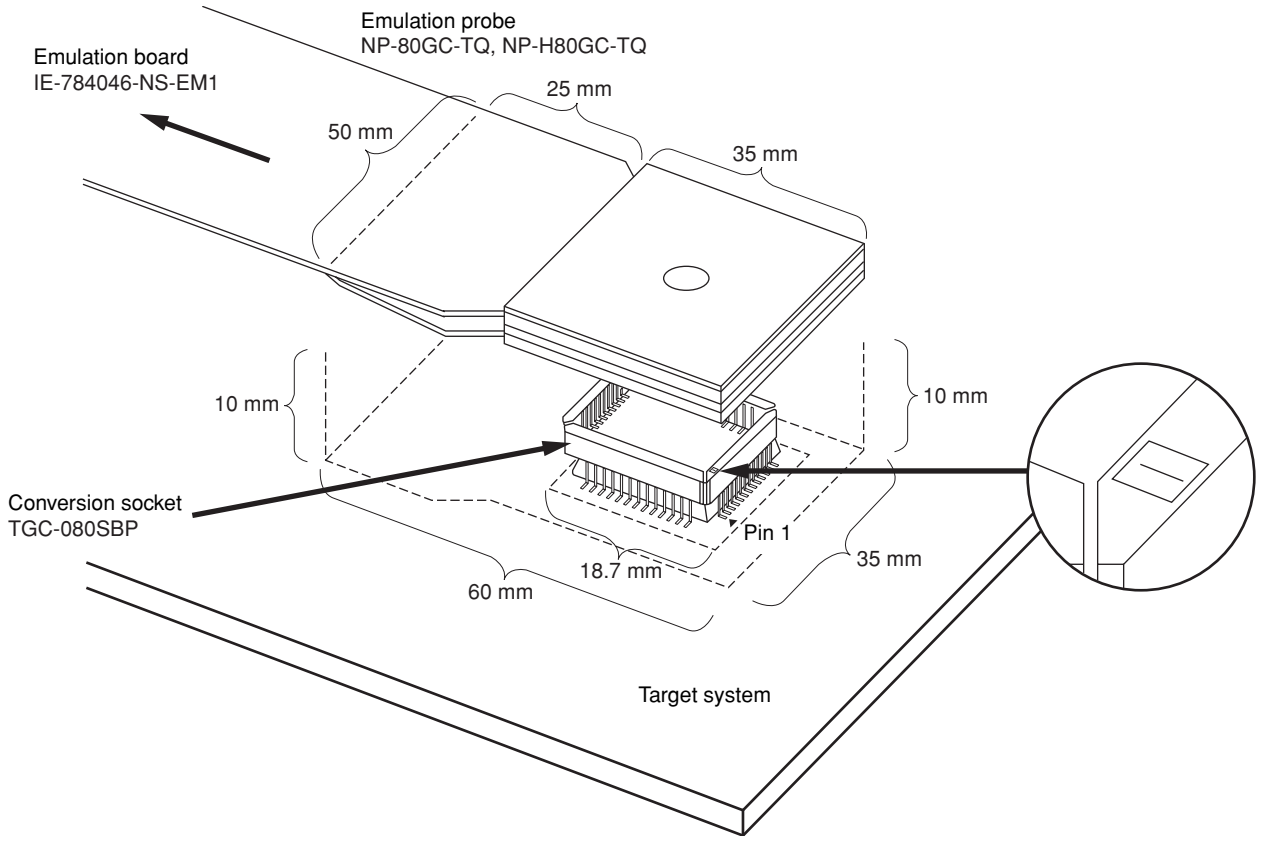
The connection condition diagrams for the emulation probe and conversion socket are shown below. Design the system considering the shape of components, etc. to be mounted on the target system in accordance with this configuration.

Figure A-2. Distance Between In-Circuit Emulator and Conversion Socket



Note 350 mm in case of the NP-H80GC-TQ.

Figure A-3. Target System Connection Conditions

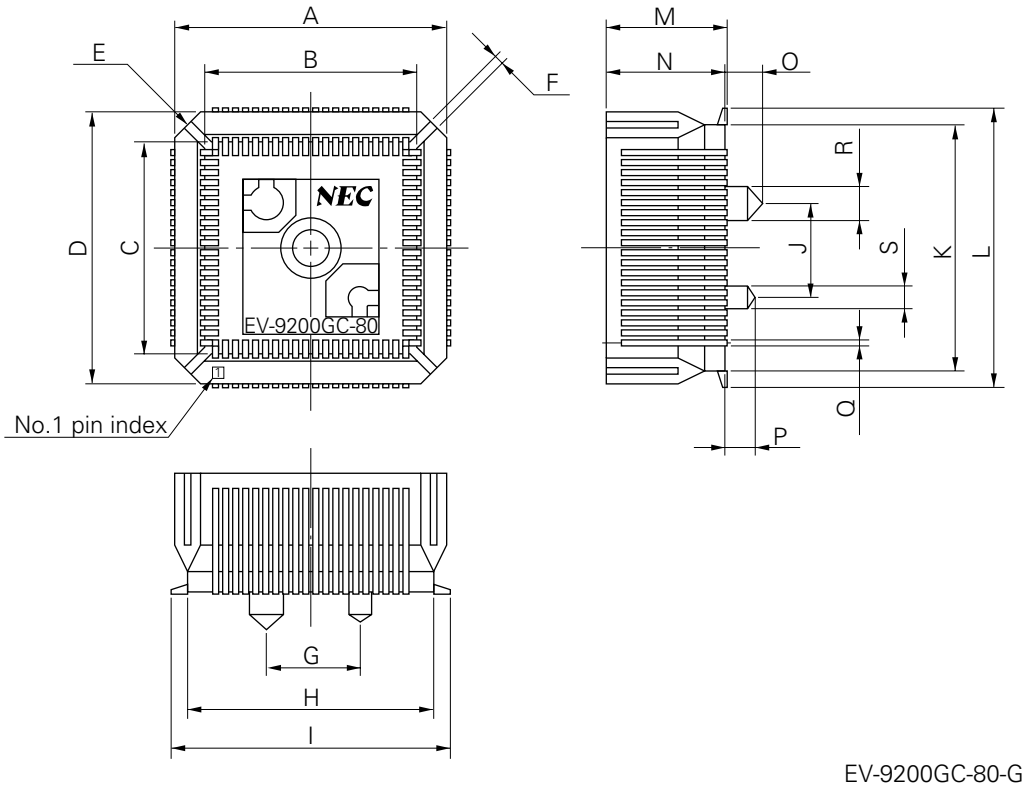


Remarks 1. NP-80GC-TQ and NP-H80GC-TQ are products made by Naito Densei Machida Mfg. Co., Ltd.

2. TGC-080SBP is a product made by Tokyo Eletech Corporation.

A.5 Deminsions of Conversion Socket (EV-9200GC-80) and Recommended Board Mounting Pattern

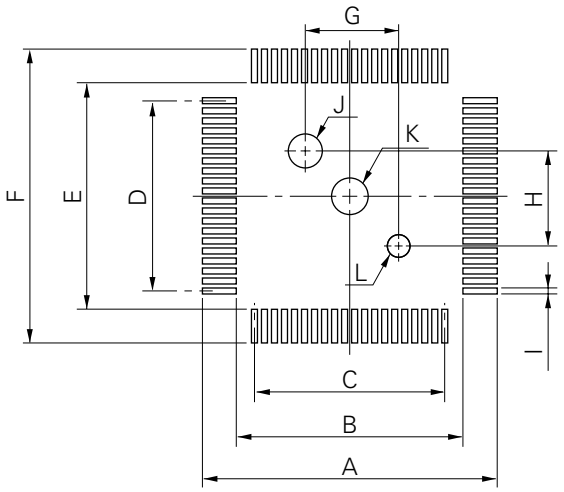
Figure A-4. Dimensions of EV-9200GC-80 (reference)



EV-9200GC-80-G1E

ITEM	MILLIMETERS	INCHES
A	18.0	0.709
B	14.4	0.567
C	14.4	0.567
D	18.0	0.709
E	4-C 2.0	4-C 0.079
F	0.8	0.031
G	6.0	0.236
H	16.0	0.63
I	18.7	0.736
J	6.0	0.236
K	16.0	0.63
L	18.7	0.736
M	8.2	0.323
N	8.0	0.315
O	2.5	0.098
P	2.0	0.079
Q	0.35	0.014
R	ϕ2.3	ϕ0.091
S	ϕ1.5	ϕ0.059

Figure A-5. Recommended Board Mounting Pattern of EV-9200GC-80 (reference)



EV-9200GC-80-P1E

ITEM	MILLIMETERS	INCHES
A	19.7	0.776
B	15.0	0.591
C	$0.65\pm0.02\times19=12.35\pm0.05$	$0.026^{+0.001}_{-0.002}\times0.748=0.486^{+0.003}_{-0.002}$
D	$0.65\pm0.02\times19=12.35\pm0.05$	$0.026^{+0.001}_{-0.002}\times0.748=0.486^{+0.003}_{-0.002}$
E	15.0	0.591
F	19.7	0.776
G	6.0 ± 0.05	$0.236^{+0.003}_{-0.002}$
H	6.0 ± 0.05	$0.236^{+0.003}_{-0.002}$
I	0.35 ± 0.02	$0.014^{+0.001}_{-0.001}$
J	$\phi2.36\pm0.03$	$\phi0.093^{+0.001}_{-0.002}$
K	$\phi2.3$	$\phi0.091$
L	$\phi1.57\pm0.03$	$\phi0.062^{+0.001}_{-0.002}$

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 Mount Manual" website (http://www.necel.com/pkg/en/mount/index.html).

★

APPENDIX B EMBEDDED SOFTWARE

For efficient development and maintenance of the μPD784054, the following embedded products are available.

RX78K4 Real-time OS	RX78K4 is a real-time OS conforming to the μITRON specifications. Tool (configurator) for generating nucleus of RX78K4 and plural information tables is supplied. Used in combination with an optional assembler package (RA78K4) and device file (DF784046). <Precaution when using RX78K4 in PC environment> The real-time OS is a DOS-based application. It should be used in the DOS Prompt when using in Windows.
	Part number: μS××××RX78K4

Caution When purchasing the RX78K4, fill in the purchase application form in advance and sign the User Agreement.

Remark ×××× and ΔΔΔ in the part number differ depending on the host machine and OS used.

μS××××RX78K4-ΔΔΔΔ

ΔΔΔΔ	Product Outline	Maximum Number for Use in Mass Production
001	Evaluation object	Do not use for mass-produced product.
100K	Mass-production object	0.1 million units
001M		1 million units
010M		10 million units
S01	Source program	Source program for mass-produced object

××××	Host Machine	OS	Supply Medium
AA13	PC-9800 Series	Windows (Japanese version)[Note]	3.5-inch 2HD FD
AB13	IBM PC/AT compatibles	Windows (Japanese version)[Note]	3.5-inch 2HC FD
BB13		Windows (English version)[Note]	
3P16	HP9000 Series 700	HP-UX (Rel. 9.05)	DAT (DDS)
3K13	SPARCstation	SunOS (Rel. 4.1.4),	3.5-inch 2HC FD
3K15		Solaris (Rel. 2.5.1)	1/4-inch CGMT

Note Can also be operated in DOS environment.

APPENDIX C REGISTER INDEX

APPENDIX D REVISION HISTORY

The revision history is described below. The “Applied to” column indicates the chapters in each edition.

(1/2)

Edition	Major Revisions from Previous Edition	Applied to
2nd edition	Change of μPD784054 from “under development” to “development completed”.	Throughout
	Addition of the following products to the relevant products: μPD784054(A), 784054(A1), 784054(A2)	
	Change of **78K/IV SERIES PRODUCT DEVELOPMENT DIAGRAM**.	**CHAPTER 1 GENERAL**
	Change of the minimum value of the supply voltage (V_{DD}) from 4.0 V to 4.5 V.	
	Addition of **1.3 Quality Grades**.	
	Addition of **1.9 Differences between μPD784054 and μPD784054(A)**.	
	Addition of **1.10 Differences between μPD784054(A), 784054(A1), and 784054(A2)**.	
	Addition of the functional description of the CLKOUT pin.	**CHAPTER 2 PIN FUNCTIONS**
	Addition of description in **(2) Capture/compare registers (CC00 through CC03)**.	**CHAPTER 7 TIMER 0**
	Addition of caution when the timer output is enabled while the active level is changed.	
	Addition of caution when the active level of the timer output is changed.	
	Addition of caution when the timer output is enabled while the active level is changed.	**CHAPTER 8 TIMER 1**
	Addition of caution when the active level of the timer output is changed.	
	Change of the description of <5> in (2) of **10.4.1 General cautions on use of watchdog timer** from “If the STOP mode or IDLE mode is entered as the result of an inadvertent program loop” to “If the STOP mode, HALT mode, or IDLE mode is entered as the result of an inadvertent program loop”.	**CHAPTER 10 WATCHDOG TIMER FUNCTION**
	Addition of caution and calculating method of the wait time if the reception completion interrupt is disabled when a reception error occurs.	**CHAPTER 12 ASYNCHRONOUS SERIAL INTERFACE/3-WIRE SERIAL I/O**
	Change of instructions in **14.9 When Interrupt Request and Macro Service Are Temporarily Held Pending**.	**CHAPTER 14 INTERRUPT FUNCTIONS**
	Change of description from “The watchdog timer must not be used to release the standby mode (STOP or IDLE mode)” to “The watchdog timer must not be used to release the standby mode (STOP, HALT, or IDLE mode)”.	**CHAPTER 16 STANDBY FUNCTION**
	Deletion of watchdog timer of “Non-maskable interrupt request (NMI pin input/ watchdog timer)”.	
	Addition of **Caution** concerning the malfunction that causes a wait for the oscillation stabilization time when the IDLE mode is released.	
	Addition of note on output of CLKOUT pin.	**CHAPTER 20 CAUTIONS ON USING DEVELOPMENT TOOLS**
	General revision for supporting IE-78K4-NS.	**APPENDIX A DEVELOPMENT TOOLS**
	Change of target host machines.	**APPENDIX B EMBEDDED SOFTWARE**
	Change of versions of OSs to be supported.	

(2/2)

Edition	Major Revisions from Previous Edition	Applied to
3rd edition	• Completion of development of the following product μPD78F4046 • Update of **78K/IV Series Product Lineup**	**CHAPTER 1 GENERAL**
	Addition of description on BWD pin in **Table 2-4 I/O Circuit Type of Each Pin and Recommended Processing of Unused Pins**	**CHAPTER 2 PIN FUNCTIONS**
	Addition of cautions on start bit during UART transmission to **12.5 Cautions**	**CHAPTER 12 ASYNCHRONOUS SERIAL INTERFACE/ 3-WIRE SERIAL I/O**
	• Modification of **Figure 16-1 Diagram of Standby Mode Transition** • Modification of description in **16.6 (5) A/D converter**	**CHAPTER 16 STANDBY FUNCTION**
	• Addition of description on Flashpro III • Addition of cautions in **18.3 Cautions**	**CHAPTER 18 μPD78F4046**
	Addition of chapter	**CHAPTER 20 ELECTRICAL SPECIFICATIONS (μPD784054)**
	Addition of chapter	**CHAPTER 21 ELECTRICAL SPECIFICATIONS (μPD784054(A))**
	Addition of chapter	**CHAPTER 22 ELECTRICAL SPECIFICATIONS (μPD784054(A1)**
	Addition of chapter	**CHAPTER 23 ELECTRICAL SPECIFICATIONS (μPD784054(A2))**
	Addition of chapter	**CHAPTER 24 TIMING CHARTS**
	Addition of chapter	**CHAPTER 25 PACKAGE DRAWING**
	Addition of chapter	**CHAPTER 26 RECOMMENDED SOLDERING CONDITIONS**
	• Addition of description on host machines and OSs • Addition of SP78K4 to **A.1 Language Processing Software,** modification of description in **Remark** • Addition of description on Flashpro III in **Remark** in **A.2 Flash Memory Writing Tools** • Addition and modification of description in **A.3.1 Hardware** • Modification of description in **Remark** in **A.3.2 Software** • Addition of **A.4 Cautions on Designing Target System**	**APPENDIX A DEVELOPMENT TOOLS**
	Modification of description	**APPENDIX B EMBEDDED SOFTWARE**
3rd edition (Modification Version)	Modification of **1.2 Ordering Information**	**CHAPTER 1 GENERAL**
	Modification of **1.3 Quality Grades**	
	Addition of **Table 26-1. Surface Mounting Type Soldering Conditions (2)**	**CHAPTER 26 RECOMMENDED SOLDERING CONDITIONS**