

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.

User's Manual

Phase-out/Discontinued

μPD78366A

16-/8-BIT SINGLE-CHIP MICROCONTROLLER

Hardware

μPD78363A

μPD78365A

μPD78366A

μPD78368A

μPD78P368A

[MEMO]

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

Note:

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

② HANDLING OF UNUSED INPUT PINS FOR CMOS

Note:

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

③ STATUS BEFORE INITIALIZATION OF MOS DEVICES

Note:

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

QTOP is a trademark of NEC Corporation.

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

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

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

SPARCstation is a trademark of SPARC International, Inc.

SunOS is a trademark of Sun Microsystems Inc.

NEWS and NEWS-OS are trademarks of Sony Corporation.

TRON stands for The Realtime Operating system Nucleus.

ITRON stands for Industrial TRON.

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

License not needed: μ PD78365A, 78P368AKL-S

The customer must judge the need for licence: μ PD78363A, 78366A, 78368A, 78P368AGF-3B9

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

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

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

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

NEC devices are classified into the following three quality grades:

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

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

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

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

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

Anti-radioactive design is not implemented in this product.

Regional Information

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

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

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

NEC Electronics Inc. (U.S.)

Santa Clara, California
Tel: 800-366-9782
Fax: 800-729-9288

NEC Electronics (Germany) GmbH

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

NEC Electronics (UK) Ltd.

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

NEC Electronics Italiana s.r.l.

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

NEC Electronics (Germany) GmbH

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

NEC Electronics (France) S.A.

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

NEC Electronics (France) S.A.

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

NEC Electronics (Germany) GmbH

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

NEC Electronics Hong Kong Ltd.

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

NEC Electronics Hong Kong Ltd.

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

NEC Electronics Singapore Pte. Ltd.

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

NEC Electronics Taiwan Ltd.

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

NEC do Brasil S.A.

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

Major Revisions in This Version

Page	Contents
Throughout	The μ PD78368A has been added as a target device. The μ PD78P368AKL-S has been developed.
P.204, 205	CHAPTER 8 A/D CONVERTER 8.2 A/D Converter Mode Register (ADM) The conversion time by setting the FR bit has been changed.
P.274, 275	CHAPTER 10 CLOCKED SERIAL INTERFACE 10.6.1 SBI data format A caution has been added regarding the bus release signal and command signal.

The mark ★ shows major revised points.

PREFACE

Users

This manual is for engineers who intend to learn the capabilities of the μ PD78366A subseries application program development.

This manual describes the following products in the μ PD78366A subseries **Note**.

- μ PD78363A, 78365A, 78366A, 78368A, 78P368A

Note In addition to the other products, the μ PD78366A subseries provides μ PD78361A, 78362A, and 78P364A. For details on the μ PD78361A, 78362A, and 78P364A, refer to the **μ PD78362A User's Manual - Hardware (U10745E)**.

Purpose

The purpose of this manual is to help understand the hardware capabilities of the μ PD78366A subseries as listed below.

Organization

The μ PD78366A subseries manual is separated into two parts: Hardware (the present manual) and Instructions.

Hardware

- Pin functions
- Internal hardware function
- Interrupts
- List of instruction set

Instructions

- CPU functions
- Addressing
- List of instruction set
- Explanation of each instruction

Caution

Cautions on the use of the μ PD78366A subseries are summarized in CHAPTER 20 CAUTIONS. Read them before using the product.
For the up-to-the-minute information about this product, please contact an NEC representative.

How to read this manual

Before using this manual, the user should have a general knowledge of the electronics, logical circuit, and microcontroller fields.

- **To use this manual as μ PD78363A, 78365A, 78368A, and 78P368A manual:**
 - Unless otherwise specified, the μ PD78366A is treated as the representative model throughout this manual. When you use the μ PD78363A, 78365A, 78368A, or 78P368A, take μ PD78366A for μ PD78363A, 78365A, 78368A, or 78P368A. In the description common to the one-time PROM and EPROM models, the term PROM is used.

- **To check the details of a register if you know the name of the register:**
→ Look it up in **APPENDIX C REGISTER INDEX**.
- **To know the detailed functions:**
→ Look it up in **APPENDIX D FUNCTION INDEX**.
- **To know the μ PD78366A subseries instruction function in detail:**
→ Refer to the **μ PD78365 User's Manual Instruction (U12117E)**.
- **To understand the general functions of the μ PD78366A subseries:**
→ Read the entire manual in the order of the table of contents.

Legend

Data weight	: Higher digits on the left side Lower digits on the right side
Active low	: $\overline{\text{xxxx}}$ (Pins and signal names are over-scored.)
Memory map address	: Low-order address on the upper side High-order address on the lower side
Note	: Explanation of the indicated part of the text
Caution	: Information requesting the user's special attention
Remark	: Supplementary information
Numeric value	: Binary: xxxxB or xxxx Decimal: xxxx Hexadecimal: xxxxH

Register representation

	7	6	5	4	③	2	1	0
EDC	B	1	0	×	A	1	0	×
Register name								

Encircled bit number indicates that the bit name is defined as a reserved word in an NEC's assembler (RA78K3) and as header file sfrbit.h in C compiler (CC78K3).

Write	Read
Write 0 or 1. Operation is not affected by either.	0 or 1 is read.
Write 0.	
Write 1.	
Write value according to function to be used.	Value according to operation status is read.

Never write the combination of codes marked “setting prohibited” in the register chart.

Confusing characters : 0 (zero), O (oh)
: 1 (numeral), l (lowercase of L), I (uppercase of i)

Related Documents

• Documents related to devices

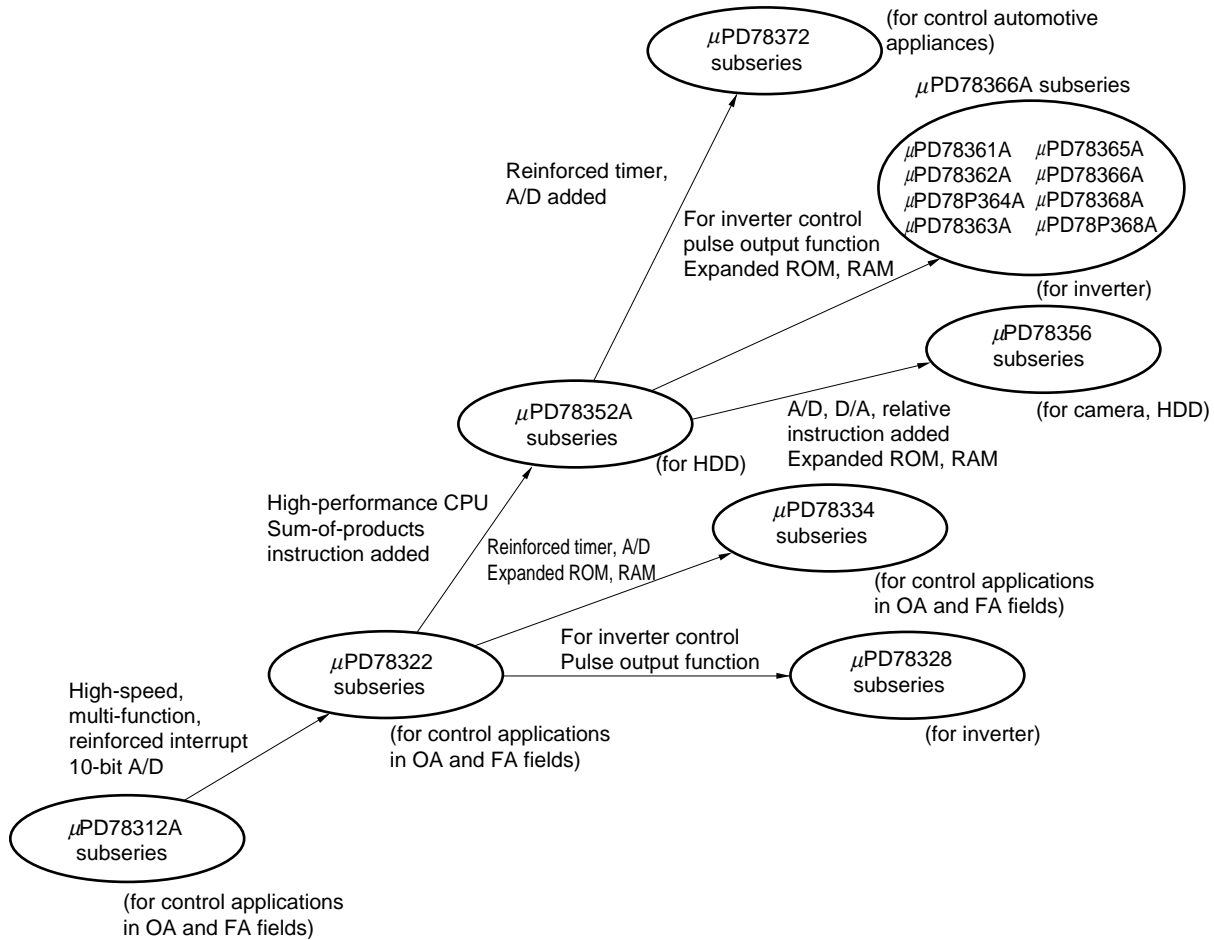
Document Name	Document No.	
	Japanese	English
μPD78362A Data Sheet	U10098J	10098E
μPD78P364A Data Sheet	U10106J	10106E
μPD78363A, 78365A, 78366A Data Sheet	U11109J	U11109E
μPD78P368A Data Sheet	U11373J	U11373E
μPD78362A User's Manual -Hardware	U10745J	U10745E
μPD78366A User's Manual -Hardware	U10205J	This manual
μPD78356 User's Manual -Instruction	U12117J	U12117E
μPD78362A Special Function Register Table	U10210J	—
μPD78366A Special Function Register Table	U10107J	—
μPD78352A Instruction Set	U11955J	—

• Documents related to development tools

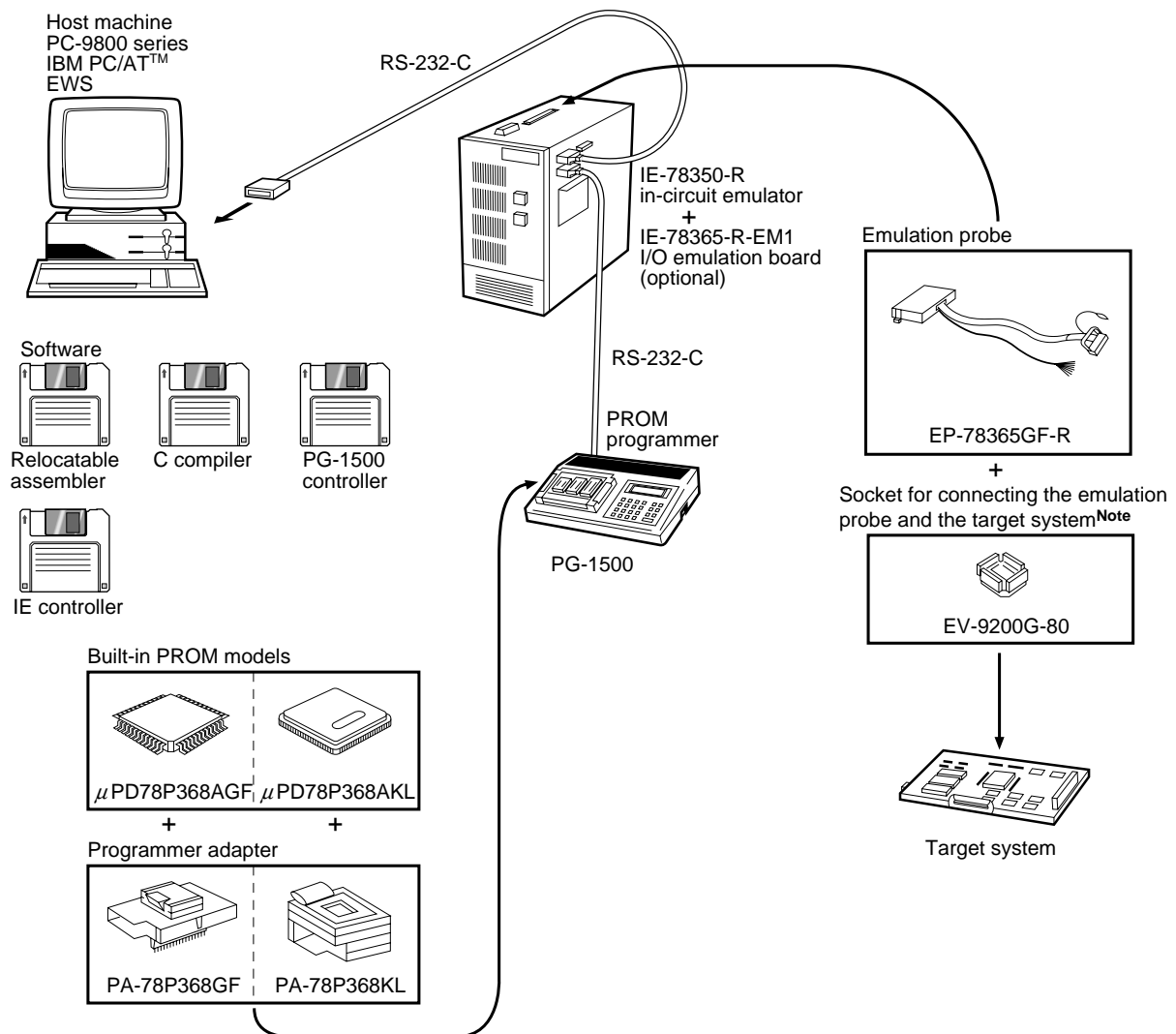
Document Name		Document No.	
		Japanese	English
IE-78350-R User's Manual	Hardware	EEU-754	EEU-1366
	Software	EEU-753	EEU-1376
IE-78365-R-EM1 User's Manual		EEU-924	EEU-1454
EP-78365GF-R User's Manual		EEU-955	EEU-1488

Caution The related documents are subject to change without notice. Be sure to use the latest version for your design work.

78K/III Series Product Development



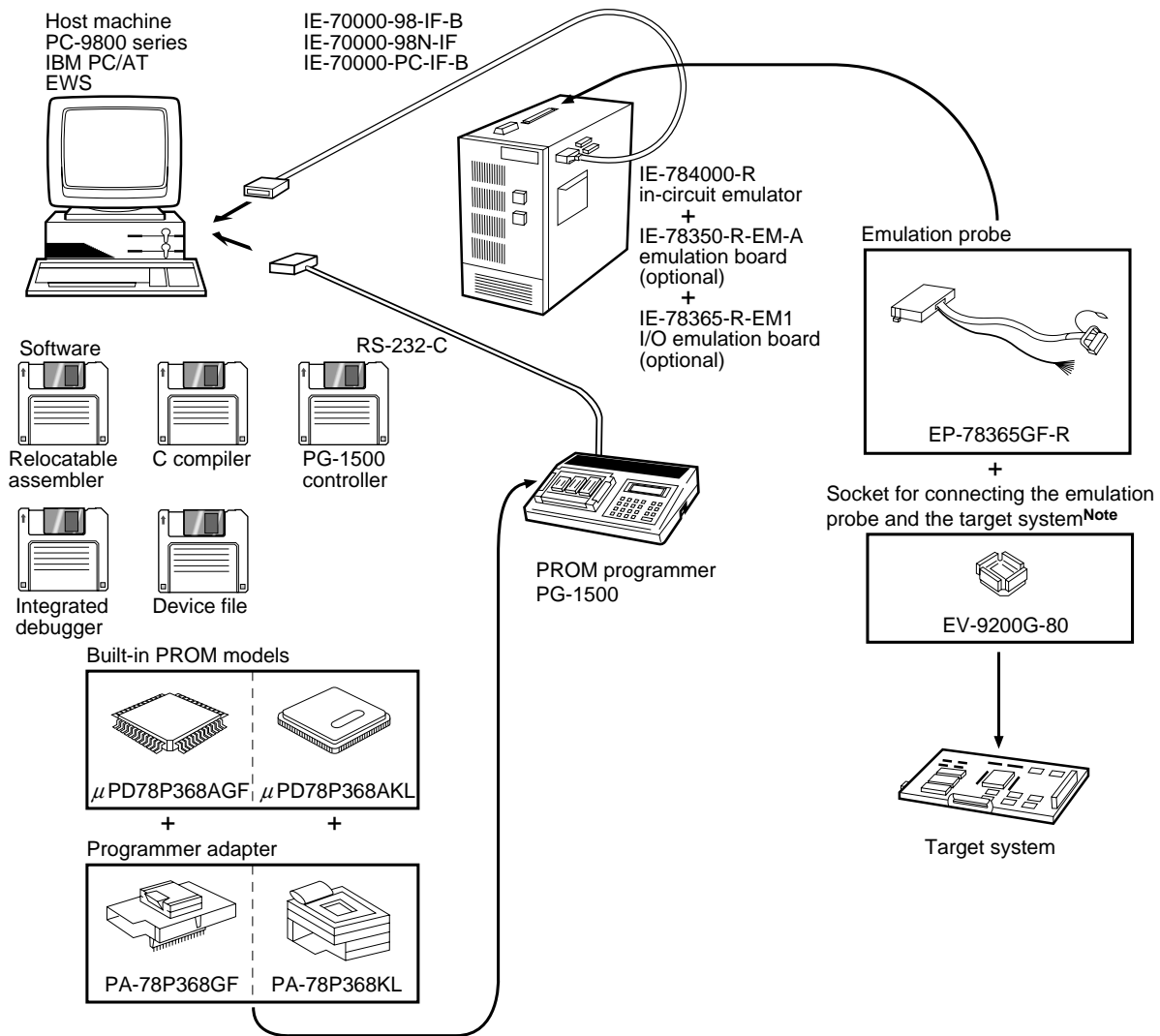
DEVELOPMENT TOOL CONFIGURATION (WHEN USING IE CONTROLLER)



Note A socket is provided with the emulation probe.

- Remarks**
1. Host machine and PG-1500 can be directly connected by RS-232-C.
 2. 3.5-inch FD represents the distributed media of software in this figure.

DEVELOPMENT TOOL CONFIGURATION (WHEN USING INTEGRATED DEBUGGER)



Note A socket is provided with the emulation probe.

- Remarks**
1. Desk top-type PC represents host machine in this figure.
 2. 3.5-inch FD represents the distributed media of software in this figure.

TABLE OF CONTENTS

CHAPTER 1 GENERAL	1
1.1 Features.....	2
1.2 Application Fields	3
1.3 Ordering Information	3
1.4 Pin Configuration (Top View)	4
1.4.1 Normal operation mode	4
1.4.2 PROM programming mode (μ PD78P368A only: MODE0/V _{PP} = H, MODE1 = L)	6
1.5 Block Diagram	8
1.6 Functional Outline	9
1.7 Differences among μPD78366A Subseries Products	11
1.8 Application Example	12
CHAPTER 2 PIN FUNCTIONS	13
2.1 Pin Function List.....	13
2.1.1 Normal operation mode	13
2.1.2 PROM programming mode (μ PD78P368A only: MODE0/V _{PP} = H, MODE1 = L)	16
2.2 Pin Functions	17
2.2.1 Normal operation mode	17
2.2.2 PROM programming mode (μ PD78P368A only).....	25
2.3 I/O Circuits and Processing of Unused Pins	26
CHAPTER 3 CPU ARCHITECTURE	29
3.1 Memory Space	29
3.1.1 Vector table area	34
3.1.2 CALLT instruction table area	35
3.1.3 CALLF instruction entry area	35
3.1.4 Internal RAM area	35
3.1.5 Special function register area	39
3.1.6 External memory area	39
3.2 Processor Register	40
3.2.1 Control register	41
3.2.2 General-purpose register	47
3.2.3 Special function register (SFR)	49
3.3 Data Memory Addressing	56
3.3.1 General-purpose register addressing	59
3.3.2 Short direct addressing	60
3.3.3 Special function register (SFR) addressing	60
CHAPTER 4 SUMMARY OF BLOCK FUNCTION	61
4.1 Execution Unit	61
4.2 Bus Control Unit	61

4.3	Program Memory and Data Memory	61
4.4	Ports	62
4.5	Real-Time Pulse Unit	62
4.6	Real-Time Output Port	62
4.7	A/D Converter	63
4.8	Serial Interface	63
4.9	PWM Output Unit	63
4.10	Watchdog Timer	63
4.11	Interrupt Controller	63
CHAPTER 5 PORT FUNCTIONS		65
5.1	Hardware Configuration	65
5.2	Port Functions	75
5.2.1	Functions and features of I/O ports	76
5.2.2	I/O mode setting	77
5.2.3	Control mode setting	79
5.2.4	Specifying pull-up resistor	86
CHAPTER 6 CLOCK GENERATOR		89
CHAPTER 7 REAL-TIME PULSE UNIT.....		93
7.1	RPU Configuration	94
7.2	Timer 0	94
7.2.1	Configuration	94
7.2.2	Control registers	102
7.2.3	Operation	110
7.3	Timer 1	141
7.3.1	Configuration	141
7.3.2	Control registers	142
7.3.3	Operation	143
7.4	Timer 2	144
7.4.1	Configuration	144
7.4.2	Control registers	146
7.4.3	Operation	150
7.5	Timer 3	156
7.5.1	Configuration	156
7.5.2	Control registers	158
7.5.3	Operation	163
7.6	Timer 4	171
7.6.1	Configuration	171
7.6.2	Control registers	173
7.6.3	Operation	180
7.7	Real-Time Output Function	195
7.7.1	Configuration	195
7.7.2	Control registers	196
7.7.3	Operation	197

CHAPTER 8 A/D CONVERTER	199
8.1 Configuration	200
8.2 A/D Converter Mode Register (ADM)	203
8.3 A/D Conversion Result Register (ADCR)	207
8.4 Operation	209
8.4.1 Basic operation of A/D converter	209
8.4.2 Operation mode of A/D converter	213
8.5 How to Read A/D Converter Characteristic Tables	222
CHAPTER 9 ASYNCHRONOUS SERIAL INTERFACE	225
9.1 Asynchronous Serial Interface Configuration	226
9.2 Selecting Pins for Serial Communication	228
9.3 Setting Pins for Serial Communication	230
9.4 Data Format Setting	232
9.5 Baud Rate Setting	234
9.5.1 Baud rate generator configuration	236
9.5.2 Specific baud rate setting	238
9.6 Transmitting Data	240
9.7 Receiving Data	242
9.8 Transmitting/Receiving Data Using Macro Service	244
9.9 If Reception Error Occurs	246
CHAPTER 10 CLOCKED SERIAL INTERFACE	249
10.1 Clocked Serial Interface Configuration	250
10.2 Setting Pins for Serial Communication	252
10.3 Baud Rate Setting	254
10.3.1 Baud rate generator configuration	256
10.3.2 Specific baud rate setting	258
10.4 Two Operation Modes of Clocked Serial Interface	260
10.5 Three-Wire Serial I/O Mode Setting	262
10.5.1 Transmission in three-wire serial I/O mode	264
10.5.2 Reception in three-wire serial I/O mode	266
10.5.3 Transmission/reception in three-wire serial I/O mode	268
10.5.4 Corrective action in case shift operation is not synchronized	271
10.6 SBI Mode Setting	272
10.6.1 SBI data format	274
10.6.2 Controlling and detecting status of serial bus	280
10.6.3 Communicating with SBI	286
10.6.4 Operation only when address is received	290
CHAPTER 11 PWM SIGNAL OUTPUT FUNCTION	293
11.1 Configuration	293
11.2 Control Register	295
11.2.1 PWM control registers (PWMC0, PWMC1)	295

11.2.2	PWM buffer registers (PWM0, PWM1)	296
11.2.3	Compare registers (CMP0, CMP1)	296
11.3	Operation	297
CHAPTER 12	WATCHDOG TIMER	299
12.1	Configuration	299
12.2	Watchdog Timer Mode Register (WDM)	300
12.3	Watchdog Timer Output Pin	302
12.4	Application Example	302
CHAPTER 13	INTERRUPT FUNCTION.....	303
13.1	Interrupt Requests	305
13.1.1	Non-maskable interrupt	305
13.1.2	Maskable interrupt	305
13.1.3	Software interrupt	305
13.1.4	Op-code trap interrupt	306
13.2	Interrupt Servicing Mode	306
13.2.1	Vectored interrupt service	306
13.2.2	Macro service	306
13.2.3	Context switching	306
13.3	Control Registers	307
13.3.1	Interrupt control registers	309
13.3.2	Interrupt mask flag registers (MK0)	313
13.3.3	Interrupt mode control register (IMC)	315
13.3.4	In-service priority register (ISPR)	316
13.3.5	Program status word (PSW)	317
13.4	Non-Maskable Interrupt Acknowledgment Operation	318
13.5	Maskable Interrupt Acknowledgment Operation	322
13.5.1	Vectored interrupt	324
13.5.2	Context switching	324
13.5.3	Maskable interrupt priority	326
13.6	Software Interrupt Acknowledgment Operation	332
13.6.1	Software interrupt acknowledgment operation by BRK instruction	332
13.6.2	Software interrupt (context switching) acknowledgment operation by BRKCS instruction	332
13.7	Op-Code Trap Interrupt Acknowledgment Operation	335
13.8	Macro Service Function	336
13.8.1	Outline of macro service	336
13.8.2	Basic function of macro service	339
13.8.3	Operation at completion of macro service	341
13.8.4	Macro service control register	342
13.8.5	Macro service mode	344
13.8.6	Macro service operation	344
13.9	Cases where Interrupt Request and Macro Service are Temporarily Held Pending	354
13.10	Instructions whose Execution Is Temporarily Suspended by Interrupts and Macro Services	356

CHAPTER 14	STANDBY FUNCTION	357
14.1	Function Overview	357
14.2	Standby Control Register (STBC)	358
14.3	Operation	360
14.3.1	HALT mode	360
14.3.2	STOP mode	363
CHAPTER 15	RESET FUNCTION	373
CHAPTER 16	BUS INTERFACE FUNCTION	377
16.1	External Device Expansion Function for μ PD78366A	377
16.2	Access to External Devices by μ PD78365A	383
16.3	Control Registers	384
16.3.1	Memory expansion mode register	384
16.3.2	Programmable wait control register	386
CHAPTER 17	PROGRAMMING FOR μPD78P368A	389
17.1	Operating Mode	390
17.2	Procedure for Writing on PROM (Page Program Mode)	391
17.3	Procedure for Writing on PROM (Byte Program Mode)	394
17.4	Procedure for Reading from PROM	397
17.5	Erase Characteristics (μ PD78P368AKL only)	398
17.6	Sticker on Erase Window (μ PD78P368AKL only)	398
17.7	Screening of One-Time PROM Products	398
CHAPTER 18	INSTRUCTION SET	399
18.1	Operand Identifier and Description	399
18.2	Legend	402
18.3	Notational Symbols in Flag Operation Field	403
18.4	Differences between μ PD78366A and μ PD78328 in Instruction Set	403
18.5	Operations of Basic Instructions	404
CHAPTER 19	INSTRUCTION EXECUTION RATE	421
19.1	Memory Space and Access Speed	421
19.1.1	Main RAM and peripheral RAM	421
19.1.2	Memory access	422
19.2	Interrupt Execution Rate	428
19.3	Calculating Number of Execution Clocks	429
CHAPTER 20	CAUTIONS	433
20.1	Cautions for CHAPTER 2 PIN FUNCTIONS	433
20.2	Cautions for CHAPTER 3 CPU ARCHITECTURE	433
20.3	Cautions for CHAPTER 5 PORT FUNCTIONS	434

20.4	Cautions for CHAPTER 6 CLOCK GENERATOR	435
20.5	Cautions for CHAPTER 7 REAL-TIME PULSE UNIT	436
20.6	Cautions for CHAPTER 8 A/D CONVERTER	438
20.7	Cautions for CHAPTER 9 ASYNCHRONOUS SERIAL INTERFACE	439
20.8	Cautions for CHAPTER 10 CLOCKED SERIAL INTERFACE.....	440
20.9	Cautions for CHAPTER 11 PWM SIGNAL OUTPUT FUNCTION	440
20.10	Cautions for CHAPTER 12 WATCHDOG TIMER	440
20.11	Cautions for CHAPTER 13 INTERRUPT FUNCTION	441
20.12	Cautions for CHAPTER 14 STANDBY FUNCTION	444
20.13	Cautions for CHAPTER 15 RESET FUNCTION	444
20.14	Cautions for CHAPTER 16 BUS INTERFACE FUNCTION	445
20.15	Cautions for CHAPTER 19 INSTRUCTION EXECUTION RATE	445
APPENDIX A DIFFERENCE BETWEEN μ PD78366A AND μ PD78328.....		447
APPENDIX B TOOLS.....		451
B.1 Development Tools		451
B.2 Built-In Software		456
APPENDIX C REGISTER INDEX		459
C.1 Register Index (In Alphabetical Order with Respect to Register Name).....		459
C.2 Register Index (In Alphabetical Order with Respect to Register Symbol)		462
APPENDIX D FUNCTION INDEX		467
APPENDIX E REVISION HISTORY		475

LIST OF FIGURES (1/6)

Figure No.	Title	Page
2-1	I/O Circuit of Each Pin	27
3-1	Memory Map (μ PD78365A, 78366A)	30
3-2	Memory Map (μ PD78368A)	31
3-3	Memory Map (μ PD78363A)	32
3-4	Memory Map (μ PD78P368A)	33
3-5	Register Configuration	40
3-6	Format of Program Status Word	42
3-7	Format of CPU Control Word	46
3-8	Manipulation Bits of General-Purpose Registers	47
3-9	Addressing Space of Data Memory (μ PD78365A, 78366A)	56
3-10	Addressing Space of Data Memory (μ PD78368A)	57
3-11	Addressing Space of Data Memory (μ PD78363A)	58
3-12	Addressing Space of Data Memory (μ PD78P368A)	59
5-1	Basic I/O Port Configuration	66
5-2	Port Specified as Output Port	67
5-3	Port Specified as Input Port	68
5-4	When Control Is Specified	69
5-5	Format of Port Read Control Register	72
5-6	Control (output port specified)	73
5-7	Port Configuration	75
5-8	Format of Port 0 Mode Register	77
5-9	Format of Port 1 Mode Register	77
5-10	Format of Port 3 Mode Register	78
5-11	Format of Port 5 Mode Register	78
5-12	Format of Port 8 Mode Register	78
5-13	Format of Port 9 Mode Register	79
5-14	Format of Port 0 Mode Control Register	82
5-15	Format of Port 3 Mode Control Register	83
5-16	Format of Port 8 Mode Control Register	84
5-17	Format of Memory Expansion Mode Register	85
5-18	Format of Pull-Up Resistor Option Register L	87
5-19	Format of Pull-Up Resistor Option Register H	87
6-1	Block Diagram of Clock Generator	89
6-2	External Circuitry of System Clock Oscillation Circuit	90
6-3	Examples of Wrong Resonator Connection Circuitry	91
7-1	Block Diagram of Timer 0 (PWM mode 0 ... symmetric triangular wave, asymmetric triangular wave)	95
7-2	Block Diagram of Timer 0 (PWM mode 0 ... toothed wave)	96
7-3	Block Diagram of Timer 0 (PWM mode 1)	97

LIST OF FIGURES (2/6)

Figure No.	Title	Page
7-4	Format of Timer Unit Mode Register 0	103
7-5	Configuration of Output Driver off Function	105
7-6	Format of Timer Control Register 0	106
7-7	Format of Timer Control Register 1	108
7-8	External Interrupt Mode Register 0 Format	109
7-9	Block Diagram of RTP Output Function of TO00-TO05 Pins	111
7-10	Operation Timing in PWM Mode 0 (symmetric triangular wave)	116
7-11	Operation Timing in PWM Mode 0 (symmetric triangular wave, BFCM0x ≥ CM03)	117
7-12	Operation Timing in PWM Mode 0 (symmetric triangular wave, BFCM0x = 0000H)	118
7-13	Operation Timing in PWM Mode 0 (asymmetric triangular wave)	123
7-14	Operation Timing in PWM Mode 0 (asymmetric triangular wave, BFCM0x ≥ CM03)	124
7-15	Operation Timing in PWM Mode 0 (asymmetric triangular wave, BFCM0x > CM03)	125
7-16	Operation Timing in PWM Mode 0 (asymmetric triangular wave, BFCM0x = 0000H) (1)	126
7-17	Operation Timing in PWM Mode 0 (asymmetric triangular wave, BFCM0x = 0000H) (2)	127
7-18	Operation Timing in PWM Mode 0 (asymmetric triangular wave, BFCM0x = CM03)	128
7-19	Operation Timing in PWM Mode 0 (toothed wave)	132
7-20	Operation Timing in PWM Mode 0 (toothed wave, BFCM0x > CM03)	133
7-21	Operation Timing in PWM Mode 0 (toothed wave, BFCM0x = CM03)	134
7-22	Operation Timing in PWM Mode 0 (toothed wave, BFCM0x = 0000H)	135
7-23	Operation Timing in PWM Mode 1	140
7-24	Block Diagram of Timer 1	141
7-25	Format of Timer Control Register 1	142
7-26	Example of Compare Operation (TM1, interval timer mode)	143
7-27	Block Diagram of Timer 2	144
7-28	Block Diagram of INTP3/INTCC20 Generation	145
7-29	Format of Timer Control Register 2	147
7-30	Format of External Interrupt Mode Register 1	148
7-31	Format of Sampling Control Register 0	149
7-32	Basic Operation of Timer 2 (TM2)	150
7-33	Example of TM2 Capture Operation (free-running operation)	152
7-34	Example of TM2 Capture Operation (interval operation)	152
7-35	Example of TM2 Compare Operation (free-running operation)	153
7-36	Example of TM2 Compare Operation (interval operation)	153
7-37	Block Diagram of Sampling Circuit (TM2)	154
7-38	Sampling Timing Chart (TM2)	154
7-39	Block Diagram of Timer 3	156
7-40	Block Diagram of INTP0/INTCC30 Generation	157
7-41	Format of Timer Control Register 3 (TMC3)	159
7-42	Format of External Interrupt Mode Register 0	160
7-43	Format of External Interrupt Mode Register 1	160
7-44	Format of Sampling Control Register 0	161
7-45	Format of Sampling Control Register 1	162
7-46	Basic Operation of Timer 3 (TM3)	163

LIST OF FIGURES (3/6)

Figure No.	Title	Page
7-47	Example of TM3 Capture Operation (free-running operation)	165
7-48	Example of TM3 Capture Operation (interval operation)	166
7-49	Example of TM3 Compare Operation (free-running operation)	167
7-50	Example of TM3 Compare Operation (interval operation)	168
7-51	Block Diagram of Sampling Circuit (TM3)	169
7-52	Sampling Timing Chart (TM3)	169
7-53	Block Diagram of Timer 4 (general-purpose timer mode)	171
7-54	Block Diagram of Timer 4 (UDC mode)	171
7-55	Format of Timer Unit Mode Register 1	174
7-56	Format of Timer Control Register 4	176
7-57	Basic Operation of Timer 4 (TM4)	180
7-58	TM4 Clear Operation (during up count in UDC mode)	184
7-59	TM4 Clear Operation (during down count in UDC mode)	184
7-60	Internal Clock Operation in UDC Mode	186
7-61	Example of Operation in Mode 1 (when valid edge of TIUD pin is rising edge)	187
7-62	Example of Operation in Mode 1 (when valid edge of TIUD pin is rising edge)	187
7-63	Example of Operation in Mode 2 (when valid edge of TIUD pin is rising edge)	188
7-64	Example of Operation in Mode 3 (when valid edge of TIUD pin is rising edge)	189
7-65	Example of Operation in Mode 3 (when valid edge of TIUD pin is rising edge)	190
7-66	Example of Operation in Mode 4	191
7-67	Block Diagram of Timer 4 (PWM output operation)	192
7-68	Example of PWM Output Operation of TO40	194
7-69	Block Diagram of Real-Time Output Port	195
7-70	Format of Real-Time Output Port Mode Register	196
7-71	Example of Real-Time Output Function Operation (P00 pin)	198
8-1	Block Diagram of A/D Converter	200
8-2	Example of Connecting Capacitor to A/D Converter Pin	201
8-3	Format of A/D Converter Mode Register	204
8-4	Word Access to ADCR Register	207
8-5	Byte Access to ADCR Register	208
8-6	A/D Conversion Basic Operation (in select mode, with software trigger)	209
8-7	A/D Conversion Basic Operation (in mixed mode, with external trigger)	210
8-8	Rewriting ADM during A/D Conversion (in scan mode, with software trigger)	211
8-9	A/D Conversion in Select Mode (1-buffer mode)	213
8-10	Example of Operation Timing in Select Mode (1-buffer mode)	214
8-11	A/D Conversion in Select Mode (4-buffer mode)	215
8-12	Example of Operation Timing in Select Mode (in 4-buffer mode)	215
8-13	A/D Conversion in Scan Mode	216
8-14	Example of Operation Timing in Scan Mode	217
8-15	A/D Conversion in Mixed Mode (select processing in 1-buffer mode)	218
8-16	Example of Operation Timing in Mixed Mode (select processing in 1-buffer mode) (software trigger)	219

LIST OF FIGURES (4/6)

Figure No.	Title	Page
8-17	Example of Operation Timing in Mixed Mode (select processing in 1-buffer mode) (external trigger or interrupt trigger)	219
8-18	A/D Conversion Timing in Mixed Mode (select processing in 4-buffer mode)	220
8-19	Example of Operation Timing in Mixed Mode (select processing in 4-buffer mode) (software trigger)	221
8-20	Example of Operation Timing in Mixed Mode (select processing in 4-buffer mode) (external trigger or interrupt trigger)	221
8-21	Total Error	223
8-22	Quantized Error	223
8-23	Zero Scale Error	224
8-24	Full Scale Error	224
8-25	Non-Linearity Error	224
9-1	Block Diagram of Asynchronous Serial Interface	227
9-2	Setting of ASIM Register (pin selection)	229
9-3	Format of Port 3 Mode Control Register	230
9-4	Format of Port 3 Mode Register	231
9-5	Format of Transmit/Receive Data of Asynchronous Serial Interface	232
9-6	Setting of ASIM Register (data format)	233
9-7	Setting of ASIM Register (serial clock)	235
9-8	Block Diagram of Baud Rate Generator	237
9-9	Format of Baud Rate Generator Control Register	237
9-10	Asynchronous Serial Interface Transmission End Interrupt Timing	241
9-11	Asynchronous Serial Interface Reception End Interrupt Timing	242
9-12	Setting of ASIM Register (reception enabled)	243
9-13	UART Transmitting/Receiving Using Macro Service	245
9-14	Reception Error Timing	246
9-15	Format of Asynchronous Serial Interface Status Register	247
10-1	Block Diagram of Clocked Serial Interface	251
10-2	Format of Port 3 Mode Control Register	252
10-3	Format of Port 3 Mode Register	253
10-4	Setting of CSIM Register (serial clock)	255
10-5	Block Diagram of Baud Rate Generator	257
10-6	Format of Baud Rate Generator Control Register	259
10-7	Example of System Configuration in Three-Wire Serial I/O Mode	260
10-8	Example of System Configuration in Serial Bus Interface (SBI) Mode	261
10-9	Timing in Three-Wire Serial I/O Mode	262
10-10	Setting of CSIM Register (three-wire serial I/O mode)	263
10-11	Timing in Three-Wire Serial I/O Mode (transmission)	264
10-12	Setting of CSIM Register (transmission enabled)	265
10-13	Timing in Three-Wire Serial I/O Mode (reception)	266
10-14	Setting of CSIM Register (reception enabled)	267

LIST OF FIGURES (5/6)

Figure No.	Title	Page
10-15	Timing in Three-Wire Serial I/O Mode (transmission/reception)	269
10-16	Setting of CSIM Register (transmission/reception enabled)	270
10-17	Example of System Configuration of Serial Bus Interface (SBI)	272
10-18	Setting of CSIM Register (SBI mode)	273
10-19	Bus Release Signal	274
10-20	Command Signal	274
10-21	Address	275
10-22	Command	275
10-23	Data	275
10-24	Acknowledge Signal	276
10-25	Busy Signal and Ready Signal	276
10-26	Format of Serial Bus Interface Control Register	281
10-27	Operations of RELT, CMDT, RELD, and CMDD	282
10-28	Operation of ACKT	282
10-29	Operation of ACKE	283
10-30	Operation of ACKD	284
10-31	Operation of BSYE	285
10-32	Setting of CSIM Register (transmission/reception enabled)	287
10-33	Address Transfer from Master Device to Slave Device	288
10-34	Command Transfer from Master Device to Slave Device	288
10-35	Data Transfer from Master Device to Slave Device	289
10-36	Data Transfer from Slave Device to Master Device	289
10-37	Example of System Configuration of Serial Bus Interface (SBI)	290
10-38	Setting of CSIM Register (wake-up function)	291
11-1	Block Diagram of PWM Unit	293
11-2	Format of PWM Control Register 0	295
11-3	Format of PWM Control Register 1	296
11-4	Operation of PWM Output Function (high-active setting)	297
12-1	Block Diagram of Watchdog Timer	299
12-2	Format of Watchdog Timer Mode Register	301
13-1	Format of Interrupt Control Registers	311
13-2	Format of Interrupt Mask Flag Register	313
13-3	Format of Interrupt Mode Control Register	315
13-4	Format of In-service Priority Register	316
13-5	Program Status Word (PSWL) Format	317
13-6	Non-Maskable Interrupt Request Acknowledgment Operation	319
13-7	Interrupt Acknowledgment Service Algorithm	323
13-8	Context Switching Operation by Generation of Interrupt Request	324
13-9	RETCS Instruction Format	325
13-10	Restoration Operation from Interrupt Using Context Switching Function by RETCS Instruction	325

LIST OF FIGURES (6/6)

Figure No.	Title	Page
13-11	Examples of Servicing an Interrupt Request Generated while Another Interrupt is being Serviced	327
13-12	Example of Servicing Interrupt Requests Generated Simultaneously	330
13-13	Differences in Level 3 Interrupt Acknowledgment Operation by Setting IMC Register	331
13-14	Context Switching Operation by Executing BRKCS Instruction	333
13-15	RETCSB Instruction Format	334
13-16	Restoration Operation from Software Interrupt Executed by BRKCS Instruction (Operation of RETCSB Instruction)	335
13-17	Differences between Vectored Interrupt and Macro Service Processing	336
13-18	Sequence Example of Macro Service Processing	339
13-19	Operation at Completion of Macro Service	341
13-20	Basic Structure of Macro Service Control Word	342
13-21	Format of Macro Service Control Word	343
14-1	Transition Diagram of Standby Modes	357
14-2	STBC Register Write Instruction	358
14-3	Format of Standby Control Register (STBC)	359
14-4	Format of Watchdog Timer Mode Register	365
14-5	Operation after Release of STOP Mode (1)	366
14-6	Operation after Release of STOP Mode (2)	367
14-7	Operation after Release of STOP Mode (3)	368
14-8	Operation after Release of STOP Mode (4)	369
14-9	Releasing STOP Mode by NMI Input (1)	370
14-10	Releasing STOP Mode by NMI Input (2)	371
15-1	Acceptance of RESET Signal	373
15-2	Reset Operation at Power-On	374
16-1	Memory Map in Expansion Mode (μ PD78368A)	379
16-2	Memory Map in Expansion Mode (μ PD78366A)	380
16-3	Memory Map in Expansion Mode (μ PD78363A)	381
16-4	Memory Map in Expansion Mode (μ PD78P368A)	382
16-5	Memory Map of μ PD78365A	383
16-6	Format of Memory Expansion Mode Register	385
16-7	Format of Programmable Wait Control Register	387
17-1	Flowchart of Procedure for Writing (page program mode)	392
17-2	Timing Chart of PROM Write and Verify Operation (page program mode)	393
17-3	Flowchart of Write Operation (byte program mode)	395
17-4	Timing Chart of PROM Write and Verify Operation (byte program mode)	396
17-5	PROM Read Timing Chart	397
19-1	Concept of Memory Access in Op-Code Fetch	423
19-2	Concept of Memory Access in Data Access	426

LIST OF TABLES (1/2)

Table No.	Title	Page
1-1	Differences among μ PD78366A Subseries Products	11
2-1	Port Pin Functions	13
2-2	Functions of Pins Other Than Port Pins	15
2-3	Functions of Pins Used in PROM Programming Mode	16
2-4	I/O Type of Each Pin and Recommended Processing	26
3-1	Vector Table Area	34
3-2	Operation in Word Access in Internal RAM Area	35
3-3	General-Purpose Register Configuration	48
3-4	Special Function Registers	51
5-1	Read Operation in Control Mode	70
5-2	Functions and Features of Ports	76
5-3	Operation of Port 4 and Port 5 (μ PD78363A, 78366A, 78368A)	80
5-4	Operation of Port 9 (μ PD78363A, 78366A, 78368A)	81
7-1	Configuration of RPU	94
7-2	Operation Modes of Timer 0 (TM0)	98
7-3	Timing of Transfer from BFCM03 to CM03	107
7-4	Operation Modes of Timer 0 (TM0)	110
7-5	List of UDC Mode Operations	183
7-6	Setting of TMC4 Register (during internal clock operation in UDC mode)	185
7-7	Up/Down Count Operation Modes	186
8-1	Example of Conversion Time Set by FR Bit	205
8-2	A/D Conversion Modes	206
8-3	Correspondence Between Analog Input and A/D Conversion Result Registers (ADCRs) (select mode, 1-buffer mode)	214
8-4	Correspondence Between Analog Input and A/D Conversion Result Registers (ADCRs) (select mode, 4-buffer mode)	215
8-5	Correspondence Between Analog Input Pins and A/D Conversion Result Registers (ADCRs) (scan mode)	216
8-6	Analog Input in Mixed Mode	218
8-7	Correspondence Between Analog Input and A/D Conversion Result Registers (ADCRs) (mixed mode: select processing in 1-buffer mode)	218
8-8	Correspondence Between Analog Input and A/D Conversion Result Registers (ADCRs) (in mixed mode: select processing in 4-buffer mode)	220
9-1	Typical Baud Rate Settings (asynchronous serial interface)	239
9-2	Causes of Reception Errors	246
10-1	Signals in SBI Mode	278

LIST OF TABLES (2/2)

Table No.	Title	Page
11-1	PWM Signal Repetition Frequencies	294
13-1	Interrupt Request Processing Modes	303
13-2	Interrupt Sources	304
13-3	Control Registers	307
13-4	Interrupt Control Register Flags for Interrupt Request Signals	308
13-5	Multiplexed Interrupt Service	326
13-6	Interrupts for which Macro Service Is Available	337
13-7	Classification of Macro Service Modes	344
13-8	Counter Mode Operation Specification	345
13-9	Specification of Block Transfer Mode Operation	346
13-10	Specification of Block Transfer Mode (with a memory pointer) Operation	348
13-11	List of Instructions that Sometimes do not Acknowledge Interrupt Requests	355
14-1	Operation States in HALT Mode	360
14-2	Acceptance of Interrupts Generated during Interrupt Servicing	361
14-3	Operations after HALT Mode Is Released by an Interrupt Request	362
14-4	Releasing HALT Mode by a Macro Service Request	362
14-5	Operation States in STOP Mode	363
14-6	Release of STOP Mode and Operation after Release	370
15-1	Hardware Statuses after Reset	375
16-1	Assigning Functions to Pins (μ PD78366A)	377
16-2	Operation of Port 5 (expansion mode)	377
16-3	Setting Pin Function (μ PD78365A)	383
17-1	Pin Functions in Programming Mode	389
17-2	Operating Modes for PROM Programming	390
18-1	Operand Identifier and Description	400
18-2	Absolute Names and Their Corresponding Function Names of 8-bit Register	401
18-3	Absolute Names and Their Corresponding Function Names of 16-bit Register	401
18-4	Notational Symbols in Flag Operation Field	403
19-1	Number of Clocks Required for Op-Code Fetch	422
19-2	Bus Control Signals to Be Output during Op-Code Fetch	424
19-3	Number of Clocks Required for Data Access	425
19-4	Bus Control Signals to Be Output during Data Access	427
19-5	Number of saddr Accesses by Instruction	430

CHAPTER 1 GENERAL

The μ PD78366A subseries is a series of NEC's 78K/III Series 16-/8-bit single-chip microcontrollers equipped with a high-speed, high-performance 16-bit CPU.

- ★ The μ PD78366A subseries provides the eight products of μ PD78361A, 78362A, 78P364A, 78363A, 78365A, 78366A, 78368A, and 78P368A. However, the manual describes the μ PD78363A, 78365A, 78366A, 78368A, and 78P368A.

The μ PD78366A has a PWM output function with a resolution higher than the existing μ PD78328, so it can provide substantially enhanced performance when used to control an inverter. In addition, the μ PD78366A subseries microcontrollers are provided with sum-of-products instructions and can be used for a wide range of applications, as high-speed, high-performance CPUs.

- ★ The μ PD78368A is an extended internal ROM model of the μ PD78366A.

The μ PD78365A is a ROM-less model of the μ PD78366A.

The μ PD78P368A is a model in the μ PD78366A series with expanded memory capacity. It has a one-time PROM or EPROM instead of the internal mask ROM of the μ PD78366A. The one-time PROM can be written only once and is suitable for small-scale production of a small quantity of application systems, or early start of production. Programs can be written to, erased from, or rewritten to the EPROM, making it ideal for system evaluation.

1.1 Features

- Internal 16-bit architecture, external 8-bit data bus
- High-speed processing backed up by pipeline control and high-speed operation clock
Minimum instruction execution time: 125 ns (internal clock: 16 MHz, external clock: 8 MHz)
- PLL control circuit: External 8 MHz → Internal 16 MHz
- 115 instructions ideal for control applications (upward compatible with the μ PD78328)
 - 16-bit arithmetic operation instructions
 - Multiplication/division instructions (16 bits \times 16 bits (with sign/without sign), 32 bits \div 16 bits)
 - Sum-of-products instructions (16 bits \times 16 bits + 32 bits)
 - Relative operation instructions
 - String instructions
 - Bit manipulation instructions, etc.
- Real-time pulse unit suitable for inverter control
 - Two timer output modes selectable (set/reset output/buffer output)
 - Can output six phases of PWM signals
 - Dead time timer
 - Output driver off function when an error occurs outside of the microcomputer
 - Up/down counter function
 - 16-bit resolution PWM signal output: 1 channel
- High-speed 10-bit resolution A/D converter: 8 channels
- 8-/9-/10-/12-bit resolution-variable PWM signal output function: 2 channels
- Two channels of independent serial interfaces
 - UART (with pin select function)
 - Clocked serial interface/SBI
- High-speed interrupt controller
 - 4 levels of priority selectable
 - Three interrupt service modes selectable
(vectored interrupt, macro service, context switching)
- Large current ports
 - Port 0 (P00-P07): $I_{OL} = 10$ mA (up to four pins can be turned on simultaneously)
 - Port 1 (P10-P17): $I_{OL} = 10$ mA (up to four pins can be turned on simultaneously)
 - Port 8 (P80-P85): $I_{OL} = 15$ mA (up to three pins can be turned on simultaneously)
- Internal memory: ROM 48K bytes (μ PD78368A) 32K bytes (μ PD78366A) 24K bytes (μ PD78363A) None (μ PD78365A)
 - PROM 48K bytes (μ PD78P368A)
 - RAM 2K bytes (μ PD78365A, 78366A, 78368A, 78P368A)
 - 768 bytes (μ PD78363A)

1.2 Application Fields

- Inverter air conditioner
- FA equipment such as robots and automatic machine tools

1.3 Ordering Information

★

Part Number	Package	Internal ROM
μ PD78363AGF-xxx-3B9	80-pin plastic QFP (14 × 20 mm)	Mask ROM
μ PD78365AGF-3B9	80-pin plastic QFP (14 × 20 mm)	None
μ PD78366AGF-xxx-3B9	80-pin plastic QFP (14 × 20 mm)	Mask ROM
μ PD78368AGF-xxx-3B9	80-pin plastic QFP (14 × 20 mm)	Mask ROM
μ PD78P368AGF-3B9	80-pin plastic QFP (14 × 20 mm)	One-time PROM
μ PD78P368AKL-S	80-pin ceramic WQFN	EPROM

Remark xxx indicates a ROM code suffix.

1.4 Pin Configuration (Top View)

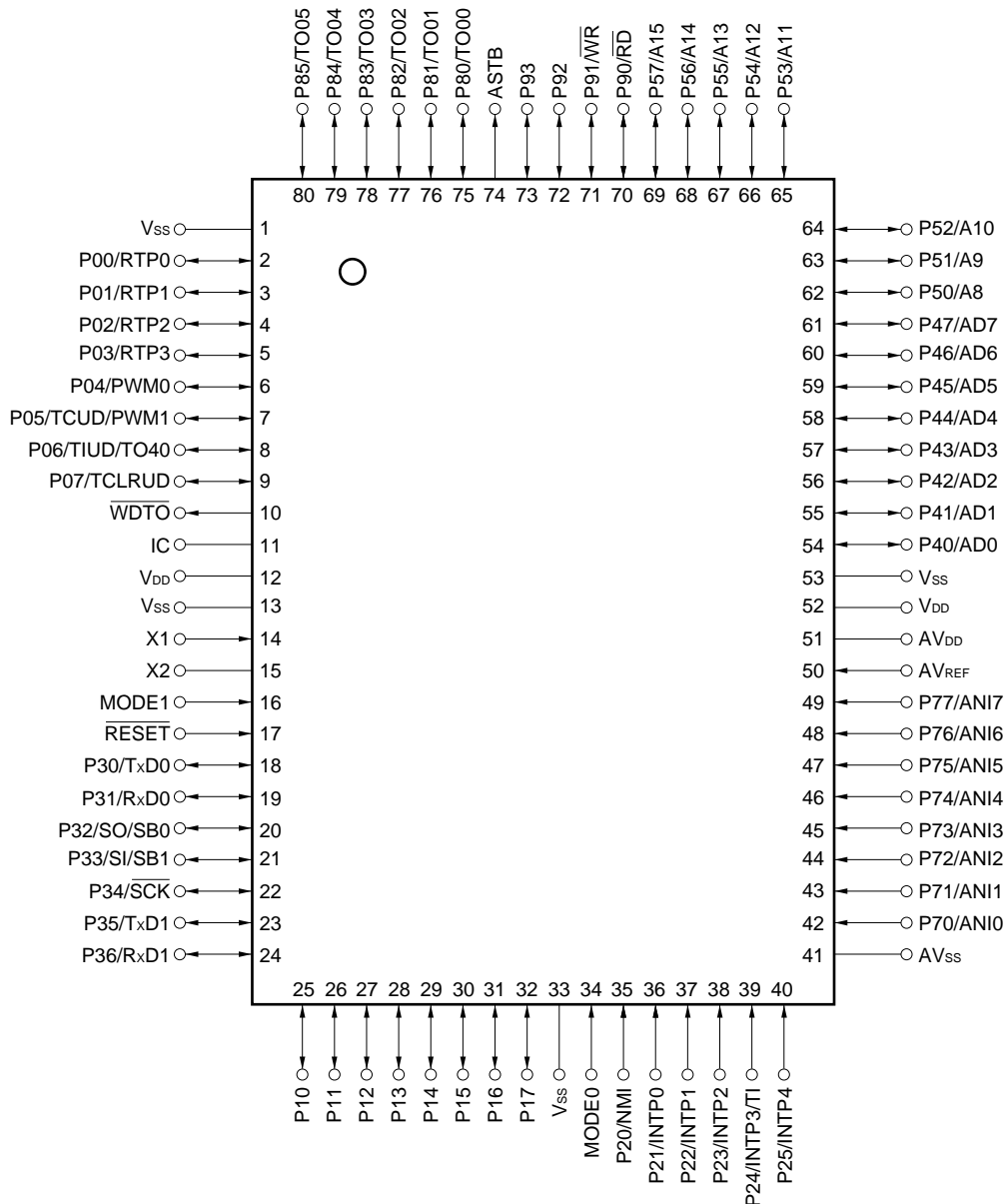
1.4.1 Normal operation mode

• 80-pin plastic QFP (14 × 20 mm)

★ μ PD78363AGF-xxx-3B9, 78365AGF-3B9, 78366AGF-xxx-3B9, 78368AGF-xxx-3B9, 78P368AGF-3B9

• 80-pin ceramic WQFN

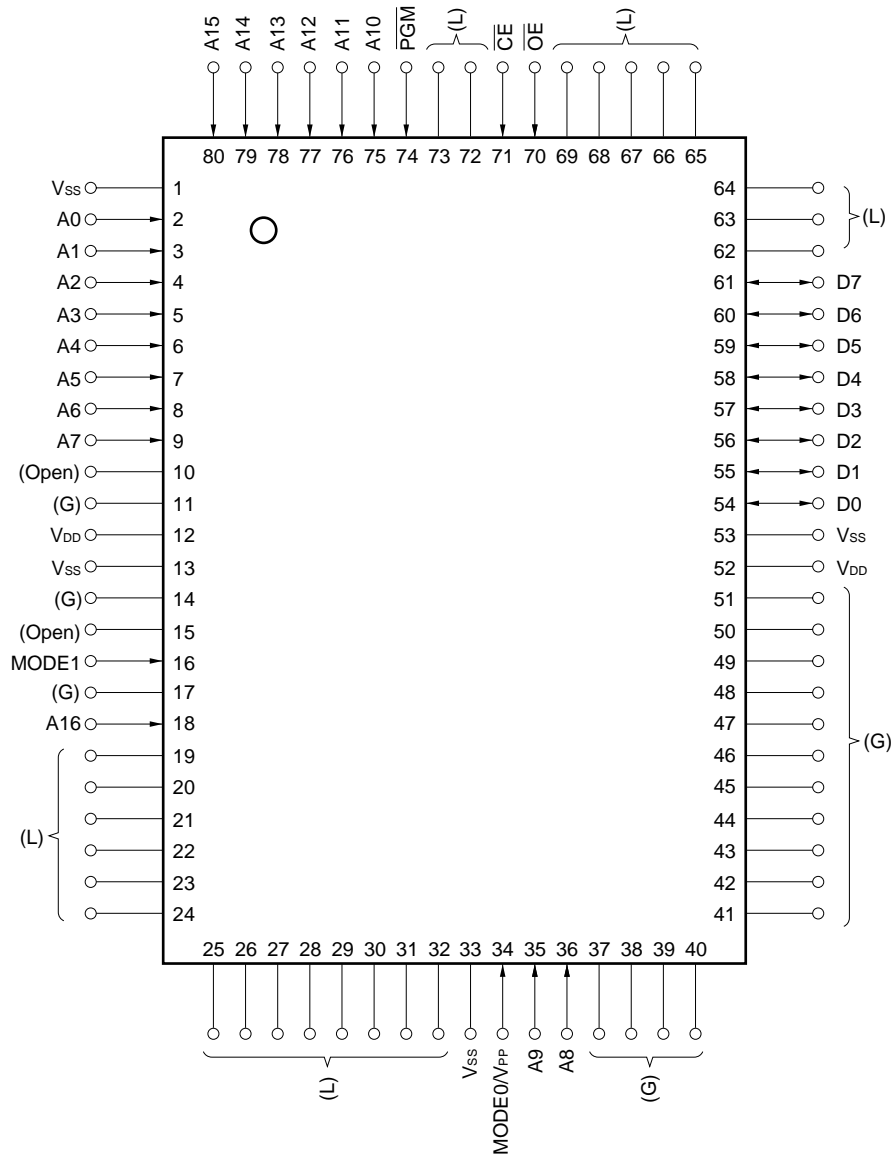
μ PD78P368AKL-S



Caution Directly connect the IC pin to the V_{SS} pin.

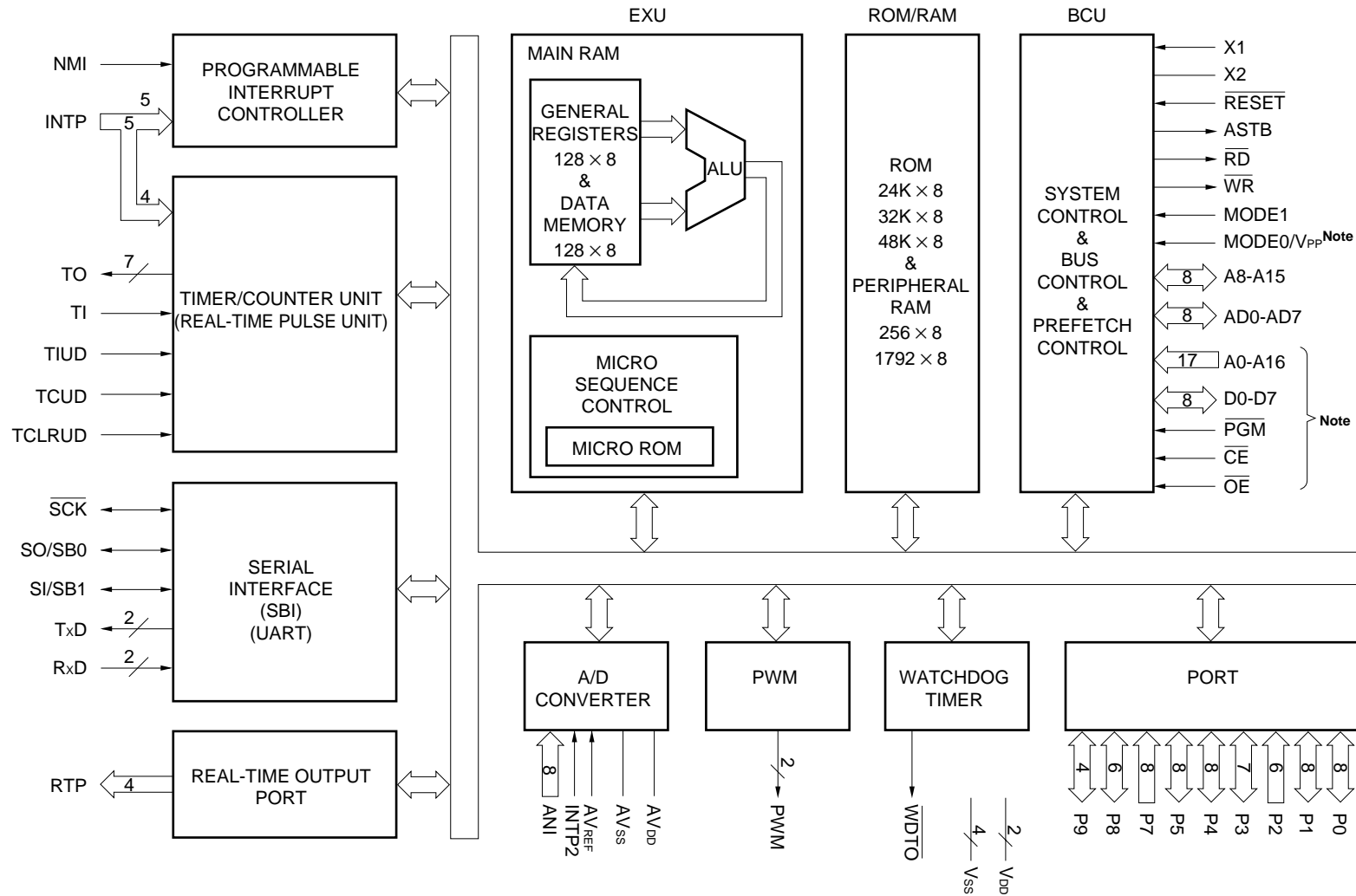
Remark xxx indicates a ROM code suffix.

A8-A15	: Address Bus	PWM0, PWM1	: Pulse Width Modulation Output
AD0-AD7	: Address/Data Bus	<u>RD</u>	: Read Strobe
ANI0-ANI7	: Analog Input	<u>RESET</u>	: Reset
ASTB	: Address Strobe	RTP0-RTP3	: Real-time Port
AV _{DD}	: Analog V _{DD}	RxD0, RxD1	: Receive Data
AV _{REF}	: Analog Reference Voltage	<u>SB0</u> , <u>SB1</u>	: Serial Bus
AV _{SS}	: Analog V _{SS}	<u>SCK</u>	: Serial Clock
IC	: Internally Connected	<u>SI</u>	: Serial Input
INTP0-INTP4	: Interrupt from Peripherals	<u>SO</u>	: Serial Output
MODE0, MODE1	: Mode	TCLRUD	: Timer Clear Up Down Counter
NMI	: Non-maskable Interrupt	TCUD	: Timer Control Up Down Counter
P00-P07	: Port0	TI	: Timer Input
P10-P17	: Port1	TIUD	: Timer Input Up Down Counter
P20-P25	: Port2	TO00-TO05, TO40:	Timer Output
P30-P36	: Port3	TxD0, TxD1	: Transmit Data
P40-P47	: Port4	V _{DD}	: Power Supply
P50-P57	: Port5	V _{SS}	: Ground
P70-P77	: Port7	<u>X1</u> , <u>X2</u>	: Crystal
P80-P85	: Port8	<u>WDTO</u>	: Watchdog Timer Output
P90-P93	: Port9	<u>WR</u>	: Write Strobe

1.4.2 PROM programming mode (μ PD78P368A only: MODE0/V_{PP} = H, MODE1 = L)• **80-pin plastic QFP (14 × 20 mm)** μ PD78P368AGF-3B9• **80-pin ceramic WQFN** μ PD78P368AKL-S**Caution (): Processing of pins not used in PROM programming mode.****L** : Individually connect these pins to V_{ss} via resistor.**G** : Connect these pins to V_{ss}.**Open** : Connect nothing to these pins.

A0-A16	: Address Bus	$\overline{\text{PGM}}$: Programming Mode
$\overline{\text{CE}}$: Chip Enable	V_{DD}	: Power Supply
D0-D7	: Data Bus	V_{PP}	: Programming Power Supply
MODE0, MODE1	: Programming Mode Set	V_{SS}	: Ground
$\overline{\text{OE}}$: Output Enable		

1.5 Block Diagram



Note In PROM programming mode of μPD78P368A.

Remark Internal ROM and RAM capacities depend on the product.

1.6 Functional Outline

(1/2)

Part No.		μPD78363A	μPD78365A	μPD78366A	μPD78368A	μPD78P368A
Item						
Minimum instruction		125 ns (internal clock: 16 MHz, external clock execution time 8 MHz)				
Internal memory	ROM	24K bytes	—	32K bytes	48K bytes	—
	PROM	—	—	—	—	48K bytes
	RAM	768 bytes	2K bytes			
Memory space		64K bytes (externally expandable)				
General-purpose register		8 bits × 16 × 8 banks				
Basic instructions		115				
Instruction set		<ul style="list-style-type: none">• 16-bit transfer/operation• Multiplication/division (16 bits × 16 bits, 32 bits ÷ 16 bits)• Bit manipulation• String• Sum-of-products (16 bits × 16 bits + 32 bits)• Relative operation				
I/O line	Input	14 (of which 8 are multiplexed with analog input)				
	I/O	49	31	49		
Real-time pulse unit		<ul style="list-style-type: none">• 16-bit timer × 1 10-bit dead time timer × 3 16-bit compare register × 4 Two output modes selectable Mode 0: set/reset output (6 channels) Mode 1: buffer output (6 channels) <ul style="list-style-type: none">• 16-bit timer × 1 16-bit compare register × 1 <ul style="list-style-type: none">• 16-bit timer × 1 16-bit capture register × 1 16-bit capture/compare register × 1 <ul style="list-style-type: none">• 16-bit timer × 1 16-bit capture register × 2 16-bit capture/compare register × 1 <ul style="list-style-type: none">• 16-bit timer × 1 16-bit compare register × 2 16-bit resolution PWM output: 1 channel				
Real-time output port		4 (4-bit unit buffer output)				
PWM unit		8-/9-/10-/12-bit resolution variable PWM output: 2 channels				
A/D converter		10-bit resolution: 8 channels				

(2/2)

Item \ Part No.	μ PD78363A	μ PD78365A	μ PD78366A	μ PD78368A	μ PD78P368A
Serial interface	With dedicated baud rate generator UART (with pin select function): 1 channel Clocked serial interface/SBI: 1 channel				
Interrupt function	<ul style="list-style-type: none"> • External: 6, internal 14 (of which 2 are multiplexed with external sources) • 4 priority levels selectable via software • 3 interrupt service modes selectable (vectored interrupt, macro service, context switching) 				
Package	<ul style="list-style-type: none"> • 80-pin plastic QFP (14 × 20 mm) • 80-pin ceramic WQFN (μPD78P368A only) 				
Others	<ul style="list-style-type: none"> • Watchdog timer • Standby functions (HALT mode and STOP mode) • PLL control circuit 				

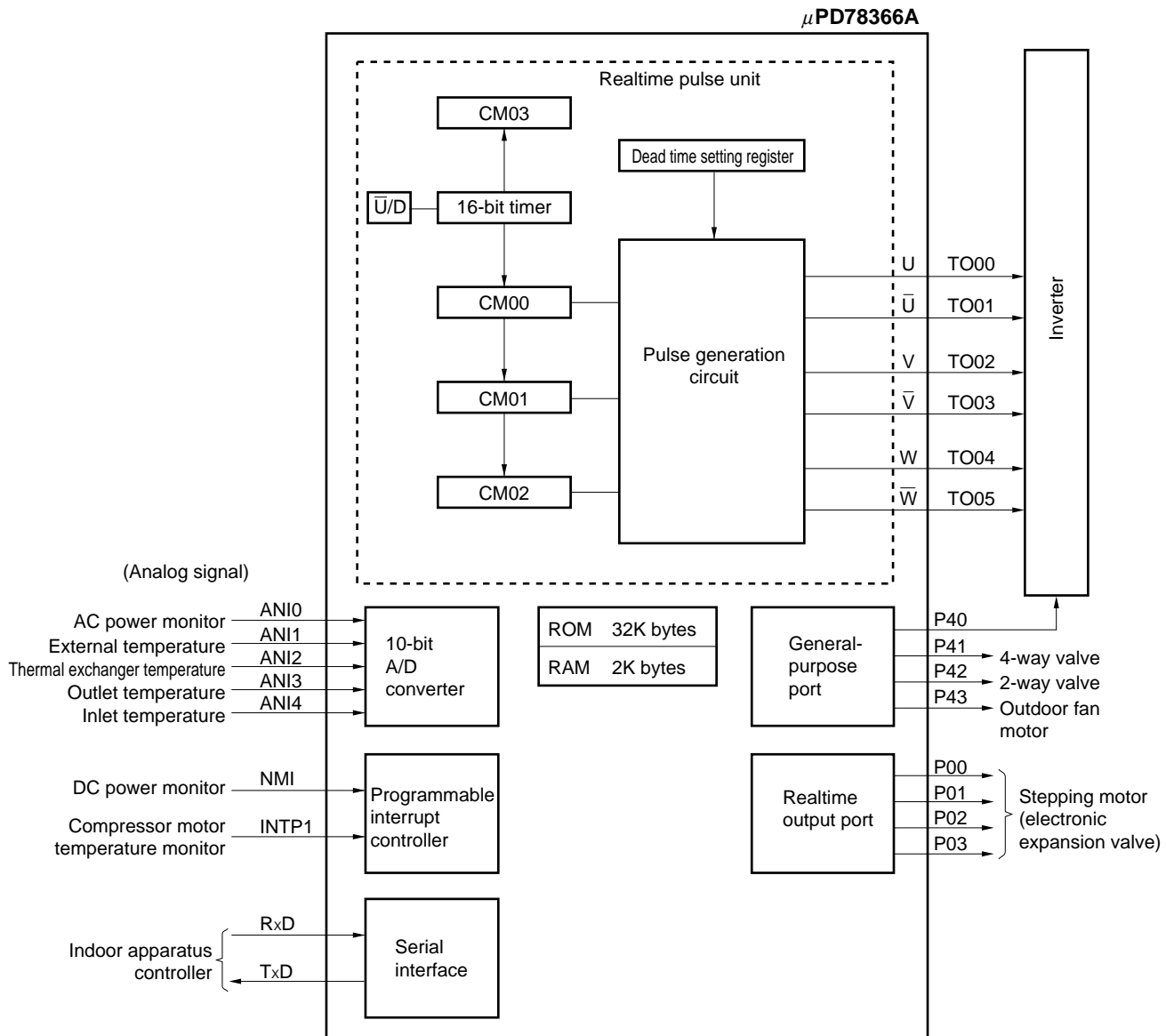
1.7 Differences among μ PD78366A Subseries Products

Table 1-1. Differences among μ PD78366A Subseries Products

Product Name		μPD78361A	μPD78362A	μPD78P364A	μPD78363A	μPD78365A	μPD78366A	μPD78368A	μPD78P368A
Parameter									
Internal memory	ROM	32K bytes	24K bytes	–	24K bytes	–	32K bytes	48K bytes	–
	PROM	–	–	48K bytes	–	–	–	–	48K bytes
	RAM	2K bytes	768 bytes	2K bytes	768 bytes	2K bytes			
Input/output lines	Input	14 (8 lines also serve as analog input)							
	I/O	38			49	31	49		
UART pin select function		None			Available				
$\overline{\text{WDTO}}$ pin		None			Available				
External device expansion function		None			Available				
ROM-less mode		None			Available	ROM-less product	Available		None
MODE setting		Be sure to set the following: MODE = L	• Normal operation mode MODE = L • Programming mode MODE = H		• Normal operation mode MODE0, 1 = LL • ROM-less mode MODE0, 1 = HH	Be sure to set the following: MODE0, 1 = HH	• Normal operation mode MODE0, 1 = LL • ROM-less mode MODE0, 1 = HH		• Normal operation mode MODE0, 1 = LL • Program-ming mode MODE0, 1 = HL
Package	With no window	64-pin plastic shrink DIP (750 mil)			80-pin plastic QFP (14 × 20 mm)				
	With window	–			–				

1.8 Application Example

To control outdoor apparatus of inverter air conditioner



CHAPTER 2 PIN FUNCTIONS

2.1 Pin Function List

2.1.1 Normal operation mode

(1) Port pins

Table 2-1. Port Pin Functions (1/2)

Pin	I/O	Function	Shared with:
P00-P03	I/O	Port 0.	RTP0-RTP3
P04		8-bit I/O port.	PWM0
P05		Can be set in input or output mode in 1-bit units.	PWM1/TCUD
P06			TO40/TIUD
P07			TCLRUD
P10-P17	I/O	Port 1. 8-bit I/O port. Can be set in input or output mode in 1-bit units.	—
P20	Input	Port 2.	NMI
P21		6-bit input port.	INTP0
P22			INTP1
P23			INTP2
P24			INTP3/TI
P25			INTP4
P30	I/O	Port 3.	TxD0
P31		7-bit I/O port.	RxD0
P32		Can be set in input or output mode in 1-bit units.	SO/SB0
P33			SI/SB1
P34			SCK
P35			TxD1
P36			RxD1
P40-P47 ^{Note}	I/O	Port 4. 8-bit I/O port. Can be set in input or output mode in 8-bit units.	AD0-AD7
P50-P57 ^{Note}	I/O	Port 5. 8-bit I/O port. Can be set in input or output mode in 1-bit units.	A8-A15

Note On the μ PD78365A, these pins do not operate as ports.

Table 2-1. Port Pin Functions (2/2)

Pin	I/O	Function	Shared with:
P70-P77	Input	Port 7. 8-bit input port.	AN10-AN17
P80-P85	I/O	Port 8. 6-bit I/O port. Can be set in input or output mode in 1-bit units.	TO00-TO05
P90 ^{Note}	I/O	Port 9. 4-bit I/O port. Can be set in input or output mode in I/O 1-bit units.	\overline{RD}
P91 ^{Note}			\overline{WR}
P92			—
P93			—

Note On the μ PD78365A, these pins do not operate as ports.

(2) Pins other than port pins

Table 2-2. Functions of Pins Other Than Port Pins (1/2)

Pin	I/O	Function	Shared with:
RTP0-RTP3	Output	Real-time output port that outputs pulse in synchronization with trigger signal from real-time pulse unit.	P00-P03
NMI	Input	Non-maskable interrupt request input	P20
INTP0		External interrupt request input	P21
INTP1			P22
INTP2			P23
INTP3			P24/TI
INTP4			P25
TI	Input	External count clock input to timer 1	P24/INTP3
TCUD		Count operation select control signal input to up/down counter (timer 4)	P05/PWM1
TIUD		External count clock input to up/down counter (timer 4)	P06/TO40
TCLRUD		Clear signal input to up/down counter (timer 4)	P07
TO00-TO05	Output	Pulse output from real-time pulse unit	P80-P85
TO40			P06/TIUD
ANI0-ANI7	Input	Analog input to A/D converter	P70-P77
TxD0	Output	Serial data output of asynchronous serial interface	P30
TxD1			P35
RxD0	Input	Serial data input of asynchronous serial interface	P31
RxD1			P36
SCK	I/O	Serial clock I/O of clocked serial interface	P34
SI	Input	Serial data input of clocked serial interface in three-wire mode	P33/SB1
SO	Output	Serial data output of clocked serial interface in three-wire mode	P32/SB0
SB0	I/O	Serial data I/O of clocked serial interface in SBI mode	P32/SO
SB1			P33/SI
PWM0	Output	PWM signal output	P04
PWM1			P05/TCUD
WDTO	Output	Signal output indicating overflow of watchdog timer (occurrence of non-maskable interrupt)	—
AD0-AD7	I/O	Multiplexed address/data bus used when external memory is connected	P40-P47
A8-A15		Address bus used when external memory is connected	P50-P57

Table 2-2. Functions of Pins Other Than Port Pins (2/2)

Pin	I/O	Function	Shared with:
ASTB	Output	Timing signal that externally latches address information output from AD0-AD7 and A8-A15 pins to access external memory	—
$\overline{\text{RD}}$		Read strobe signal to external memory	P90
$\overline{\text{WR}}$		Write strobe signal to external memory	P91
MODE0	Input	Control signal input to set operation mode. With $\mu\text{PD78363A}$, 78366A, 78368A, and 78P368A, MODE0 and MODE1 are V_{SS} . With $\mu\text{PD78365A}$, MODE0 and MODE1 are always connected to V_{DD} .	—
MODE1			—
$\overline{\text{RESET}}$	Input	System reset input	—
X1	Input	System clock oscillation crystal connecting pins. Input external clock to X1 pin. Leave X2 open.	—
X2	—		—
AV_{REF}	Input	A/D converter reference voltage input	—
AV_{DD}	—	A/D converter analog power supply	—
AV_{SS}	—	A/D converter GND	—
V_{DD}	—	Positive power supply	—
V_{SS}	—	GND	—
IC	—	Internally connected. Connect this pin directly to V_{SS}	—

2.1.2 PROM programming mode ($\mu\text{PD78P368A}$ only: MODE0/ V_{PP} = H, MODE1 = L)

Table 2-3. Functions of Pins Used in PROM Programming Mode

Pin	I/O	Function
MODE0/ V_{PP}	Input	PROM programming mode setting/writing power
MODE1		PROM programming mode setting
A0-A16		Address bus
D0-D7	I/O	Data bus
$\overline{\text{PGM}}$	Input	Program input
$\overline{\text{CE}}$		PROM enable input
$\overline{\text{OE}}$		Read strobe to PROM
V_{DD}	—	Positive power supply
V_{SS}		GND

2.2 Pin Functions

2.2.1 Normal operation mode

(1) P00-P07 (Port 0) ... three-state I/O

These pins form an 8-bit I/O port. They serve as not only a general-purpose I/O port, but also a real-time output port, control signal input pins of the real-time pulse unit, and PWM signal output pins.

Port 0 can be set in the following operation modes in 1-bit units by the port 0 mode control register (PMC0) (refer to **5.2 Port Functions**).

(a) Port mode

In this mode, port 0 functions as an 8-bit general-purpose I/O port, and can be set in the input or output mode in 1-bit units by the port 0 mode register (PM0).

(b) Control mode

In this mode, each bit of port 0 serve as a control signal input or output pin, as follows:

(i) RTP0-RTP3

These pins function as a real-time output port.

(ii) PWM0 and PWM1

These pins output the PWM signal.

(iii) TO40

This pin serves as the timer output pin of timer 4 of the real-time pulse unit.

(iv) TIUD

This pin inputs an external count clock to timer 4 of the real-time pulse unit.

(v) TCUD

This pin inputs the count operation select control signal to timer 4 of the real-time pulse unit.

(vi) TCLRUD

This pin inputs a clear signal to timer 4 of the real-time pulse unit.

Caution Each pin is set in the input port mode when the $\overline{\text{RESET}}$ signal has been input (output high impedance). At this time, the contents of the output latch become undefined.

(2) P10-P17 (Port 1) ... three-state I/O

These pins form an 8-bit I/O port that functions as a general-purpose I/O port.

This port can be set in the input or output mode in 1-bit units by the port 1 mode register (PM1).

Caution Each pin is set in the input port mode when the $\overline{\text{RESET}}$ signal has been input (output high impedance). At this time, the contents of the output latch become undefined.

(3) P20-P25 (Port 2) ... input

These pins form a 6-bit input port. They also function as external interrupt signal and external count clock input pins.

(a) Port mode

Port 2 is fixed in the control mode. However, the status of each pin can be read by executing a read instruction to port 2.

(b) Control mode

The functions of P20-P25 as control signal pins are as follows:

(i) NMI

Inputs an edge-detected external non-maskable interrupt request.

(ii) INTP0-INTP4

Input an edge-detected, external interrupt request.

(iii) TI

Inputs an external count clock to timer 1 of the real-time pulse unit.

(4) P30-P36 (Port 3) ... three-state I/O

These pins form a 7-bit I/O port. They also function as the I/O pins of the serial interface.

Port 3 can be set in the following operation modes in 1-bit units by the port 3 mode control register (PMC3) (refer to **5.2 Port Functions**):

(a) Port mode

In this mode, port 3 functions as a 7-bit general-purpose I/O port, which can be set in the input or output mode in 1-bit units by the port 3 mode register (PM3).

(b) Control mode

In this mode, each bit of port 3 functions as a control signal pin, as follows:

(i) Rx/D0, Rx/D1, Tx/D0, Tx/D1

Serial data I/O pins of the asynchronous serial interface (UART).

(ii) SO/SB0, SI/SB1

Serial data I/O pins of the clocked serial interface.

(iii) $\overline{\text{SCK}}$

Serial clock I/O pins of the clocked serial interface.

Caution Each pin is set in the input port mode when the $\overline{\text{RESET}}$ signal has been input (output high impedance). At this time, the contents of the output latch become undefined.

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

The functions of these pins differ depending on whether it is the μ PD78363A, 78365A, 78366A, 78368A, or 78P368A.

(a) μ PD78365A (MODE0, MODE1 = HH)

Port 4 of the μ PD78365A is fixed in the external memory expansion mode, and always functions as an 8-bit address/data bus (AD0-AD7). It does not have a port function.

(b) μ PD78363A, 78366A, 78368A, and 78P368A (MODE0, MODE1 = LL)

Port 4 of the μ PD78363A, 78366A, 78368A, and 78P368A serves as an 8-bit I/O port. It also functions as an address/data bus.

This port can be set in the following operation modes by the memory expansion mode register (MM) (refer to **CHAPTER 16 BUS INTERFACE FUNCTION**):

(i) Port mode

In this mode, port 4 functions as an 8-bit general-purpose I/O port, which can be set in the input or output mode in 8-bit units by the memory expansion mode register (MM).

(ii) External memory expansion mode

In this mode, port 4 functions as an address/data bus (AD0-AD7) through which the external memory is to be accessed. In this case, the bus is not influenced by the value of the port 4 register.

Caution Each pin is set in the input port mode when the $\overline{\text{RESET}}$ signal has been input (output high impedance) in either of port mode or external memory expansion mode. At this time, the contents of the output latch become undefined.

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

The functions of these pins differ depending on whether it is the μ PD78363A, 78365A, 78368A, or 78P368A.

(a) μ PD78365A (MODE0, MODE1 = HH)

Port 5 of the μ PD78365A is fixed in the external memory expansion mode, and always functions as an 8-bit address bus (A8-A15). It does not have a port function.

(b) μ PD78363A, 78366A, 78368A, and 78P368A (MODE0, MODE1 = LL)

Port 5 of the μ PD78363A, 78366A, and 78P368A serves as an 8-bit I/O port. It also functions as an address bus.

This port can be set in the following operation modes by the memory expansion mode register (MM) (refer to **CHAPTER 16 BUS INTERFACE FUNCTION**):

(i) Port mode

In this mode, port 5 functions as a 2-/4-/8-bit general-purpose I/O port, which can be set in the input or output mode in 1-bit units by the port 5 mode register (PM5).

(ii) External memory expansion mode

In this mode, port 5 functions as an address bus (A8-A15) (4-, 6-, or 8-bit wide) through which the external memory is to be accessed.

Depending on the size of the external memory to be connected, the width of the port consisting of P50-P57 (i.e., the number of port bits) can be specified, and the excess pins, if any, can be used as general-purpose I/O port pins.

Caution Each pin is set in the input port mode when the $\overline{\text{RESET}}$ signal has been input (output high impedance) in either of port mode or external memory expansion mode. At this time, the contents of the output latch become undefined.

(7) P70-P77 (Port 7) ... input

These pins form an 8-bit input port. They also serve as the analog signal input pins of the A/D converter.

(a) Port mode

Port 7 is fixed in the control mode. The status of each pin can be read by executing a read instruction to port 7.

(b) Control mode

In this mode, port 7 functions as the analog signal input pins of the A/D converter (ANI0-ANI7).

(8) P80-P85 (Port 8) ... three-state I/O

These pins form a 6-bit I/O port. They also serve as the timer output pins of the real-time pulse unit.

Port 8 can be set in the following operation modes in 1-bit units by the port 8 mode control register (PMC8) (refer to **5.2 Port Functions**).

(a) Port mode

In this mode, port 8 serves as an 8-bit general-purpose I/O port, which can be set in the input or output mode in 1-bit units by the port 8 mode register (PM8).

(b) Control mode

In this mode, the port 8 pins function as the timer output pins (TO00-TO05) of timer 0 of the real-time pulse unit.

Caution Each pin is set in the input port mode when the $\overline{\text{RESET}}$ signal has been input (output high impedance). At this time, the contents of the output latch become undefined.

(9) P90-P93 (Port 9) ... three-state I/O

The functions of these pins differ depending on whether it is the $\mu\text{PD78363A}$, 78365A, 78366A, 78368A, or 78P368A.

(a) $\mu\text{PD78365A}$ (MODE0, MODE1 = HH)

P90 and P91 of the $\mu\text{PD78365A}$ are fixed in the external memory expansion mode, and always function as $\overline{\text{RD}}$ and $\overline{\text{WR}}$ pins. They do not have a port function.

P92 and P93 function as general-purpose I/O port pins, which can be set in the input or output mode by the port 9 mode register (PM9).

(b) $\mu\text{PD78363A}$, 78366A, 78368A, and 78P368A (MODE0, MODE1 = LL)

Port 9 of the $\mu\text{PD78363A}$, 78366A, 78368A, and 78P368A serves as a 4-bit I/O port. It also outputs control signals to the external memory.

Port 9 can be set in the following operation modes by the memory expansion mode register (MM) and programmable wait control register (PWC) (refer to **CHAPTER 16 BUS INTERFACE FUNCTION**).

(i) Port mode

In this mode, port 9 functions as a 4-bit general-purpose I/O port, which can be set in the input or output mode in 1-bit units by the port 9 mode register (PM9).

(ii) External memory expansion mode (P90 and P91 only)

Only P90 and P91 can be set in this mode to output the following control signals to the external memory:

<1> \overline{RD}

Read signal output to the external memory.

<2> \overline{WR}

Write signal output to the external memory.

Caution Each pin is set in the input port mode when the \overline{RESET} signal has been input (output high impedance) in either of port mode or external memory expansion mode. At this time, the contents of the output latch become undefined.

(10) ASTB (Address Strobe) ... output

This pin outputs a timing signal with which an external memory is to be accessed. It is used to latch an address output from the P40/AD0-P47/AD7 pins externally.

(11) MODE0 and MODE1 (Mode) ... input

These pins input control signals that specify an operation mode. The setting of these pins differs as shown in the table below for the μ PD78363A, 78365A, 78366A, 78368A, and 78P368A. Note that the levels of the MODE0 and MODE1 pins must not be changed during operation.

μ PD78365A	μ PD78363A, 78366A, 78368A	μ PD78P368A
Be sure to set as follows: MODE0, 1 = HH	<ul style="list-style-type: none"> Normal operation mode MODE0, 1 = LL ROM-less mode MODE0, 1 = HH 	<ul style="list-style-type: none"> Normal operation mode MODE0, 1 = LL Programming mode MODE0, 1 = HL

- Cautions**
1. Be sure to connect the MODE0 and MODE1 pins directly to V_{DD} or V_{SS} .
 2. Do not set the MODE0 and MODE1 pins other than above.
 3. The μ PD78P368A cannot be set in the ROM-less mode.

(12) \overline{RESET} (Reset) ... input

This pin inputs a low-active system reset signal.

(13) X1 and X2 (Crystal)

Connect a crystal resonator for system clock oscillation across these pins. To supply an external clock, input the clock to the X1 pin and leave the X2 pin open.

(14) $\overline{\text{WDTO}}$ (Watchdog Timer Output) ... output

This pin outputs a signal that indicates that the watchdog timer has generated a non-maskable interrupt.

(15) AV_{REF} (Analog Reference Voltage) ... input

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

(16) AV_{DD} (Analog V_{DD})

This pin is the power supply pin of the A/D converter.

(17) AV_{SS} (Analog V_{SS})

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

(18) V_{DD} (Power Supply)

Positive power supply pin

(19) V_{SS} (Ground)

Ground potential pin

(20) IC (Internally Connected)

This pin is internally connected and should be directly connected to V_{SS} .

2.2.2 PROM programming mode (μ PD78P368A only)**(1) MODE0 and MODE1 ... input**

These pins of the μ PD78P368A set the PROM programming mode, which is set when MODE0 = H and MODE1 = L.

(2) A0-A16 (Address Bus) ... input

These pins form an address bus which selects an address of the internal PROM (0000H-BFFFH).

(3) D0-D7 (Data Bus) ... I/O

These pins form a data bus through which the program in the internal PROM is written or read.

(4) $\overline{\text{PGM}}$ (Programming Mode) ... input

This pin inputs the operation mode control signal of the internal PROM.

When this signal is active, the internal PROM can be written.

When this signal is inactive, the internal PROM can be read.

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

This pin inputs an internal PROM enable signal.

When $\overline{\text{CE}} = \overline{\text{OE}} = \text{H}$, $\overline{\text{PGM}} = \text{L}$, one page (4 bytes) of a program can be written in 1-byte units.

When $\overline{\text{CE}} = \text{L}$, $\overline{\text{OE}} = \text{H}$, $\overline{\text{PGM}} = \text{L}$, 1 byte of program can be written at a time.

If $\overline{\text{OE}}$ is made low when $\overline{\text{CE}} = \text{L}$, the contents of the PROM can be read.

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

This pin inputs a read strobe signal to the internal PROM.

If this signal is made active when $\overline{\text{CE}} = \text{L}$, the contents of PROM addressed by A0-A16 can be read to D0-D7 in 1-byte units.

(7) V_{PP} (Programming Power Supply)

Power supply to write a program.

When $\overline{\text{OE}} = \text{H}$ and $\overline{\text{CE}} = \text{L}$ while $V_{\text{PP}} = 12.5 \text{ V}$, the program on D0-D7 can be written to the internal PROM addressed by A0-A16.

(8) V_{DD} (Power Supply)

Positive power supply pin

(9) V_{SS} (Ground)

Ground potential pin

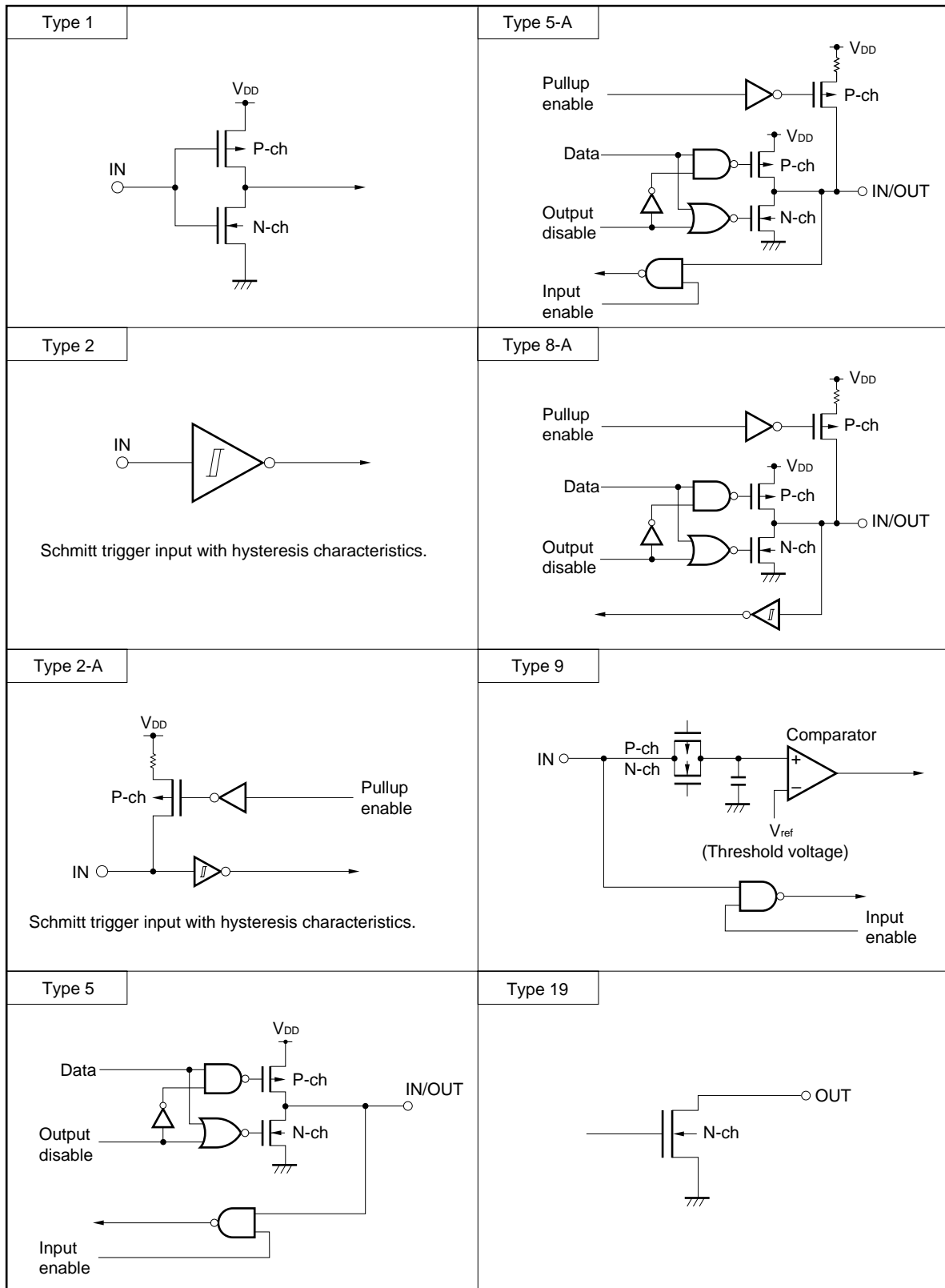
2.3 I/O Circuits and Processing of Unused Pins

Table 2-4 shows the I/O type of a pin with a function, and the recommended processing of the pin when the pin function is not used. Figure 2-1 shows the circuit of each I/O type.

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

Pin Name	I/O Circuit Type	Recommended Processing of Unused Pins
P00/RTP0-P03/RTP3	5-A	Input: Individually connect to V _{DD} or V _{SS} via resistor
P04/PWM0		Output: Leave unconnected
P05/TCUD/PWM1		
P06/TIUD/TO40		
P07/TCLRUD		
P10-P17		
P20/NMI	2	Connect to V _{SS}
P21/INTP0-P23/INTP2	2-A	
P24/INTP3/TI		
P25/INTP4		
P30/TxD0	5-A	Input: Individually connect to V _{DD} or V _{SS} via resistor
P31/RxD0		Output: Leave unconnected
P32/SO/SB0	8-A	
P33/SI/SB1		
P34/ \overline{SCK}		
P35/TxD1	5-A	
P36/RxD1		
P40/AD0-P47/AD7		
P50/A8-P57/A15		
P70/ANI0-P77/ANI7	9	Connect to V _{SS}
P80/TO00-P85/TO05	5-A	Input: Individually connect to V _{DD} or or V _{SS} via resistor
P90/ \overline{RD}		Output: Leave unconnected
P91/ \overline{WR}		
P92, P93		
ASTB	5	
\overline{WDTO}	19	Connect to V _{SS}
MODE0, MODE1	1	—
\overline{RESET}	2	
AV _{REF} , AV _{SS}	—	Connect to V _{SS}
AV _{DD}		Connect to V _{DD}

Figure 2-1. I/O Circuit of Each Pin



[MEMO]

CHAPTER 3 CPU ARCHITECTURE

3.1 Memory Space

The μ PD78366A can access memory of up to 64K bytes. The μ PD78366A can also select the internal ROM or an external memory (ROM-less mode) for access by using the MODE0 and 1 pins.

Each device has a different program memory map. Both devices have the same data memory map.

(1) μ PD78365A (MODE0, 1 = HH)

Program memory is mapped into a 63232-byte area at addresses 0000H to F6FFH in the external memory. This area can also be used as data memory.

Data memory is mapped in a 2048-byte area at addresses F700H to FEFFH in the internal RAM.

Always set MODE0 and 1 = HH.

★

(2) μ PD78368A (MODE0, 1 = LL)

Program memory is mapped into a 49152-byte area at addresses 0000H to BFFFH in the internal ROM and a 14080-byte area at addresses C000H to F6FFH in the external memory. The external memory is accessed in the external memory expansion mode. The area mapped in the external memory can also be used as data memory.

Data memory is mapped into a 2048-byte area at addresses F700H to FEFFH in the internal RAM.

Usually, set MODE0 and 1 = LL. To set the ROM-less mode, set MODE0 and 1 = HH.

(3) μ PD78366A (MODE0, 1 = LL)

Program memory is mapped into a 32768-byte area at addresses 0000H to 7FFFH in the internal ROM and a 30464-byte area at addresses 8000H to F6FFH in the external memory. The external memory is accessed in the external memory expansion mode. The area mapped in the external memory can also be used as data memory.

Data memory is mapped into a 2048-byte area at addresses F700H to FEFFH in the internal RAM.

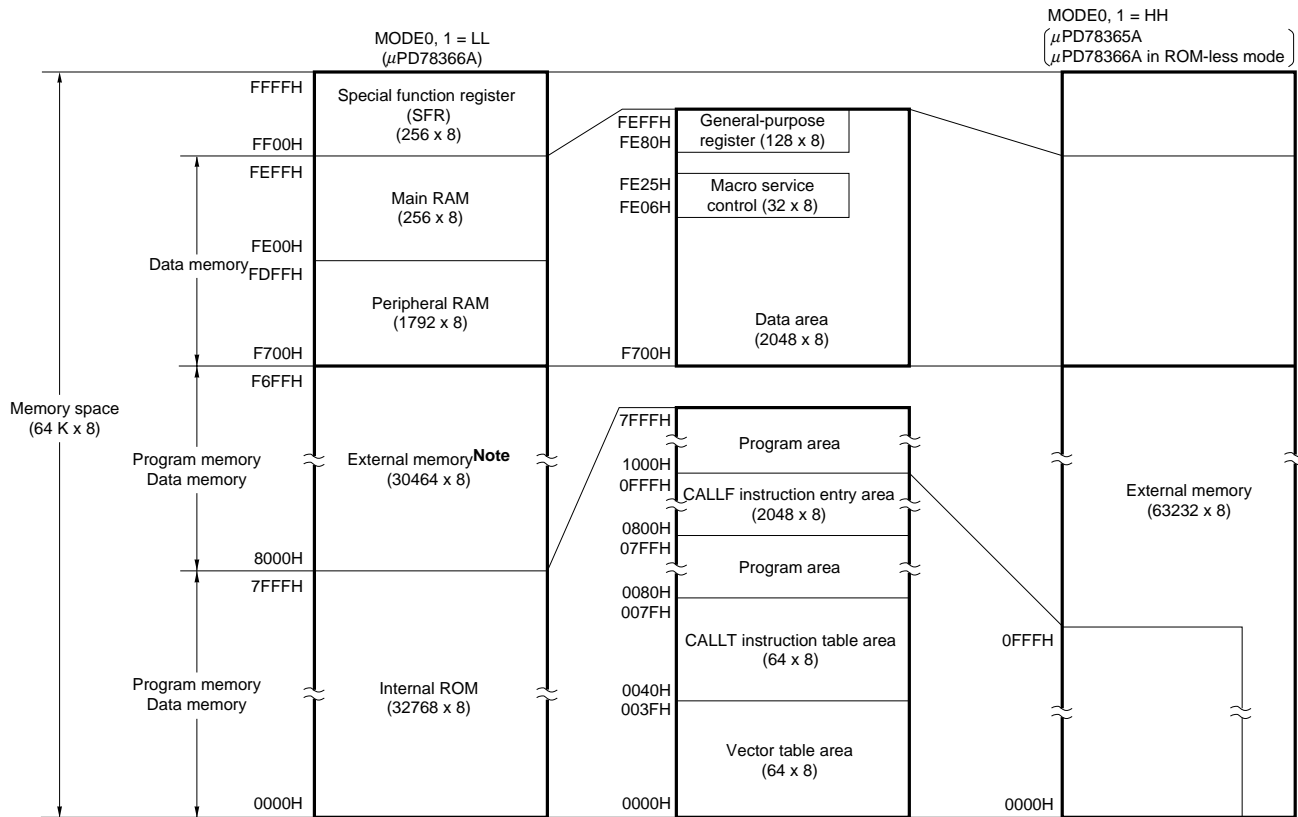
Usually, set MODE0 and 1 = LL. To set the ROM-less mode, set MODE0 and 1 = HH.

(4) μ PD78363A (MODE0, 1 = LL)

Program memory is mapped into a 24576-byte area at addresses 0000H to 5FFFH in the internal ROM and a 39936-byte area at addresses 6000H to FBFFH in the external memory. The external memory is accessed in the external memory expansion mode. The area mapped in the external memory can also be used as data memory.

Data memory is mapped into a 768-byte area at addresses FC00H to FEFFH in the internal RAM.

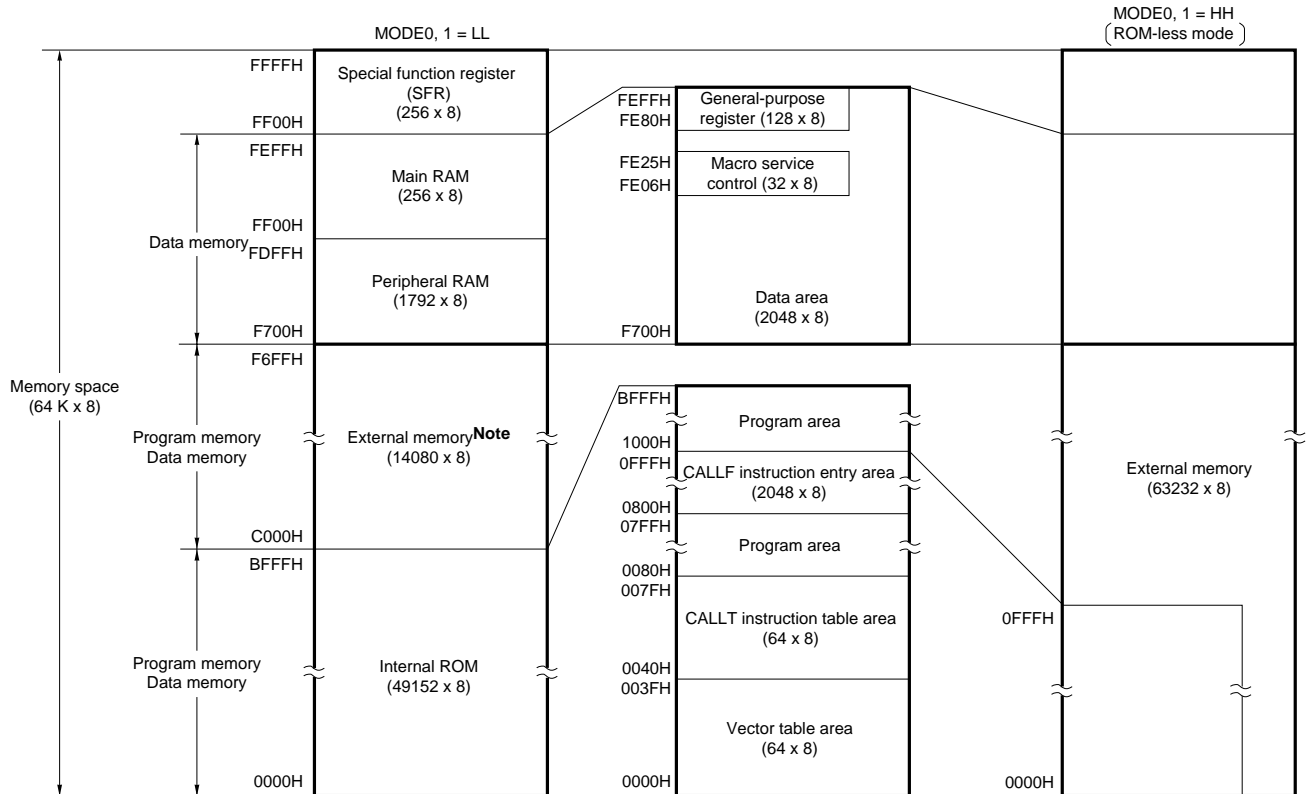
Usually, set MODE0 and 1 = LL. To set the ROM-less mode, set MODE0 and 1 = HH.

Figure 3-1. Memory Map (μ PD78365A, 78366A)

Note Accessed in external memory expansion mode.

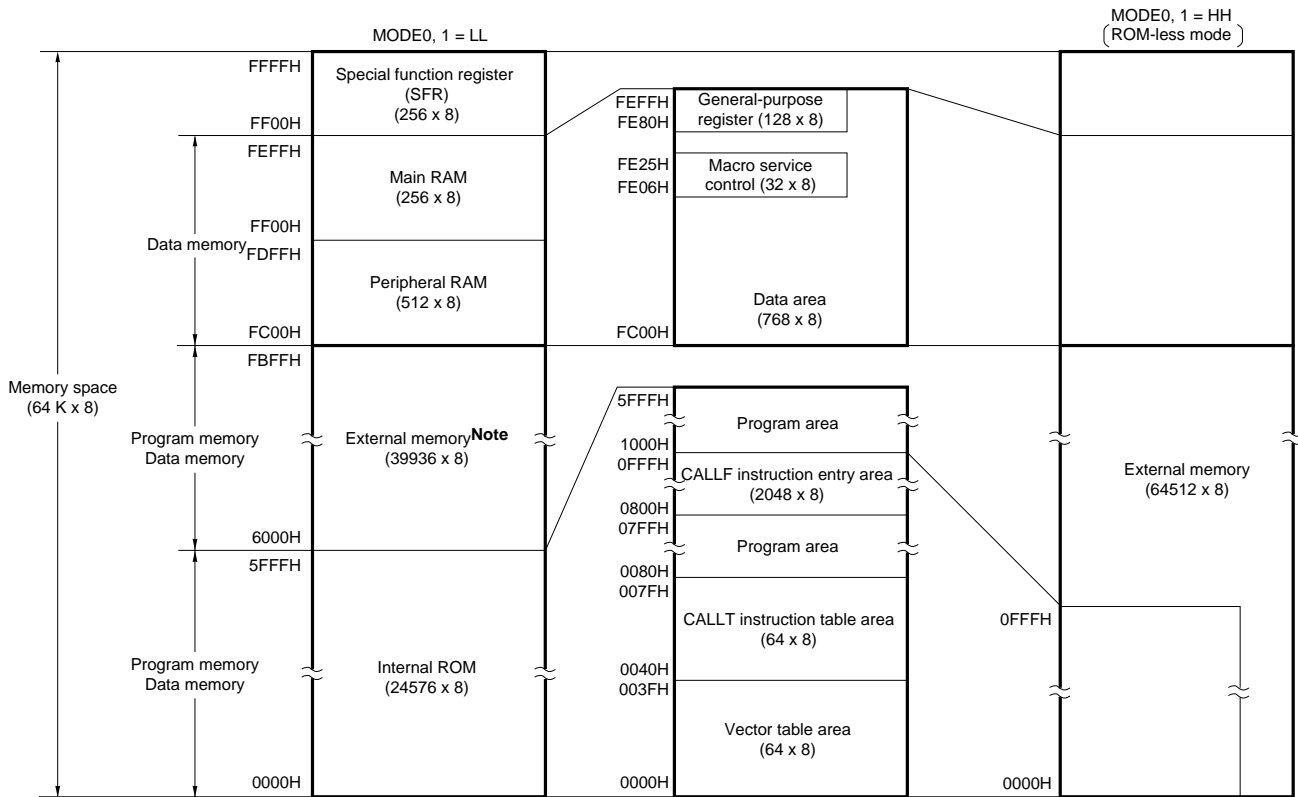
Caution For word access (including stack operations) to the main RAM area (FE00H - FEFFH), the address that specifies the operand must be an even value.

★

Figure 3-2. Memory Map (μ PD78368A)

Note Accessed in external memory expansion mode.

Caution For word access (including stack operations) to the main RAM area (FE00H - FEFFH), the address that specifies the operand must be an even value.

Figure 3-3. Memory Map (μ PD78363A)

Note Accessed in external memory expansion mode.

Caution For word access (including stack operations) to the main RAM area (FE00H - FEFFH), the address that specifies the operand must be an even value.

(5) μ PD78P368A (MODE0, 1 = LL)

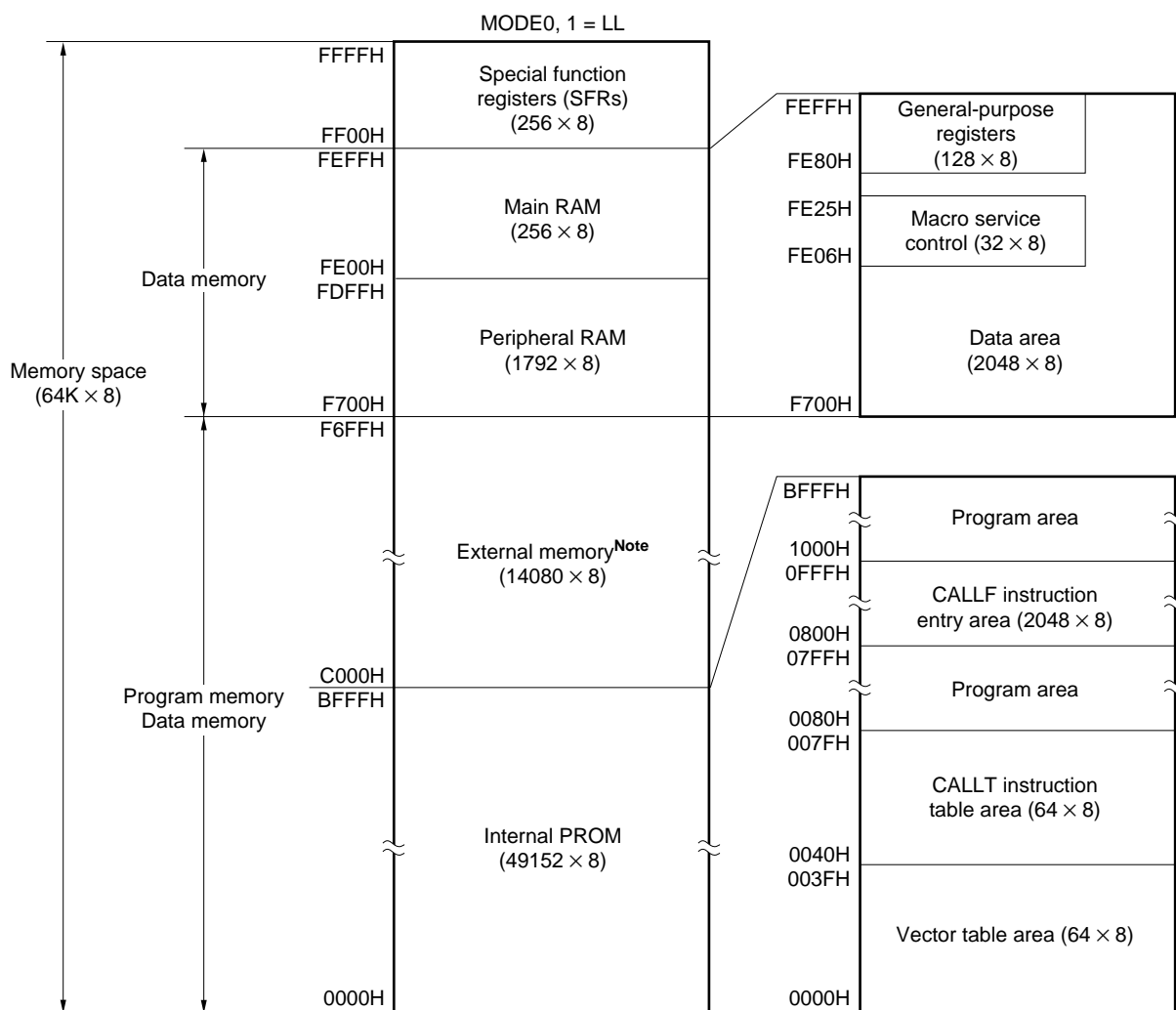
Program memory is mapped into a 49152-byte area at addresses 0000H to BFFFH in the internal PROM and a 14080-byte area at addresses C000H to F6FFH in the external memory. The external memory is accessed in the external memory expansion mode. The area mapped in the external memory can also be used as data memory.

Data memory is mapped into a 2048-byte area at addresses F700H to FEFFH.

Usually, set MODE0 and 1 = LL. To set the programming mode, set MODE0 and 1 = HL.

Caution The μ PD78P368A cannot be set in the ROM-less mode.

Figure 3-4. Memory Map (μ PD78P368A)



Note Access in the external memory expansion mode

Caution When a word access to the main RAM area (FE00H to FEFFH) (containing stack handling) is executed, addresses specified in operands are limited to even addresses.

3.1.1 Vector table area

Interrupt requests from peripheral hardware, reset inputs, external interrupt requests, and branch addresses interrupted by a break instruction, are stored in the 64-byte area from 0000H to 003FH.

When an interrupt request is issued, the contents of each vector table are set to the program counter (PC) before branching. The contents of the even addresses are set to low-order eight bits of the PC and the contents of the odd addresses are set to the high-order eight bits.

When the TPF bit of the CPU control word (CCW) is set to 1, the external memory area 8002H to 803FH, instead of the area 0002H to 003FH, is used as an interrupt vector table.

Table 3-1. Vector Table Area

Interrupt Source		Vector Table Address	
Interrupt Request	Interrupt Source/Unit	TPF = 0	TPF = 1
RESET	RESET pin input	0000H	
NMI	NMI pin input	0002H	8002H
INTWDT	Watchdog timer	0004H	8004H
INTOV3	Real-time pulse unit	0006H	8006H
INTP0/INTCC30	INTP0 pin input/real-time pulse unit	0008H	8008H
INTP1	INTP1 pin input	000AH	800AH
INTP2	INTP2 pin input	000CH	800CH
INTP3/INTCC20	INTP3 pin input/real-time pulse unit	000EH	800EH
INTP4	INTP4 pin input	0010H	8010H
INTTM0	Real-time pulse unit	0012H	8012H
INTCM03		0014H	8014H
INTCM10		0016H	8016H
INTCM40		0018H	8018H
INTCM41		001AH	801AH
INTSER	Asynchronous serial interface	001CH	801CH
INTSR		001EH	801EH
INTST		0020H	8020H
INTCSI	Clocked serial interface	0022H	8022H
INTAD	A/D converter	0024H	8024H
OP code trap	—	003CH	
BRK instruction	—	003EH	

3.1.2 CALLT instruction table area

Thirty-two tables of the address called by a single-byte call instruction (CALLT) can be stored in the 64-byte area from 0040H to 007FH. This area is the CALLT table area.

When the TPF bit of the CPU control word (CCW) is set to 1, an external memory area 8040H to 807FH, instead of the area 0040H to 007FH, is used as a CALLT instruction table.

3.1.3 CALLF instruction entry area

A subroutine can be directly called from the area 0800H to 0FFFFH by using a double-byte call instruction (CALLF).

3.1.4 Internal RAM area

2048-byte (768-byte for μ PD78363A) RAM is built into the area F700H to FEFFH (FC00H to FEFFH for μ PD78363A). This area consists of the following two components:

- Peripheral RAM : F700H to FDFFH (1792 bytes) μ PD78365A, 78366A, 78368A, 78P368A
FC00H to FDFFH (512 bytes) μ PD78363A
- Main RAM : FE00H to FEFFH (256 bytes)

High-speed access to the main RAM is possible. Macro service control words are mapped in the 32-byte area from FE06H to FE25H in the main RAM area. General-purpose registers are mapped in the 128-byte area from FE80H to FEFFH in the main RAM area. The general-purpose registers consist of eight banks.

- Cautions**
1. When a word access to the main RAM area (FE00H to FEFFH) (containing stack handling) is executed, the access operation varies, depending on whether the reference address is even or odd. (See Table 3-2). Therefore, if an access to an even address and an access to an odd address are mixed, an error is caused. Specify only even reference addresses. (See Examples 1 and 2.) To execute a 16-bit data transfer instruction, specify even addresses in operands. If odd addresses are specified, an error occurs in the assembler package (RA78K3).
 2. Do not make a word access across the peripheral RAM area and main RAM area. (See Example 3).

Table 3-2. Operation in Word Access in Internal RAM Area

Reference Address (n) \ Access Area	Even	Odd
Main RAM	○	×
Peripheral RAM	○	○

Remark ○: Access to addresses n and $n + 1$
 ×: Access to addresses n and $n - 1$

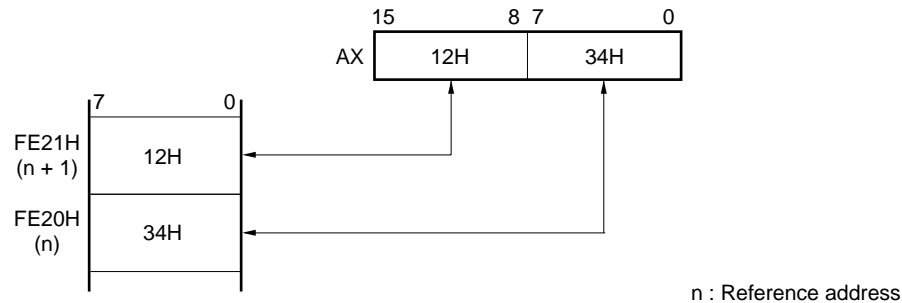
Examples of word access in the internal RAM area are given in Examples 1-5.

Examples 1. To write/read word data into/from an even address (FE20H) in the main RAM area

When word data is written into an even address (address n) in the main RAM area, the low-order eight bits of the word data are written into the even address (address n) and the high-order eight bits are written into the odd address (address $n + 1$).

When word data is read from an even address (address n) in the main RAM area, it is read from addresses n and $n + 1$.

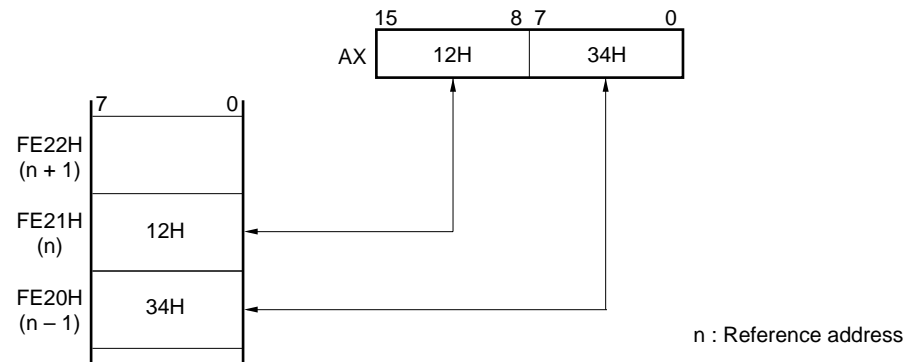
```
MOVW  AX, #1234H
MOVW  0FE20H, AX; Write word data into FE20H
MOVW  AX, 0FE20H; Read word data from FE20H
```

**2. To write/read word data into/from an odd address (FE21H) in the main RAM area**

When word data is written into an odd address in the main RAM area, the high-order eight bits of the word data are written into the odd address (address n) and the low-order eight bits are written into the even address (address $n - 1$).

When word data is read from an odd address (address n) in the main RAM area, it is read from addresses n and $n - 1$.

```
MOVW  AX, #1234H
MOVW  DE, #0FE21H
MOVW  [DE], AX; Write word data into FE21H
MOVW  AX, [DE]; Read word data from FE21H
```



Example 3. To write/read word data across the peripheral RAM area and main RAM area

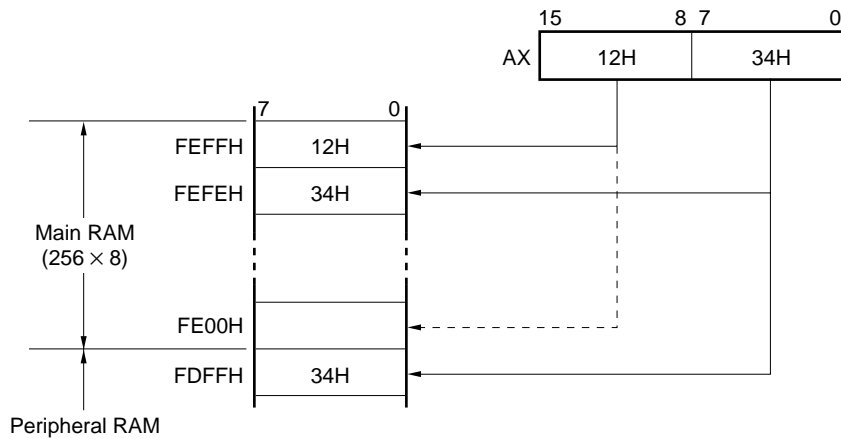
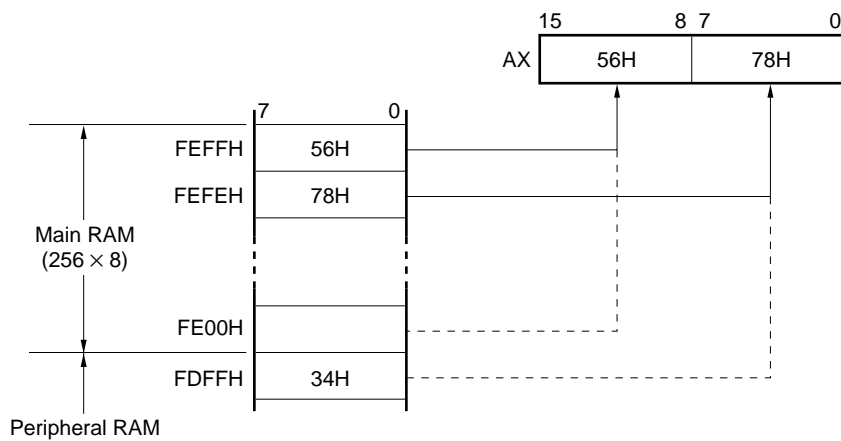
If word data is written across the peripheral RAM area and main RAM area, it is written into the 256-byte apart address, causing an error to occur.

If word data is read from the end address of the peripheral RAM area (FDFFH), it is read from FEFEH and FEFFH which are 256 bytes apart from FDFFH.

```

MOVW    AX, #1234H
MOVW    DE, #0FDFFH
MOVW    [DE], AX; Write word data into peripheral RAM (FDFFH)
.
.
.
MOVW    DE, #0FDFFH
MOVW    AX, [DE]; Read word data from peripheral RAM (FDFFH)

```

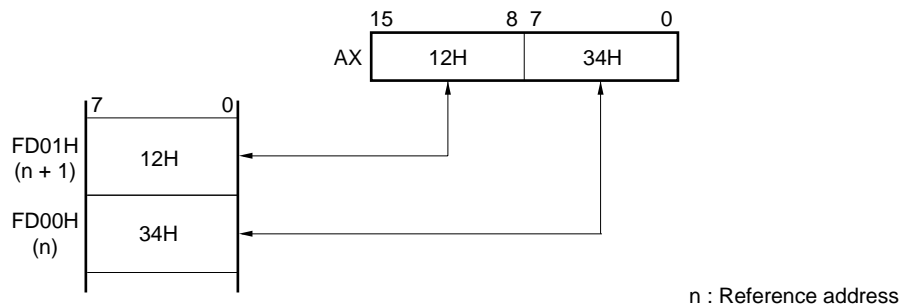
(Write)**(Read)**

Example 4. To write/read word data into/from an even address (FD00H) in the peripheral RAM area

When word data is written into an even address in the peripheral RAM area, the low-order eight bits of the word data are written into the even address (address n) and the high-order eight bits are written into the odd address (address $n + 1$).

When word data is read from an even address (address n) in the peripheral RAM area, it is read from addresses n and $n + 1$.

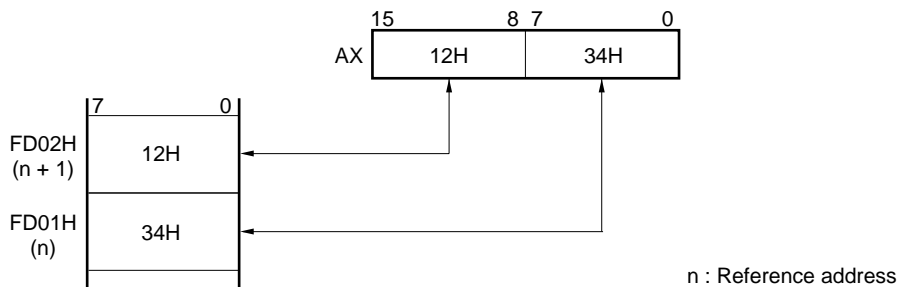
```
MOVW  AX, #1234H
MOVW  DE, #0FD00H
MOVW  [DE], AX; Write word data into FD00H
MOVW  AX, [DE]; Read word data from FD00H
```

**5. To write/read word data into/from an odd address (FD01H) in the peripheral RAM area**

When word data is written into an odd address in the peripheral RAM area, the low-order eight bits of the word data are written into the odd address (address n) and the high-order eight bits are written into the even address (address $n + 1$).

When word data is read from an odd address (address n) in the peripheral RAM area, it is read from addresses n and $n + 1$.

```
MOVW  AX, #1234H
MOVW  DE, #0FD01H
MOVW  [DE], AX; Write word data into FD01H
MOVW  AX, [DE]; Read word data from FD01H
```



3.1.5 Special function register area

Registers having special functions, such as mode and control registers for the peripheral hardware, are mapped in the area from FF00H to FFFFH.

Caution Unmapped addresses of the special function register cannot be accessed (except the external access area).

3.1.6 External memory area

The μ PD78366A can expand an external memory (ROM or RAM) step by step in the 30K-byte area (8000H to F6FFH)^{Note}.

The μ PD78365A can connect an external memory (ROM or RAM) in the 64K-byte area (0000H to F6FFH).

To access an external memory, the pins P40/AD0 to P47/AD7 (address/data bus), P50/A8 to P57/A15 (address bus), \overline{RD} , \overline{WR} , and ASTB are used. An external access area is mapped in the 16-byte area from FFD0H to FFDFH of the special function register (SFR) area. SFR addressing can access an external memory in this area.

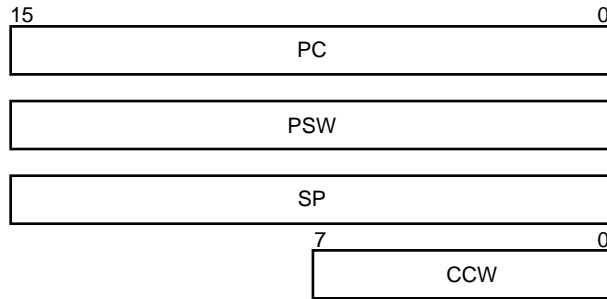
Note 40K-byte area (6000H to FBFFH) for μ PD78363A, 14K-byte area (C000H to F6FFH) for μ PD78368A and 78P368A.

3.2 Processor Register

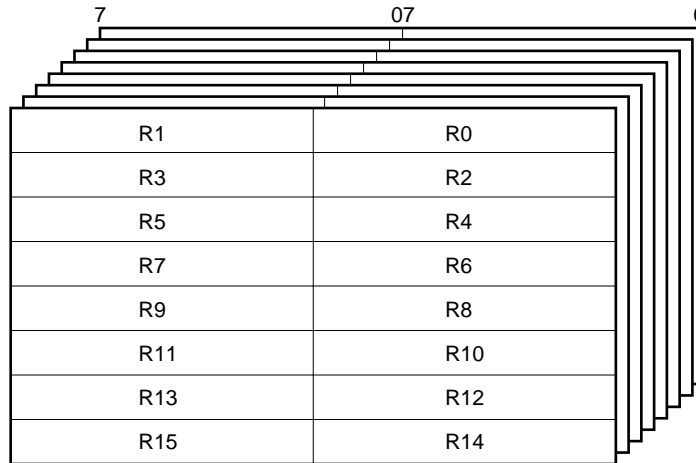
There are three groups of registers: a control register group (consisting of one 8-bit register and three 16-bit registers), a general-purpose register group (consisting of eight banks, each of which consists of sixteen 8-bit registers), and a special function register group (consisting of registers having special functions such as I/O mode registers for the peripheral hardware).

Figure 3-5. Register Configuration

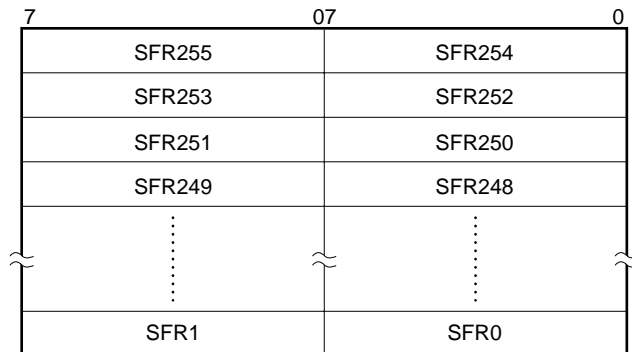
Control register



General-purpose register



Special function register



Remark The CCW of the control register group is mapped in the special function register (SFR) area.

3.2.1 Control register

The control register group controls the program sequence, status, and stack memory and modifies operand addressing.

This group consists of an 8-bit register and three 16-bit registers.

(1) Program counter (PC)

The program counter (PC) is a 16-bit register which holds address information of the program to be executed next.

The PC operates as follows:

- **In normal operation**

Automatically incremented according to the number of bytes of the instruction to be fetched.

- **When a branch instruction is executed**

The contents of the immediate data or register are set.

Input to the $\overline{\text{RESET}}$ pin sets the data in the reset vector table at 0000H and 0001H in the PC and makes a branch.

(2) Program status word (PSW)

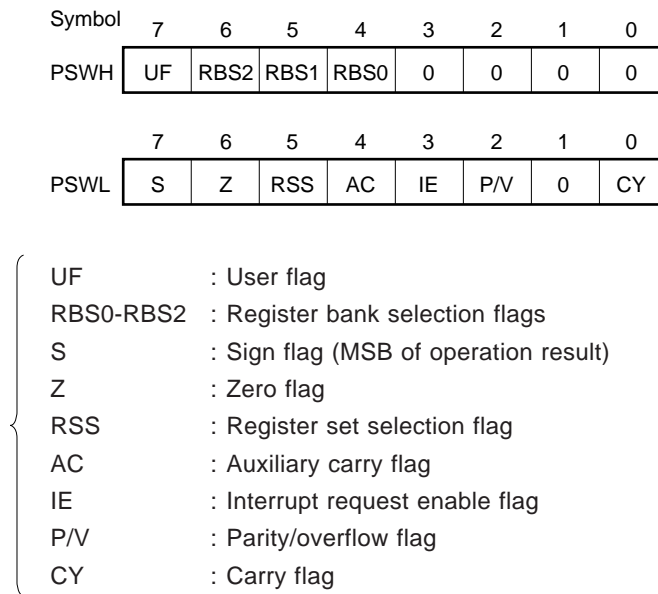
The program status word (PSW) is a 16-bit register consisting of flags set or reset according to the result of executing an instruction.

Read and write operations are performed by the high-order eight bits (PSWH) or low-order eight bits (PSWL). Flags are operated by the bit manipulation instructions.

When an interrupt request is issued and when a BRK instruction is executed, the contents of the PSW is automatically saved in the stack. When an RETI or RETB instruction is executed, the contents are automatically restored.

Input to the $\overline{\text{RESET}}$ pin resets all bits to 0.

Figure 3-6. Format of Program Status Word



The flags are explained below.

(a) User flag (UF)

This flag controls the program. The flag is set or reset on the user program.

(b) Register bank selection flag (RBS0-RBS2)

This 3-bit flag selects one of eight register banks (register bank 0 to register bank 7).

(c) Sign flag (S)

The sign flag indicates that the most significant bit after an arithmetic/logical operation is 1.

This flag is set to 1 when the most significant bit after the operation is 1. Otherwise, this flag is reset to 0. This flag can be tested with a conditional branch instruction.

(d) Zero flag (Z)

The zero flag indicates that the result of an arithmetic/logical operation is 0.

This flag is set to 1 when the result is 0. Otherwise, this flag is reset to 0.

This flag can be tested with a conditional branch instruction.

(e) Register set selection flag (RSS)

This flag specifies general-purpose registers (8 bits each) functioning as X, A, C, and B and general-purpose register pairs (16 bits each) functioning as AX and BC.

The RSS flag values correspond to function names and absolute names enclosed in parentheses as follows (refer to **Table 3-3. General-Purpose Register Configuration**).

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

To set or reset the RSS flag, be sure to write an RSS directive just before or immediately after the instruction for setting or resetting the RSS flag as shown in the example below:

<Program example>

- **To reset the RSS flag (RSS = 0)**

```
RSS 0      ; RSS directive
CLR1 PSWL.5
MOV  B, A   ; Corresponds to "MOV R3, R1".
```

- **To set the RSS flag (RSS = 1)**

```
RSS 1      ; RSS directive
SET1 PSWL.5
MOV  B, A   ; Corresponds to "MOV R7, R5".
```

Switching the RSS flag between the values has the same effect as using two register sets. Registers and register pairs not specified by the RSS flag can be accessed by writing the absolute names in the program.

(f) Auxiliary carry flag (AC)

The auxiliary carry flag is used for decimal adjustment and indicates that an underflow or overflow has occurred for bit 3.

This flag is set to 1 when the result of executing an arithmetic/logical instruction generates a carry of bit 3 (overflow) or a borrow into bit 3 (underflow). Otherwise, this flag is reset to 0.

This flag can be tested with a conditional branch instruction.

(g) Interrupt request enable flag (IE)

This flag enables or disables an interrupt request.

Executing an EI instruction sets this flag to 1. Executing a DI instruction or receiving an interrupt resets this flag to 0.

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

When an arithmetic/logical instruction is executed, this flag operates as follows.

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

- **Parity flag operation**

This flag is set to 1 if the number of set bits (set to 1) as the result of executing a logical instruction is an even number. Otherwise, this flag is reset to 0. The parity flag depends on only the low-order eight bits of the logical operation result whether the logical operation is performed in units of 8 or 16 bits.

- **Overflow flag operation**

This flag is set to 1 if the result of executing an arithmetic instruction exceeds the two's complement range. Otherwise, the flag is reset to 0.

For example, the two's complement range for 8-bit arithmetic operation is from 80H (−128) to 7FH (+127). The flag is set to 1 if the result exceeds the range. The flag is reset to 0 if the result is within the range.

Example The overflow flag operates as follows while executing an 8-bit add instruction.

When 78H (+120) is added to 69H (+105), the result is E1H (+225). The P/V flag is then set to 1 because the result exceeds the upper limit of the two's complement range. E1H is represented as -31 in two's complement.

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

When the following two negative numbers are added, the P/V flag is reset to 0 because the result is within the two's complement range.

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

(i) Carry flag (CY)

The carry flag indicates that an overflow or underflow has occurred in an arithmetic/logical operation. This flag is set to 1 when an arithmetic/logical operation results in a carry (overflow) or a borrow (underflow) for bit 7. In word operations, this flag is set to 1 when a carry (overflow) or borrow (underflow) occurs for bit 15. Otherwise, this flag is reset to 0.

This flag can be tested with a conditional branch instruction. This flag also functions as a bit accumulator when a bit manipulation instruction is executed.

(3) Stack pointer (SP)

The stack pointer (SP) is a 16-bit register which holds the first address of the memory stack area (LIFO format).

The SP is manipulated by a dedicated instruction (Stack manipulation instruction).

The stack pointer is decremented before data is written (saved) into the stack memory, and is incremented after data is read (restored) from the stack memory.

As input to the $\overline{\text{RESET}}$ pin causes the SP to become undefined, the SP must be set before calling a subroutine.

Caution When a word access to the main RAM area (FE00H to FEFFH) is executed, addresses specified in operands are limited to even addresses.

(4) CPU control word (CCW)

The CPU control word (CCW) is an 8-bit register which consists of flags related to CPU control.

The CCW is mapped in the special function register area (FFC1H) and can be controlled by the software.

Input to the $\overline{\text{RESET}}$ pin resets all bits to 0.

Figure 3-7. Format of CPU Control Word

Symbol	7	6	5	4	3	2	1	0	Address	When reset	R/W
CCW	0	0	0	0	0	0	TPF	0	FFC1H	00H	R/W

TPF : Table position flag

The table position flag (TPF) specifies the location of a vector table referenced by a CALLT instruction or interrupt request. The TPF flag switches the vector table location as follows:

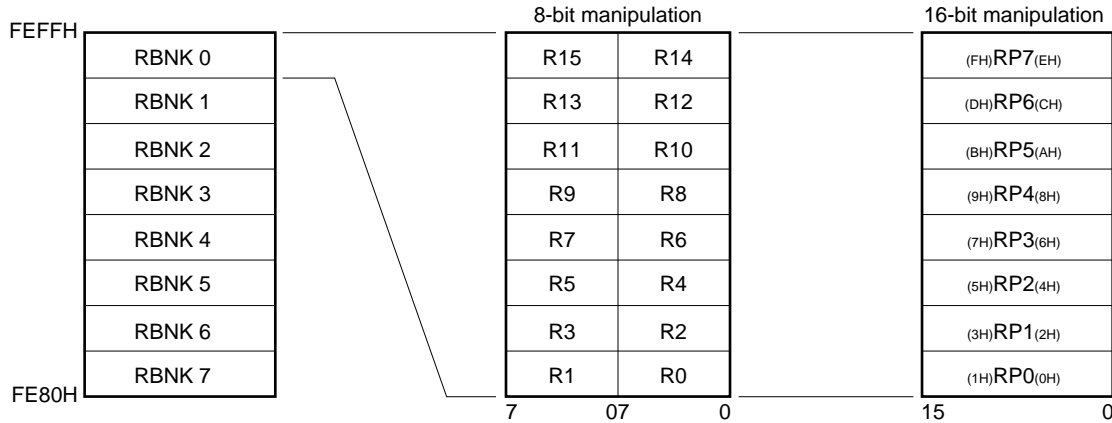
- TPF = 0 (reset)
0000H-007FH
- TPF = 1 (set)
8000H-807FH

Caution The vector tables for the $\overline{\text{RESET}}$ input, BRK instruction, and op-code trap interrupt are fixed to 0000H, 003EH, and 003CH, respectively. They are not affected by the TPF.

3.2.2 General-purpose register

The 128-byte general-purpose register group which consists of eight banks is mapped in the specific area (FE80H to FEFFH) of the internal RAM space. Each bank consists of 16 8-bit general-purpose registers.

Figure 3-8. Manipulation Bits of General-Purpose Registers



A pair of 8-bit registers can function as eight 16-bit register pairs (RP0-RP7).

A function name listed in Table 3-3, as well as an absolute name, is assigned to each 8-bit register (16 registers). The X register functions as the low-order bits of a 16-bit accumulator. The A register functions as an 8-bit accumulator or the high-order bits of a 16-bit accumulator. The B and C registers function as a counter. The DE, HL, VP, and UP registers, in a pair, function as an address register. The VP register functions as a base register and the UP register functions as a user stack pointer.

The value of the register set selection flag (RSS) in the program status word (PSW) changes the register having a specific function, as shown in Table 3-3.

If the program has been coded with the function names, operating the RSS flag has the same effect as using two sets of registers (X, A, B, C, AX, and BC). A register not specified by the RSS flag can be accessed by writing the absolute name in a program. When the RSS flag is 0, for example, register R4 can be accessed by specifying the absolute name, that is, R4 in the program.

The μ PD78366A can implement two types of addressing: implied addressing and register addressing. Implied addressing is performed as process data addressing by a function name which places much importance on the specific function of each register. Register addressing is performed by an absolute name which aims to create a program which is easy to describe and which performs less data transfer operations, enabling high-speed data processing.

Table 3-3. General-Purpose Register Configuration**(a) Correspondence between absolute names and function names for 8-bit registers**

Absolute Name	Function Name	
	RSS = 0	RSS = 1
R0	X	
R1	A	
R2	C	
R3	B	
R4		X
R5		A
R6		C
R7		B
R8	V _{PL}	V _{PL}
R9	V _{PH}	V _{PH}
R10	U _{PL}	U _{PL}
R11	U _{PH}	U _{PH}
R12	E	E
R13	D	D
R14	L	L
R15	H	H

(b) Correspondence between absolute names and function names for 16-bit register pairs

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

3.2.3 Special function register (SFR)

Unlike the general registers, the special function registers (SFRs) have special functions. The SFRs are assigned to the memory space at addresses FF00H to FFFFH, namely, a 256-byte special function register area.

Short direct addressing is available for a 32-byte area at addresses FF00H to FF1FH. Data in the SFRs assigned to this area can be processed in fewer clock cycles because the word length of the SFRs in the area is less than that of the SFRs in other areas. These assigned SFRs consist of capture register, compare register, and port, and they are frequently accessed.

The 16-byte area at addresses FFD0H to FFDFH is used to access the external storage medium by SFR addressing. Instructions in short word length enable access to external memory or bit manipulation in the external device.

The SFRs can be manipulated by arithmetic/logical instructions, transfer instructions, bit manipulation instructions, or such like in the same way as general-purpose registers. The manipulatable bit units (1, 8, or 16 bit units) vary according to the SFR (refer to **Table 3-4**).

The following describes the methods for specifying the SFRs corresponding to manipulatable bit units:

- **Bit manipulation**

Specify the abbreviation for the operand (sfr.bit) of a bit manipulation instruction. The SFR can also be addressed.

- **8-bit manipulation**

Specify the abbreviation for the operand (sfr) of an 8-bit manipulation instruction. The SFR can also be addressed.

- **16-bit manipulation**

Specify the abbreviation for the operand (sfrp) of a 16-bit manipulation instruction. A 16-bit SFR is assigned to a two-byte area at consecutive even and odd addresses. Specify an even address when addressing the SFR.

Table 3-4 lists the special function registers (SFRs). The items in Table 3-4 mean:

- Abbreviation A symbol indicating the address of a on-chip SFR. This can be specified in the operand field of an instruction.
Reserved words in the NEC Assembler (RA78K3) are already defined.
They can be used as sfr variable by #pragma sfr instruction in the C compiler (CC78K3).
- R/W Indicates whether data can be read from the special function register and/or data can be written into the register.
R/W : Can be read and written.
R : Read only^{Note}.
W : Write only.
- Manipulatable bit unit..... Indicates the unit of bits (1, 8, or 16) when manipulating the special function register (indicated by ○).
The SFR which can be manipulated in units of 16 bits can be specified in the sfrp operand. An even address is specified for the address specification.
The SFR which can be manipulated bit by bit can be specified by a bit manipulation instruction.
- When reset Indicates the status of each register for the input to the $\overline{\text{RESET}}$ pin.

Note Read-only register. The bits of the register can be tested.

- Cautions**
1. Write 0 or 1 into any SFR bit correctly whenever it is predetermined to be so.
 2. Do not write data into the register which is only used for data reading. Writing data into such registers may result in an error.
 3. The SFR area addresses (FF00H to FFFFH) to which a special function register is not assigned cannot be accessed (except the external access area). Accessing these addresses may result in an error.
 4. When the read data is used as byte data, handle undefined bits before the data is used.
 5. TOUT and TXS are write-only registers. Do not read from them.
 6. Bits 0, 1 and 4 of SBIC are write-only bits. If these bits are read, the value read is 0.

Table 3-4. Special Function Registers (1/5)

Address	Special Function Register (SFR) Name	Abbreviation	R/W	Manipulatable Bit Unit			When Reset
				1	8	16	
FF00H	Port 0	P0	R/W	○	○	—	Undefined
FF01H	Port 1	P1		○	○	—	
FF02H	Port 2	P2	R	—	○	—	
FF03H	Port 3	P3	R/W	○	○	—	
FF04H	Port 4	P4Note		○	○	—	
FF05H	Port 5	P5Note		○	○	—	
FF07H	Port 7	P7	R	—	○	—	
FF08H	Port 8	P8	R/W	○	○	—	
FF09H	Port 9	P9		○	○	—	
FF10H	Compare register 00	CM00		—	—	○	
FF11H							
FF12H	Compare register 01	CM01		—	—	○	
FF13H							
FF14H	Compare register 02	CM02		—	—	○	
FF15H							
FF16H	Compare register 03	CM03		—	—	○	
FF17H							
FF18H	Buffer register CM00	BFCM00		—	—	○	
FF19H							
FF1AH	Buffer register CM01	BFCM01		—	—	○	
FF1BH							
FF1CH	Buffer register CM02	BFCM02		—	—	○	
FF1DH							
FF1EH	Timer register 0	TM0	R	—	—	○	0000H
FF1FH							
FF20H	Port 0 mode register	PM0	R/W	○	○	—	FFH
FF21H	Port 1 mode register	PM1		○	○	—	
FF23H	Port 3 mode register	PM3		○	○	—	×111 1111B
FF25H	Port 5 mode register	PM5Note		○	○	—	FFH
FF28H	Port 8 mode register	PM8		○	○	—	××11 1111B
FF29H	Port 9 mode register	PM9		○	○	—	×××× 1111B
FF2CH	Reload register	DTIME	R/W	—	—	○	Undefined
FF2DH							

Note Not contained in μ PD78365A.

Table 3-4. Special Function Registers (2/5)

Address	Special Function Register (SFR) Name	Abbreviation	R/W	Manipulatable Bit Unit			When Reset	
				1	8	16		
FF2EH	Timer unit mode register 0	TUM0	R/W	○	○	—	00H	
FF2FH	Timer unit mode register 1	TUM1		○	○	—		
FF30H	Compare register 10	CM10		—	—	○	Undefined	
FF31H								
FF32H	Timer register 1	TM1	R	—	—	○	0000H	
FF33H								
FF34H	Capture/compare register 20	CC20	R/W	—	—	○	Undefined	
FF35H								
FF36H	Capture register 20	CT20	R	—	—	○		
FF37H								
FF38H	Timer register 2	TM2		—	—	○	0000H	
FF39H								
FF3AH	Buffer register CM03	BFCM03	R/W	—	—	○	Undefined	
FF3BH								
FF3CH	External interrupt mode register 0	INTM0		○	○	—	00H	
FF3DH	External interrupt mode register 1	INTM1		○	○	—		
FF40H	Port 0 mode control register	PMC0		○	○	—		
FF43H	Port 3 mode control register	PMC3		○	○	—	×000 0000B	
FF44H	Pull-up resistor option register L	PUOL		○	○	—	00H	
FF45H	Pull-up resistor option register H	PUOH		○	○	—		
FF48H	Port 8 mode control register	PMC8		○	○	—	××00 0000B	
FF4EH	Sampling control register 0	SMPC0		○	○	—	00H	
FF4FH	Sampling control register 1	SMPC1		○	○	—		
FF50H	Capture/compare register 30	CC30	R	—	—	○	Undefined	
FF51H								
FF52H	Capture register 30	CT30		—	—	○		
FF53H								
FF54H	Capture register 31	CT31		—	—	○		
FF55H								
FF56H	Timer register 3	TM3		—	—	○	0000H	
FF57H								
FF58H	Compare register 40	CM40	R/W	—	—	○	Undefined	
FF59H								
FF5AH	Compare register 41	CM41		—	—	○		
FF5BH								

Table 3-4. Special Function Registers (3/5)

Address	Special Function Register (SFR) Name	Abbreviation	R/W	Manipulatable Bit Unit			When Reset
				1	8	16	
FF5CH	Timer register 4	TM4	R	–	–	○	0000H
FF5DH							
FF5EH	Timer control register 4	TMC4	R/W	○	○	–	00H
FF5FH	Timer out register	TOUT	W	–	○	–	xx01 0101B
FF60H	Real-time output port register	RTP	R/W	○	○	–	Undefined
FF61H	Real-time output port mode register	RTPM		○	○	–	00H
FF62H	Port read control register	PRDC		○	○	–	
FF68H	A/D converter mode register	ADM		○	○	–	
FF70H	Slave buffer register 0	SBUF0		○	○	–	Undefined
FF71H	Slave buffer register 1	SBUF1		○	○	–	
FF72H	Slave buffer register 2	SBUF2		○	○	–	
FF73H	Slave buffer register 3	SBUF3		○	○	–	
FF74H	Slave buffer register 4	SBUF4		○	○	–	
FF75H	Slave buffer register 5	SBUF5		○	○	–	
FF76H	Master buffer register 0	MBUF0		○	○	–	
FF77H	Master buffer register 1	MBUF1		○	○	–	
FF78H	Master buffer register 2	MBUF2		○	○	–	
FF79H	Master buffer register 3	MBUF3		○	○	–	
FF7AH	Master buffer register 4	MBUF4		○	○	–	
FF7BH	Master buffer register 5	MBUF5		○	○	–	
FF7CH	Timer control register 0	TMC0	R/W	○	○	–	00H
FF7DH	Timer control register 1	TMC1		○	○	–	
FF7EH	Timer control register 2	TMC2		○	○	–	
FF7FH	Timer control register 3	TMC3		○	○	–	
FF80H	Clocked serial interface mode register	CSIM		○	○	–	
FF82H	Serial bus interface control register	SBIC	R/W ^{Note}	○	○	–	Undefined
FF84H	Baud rate generator control register	BRGC	R/W	○	○	–	
FF85H	Baud rate generator compare register	BRG		–	○	–	
FF86H	Serial I/O shift register	SIO		○	○	–	
FF88H	Asynchronous serial interface mode register	ASIM		○	○	–	80H

Note Bits 7 and 5 : read/write
 Bits 6, 3, and 2 : read only
 Bits 4, 1, and 0 : write only

Table 3-4. Special Function Registers (4/5)

Address	Special Function Register (SFR) Name	Abbreviation	R/W	Manipulatable Bit Unit			When Reset
				1	8	16	
FF8AH	Asynchronous serial interface status register	ASIS	R	–	○	–	00H
FF8CH	Serial receive buffer : UART	RXB		–	○	–	Undefined
FF8EH	Serial transmit shift register : UART	TXS	W	–	○	–	
FFA0H	PWM control register 0	PWMC0	R/W	○	○	–	00H
FFA1H	PWM control register 1	PWMC1		○	○	–	Undefined
FFA2H	PWM register 0L	PWM0L		○	○	–	
FFA2H	PWM register 0	PWM0		–	–	○	
FFA3H							
FFA4H	PWM register 1L	PWM1L		○	○	–	
FFA4H	PWM register 1	PWM1		–	–	○	
FFA5H							
FFA8H	In-service priority register	ISPR		R	○	○	–
FFAAH	Interrupt mode control register	IMC	R/W	○	○	–	80H
FFACH	Interrupt mask register 0L	MK0L		○	○	–	FFH
FFACH	Interrupt mask register 0	MK0		–	–	○	FFFFH
FFADH							
FFADH	Interrupt mask register 0H	MK0H		○	○	–	FFH
FFB0H	A/D conversion result register 0	ADCR0	R	–	–	○	Undefined
FFB1H							
FFB1H	A/D conversion result register 0H	ADCR0H		–	○	–	
FFB2H	A/D conversion result register 1	ADCR1		–	–	○	
FFB3H							
FFB3H	A/D conversion result register 1H	ADCR1H		–	○	–	
FFB4H	A/D conversion result register 2	ADCR2		–	–	○	
FFB5H							
FFB5H	A/D conversion result register 2H	ADCR2H		–	○	–	
FFB6H	A/D conversion result register 3	ADCR3		–	–	○	
FFB7H							
FFB7H	A/D conversion result register 3H	ADCR3H		–	○	–	
FFB8H	A/D conversion result register 4	ADCR4		–	–	○	
FFB9H							
FFB9H	A/D conversion result register 4H	ADCR4H		–	○	–	
FFBAH	A/D conversion result register 5	ADCR5		–	–	○	
FFBBH							
FFBBH	A/D conversion result register 5H	ADCR5H		–	○	–	

Table 3-4. Special Function Registers (5/5)

Address	Special Function Register (SFR) Name	Abbreviation	R/W	Manipulatable Bit Unit			When Reset		
				1	8	16			
FFBCH	A/D conversion result register 6	ADCR6	R	–	–	○	Undefined		
FFBDH									
FFBDH				A/D conversion result register 6H	ADCR6H	–		○	–
FFBEH				A/D conversion result register 7	ADCR7	–		–	○
FFBFH									
FFBFH	A/D conversion result register 7H	ADCR7H	–	○	–				
FFC0H	Standby control register	STBC ^{Note 1}	R/W	–	○	–	0000 ×000B		
FFC1H	CPU control word	CCW		○	○	–	00H		
FFC2H	Watchdog timer mode register	WDM ^{Note 1}		–	○	–			
FFC4H	Memory expansion mode register	MM		○	○	–	Note 2		
FFC6H	Programmable wait control register	PWC		–	–	○	C0AAH		
FFC7H									
FFD0H to FFD FH	External SFR area	–		○	○	–	Undefined		
FFE0H	Interrupt control register (INTOV3)	OVIC3		○	○	–	43H		
FFE1H	Interrupt control register (INTP0/INTCC30)	PIC0		○	○	–			
FFE2H	Interrupt control register (INTP1)	PIC1		○	○	–			
FFE3H	Interrupt control register (INTP2)	PIC2		○	○	–			
FFE4H	Interrupt control register (INTP3/INTCC20)	PIC3		○	○	–			
FFE5H	Interrupt control register (INTP4)	PIC4		○	○	–			
FFE6H	Interrupt control register (INTTM0)	TMIC0		○	○	–			
FFE7H	Interrupt control register (INTCM03)	CMIC03		○	○	–			
FFE8H	Interrupt control register (INTCM10)	CMIC10		○	○	–			
FFE9H	Interrupt control register (INTCM40)	CMIC40		○	○	–			
FFEAH	Interrupt control register (INTCM41)	CMIC41		○	○	–			
FFEBH	Interrupt control register (INTSER)	SERIC		○	○	–			
FFECH	Interrupt control register (INTSR)	SRIC		○	○	–			
FFEDH	Interrupt control register (INTST)	STIC		○	○	–			
FFEEH	Interrupt control register (INTCSI)	CSIIC		○	○	–			
FFEFH	Interrupt control register (INTAD)	ADIC		○	○	–			

Notes 1. Can be written by a special instruction.

2. The state of the memory expansion mode register (MM) after a reset depends on the product.

μPD78363A...60H

μPD78365A, 78366A...20H

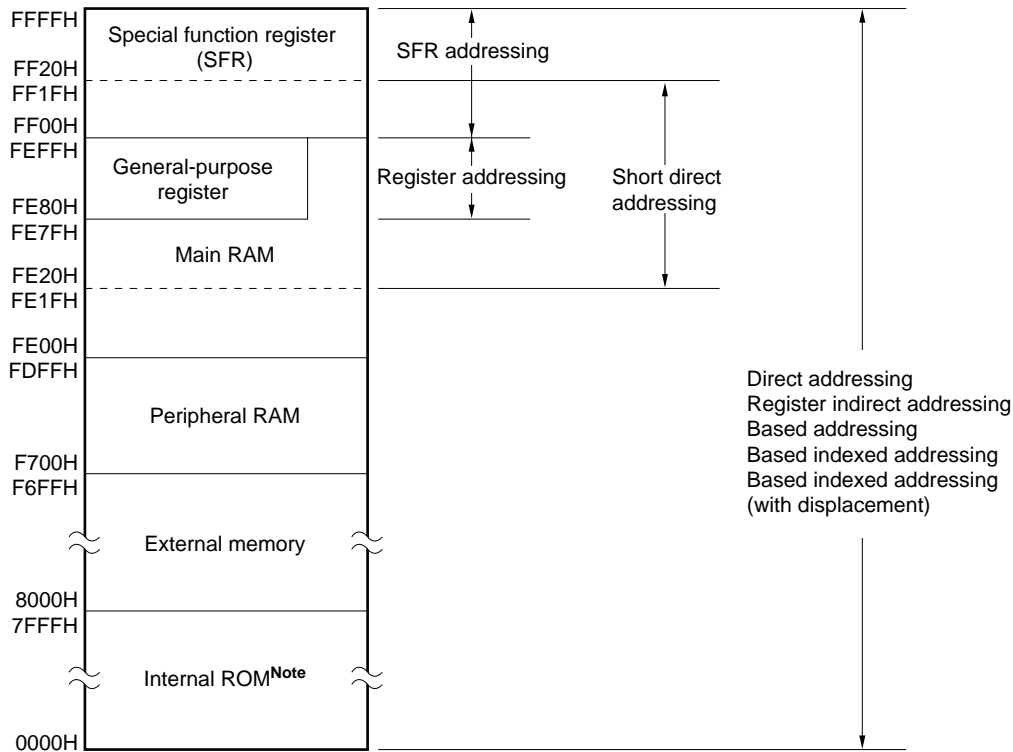
μPD78368A, 78P368A...00H

3.3 Data Memory Addressing

Various addressing modes are provided for the μ PD78366A to improve memory operability or to enable the use of a high-level language. Special addressing is applicable, in particular, to the space of data memory from F700H to FFFFH (FC00H to FFFFH for μ PD78363A) according to each function of the special function register (SFR) group and general-purpose register group.

Figure 3-9 through 3-12 show the addressing space of data memory.

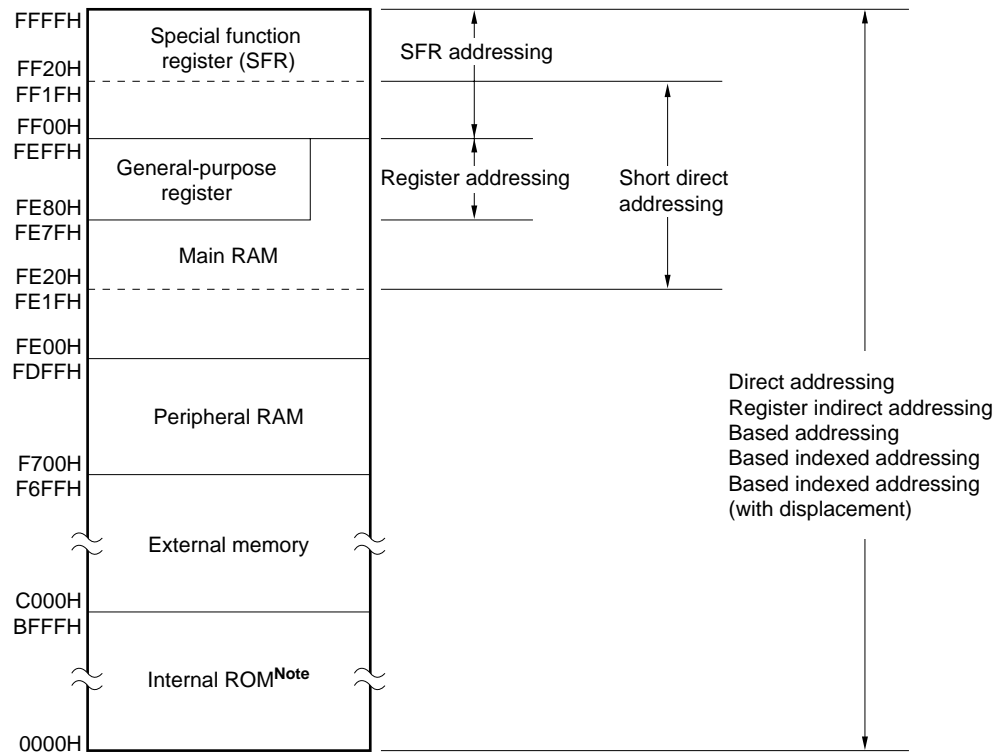
Figure 3-9. Addressing Space of Data Memory (μ PD78365A, 78366A)



Note Only the external memory can be used in the ROM-less mode of the μ PD78365A or μ PD78366A.

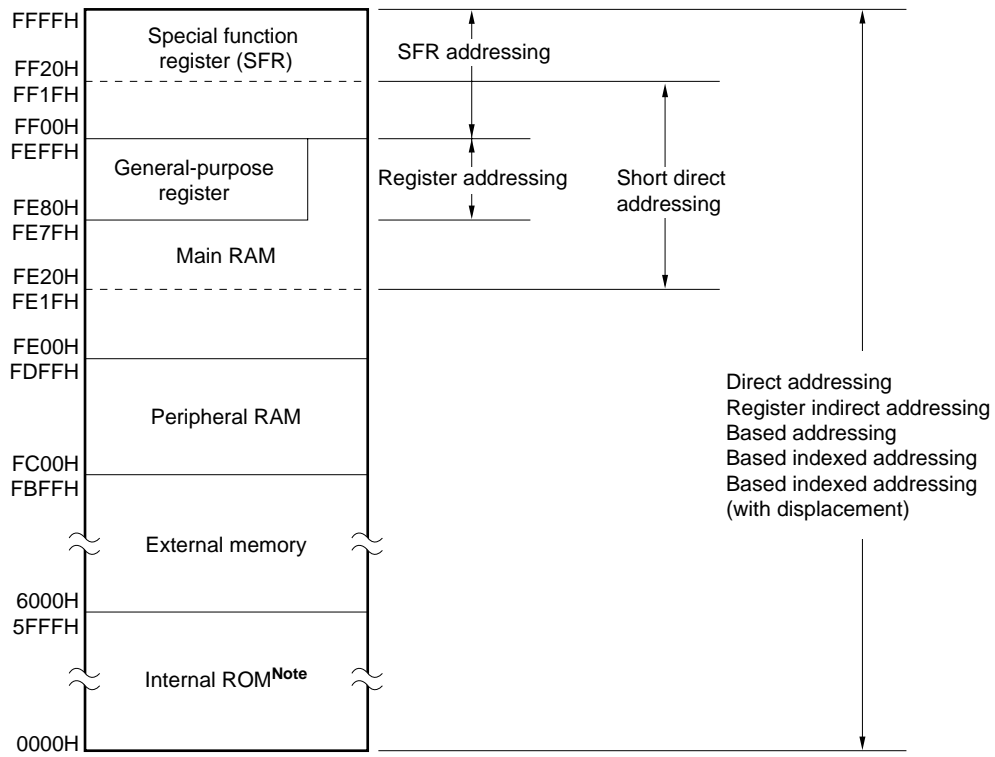
Caution When a word access to the main RAM area (FE00H to FEFFH) (containing stack manipulation) is executed, addresses specified in operands are limited to even addresses.

★

Figure 3-10. Addressing Space of Data Memory (μ PD78368A)

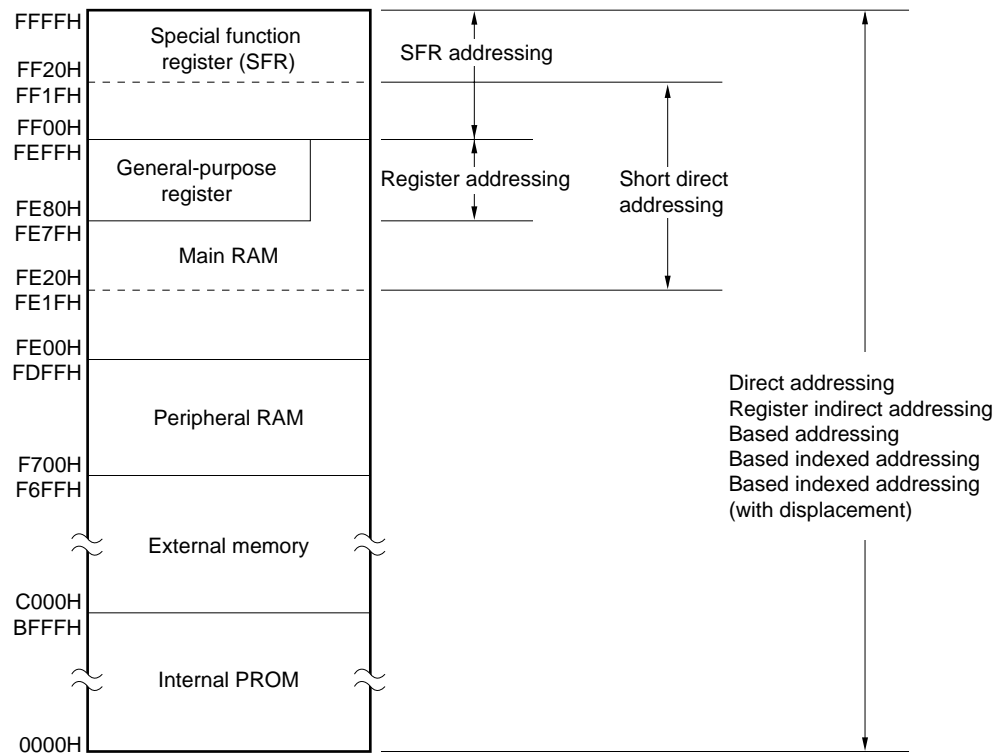
Note Only the external memory can be used in the ROM-less mode.

Caution When a word access to the main RAM area (FE00H to FEFFH) (containing stack manipulation) is executed, addresses specified in operands are limited to even addresses.

Figure 3-11. Addressing Space of Data Memory (μ PD78363A)

Note Only the external memory can be used in the ROM-less mode.

Caution When a word access to the main RAM area (FE00H to FEFFH) (containing stack manipulation) is executed, addresses specified in operands are limited to even addresses.

Figure 3-12. Addressing Space of Data Memory (μ PD78P368A)

Caution When a word access to the main RAM area (FE00H to FEFFH) (containing stack manipulation) is executed, addresses specified in operands are limited to even addresses.

3.3.1 General-purpose register addressing

(1) Implied addressing

The instruction automatically addresses the register that functions as an accumulator (A or AX) or loop counter (B or C) assigned to the general-register area.

Coding example MULU r

If the value stored in register B is used as a multiplier for a multiply instruction of 8 bits by 8 bits, write the following. The instruction performs multiply operation between the accumulator (register A) and register B and stores the result in the 16-bit accumulator (register AX).

MULU B ; $AX \leftarrow A \times B$

(2) Register addressing

The instruction directly addresses the desired registers.

Coding example ADD r, r

To specify registers D and E storing the target values for the 8-bit add instruction, write the following:

ADD D, E ; $D \leftarrow D + E$

3.3.2 Short direct addressing

This addressing is used for accessing the internal RAM area at addresses FE20H to FEFFH and the SFR area at addresses FF00H to FF1FH. Short direct addressing enables high-speed access to these areas by a short instruction code.

Specify an even address when manipulating 16-bit data.

Coding example ADD A, saddr

If one target value of the 8-bit add instruction is already stored in the location at address FE80H in internal data memory, code the following:

ADD A, 0FE80H ; $A \leftarrow A + (FE80H)$

3.3.3 Special function register (SFR) addressing

This addressing is used for manipulating SFRs mapped in the SFR area at addresses FF00H to FFFFH.

Coding example MOV A, sfr

If a special function register is specified as a transfer source for port 0 in the SFR area, write the following 8-bit transfer instruction:

MOV A, P0 ; $A \leftarrow P0$

CHAPTER 4 SUMMARY OF BLOCK FUNCTION

4.1 Execution Unit

The execution unit (EXU) controls address calculation, arithmetic/logical operations, and data transfer by a microprogram.

The EXU contains 256-byte main RAM. Eight register banks are addressed to the main RAM in the EXU.

4.2 Bus Control Unit

The bus control unit (BCU) activates a required bus cycle according to the physical address obtained from the execution unit (EXU). When the EXU does not issue a bus cycle activation request, the BCU generates an address required for prefetching an instruction. The prefetched instruction code is fetched into the instruction queue.

The number of bytes held in the instruction queue depends on the area from which the instruction is fetched.

- Fetched from internal memory^{Note}: 5 bytes
- Fetched from external memory: 3 bytes

Note Internal memory: Internal ROM (only in the μ PD78363A, 78366A, 78368A, and 78P368A), peripheral RAM

4.3 Program Memory and Data Memory

The program memory and data memory capacities depend on the product.

The μ PD78363A contains a 24K-byte program memory (ROM) and a 768-byte data memory (RAM).

The μ PD78366A contains a 32K-byte program memory (ROM) and a 2048-byte data memory (RAM).

The μ PD78368A contains a 48K-byte program memory (ROM) and a 2048-byte data memory (RAM).

However, the μ PD78365A does not contain ROM and contains a 2048-byte data memory (RAM) only.

The μ PD78P368A contains a 48K-byte program memory (ROM) and a 2048-byte data memory (RAM).

The 2048-byte (768-byte for μ PD78363A) data memory consists of 256-byte main RAM contained in the execution unit (EXU) and 1792-byte (512-byte for μ PD78363A) peripheral RAM.

★

4.4 Ports

A port has a function as a general-purpose port and a function of control pins as shown in the following table.

Port Name	I/O	Compound Function	
Port 0	8-bit I/O	General-purpose port	Real-time output port, RPU control signal input, PWM signal output
Port 1	8-bit I/O		—
Port 2	6-bit input		External interrupt input, RPU count pulse input
Port 3	7-bit I/O		Serial interface I/O
Port 4	8-bit I/O		Address/data bus (AD0-AD7)
Port 5	8-bit I/O		Address bus (A8-A15)
Port 7	8-bit input		A/D converter analog input
Port 8	6-bit I/O		RPU timer output
Port 9	4-bit I/O		External access control signal output

Remark RPU: real-time pulse unit

4.5 Real-Time Pulse Unit

The real-time pulse unit (RPU) can output a programmable pulse and measure a pulse width and frequency. It also controls the output timing of the real-time output port.

RPU consists of the following hardware.

- 16-bit timer $\times 5$
- 16-bit compare registers $\times 7$
- 16-bit capture registers $\times 3$
- 16-bit capture/compare register $\times 2$

4.6 Real-Time Output Port

This port controls the output timing of ports by using a signal sent from the RPU as a trigger, and can output data in 4-bit units. It is multiplexed with port 0 (P00 to P03) and can output four real-time pulses.

In addition, the output of P00 to P03 can be modulated for PWM.

4.7 A/D Converter

This 10-bit A/D converter has eight analog input pins and is of successive approximation type with a high operating speed and resolution.

4.8 Serial Interface

The following two channels of serial interfaces, each independent of the other, are provided. In addition, a baud rate generator that can be commonly used with the two channels is also provided. The clocked serial interface can operate in two operation modes.

- Asynchronous serial interface (UART) (with pin select function)
- Clocked serial interface
 - Serial bus interface mode (SBI mode)
 - Three-wire serial I/O mode

4.9 PWM Output Unit

The μ PD78366A has two PWM signal outputs of 8-/9-/10-/12-bit resolution. By externally connecting a low-pass filter, a PWM output can be used as a digital-to-analog conversion output. The PWM outputs are most suitable, for example, for a control signal for the actuator of a motor.

4.10 Watchdog Timer

The watchdog timer is a free-running counter with a nonmaskable interrupt function designed to prevent crashes or deadlocks. A program error can be detected when a watchdog timer overflow interrupt (INTWDT) is generated or when the watchdog timer output pin ($\overline{\text{WDTO}}$) goes low. By connecting this output to the $\overline{\text{RESET}}$ pin, abnormal application system operation caused by a program error can be prevented.

4.11 Interrupt Controller

The interrupt controller processes various interrupt requests (NMI and INTP0 to INTP4) issued from peripheral hardware and external device with the vectored interrupt, macro service, or context switching. The interrupt controller also allows programmable specification of the 4-level interrupt priority by software.

[MEMO]

CHAPTER 5 PORT FUNCTIONS

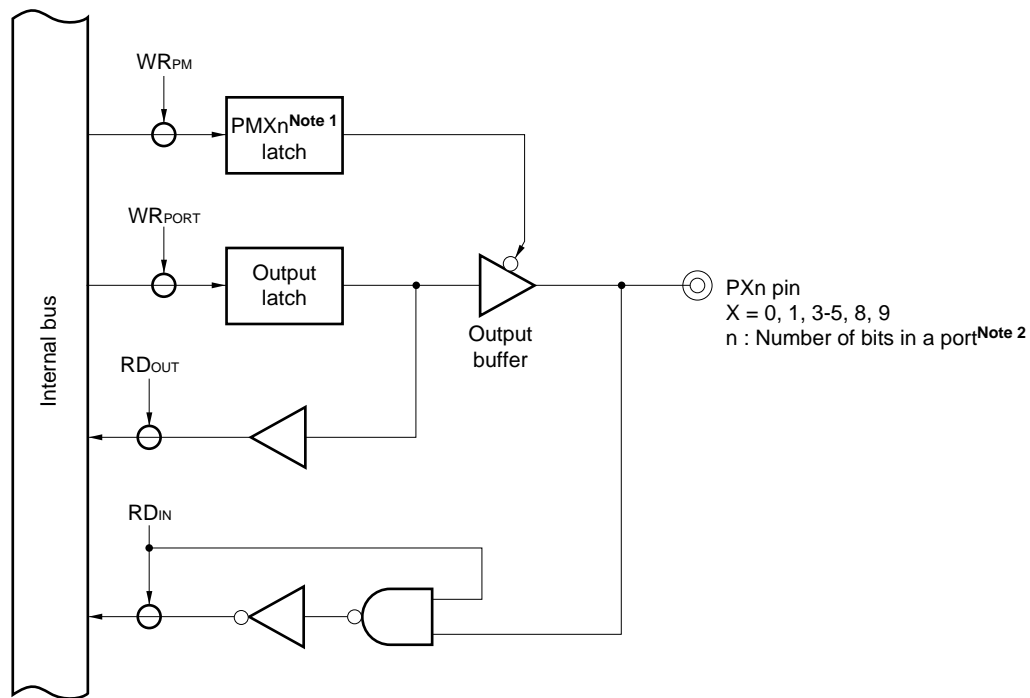
5.1 Hardware Configuration

As shown in Figure 5-1, three-state bidirectional ports are basically used for the ports of the μ PD78366A. (Ports 2 and 7 are used only for input).

A $\overline{\text{RESET}}$ input signal sets each bit of a port mode register to 1, specifying the port as an input port. All the port lines go into a high-impedance state. The contents of the output latch become undefined by a $\overline{\text{RESET}}$ input signal.

When using the port as an output port, set data in the output latch before specifying the port as an output port.

Caution In the μ PD78365A, μ PD78363A, 78366A, and 78368A in the ROM-less mode, ports 4, 5, and 9 (lower 2 bits) do not function as ports. For details, refer to 5.2 Port Functions.

Figure 5-1. Basic I/O Port Configuration

Notes 1. PMXn latch: Bit n in the port mode register PMX ($X = 0, 1, 3, 5, 8, 9$)

2. When $X = 0, 1, 4, 5$, $n = 0$ to 7

When $X = 3$, $n = 0$ to 6

When $X = 8$, $n = 0$ to 5

When $X = 9$, $n = 0$ to 3

Remarks 1. Ports 2 and 7 are input ports.

2. Port 4 can be set in input or output mode by memory expansion mode register (MM).

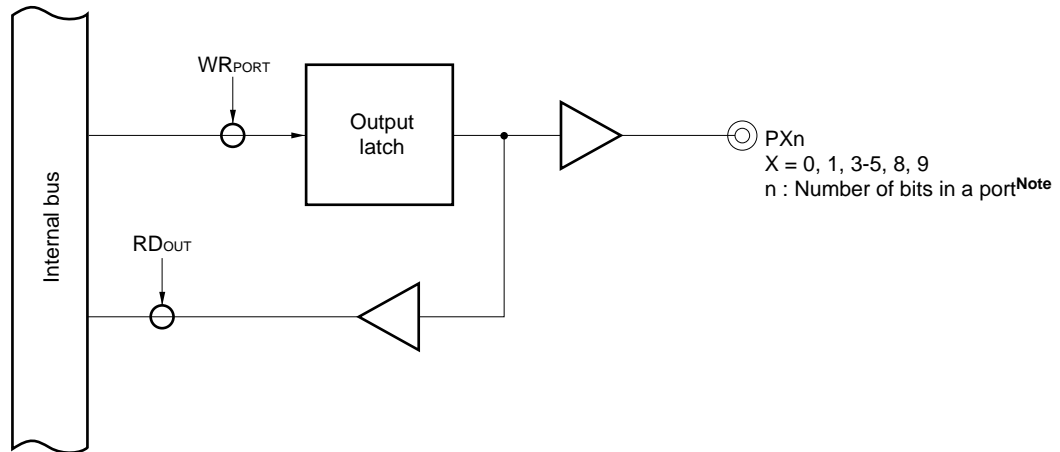
(1) When a port is specified as an output port

The output latch is enabled, and a transfer instruction can transfer data between the output latch and accumulator.

The contents of the output latch can be set or reset bit by bit. Data once written to the output latch are held until another instruction is executed to operate the port.

When a port specified as an output port is read using an instruction such as a transfer instruction, the contents of the output latch are read.

Figure 5-2. Port Specified as Output Port



Note When $X = 0, 1, 4, 5$, $n = 0$ to 7
 When $X = 3$, $n = 0$ to 6
 When $X = 8$, $n = 0$ to 5
 When $X = 9$, $n = 0$ to 3

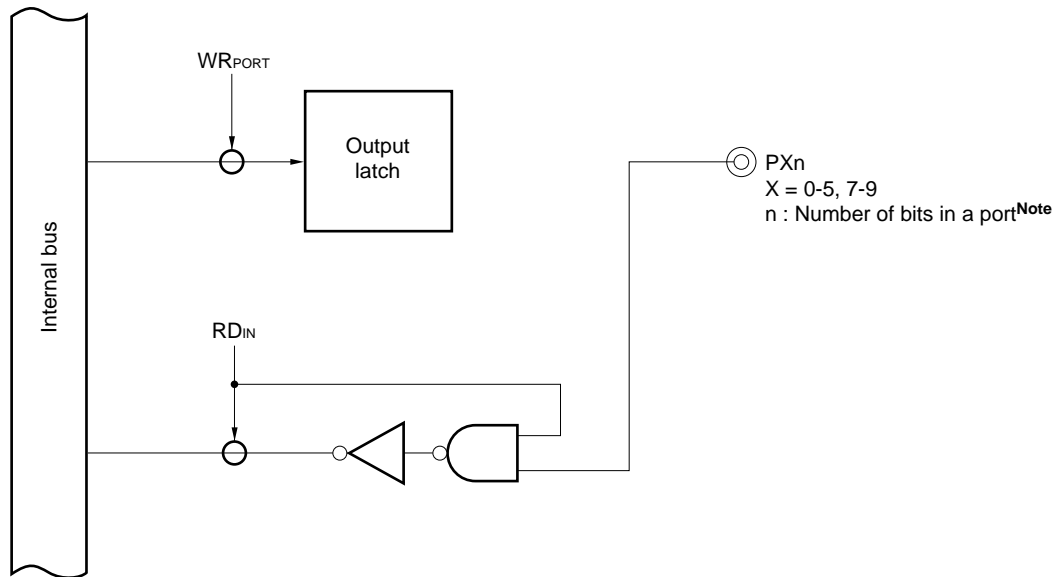
(2) When a port is specified as an input port

The levels of the port pins can be loaded into the accumulator with a transfer instruction. Even when the port is specified as an input port, writing to the output latch is possible. Data transferred from the accumulator with a transfer instruction are all stored in the output latch, regardless of whether the port is specified as an input port or output port.

However, the data is not output to the port pins because the output buffer for the bits have become high-impedance. (When the bits specified as an input are switched to an output port, the contents of the output latch are output to the port pins).

The contents of the output latch of the bits specified as an input port cannot be loaded into the accumulator (refer to **Figure 5-3**).

Figure 5-3. Port Specified as Input Port

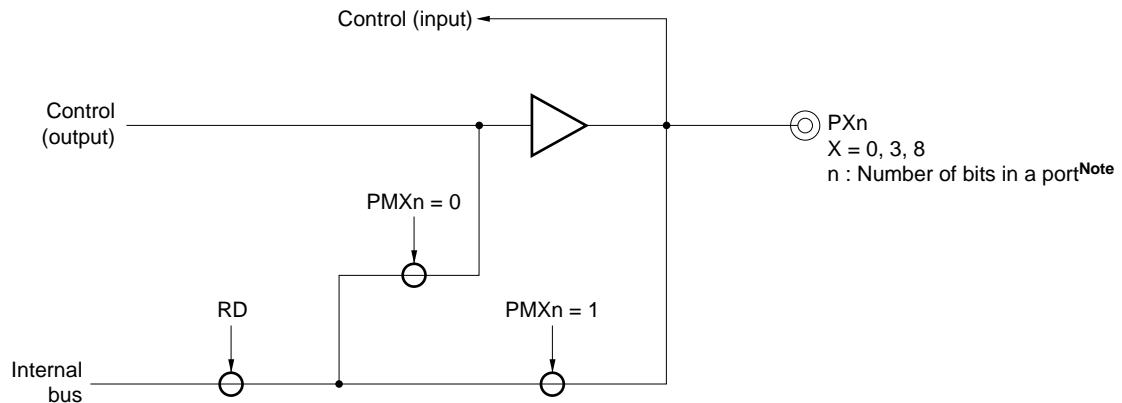


Note When $X = 0, 1, 4, 5, 7$, $n = 0$ to 7
 When $X = 2, 8$, $n = 0$ to 5
 When $X = 3$, $n = 0$ to 6
 When $X = 9$, $n = 0$ to 3

(3) When a port is specified as a control signal input or output

Ports 0, 3, and 8 can be used to input or output control signals on a bit-by-bit basis by setting the desired bit(s) of the corresponding port mode control register (PMC0, PMC3, or PMC8) to 1. In this case, the setting of the port mode register (PM0, PM3, or PM8) has no effect.

When a pin or pins are used for a control signal, the state of the control signal can be checked by executing a port read instruction.

Figure 5-4. When Control Is Specified

Note When $X = 0$, $n = 0$ to 7
 When $X = 3$, $n = 0$ to 6
 When $X = 8$, $n = 0$ to 5

(a) When port outputs control signals

The pin state of a control signal can be read by executing a port read instruction when the desired bit of the port mode register is set to 1.

The state of an internal control signal can be read by executing a port read instruction when the desired bit of the port mode register is reset to 0.

(b) When port inputs control signals

The pin status of a control signal can be read by executing a port read instruction only when the corresponding bit of the port mode register is set to 1.

When the desired bit of the port mode register is reset to 0, if a port read instruction is executed, 0 is always read.

Caution The pins functioning as input pins in the control mode may operate erroneously if the corresponding bits of the port mode control register are rewritten during the operation. Therefore, write into the port mode control register when the system is initialized, etc. Do not rewrite the bits dynamically during the operation.

Table 5-1. Read Operation in Control Mode (1/2)

Pin Name	Control Function (I/O)	PMXn	Read Operation
P00-P03	RTP0-RTP3 (O)	1	Pin status
		0	Internal control signal
P04	PWM0 (O)	1	Pin status
		0	Internal control signal
P05	CMD = 0 : PWM1 (O)	1	Pin status
		0	Internal control signal
	CMD = 1 : TCUD (I)	1	Pin status
		0	Internal control signal (PWM1 signal)
P06	CMD = 0 : TO40 (O)	1	Pin status
		0	Internal control signal
	CMD = 1 : TIUD (I)	1	Pin status
		0	Internal control signal (TO40 signal)
P07	CMD = 0 : "0" (O)	1	Pin status (fixed to "0")
		0	Internal control signal (fixed to "0")
	CMD = 1 : TCLRUD (I)	1	Pin status
		0	Internal control signal (fixed to "0")
P30	SPS = 0 : TxD0 (O)	1	Pin status
		0	Internal control signal
	SPS = 1 : "1" (O)	1	Pin status (fixed to "1")
		0	Internal control signal (fixed to "1")
P31	SPS = 0 : RxD1 (I)	1	Pin status
		0	Internal control signal (fixed to "0")
	SPS = 1 : "Hi-Z"	1	Pin status (high impedance)
		0	Internal control signal (fixed to "0")
P32	MOD1 = 1, MOD2 = 0 : SB0 (I/O)	1	Pin status
		0	Internal control signal
	MOD1 = MOD2 = 1 : "Hi-z"	1	Pin status (high impedance)
		0	Internal control signal
	MOD1 = 0, MOD2 = x : SO (O)	1	Pin status
		0	Internal control signal

- Remarks**
1. PMXn : port mode register (X = 0 or 3, n: number of bits of port)
 2. CMD : bit 7 of TUM1 register SPS : bit 1 of ASIM register
MOD1 : bit 3 of CSIM register MOD2 : bit 4 of CSIM register
 3. x: don't care

Table 5-1. Read Operation in Control Mode (2/2)

Pin Name	Control Function (I/O)	PMXn	Read Operation
P33	MOD1 = 1, MOD2 = 0 : "Hi-Z"	1	Pin status (high impedance)
		0	Internal control signal
	MOD1 = MOD2 = 1 : SB1 (I/O)	1	Pin status
		0	Internal control signal
	MOD1 = 0, MOD2 = × : SI (I)	1	Pin status
		0	Internal control signal
P34	$\overline{\text{SCK}}$ (I/O)	1	Pin status
		0	Internal control signal
P35	SPS = 0 : "1" (O)	1	Pin status (fixed to "1")
		0	Internal control signal (fixed to "1")
	SPS = 1 : TxD1 (O)	1	Pin status
		0	Internal control signal
P36	SPS = 0 : "Hi-Z"	1	Pin status (high impedance)
		0	Internal control signal (fixed to "0")
	SPS = 1 : RxD1 (I)	1	Pin status
		0	Internal control signal (fixed to "0")
P80-P85	TO00-TO05 (O)	1	Pin status
		0	Internal control signal

Remarks 1. PMXn : port mode register (X = 3 or 8, n: number of bits of port)

2. MOD1 : bit 3 of CSIM register

MOD2 : bit 4 of CSIM register

SPS : bit 1 of ASIM register

3. ×: don't care

(4) Port output data check function

The μ PD78366A has a function that enables pin state to be read in the port output mode so that application system reliability can be improved (pin access mode). With this function, output data and actual pin state can be checked as required. If a mismatch is found, an action such as replacement with another system can be taken.

Before pin state can be read, bit 0 of the port read control register (PRDC) must be set to 1.

A $\overline{\text{RESET}}$ input signal sets register PRDC to 00H.

Figure 5-5. Format of Port Read Control Register

Symbol	7	6	5	4	3	2	1	0	Address	When reset	R/W
PRDC	0	0	0	0	0	0	0	PRDC0	FF62H	00H	R/W

PRDC0	Operation mode
0	Normal mode
1	Pin access mode

Caution Bits 7 to 1 of the PRDC register are fixed to “0” by hardware. Even if “1” is written to them, they remain “0”.

Example A sample program is given below which checks output data to port 0 (P0), port 1 (P1), and port 3 (P3) by using the pin access mode.

```

TEST    DI                      ; Disables interrupt.
        MOV    A,#5AH           ; Test data = 5AH
        MOV    P0,A             ; Sets 5AH in output latch.
        MOV    P1,A
        MOV    P3,A
        SET1   PRDC.0           ; Sets pin access mode (sets PRDC).
        CMP    A,P0             ; Compares pin level with output latch contents.
        BNE    $ERR0           ; Performs error processing if mismatch occurs.
        CMP    A,P1
        BNE    $ERR1
        CMP    A,P3
        BNE    $ERR3
        CLR1   PRDC.0           ; Normal mode (resets PRDC)
        EI                      ; Enables interrupt.

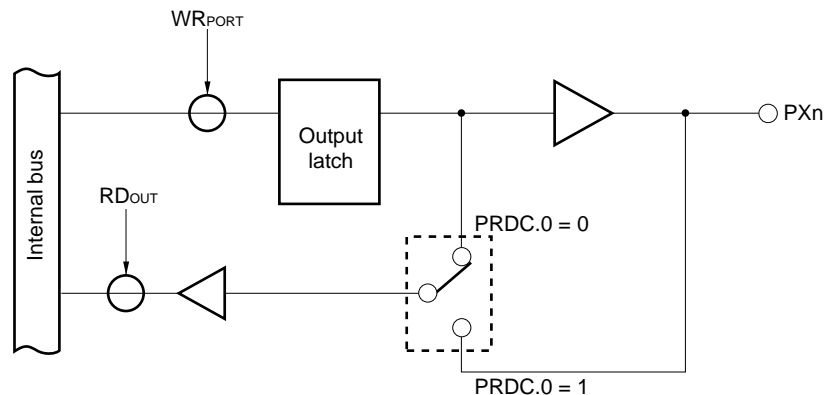
```

- Cautions**
1. In the pin access mode ($PRDC0 = 1$), no bit manipulation instruction for a port operates normally. After a port check is completed, be sure to reset the mode to the normal mode ($PRDC0 = 0$).
 2. If an interrupt occurs in the pin access mode, a bit manipulation instruction may be executed in the same mode. This will cause an error. Before starting a check, be sure to set the DI state.
In addition, do not use macro services that manipulate ports.
 3. Non-maskable interrupts are unavoidable. So, the following provisions should be made in the program as required:
 - The non-maskable interrupt routine is to perform no port manipulation.
 - The level of $PRDC.0$ is to be saved at the start of the non-maskable interrupt routine, then is restored when control is returned.
 4. The pin access mode is a function to access the pin status of an output port. If a pin set in the input port mode ($PMXn = 1$) is read in the pin access mode ($PRDC0 = 1$), “0” is read regardless of the input level.

When $PRDC.0$ is set to 1, the switch enclosed in dotted lines in the figure on the next page is connected to the pin and the pin state is read. If a bit manipulation instruction is executed in this state, the pin state is read and a bit operation is executed. This may adversely affect the contents of the output latch.

When $PRDC.0$ is reset to 0, the system enters the normal mode.

Figure 5-6. Control (output port specified)



In addition, instructions (CHKL, CHKLA) dedicated to frequent port state checking are available. By EXCLUSIVE ORing, these instructions compare the pin state with the contents of the output latch (in the port mode) or the pin state with the level of the internal control output signal (in the control mode).

Example A sample program is given below which checks pin state and the contents of an output latch using instructions CHKL and CHKLA.

```

TEST:   SET1   P0.3      ; Sets bit 3 of port 0.
        CHKL   P0        ; Checks port 0.
        BNE    $ERR1     ; Branches to error processing (ERR1)
                               if mismatch with output latch contents occurs.
        .
        .
        .
ERR1:   CHKLA  P0        ; Checks incorrect bits.
        BT     A.7,$BIT07 ; Bit 7?
        BT     A.6,$BIT06 ; Bit 6?
        .
        .
        .
        BT     A.1,$BIT01 ; Bit 1?
        BR     $BIT00     ; Bit 0 is incorrect if all above bits are correct.

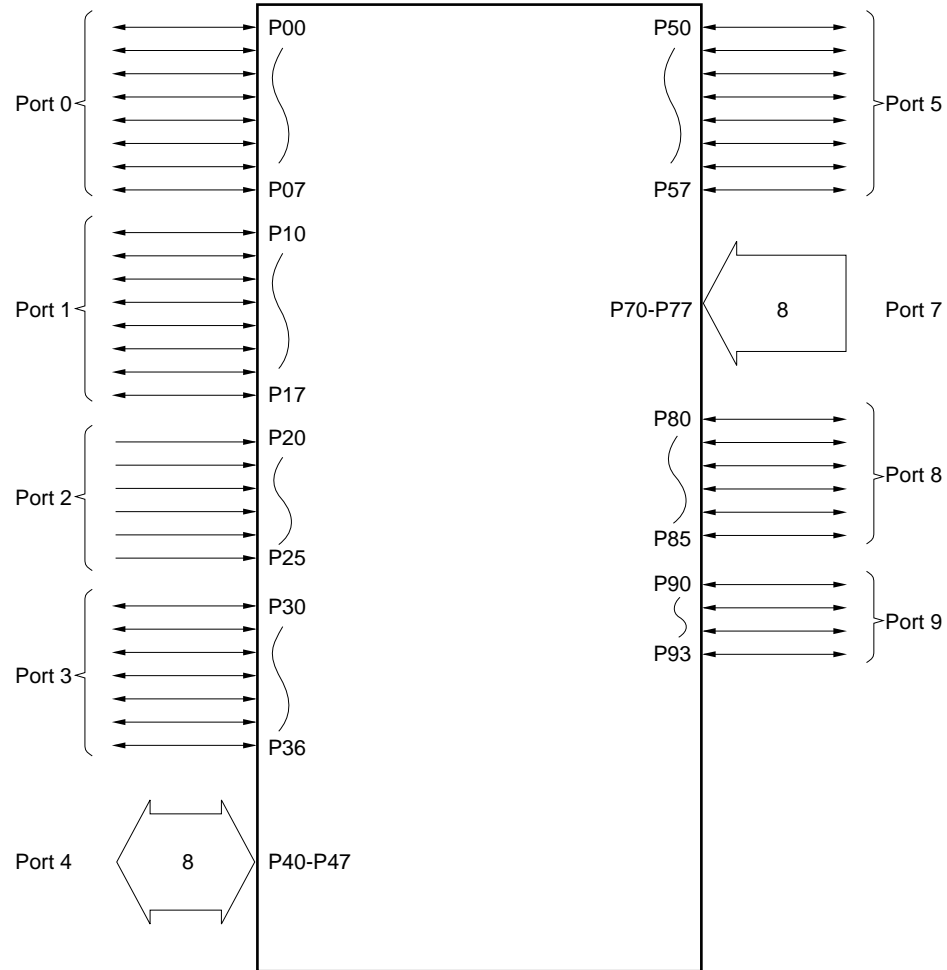
```

- Cautions**
1. Use the CHKL and CHKLA instructions only when the PRDC0 bit of the PRDC register is set to 0 (normal mode).
 2. In the case of those input/output port pins that are set in the input port mode, the results of the CHKL or CHKLA instructions always match regardless of the setting of the port/control mode. In the case of the dedicated input port, because it is not provided with an output latch, the input pin level is read when the CHKL or CHKLA instruction is executed. Therefore, the CHKL or CHKLA instruction is actually invalid for the dedicated input port and these instructions should not be used.
 3. If the CHKL or CHKLA instruction is executed with port 4 set to the input port mode or expansion mode, a mismatch may be generated (a mismatch is generated if the pin level changes during execution of the CHKL or CHKLA instruction). Therefore, while port 4 is set to the input port mode or expansion mode, do not execute these instructions.
 4. Set control output pins to the input mode before executing the CHKL or CHKLA instruction to check the output level of the pin of a port where control and port output pins are used together. (The output level of a control pin cannot be checked with the CHKL or CHKLA instruction because the output level varies asynchronously).

5.2 Port Functions

The μ PD78366A is provided with the ports shown in Figure 5-7 and can perform various control operations.

Figure 5-7. Port Configuration



5.2.1 Functions and features of I/O ports

Table 5-2 lists the I/O ports. These ports function not only as I/O ports but also as I/O pins of the internal hardware.

Table 5-2. Functions and Features of Ports

Port Name	Port Function	Compound Function
Port 0	8-bit I/O port. Can be set in input or output mode in 1-bit units	Inputs control signals of real-time output port (RTP) and real-time pulse unit (RPU), and outputs PWM signal in control mode
Port 1	8-bit I/O port. Can be set in input or output mode in 1-bit units	—
Port 2	6-bit input port	Inputs external interrupt and count pulse input to real-time pulse unit (RPU)(fixed in control mode)
Port 3	7-bit I/O port. Can be set in input or output mode in 1-bit units	I/O of serial interface (UART, CSI) in control mode
Port 4	8-bit I/O port. Can be set in input or output mode in 8-bit units	Address/data bus (AD0-AD7) when memory is expanded
Port 5	8-bit I/O port. Can be set in input or output mode in 1-bit units	Address bus (A8-A15) when memory is expanded
Port 7	8-bit input port	Analog input of A/D converter (fixed in control mode)
Port 8	6-bit I/O port. Can be set in input or output mode in 1-bit units	Timer output of real-time pulse unit (RPU) in control mode
Port 9	4-bit I/O port. Can be set in input or output mode in 1-bit units	Outputs control signals when memory is expanded

5.2.2 I/O mode setting**(1) Port n (n = 0, 1, 3, 5, 8, 9)**

The I/O mode of each port is set bit by bit with a port mode register (PM) (refer to **Figures 5-8 to 5-13**). Each pin of a port functions as an input or output port, depending on the setting of the corresponding bit in the PM register. It is an input port when the bit is one, and an output port when the bit is zero.

The contents of each PM register are specified by an 8-bit manipulation instruction.

(2) Ports 2 and 7

Ports 2 and 7 function only as input ports. These ports have no port mode register.

(3) Port 4

The I/O mode of port 4 only is set with memory expansion mode register (MM) in units of eight bits (refer to **Figure 5-17**).

Register MM is set with a bit or 8-bit manipulation instruction.

Figures 5-8 to 5-13 show the format of each port mode instruction register.

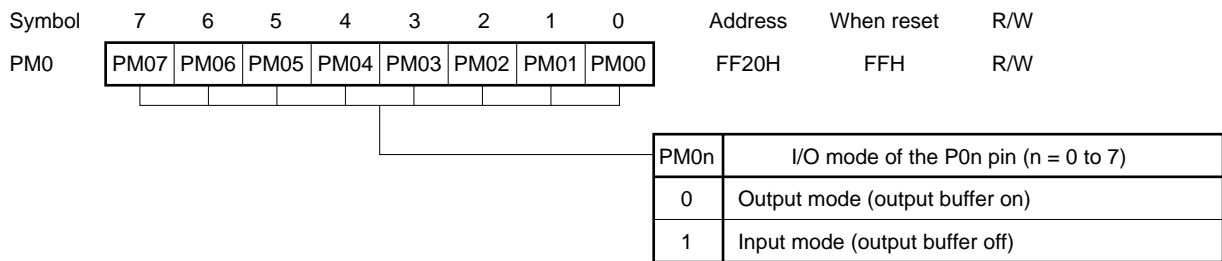
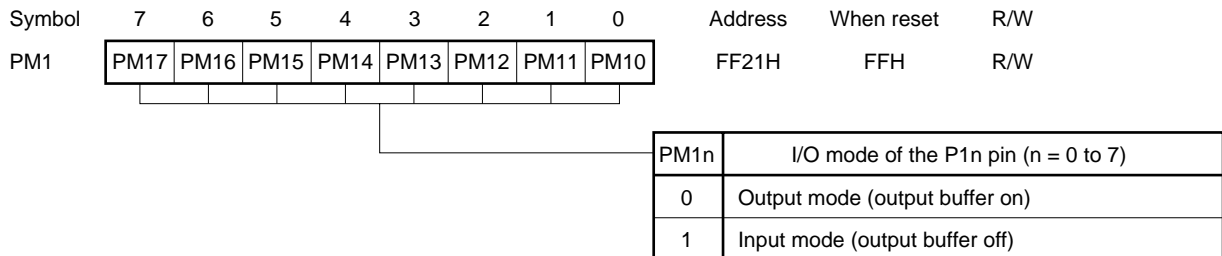
Figure 5-8. Format of Port 0 Mode Register**Figure 5-9. Format of Port 1 Mode Register**

Figure 5-10. Format of Port 3 Mode Register

Symbol	7	6	5	4	3	2	1	0	Address	When reset	R/W
PM3	×	PM36	PM35	PM34	PM33	PM32	PM31	PM30	FF23H	×111 1111B	R/W

PM3n	I/O mode of P3n pin (n = 0 to 6)
0	Output mode (output buffer on)
1	Input mode (output buffer off)

Remark ×: don't care**Figure 5-11. Format of Port 5 Mode Register**

Symbol	7	6	5	4	3	2	1	0	Address	When reset	R/W
PM5	PM57	PM56	PM55	PM54	PM53	PM52	PM51	PM50	FF25H	FFH	R/W

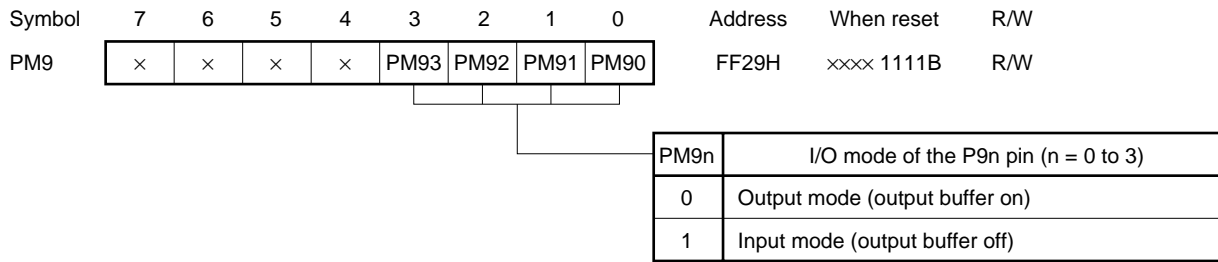
PM5n	I/O mode of the P5n pin (n = 0 to 7)
0	Output mode (output buffer on)
1	Input mode (output buffer off)

Figure 5-12. Format of Port 8 Mode Register

Symbol	7	6	5	4	3	2	1	0	Address	When reset	R/W
PM8	×	×	PM85	PM84	PM83	PM82	PM81	PM80	FF28H	××11 1111B	R/W

PM8n	I/O mode of the P8n pin (n = 0 to 5)
0	Output mode (output buffer on)
1	Input mode (output buffer off)

Remark ×: don't care

Figure 5-13. Format of Port 9 Mode Register

Remark ×: don't care

5.2.3 Control mode setting

(1) Port n (n = 0, 3, 8)

The control mode of each port is set with the corresponding port mode control register (PMC) bit by bit (refer to **Figures 5-14 to 5-16**).

Each pin of a port functions as a control pin when the corresponding bit of the PMC register is set to 1. In this case, the previous states of the port and the set value in the PM register have no effect.

The PMC register is set with an 8-bit manipulation instruction.

(2) Ports 2 and 7

The pins of ports 2 and 7 are always set in the control mode. These ports have no port mode control register.

The pin state of each port can be read by executing a port read instruction.

(3) Ports 4 and 5

(a) μ PD78363A, 78366A, and 78368A

The control mode is set according to the contents of the memory expansion mode (MM) register (refer to **Figure 5-17**). As shown in Table 5-3, the operations of ports 4 and 5 depend on the expansion mode of the MM register. When the MM register is in the expansion mode, port 4 functions as an address/data bus and some bits of port 5 function as address bus.

When data is written or verified with the μ PD78P368A (in the PROM programming mode), port 4 functions as I/O port for the data.

The contents of the MM register are specified by a bit manipulation instruction or an 8-bit manipulation instruction.

Table 5-3. Operation of Port 4 and Port 5 (μ PD78363A, 78366A, 78368A)

Setting of MM register		Operation of port 4	Operation of port 5							
		P40 - P47	P50	P51	P52	P53	P54	P55	P56	P57
Port mode (single-chip mode)		Input port	General-purpose I/O port							
		Output port								
Expansion mode	256-byte expansion	Address/data bus	A8	A9	A10	A11	General-purpose I/O port			
	4K-byte expansion						A12	A13	General-purpose I/O port	
	16K-byte expansion								A14	A15
	Full expansion ^{Note}									

Note 32K-byte expansion in case of μ PD78363A

Remark An (n = 8 to 15): Address bus

(b) μ PD78365A

Ports 4 and 5 operate as follows. Each pin of the ports does not function as a port pin.

- Port 4: Address/data bus (AD0 to AD7)
- Port 5: Address bus (A8 to A15)

The same applies when the μ PD78363A, 78366A, and 78368A are used in the ROM-less mode (MODE0, 1 = HH).

(4) Port 9

The control mode is set according to the statuses of the MODE0 and 1 pins and the contents of the memory expansion mode (MM) register.

(a) μ PD78363A, 78366A, and 78368A

Port 9 operates as shown in Table 5-4.

When the MM register is in the expansion mode, P90 functions as the $\overline{\text{RD}}$ pin and P91 functions as the $\overline{\text{WR}}$ pin. When the μ PD78363A, 78366A, and 78368A are set in the ROM-less mode (MODE0, 1 = HH), P90 and P91 always function as $\overline{\text{RD}}$ and $\overline{\text{WR}}$, respectively. In this case, the setting of the MM0-MM2 bits is invalid.

Table 5-4. Operation of Port 9 (μ PD78363A, 78366A, 78368A)

MODE0	MODE1	Setting of MM Register	Port 9 Operation	
		MM0-MM2	P90	P91
L	L	Port mode	General-purpose port	
		Expansion mode	$\overline{\text{RD}}$	$\overline{\text{WR}}$
H	H	—	$\overline{\text{RD}}$	$\overline{\text{WR}}$

(b) μ PD78365A

The P90 and P91 pins always function as the $\overline{\text{RD}}$ and $\overline{\text{WR}}$ pins.

Fix both the MODE0 and MODE1 pins to the high level.

The setting of the lower 3 bits (MM0-MM2) of the MM register is invalid.

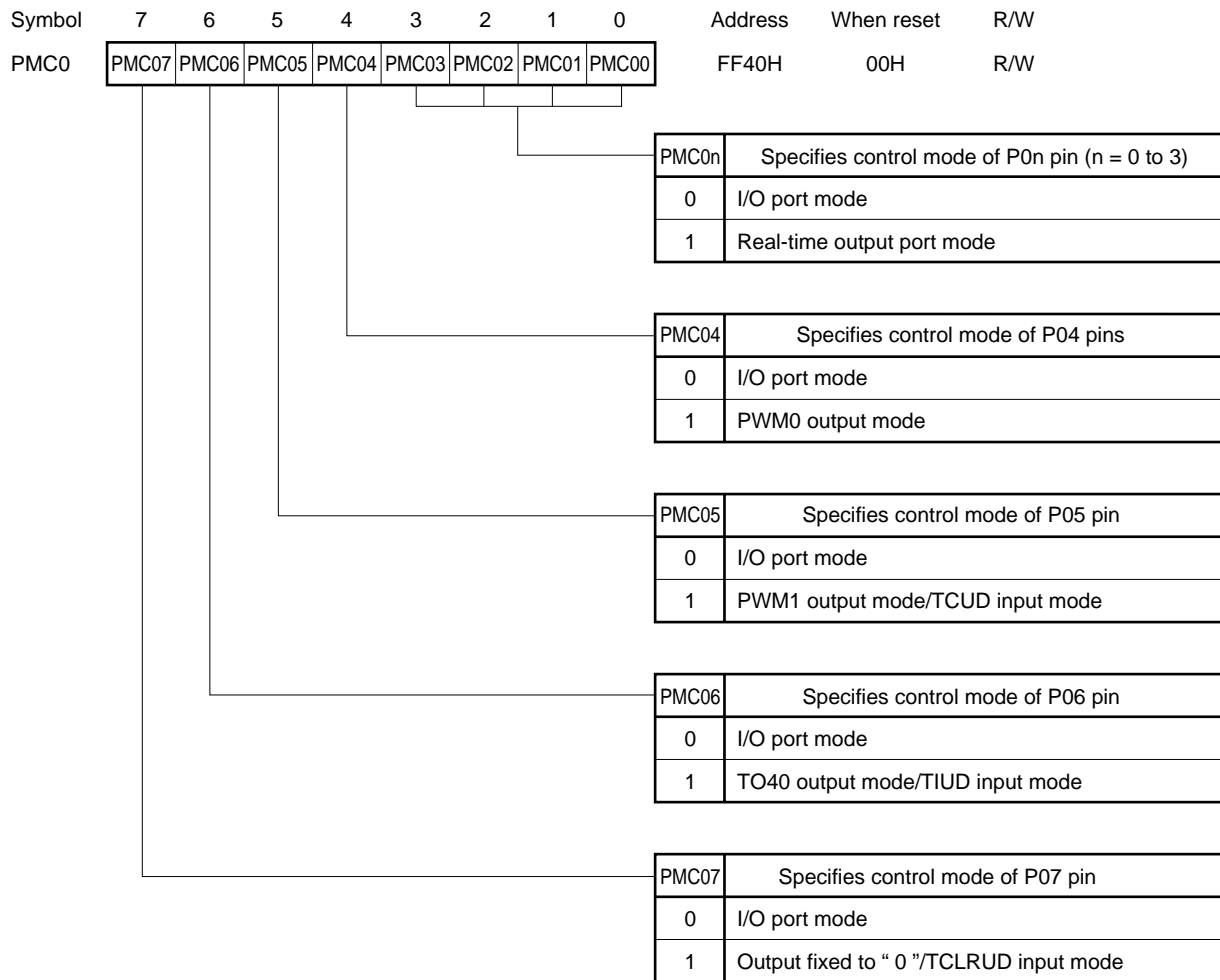
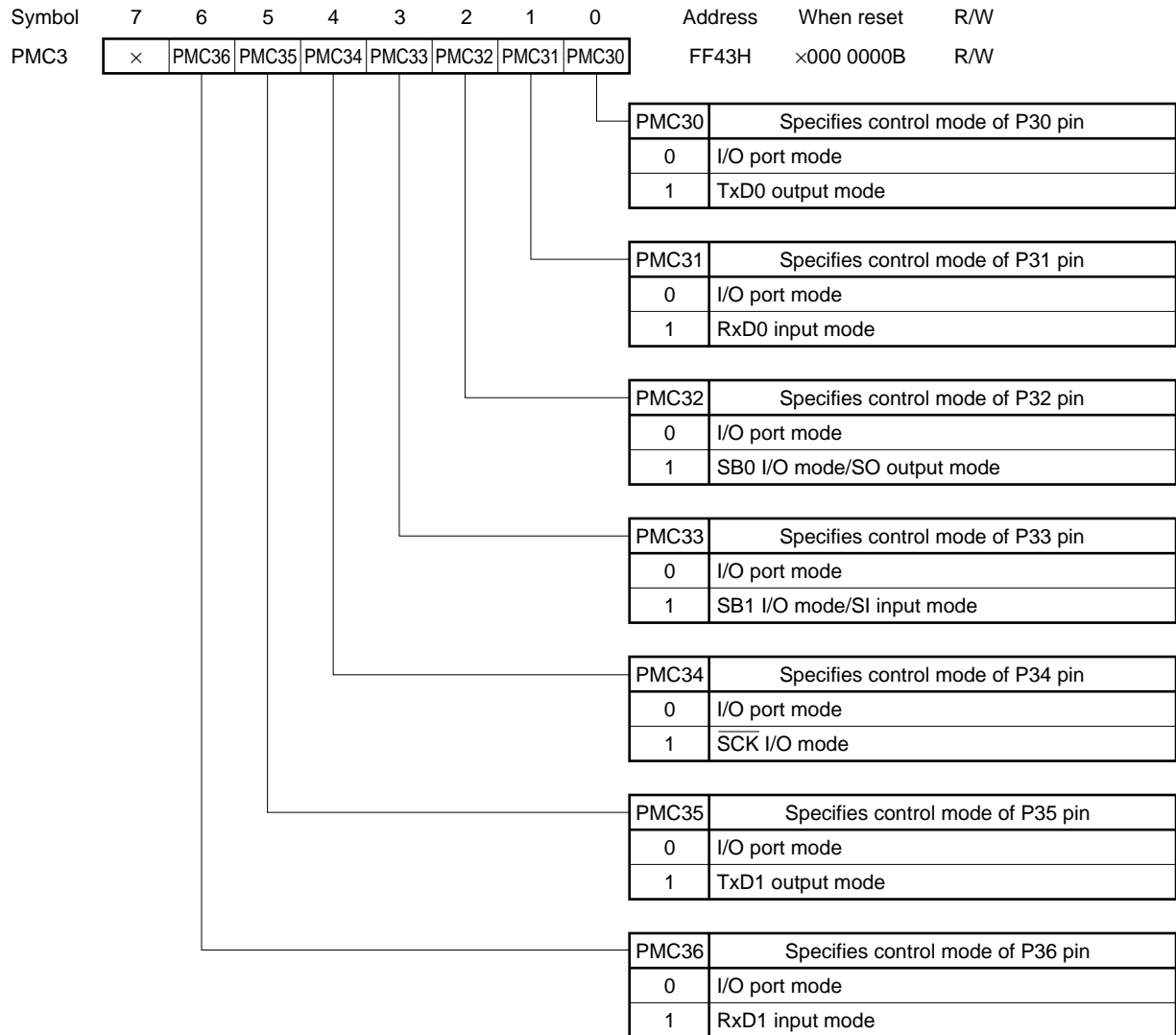
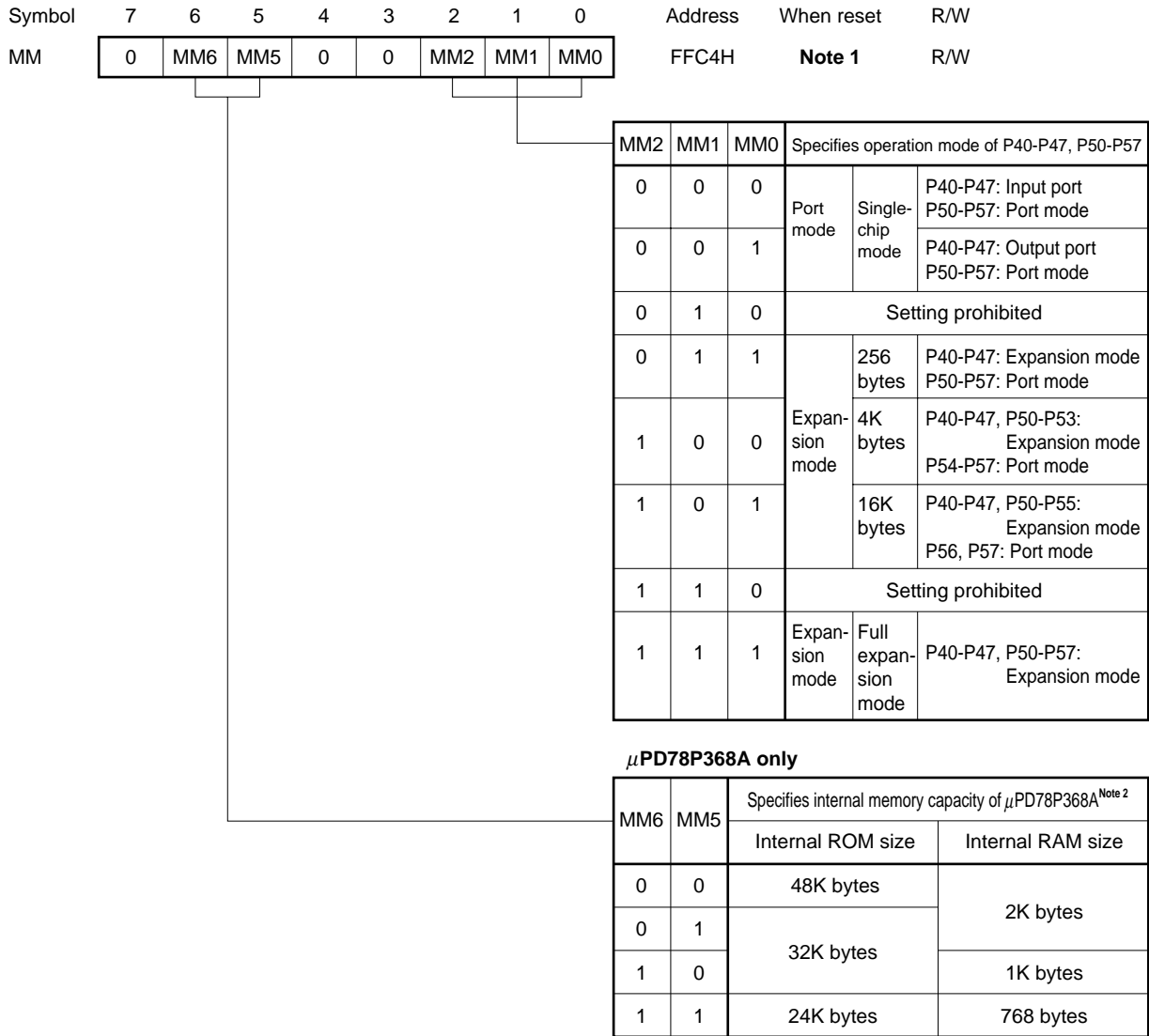
Figure 5-14. Format of Port 0 Mode Control Register

Figure 5-15. Format of Port 3 Mode Control Register**Remark** ×: don't care

Symbol	7	6	5	4	3	2	1	0	Address	When reset	R/W
PMC8	×	×	PMC85	PMC84	PMC83	PMC82	PMC81	PMC80	FF48H	××00 0000B	R/W

PMC8n	Specifies control mode of P8n pin (n = 0 to 5)
0	I/O port mode
1	TOn output mode

Remark \times : don't care

Figure 5-17. Format of Memory Expansion Mode Register

Notes 1. The value of the MM register after a reset depends on the product.

μPD78363A...60H

μPD78365A, 78366A...20H

μPD78368A, 78P368A...00H

2. A function to change the capacity of the internal memory of the μPD78P368A. With the μPD78363A, 78365A, 78366A, and 78368A, the internal memory is fixed to the status after reset.

Cautions 1. Bits 3, 4, and 7 of the MM register are fixed to 0 by hardware. Even if 1 is written to them, they remain 0.

2. Do not write the setting prohibited combination of codes.

5.2.4 Specifying pull-up resistor

The μ PD78366A is provided with pull-up resistors that can be internally connected to each pin of ports 0-5, 8, and 9 through software (except P20 pin).

(1) Ports 0, 1, 3, and 8

Connection of the pull-up resistor to each pins can be specified by pull-up resistor option registers (PUOL and PUOH), port mode register (PM), and port mode control register (PMC).

When the PMC register is "0" (port mode), set the PUOL and PUOH registers to "1" and the corresponding PM register to "1" (input port mode). The pin set in the input port mode will be connected to an internal pull-up resistor.

The pin set to "1" by the PMC register (control mode) is not connected to the internal pull-up resistor, regardless of the specification of the PUOL, PUOH, and PM registers.

(2) Port 2

When the PUO2 bit of the PUOL register is set to "1", all the five pins of the port (P21-P25) are connected to an internal pull-up resistor (Cannot be specified bit-wise).

Caution The P20/NMI pin does not contain a pull-up resistor on hardware. Therefore, even if PUO2 is set to 1, no internal pull-up resistor is set in the P20/NMI pin.

(3) Port 4

If the PUO4 bit of the PUOL register is set to "1" when port 4 is set in the input mode by the memory expansion mode register (MM), all the eight pins of the port (P40-P47) are connected to an internal pull-up resistor (Cannot be specified bit-wise).

(4) Port 5

If the PUO5 bit of the PUOL register is set to "1" when port 5 is set in the input mode by the memory expansion mode register (MM) and port 5 mode register (PM5), the internal pull-up resistor becomes valid.

(5) Port 9

If the PUO9 bit of the PUOH register is set to "1" when port 9 is set in the input mode by the memory expansion mode register (MM) and port 9 mode register (PM9), the internal pull-up resistor becomes valid.

The internal pull-up resistors of ports 4, 5 and 9 are invalid regardless of the specification by the PM register and PUO bit, when the expansion mode is set by the MM register.

Caution When emulating the μ PD78366A with the IE-78350-R, the internal pull-up resistors of ports 1, 4, 5, and 9 are invalid, even if the PU01, PU04, PU05, and PU09 bits of the PUOL and PUOH registers are set to “1”. To use the pull-up resistor, set the corresponding bit to “1”, and then connect an external pull-up resistor, in order to share software between the IE-78350-R and μ PD78366A.

Figure 5-18. Format of Pull-Up Resistor Option Register L

Symbol	7	6	5	4	3	2	1	0	Address	When reset	R/W
PUOL	0	0	PU05	PU04	PU03	PU02	PU01	PU00	FF44H	00H	R/W

PUOn	Pull-up resistor of port n (n = 0 to 5)
0	Not used with port n
1	Used with port n

Figure 5-19. Format of Pull-Up Resistor Option Register H

Symbol	7	6	5	4	3	2	1	0	Address	When reset	R/W
PUOH	0	0	0	0	0	0	PU09	PU08	FF45H	00H	R/W

PMOn	Pull-up resistor of port n (n = 8, 9)
0	Not used with port n
1	Used with port n

Caution Bits 7 and 6 of the PUOL register and bits 7 to 2 of the PUOH register are fixed to 0 by hardware. Even if 1 is written to them they remain 0.

Remark To enter the STOP mode, 00H should be set in the PUOL/PUOH register to reduce consumption current.

[MEMO]

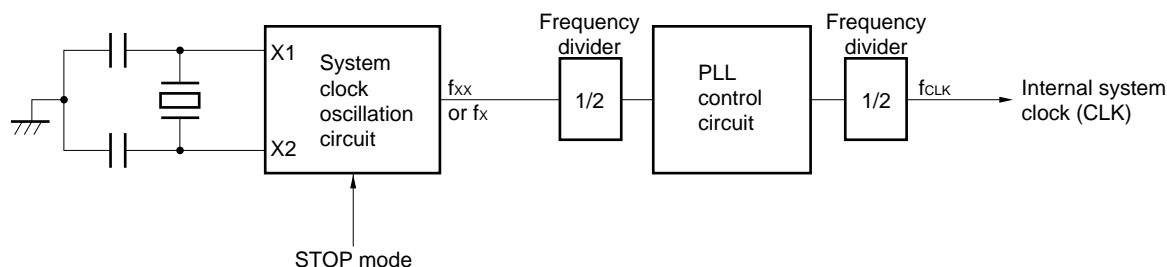
CHAPTER 6 CLOCK GENERATOR

The clock generator generates and controls an internal system clock (CLK) supplied to the CPU.

An internal system clock of the maximum frequency of 16 MHz (f_{CLK}) is generated when an 8-MHz crystal resonator is connected across the X1 and X2 pins of the μ PD78366A.

The clock generator is configured as shown in Figure 6-1.

Figure 6-1. Block Diagram of Clock Generator

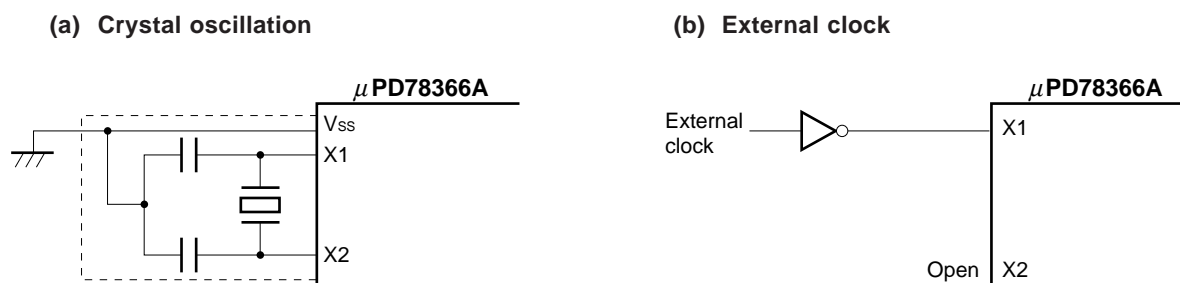


- Remarks**
1. f_{xx} : Crystal resonator frequency
 2. f_x : External clock frequency
 3. f_{CLK} : Internal system clock frequency

The system clock oscillation circuit generates a clock signal with a crystal resonator. The system clock oscillation circuit stops oscillation when it is set to the standby mode (STOP mode). (refer to **CHAPTER 14 STANDBY FUNCTION**).

An external clock can be also applied. In this case, a clock signal is to be applied to the X1 pin. Leave the X2 pin open.

Caution To use the external clock, do not set the STOP mode.

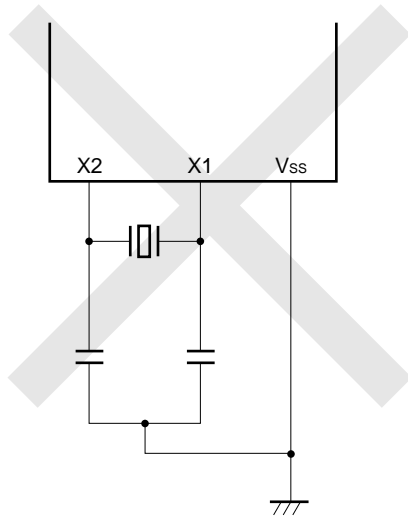
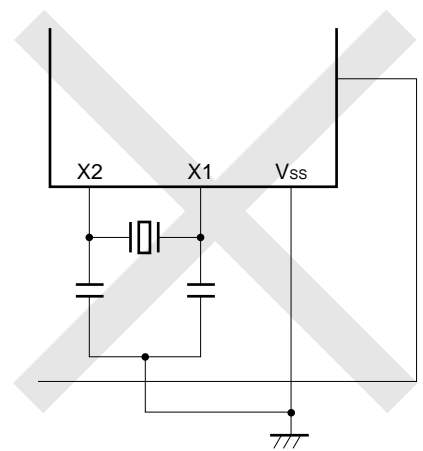
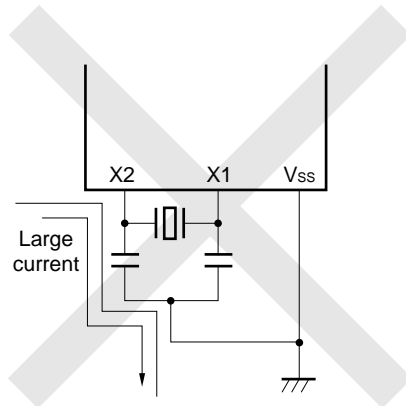
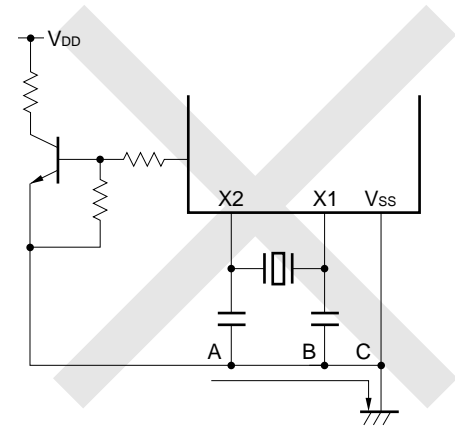
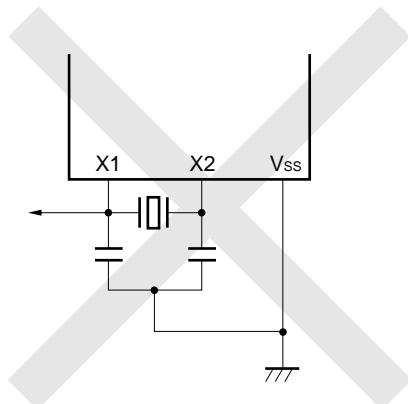
Figure 6-2. External Circuitry of System Clock Oscillation Circuit

Cautions 1. When using a system clock oscillation circuit, wire the portion enclosed by the dotted line in Figure 6-2 as follows to prevent adverse influence 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 in the vicinity of line through which a high alternating current flows.
- Always keep the ground potential for the capacitor in the oscillation circuit at the same potential as Vss. Do not ground the capacitor to a ground pattern through which a high current flows.
- Do not extract any signal from the oscillation circuit.

2. Make sure that a load such as wiring capacitance is not connected to the X2 pin when inputting an external clock.

Figure 6-3 shows examples of wrong resonator connection circuitry.

Figure 6-3. Examples of Wrong Resonator Connection Circuitry**(a) Connection circuit wiring is too long.****(b) There is another signal line crossing.****(c) A high varying current flows near a signal line.****(d) A current flows over the ground line of the oscillation circuit.**
(The potentials of points A, B, and C change).**(e) A signal is extracted.**

[MEMO]

CHAPTER 7 REAL-TIME PULSE UNIT

The real-time pulse unit (RPU) is used to measure pulse intervals or frequency, or to output programmable pulses (six channels of PWM control signals).

The RPU consists of five 16-bit timers: timers 0 through 4. One of these timers is provided with a 10-bit dead time timer and is suitable for inverter control application. In addition, a function to turn off output through software or external interrupt is also provided.

Each timer has the following features:

- Timer 0 ... Controls the PWM cycle of TO00 through TO05 output pins. Also operates as a general-purpose interval timer. The following five operating modes are available for timer 0.
 - General-purpose interval timer mode
 - PWM mode 0 (symmetric triangular wave)
 - PWM mode 0 (asymmetric triangular wave)
 - PWM mode 0 (toothed wave)
 - PWM mode 1
- Timer 1 ... Operates as a general-purpose interval timer.
- Timers 2 & 3 ... Provided with a programmable input sampling circuit that rejects noise superimposed on the input signal, and a capture function.
- Timer 4 ... Operates as a general-purpose timer or an up-down counter. When used as a general-purpose timer, controls the PWM cycle of the TO40 output pin. The following two operating modes are available for timer 4.
 - General-purpose timer mode
 - Up/down counter mode (UDC mode)

7.1 RPU Configuration

Table 7-1 shows the configuration of the RPU.

Table 7-1. Configuration of RPU

	Timer Register	Register	Compare Register Coincidence Interrupt	Capture Trigger	Timer Output	Timer Clear
Timer 0	16-bit timer (TM0)	16-bit compare register (CM00)	—	—	6	INTCM03
		16-bit compare register (CM01)	—			
		16-bit compare register (CM02)	—			
		16-bit compare register (CM03)	INTCM03			
Timer 1	16-bit timer (TM1)	16-bit compare register (CM10)	INTCM10	—	—	INTCM10
Timer 2	16-bit timer (TM2)	16-bit capture/compare register (CC20)	INTCC20	INTP3	—	INTCC20
		16-bit capture register (CT20)	—			
Timer 3	16-bit timer (TM3)	16-bit capture/compare register (CC30)	INTCC30	INTP0	—	INTCC30
		16-bit capture register (CT30)	—	INTP1		
		16-bit capture register (CT31)	—	INTP4		
Timer 4	16-bit timer (TM4)	16-bit compare register (CM40)	INTCM40	—	1	TCLRUD
		16-bit compare register (CM41)	INTCM41			INTCM40

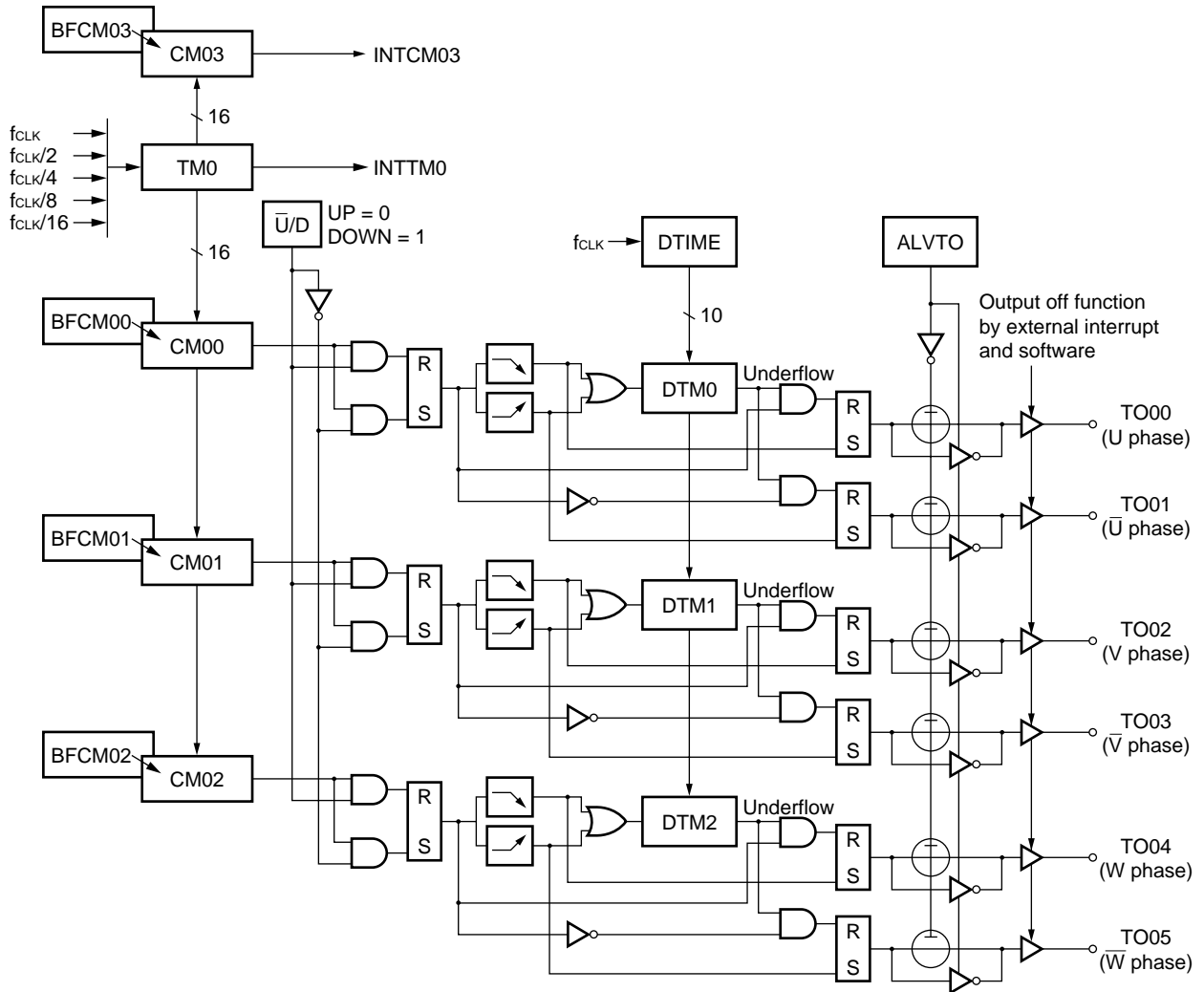
7.2 Timer 0

7.2.1 Configuration

Timer 0 consists of a 16-bit timer 0 (TM0) and four 16-bit compare registers (CM00-CM03).

Figures 7-1 through 7-3 show the block diagrams of timer 0.

Figure 7-1. Block Diagram of Timer 0 (PWM mode 0 ... symmetric triangular wave, asymmetric triangular wave)

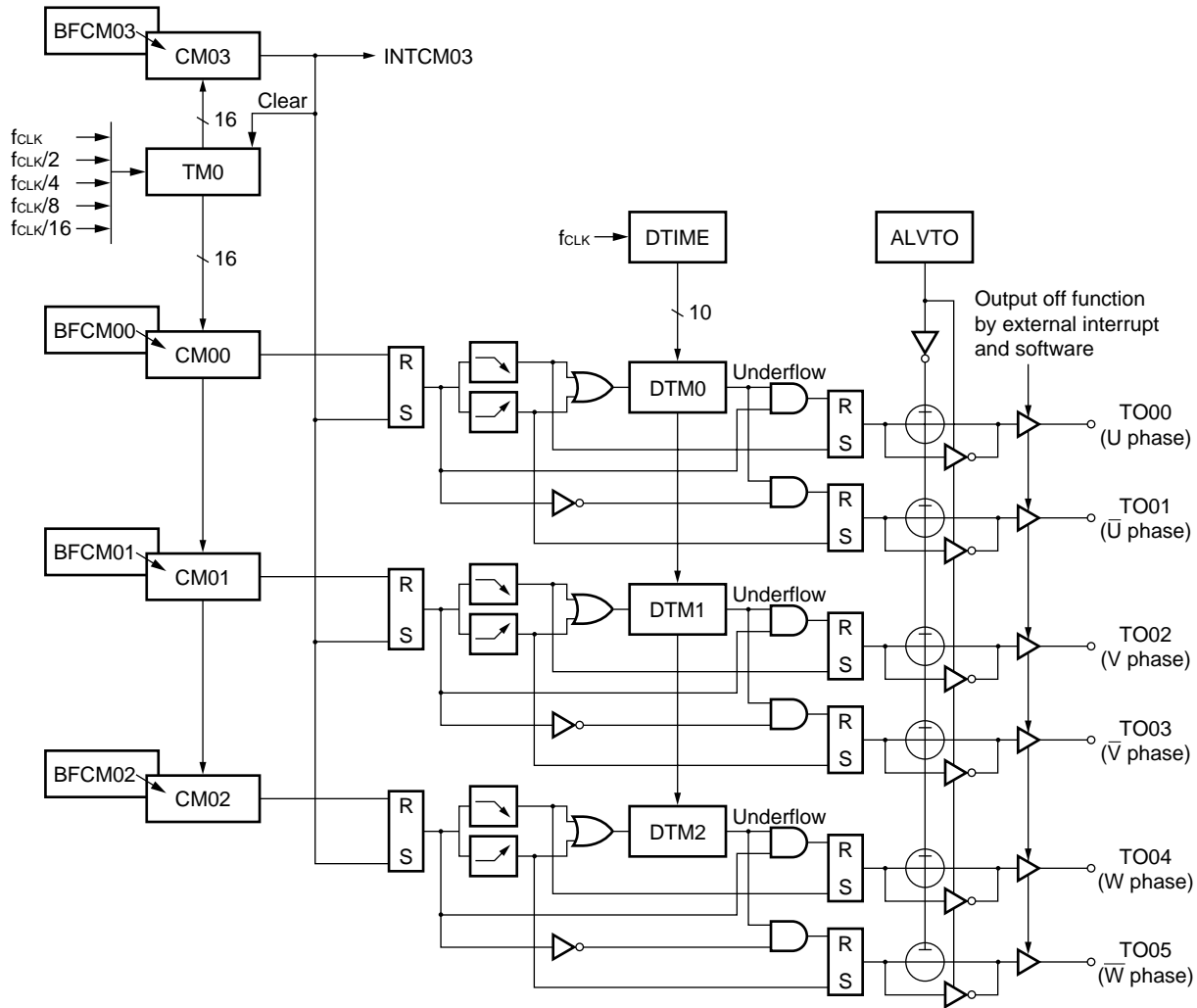


TM0 : Timer register
 CM00-CM03 : Compare registers
 BFCM00-BFCM03: Buffer registers
 DTIME : Reload register

DTM0-DTM2 : Dead time timers
 ALVTO : Bit 2 of TUM0 register
 \bar{U}/D : Bit 3 of TMC0 register

Remark fCLK: internal system clock

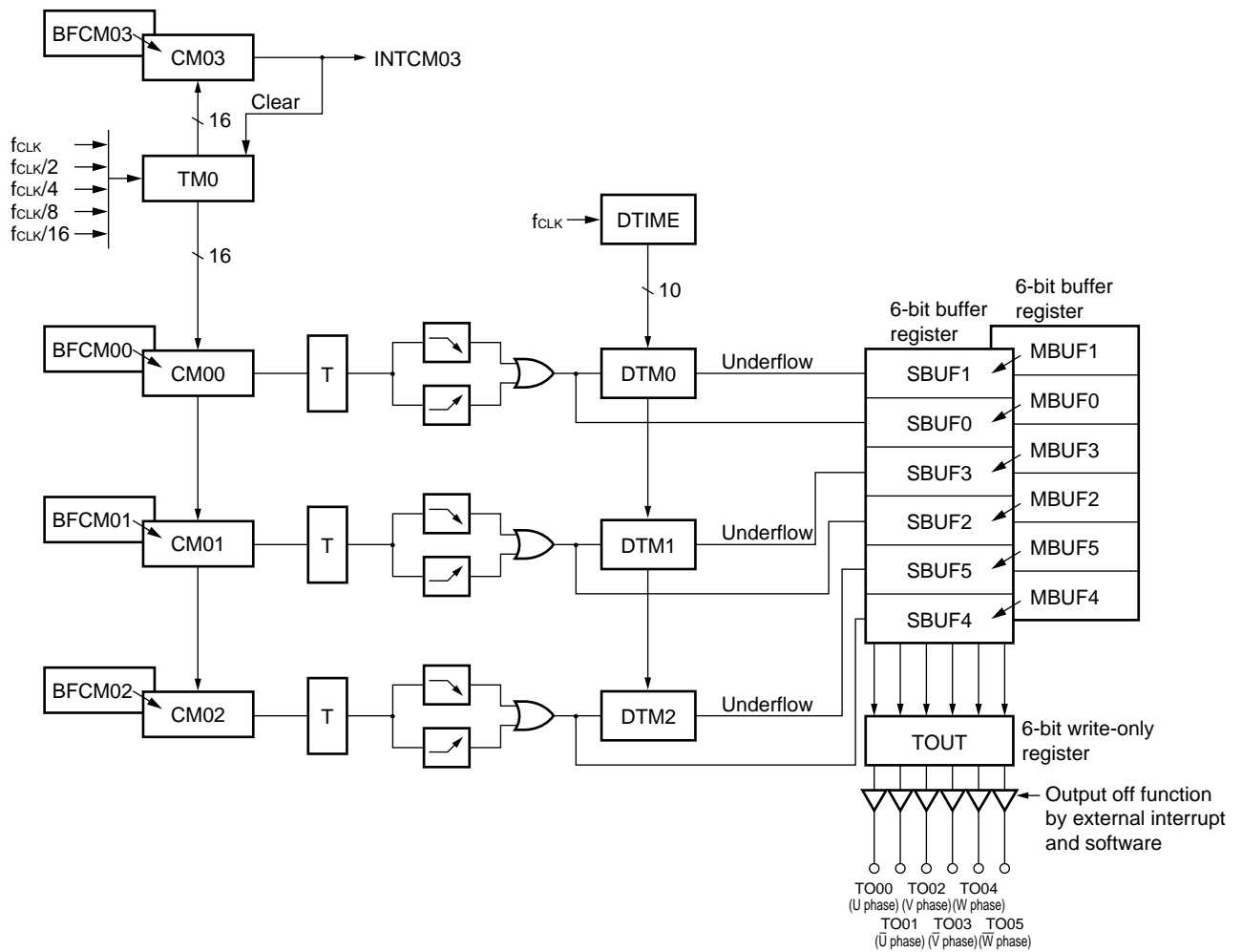
Figure 7-2. Block Diagram of Timer 0 (PWM mode 0 ... toothed wave)



TM0 : Timer register
 CM00-CM03 : Compare registers
 BFCM00-BFCM03: Buffer registers
 DTIME : Reload register
 DTM0-DTM2 : Dead time timers
 ALVTO : Bit 2 of TUM0 register

Remark fCLK: internal system clock

Figure 7-3. Block Diagram of Timer 0 (PWM mode 1)



TM0 : Timer register
 CM00-CM03 : Compare registers
 BFCM00-BFCM03: Buffer registers
 DTIME : Reload register
 DTM0-DTM2 : Dead time timers

MBUF0-MBUF5 : Master buffer registers
 SBUF0-SBUF5 : Slave buffer registers
 TOUT : Timer out register

Remark f_{CLK}: internal system clock

(1) 16-bit timer 0 (TM0)

TM0 is a 16-bit up/down timer and operates as an up timer. Its cycle is controlled by a compare register (CM03). TM0 can operate in the five operation modes as shown in Table 7-2. These operation modes can be selected by timer unit mode register 0 (TUM0).

Table 7-2. Operation Modes of Timer 0 (TM0)

Operation Mode	Count Operation	Timer Clear	BFCM0n → CM0n Transfer Timing
General-purpose interval timer	Up	INTCM03	—
PWM mode 0 (symmetric triangular wave)	Up/down	—	INTTM0
PWM mode 0 (asymmetric triangular wave)	Up/down	—	INTTM0 INTCM03
PWM mode 0 (toothed wave)	Up	INTCM03	INTCM03
PWM mode 1	Up	INTCM03	INTCM03

Remark n = 0-2

TM0 can be cleared by an interrupt that occurs when the current value of the timer coincides (INTCM03) with the value of the compare register (CM03) (in the general-purpose interval timer mode, PWM mode 0 (toothed wave), and PWM mode 1).

TM0 is a register that can be read only by a 16-bit manipulation instruction.

All the bits of TM0 are cleared to 0 by the $\overline{\text{RESET}}$ input.

(2) 10-bit dead time timers 0-2 (DTM0-DTM2)

DTM0-DTM2 are 10-bit down timers that generate dead time suitable for inverter control application. These timers operate as one-shot timers.

Counting by a dead time timer is enabled or disabled by the CED bit of timer control register 1 (TMC1), and cannot be controlled (i.e., started or stopped) through software. The dead time timer starts and stops counting hardware-wise.

The dead time timer starts counting down when the value of the reload register (DTIME) is reloaded to it in synchronization with the compare coincidence timing of CM00 to CM02:

When the value of a dead time timer changes from 000H to 3FFH, the dead time timer generates an underflow signal, and the timer stops holding the value 3FFH.

If the value of a dead time timer coincides with the value of the corresponding compare register before the underflow of the dead time timer takes place, the value of DTIME is reloaded to the dead time timer again, and the timer starts down counting.

The count clock of the dead time timer is fixed to f_{CLK} , and the dead time width is (Set value of DTIME + 1) $\times f_{CLK}$.

If TM0 operates in the PWM mode 0 with the dead time timer disabled from counting, a True Bar signal without dead time is output to TO00 and TO01, TO02 and TO03, and TO04 and TO05.

DTM0 to DTM2 cannot be read/written (and cannot be controlled by software).

DTM0 to DTM2 are initialized to 3FFH by the \overline{RESET} input.

(3) 10-bit reload register (DTIME)

DTIME is a 10-bit register that is commonly used to set the values of the three dead time timers (DTM0-DTM2). However, a value is reloaded from DTIME to each dead time timer independently.

DTIME can be read/written by a 16-bit manipulation instruction. The low-order 10 bits of DTIME are valid data and the high-order 6 bits are fixed to "0" by hardware.

\overline{RESET} input makes DTIME undefined.

(4) 16-bit compare registers 00-02 (CM00-CM02)

CM00-CM02 are 16-bit compare registers that always compare their own values with the value of TM0. If the value of a compare register coincides with the value of TM0, the compare register outputs a trigger signal, and changes the content of the flip-flop connected to the register.

Each of CM00-CM02 is provided with a buffer register (BFCM00-BFCM02), so that the contents of the buffer can be transferred to the corresponding compare register at any time.

CM00 to CM02 can be read/written by a 16-bit manipulation instruction. $\overline{\text{RESET}}$ input makes them undefined. Writing is performed as follows by setting the CMWE bit of timer control register 0 (TMC0).

- CMWE = 0: When a write instruction is executed, values of the buffer registers (BFCM00 to BFCM02) corresponding to the respective compare registers are transferred to the compare registers.
- CMWE = 1: Data can be written directly to the compare registers by a write instruction.

For reading, data can be read directly irrespective of the TMC0 register setting.

(5) 16-bit compare register 03 (CM03)

CM03 is a 16-bit register and performs compare operations together with TM0. When a coincidence is detected, it generates an interrupt signal (INTCM03). CM03 controls the count higher limit value of TM0, and if the values coincide, it switches the up/down of TM0 or clears the timers with the next count clock.

Furthermore, CM03 is provided with a buffer register (BFCM03) and transfers the buffer contents to CM03 with arbitrary timing. Transfer enable/disable is controlled by the B3TR bit of the TMC0 register.

CM03 can be read/written by a 16-bit manipulation instruction. $\overline{\text{RESET}}$ input makes it undefined.

(6) 16-bit buffer registers CM00-CM02 (BFCM00-BFCM02)

BFCM00-BFCM02 are 16-bit registers each of which transfers data to the corresponding compare register (CM00-CM02) when an interrupt signal (INTCM03/INTTM0) is generated.

BFCM00 to BFCM02 can be read/written by a 16-bit manipulation instruction. $\overline{\text{RESET}}$ input makes them undefined.

(7) 16-bit buffer register CM03 (BFCM03)

BFCM03 is a 16-bit register and it transfers data to the compare registers with arbitrary timing. Transfer enable/disable is controlled by the B3TR bit of the TMC0 register.

(8) 6-bit master buffer registers 0-5 (MBUF0-MBUF5)

MBUF0 to MBUF5 are 6-bit registers to which a value is set when an interrupt signal (INTCM03) is generated. Timing to transfer data from an MBUF to the corresponding SBUF is controlled by INTCM03.

MBUF0 to MBUF5 can be read/written by a bit manipulation instruction or 8-bit manipulation instruction. The low-order 6 bits of MBUF0 to MBUF5 are valid data and the high-order 2 bits are fixed to "0" by hardware. $\overline{\text{RESET}}$ input makes them undefined.

This buffer register is used in PWM mode 1.

MBUF0 is also used for the RTP output in the general-purpose interval timer mode.

(9) 6-bit slave buffer registers 0-5 (SBUF0-SBUF5)

SBUF0 to SBUF5 are 6-bit registers that output data to the TOUT register in synchronization with the output timing determined taking the dead time into consideration by coincidence of CM00-CM02 compare values. The timing at which the contents of an MBUF are transferred to the corresponding SBUF is controlled by INTCM03.

SBUF0 to SBUF5 can be read/written by a bit manipulation instruction or 8-bit manipulation instruction. The low-order 6 bits of SBUF0 to SBUF5 are valid data and the high-order 2 bits are fixed to "0" by hardware. $\overline{\text{RESET}}$ input makes them undefined.

This buffer register is used in PWM mode 1.

SBUF0 is also used for the RTP output in the general-purpose interval timer mode.

(10) 6-bit timer out register (TOUT)

TOUT is a 6-bit register that can be written by an 8-bit manipulation instruction. It is an output latch that can be used only when TM0 is in the following modes.

<1> General-purpose interval timer mode

<2> PWM mode 1

When TM0 is in any mode other than the above, writing is disabled by hardware and it is not possible to write to it even by executing a write instruction.

The TOUT data is output directly to port 8 (P80 to P85). Therefore, if port 8 is read with bits PM8.0 to PM8.5 of the port 8 mode register (PM8) set to "0" (output port), it is possible to read the output data of TOUT. If port 8 is read with bits PM8.0 to PM8.5 set to "1" (input port), the pin statuses are read.

Since the high-order 2 bits of TOUT have no latch to retain data by hardware, they are undefined. When port 8 is read, the high-order 2 bits become undefined in the same way.

Caution The condition under which the TOUT data is output to port 8 is that port 8 is in control mode (TO00 to TO05 outputs) and that TM0 is in the general-purpose interval timer mode or PWM mode 1. Otherwise, the TOUT data is not output to port 8 and therefore it is not possible to read the TOUT contents.

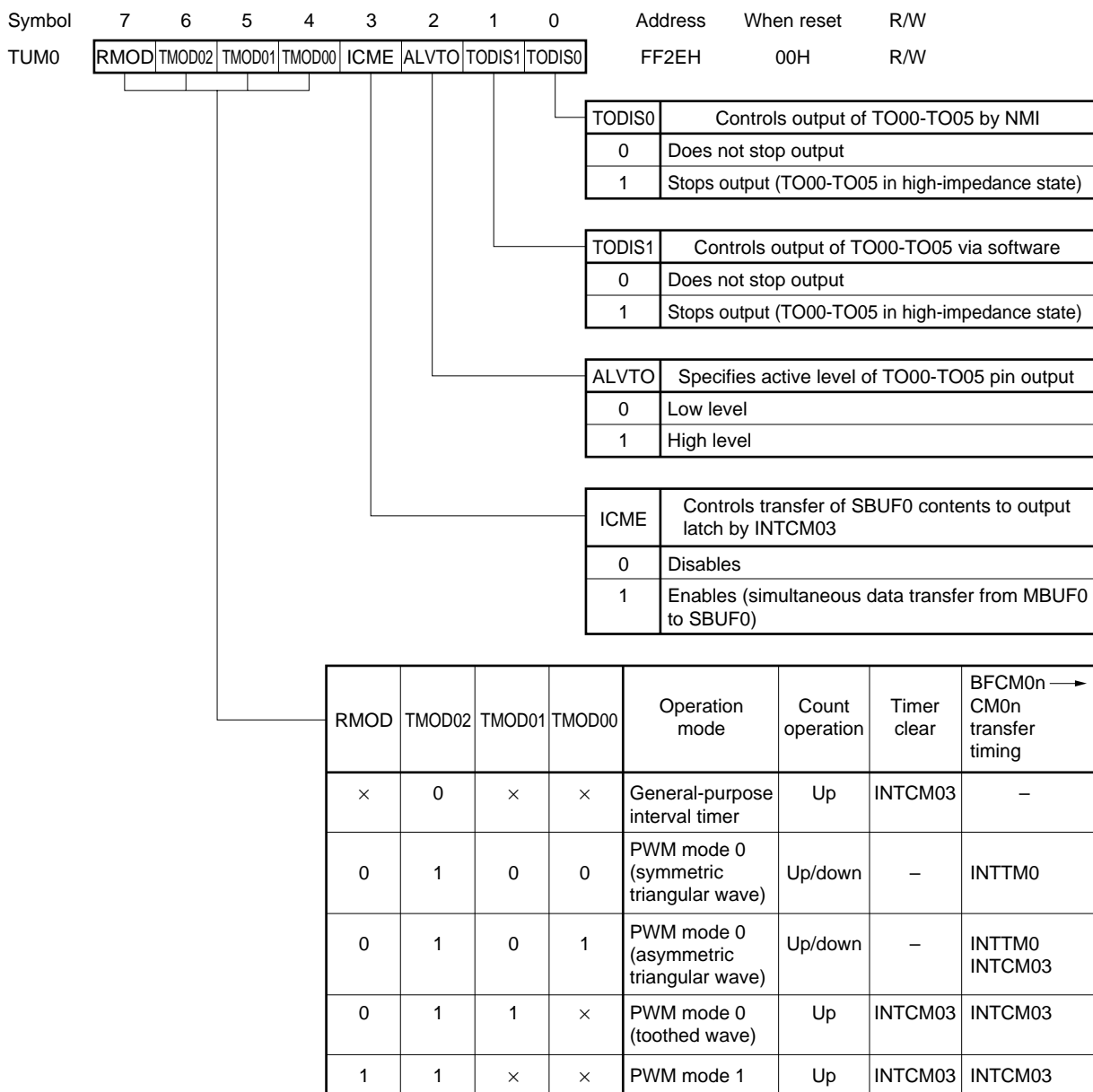
7.2.2 Control registers

(1) Timer unit mode register 0 (TUM0)

Timer unit mode register 0 (TUM0) is an 8-bit register that specifies the operation modes of TM0, and controls the output of TO00 to TO05.

TUM0 can be read or written by a bit manipulation instruction or an 8-bit manipulation instruction.

This register is cleared to 00H by the RESET input.

Figure 7-4. Format of Timer Unit Mode Register 0

- Cautions**
1. The specification of the ALVTO bit is valid only for the pins set to the control mode (TO00 to TO05) by the PMC8 register when TM0 is in PWM mode 0 (symmetrical triangular wave, asymmetrical triangular wave, toothed wave).
 2. The ICME bit is valid only when TM0 is in the general-purpose interval timer mode (TMOD02 = 0).
 3. It is prohibited to change bits RMOD, TMOD02 through TMOD00 and ALVTO during operation of TM0 (CE0 = 1) and DTM0 to DTM2 (CED = 1).

- Remarks**
1. n = 0-2
 2. x: don't care

[Output driver off function]

When an external error occurs, the μ PD78366A can turn off the output pin (TO00-TO05) driver.

The function can be executed under the control of the TODIS0, TODIS1 bit of the TUM0 register, as described below:

<1> External interrupt (NMI) (TODIS0 = 1)

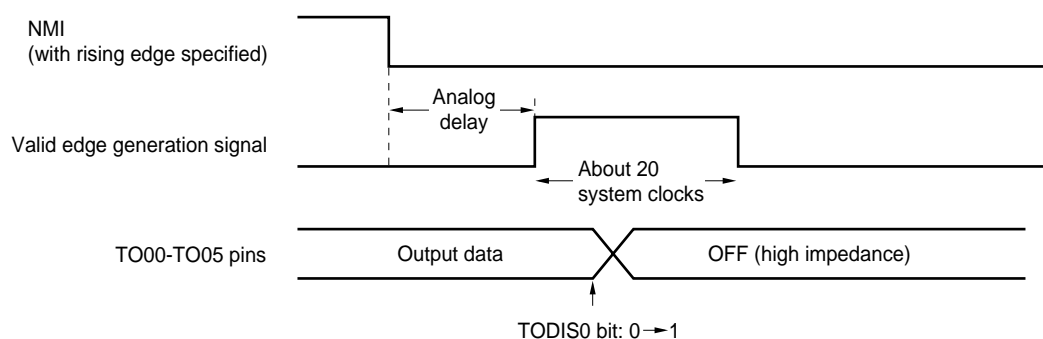
When an interrupt active edge specified at the NMI pin occurs, the TO00-TO05 pin driver can be turned off.

The NMI pin active edge can be specified in the external interrupt mode register 0 (INTM0).

Even if a NMI pin edge detection pulse goes off, the TO00-TO05 pin driver remains off. To restore the operation, change the TODIS0 bit from 1 to 0.

Caution After the valid edge of NMI has been generated, the valid edge generation signal is retained for the duration of about 20 system clocks. If the TODIS0 bit is changed from “0” to “1” during this period, the TO00 through TO05 pins are set to OFF by the NMI that has been already generated.

It is recommended that the TODIS0 bit be manipulated in the NMI routine except when the system is initialized.

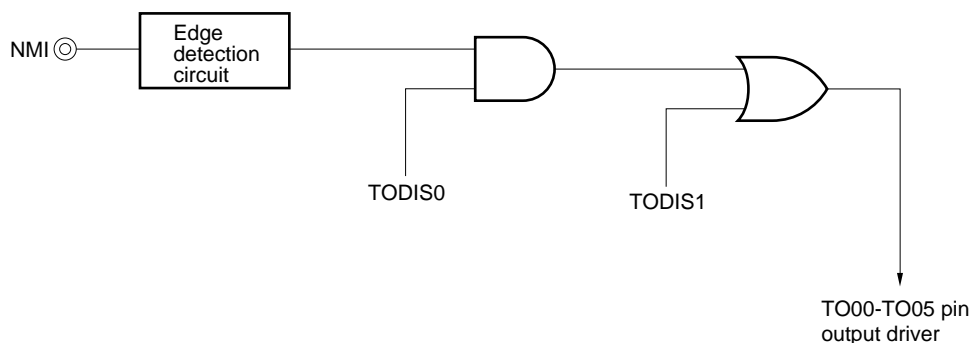


Remark Execution branches to the NMI routine after the valid edge generation signal has changed from “1” to “0”.

<2> Software control (TODIS1 = 1)

The TO00-TO05 pin driver can be turned off under software control regardless of the NMI pin.

Figure 7-5. Configuration of Output Driver off Function

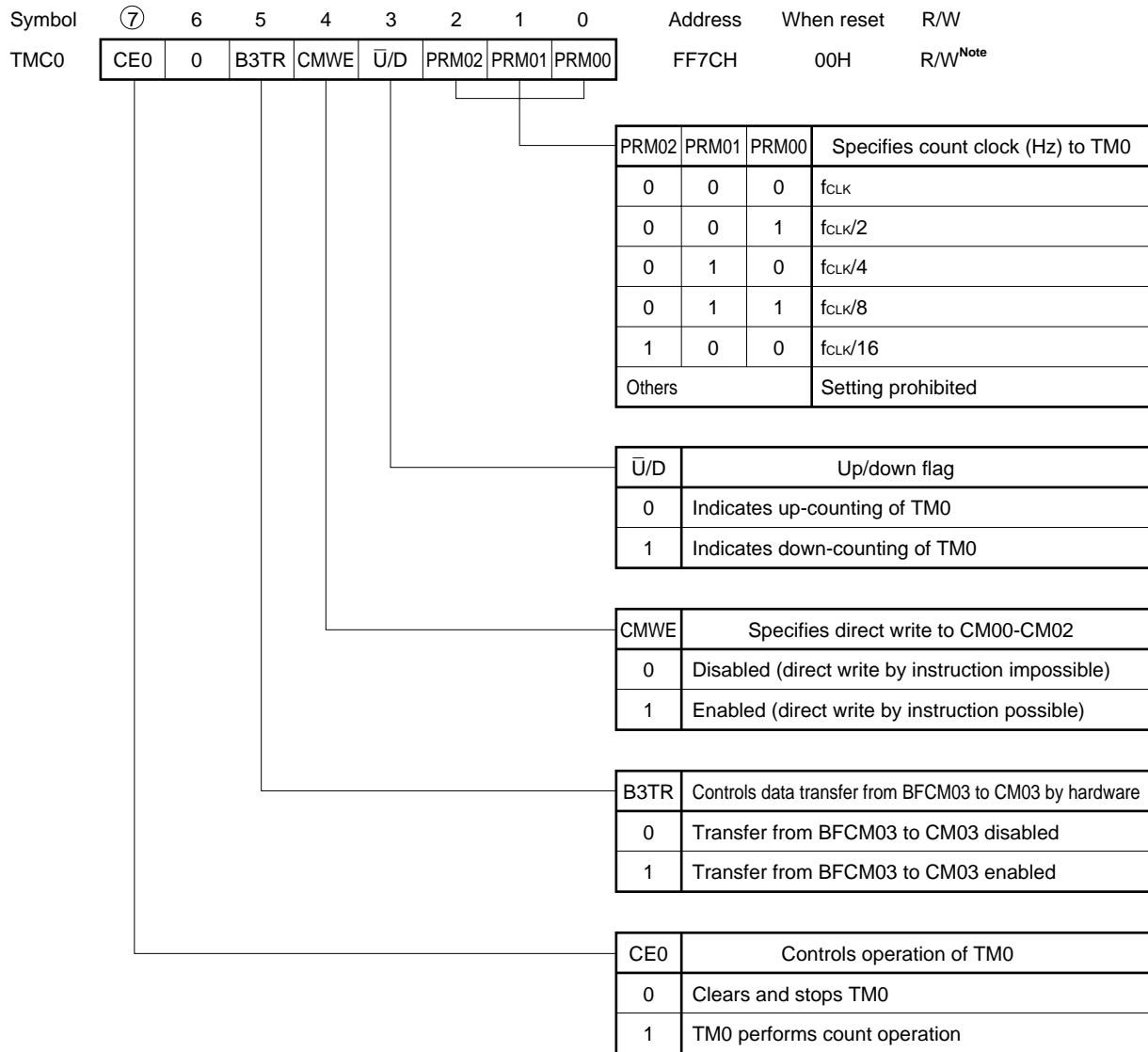


(2) Timer control register 0 (TMC0)

Timer control register 0 (TMC0) is an 8-bit register that controls the operation of TM0, CM00 to CM03.

This register can be read or written by a bit manipulation instruction or an 8-bit manipulation instruction.

TMC0 is cleared to 00H by the $\overline{\text{RESET}}$ input.

Figure 7-6. Format of Timer Control Register 0

Note The $\overline{\text{U/D}}$ bit is a read-only bit.

Remark f_{CLK} : internal system clock

- Cautions**
1. Bit 6 of the TMC0 register is fixed to “0” by hardware. Even if “1” is written, it remains “0”.
 2. It is prohibited to change bits B3TR and PRM02 through PRM00 during operation of TM0 (CE0 = 1)
 3. If CE0 is set to 0 (TM0 stops), $\overline{U/D}$ flag is cleared to 0.

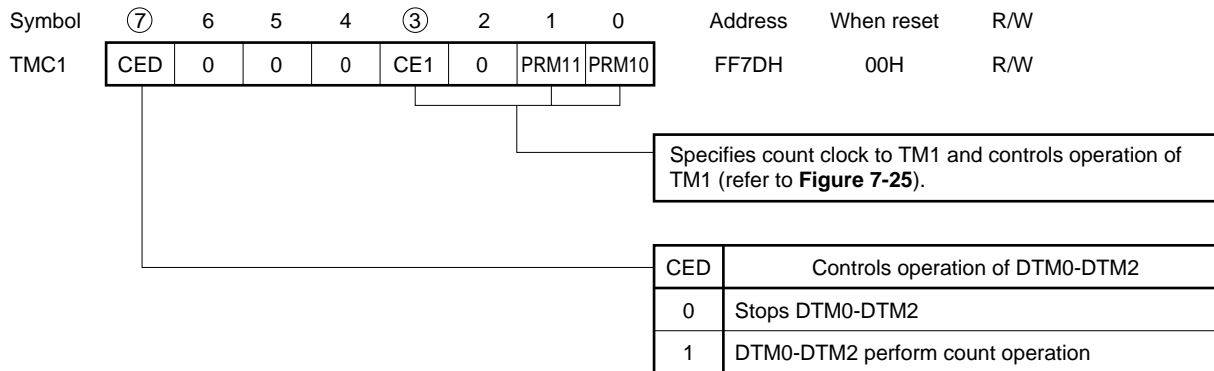
Table 7-3. Timing of Transfer from BFCM03 to CM03

B3TR	TM0 Operating Mode	Timing of Transfer from BFCM03 to CM03
0	All modes	Not transferred
1	General-purpose interval timer	Not transferred
	PWM mode 0 (symmetric triangular wave)	INTTM0
	PWM mode 0 (asymmetric triangular wave)	INTTM0
	PWM mode 0 (toothed wave)	INTCM03
	PWM mode 1	INTCM03

Caution CM03 can be read/written directly.

(3) Timer control register 1 (TMC1)

Timer control register 1 (TMC1) is an 8-bit register that controls the operations of TM1 and DTM0-DTM2. This register can be read or written by a bit manipulation instruction or an 8-bit manipulation instruction. TMC1 is cleared to 00H by the $\overline{\text{RESET}}$ input.

Figure 7-7. Format of Timer Control Register 1

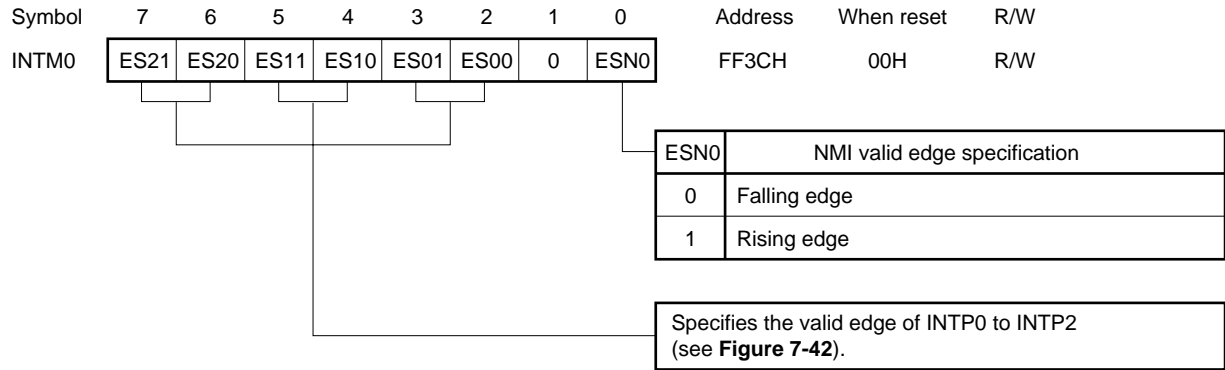
- Cautions**
1. The CED bit is common to the three dead time timers (DTM0-DTM2).
 2. Bits 6 to 4 and 2 of the TMC1 register are fixed to “0” by hardware. Even if “1” is written, they remain “0”.
 3. It is prohibited to change the CED bit during the operation of TM0 (CE0 = 1)

When TM0 is operated in PWM mode 0 with CED = 0 (dead time timer stopped), true bar signals with no dead time are output to TO00 and TO01, TO02 and TO03, and TO04 and TO05.

(4) External interrupt mode register 0 (INTM0)

External interrupt mode register 0 (INTM0) is an 8-bit register for specifying the valid edge of NMI and INTP0 to INTP2.

The INTM0 register can be read/written by a bit manipulation instruction or 8-bit manipulation instruction. $\overline{\text{RESET}}$ input sets it to 00H.

Figure 7-8. External Interrupt Mode Register 0 Format

Caution Bit 1 of the INTM0 register is fixed to “0” by hardware. Even if “1” is written, it remains “0”.

7.2.3 Operation

(1) Basic operation

Timer 0 (TM0) is a 16-bit interval timer that operates as an up/down or up timer. The cycle is controlled by a compare register (CM03).

All the bits of TM0 are cleared to 0 by the $\overline{\text{RESET}}$ input and the timer stops counting.

Counting is enabled or disabled by the CE0 bit of timer control register 0 (TMC0). When the CE0 bit is set to 1 through software, the timer starts counting; when the bit is reset to 0, TM0 is cleared and stops counting. When the value set in advance to the compare register (CM03) coincides with the current count value of TM0, a coincidence interrupt (INTCM03) occurs, and TM0 is cleared.

Five internal clocks can be selected by the TMC0 register as the count clock to TM0.

When TM0 is used as an up/down timer, and when TM0 = 0000H as a result of the down-count operation, an underflow interrupt (INTTM0) occurs.

TM0 can operate in the following five operation modes which are selected by timer unit mode register (TUM0):

- General-purpose interval timer
- PWM mode 0
 - triangular wave modulation (symmetric triangular wave control)
 - triangular wave modulation (asymmetric triangular wave control)
 - toothed wave modulation control
- PWM mode 1

Table 7-4. Operation Modes of Timer 0 (TM0)

TUM0 Register				Operation Mode	Count Operation	Timer Clear	BFCM0n → CM0n Transfer Timing
RMOD	TMOD02	TMOD01	TMOD00				
×	0	×	×	General-purpose interval timer	Up	INTCM03	—
0	1	0	0	PWM mode 0 (symmetric triangular wave)	Up/down	—	INTTM0
0	1	0	1	PWM mode 0 (asymmetric triangular wave)	Up/down	—	INTTM0 INTCM03
0	1	1	×	PWM mode 0 (toothed wave)	Up	INTCM03	INTCM03
1	1	×	×	PWM mode 1	Up	INTCM03	INTCM03

Remarks 1. n = 0-2

2. ×: don't care

Each operation mode of TM0 is described on the following pages:

(2) General-purpose interval timer mode

If TM0 is set to the general-purpose interval timer mode, TM0 and CM03 always perform comparison operations and whenever they detect a coincidence, they generate an interrupt signal (INTCM03). The result of a compare coincidence is retained by hardware and TM0 is cleared (0000H) with the next count clock after the coincidence. Furthermore, when the next count clock is input, TM0 is counted up to 0001H.

$$\text{Interval cycle} = (\text{value of CM03} + 1) \times \text{TM0 count clock rate}$$

If TM0 is set to this mode, it is possible to operate pins TO00 to TO05 as a real-time output port (RTP).

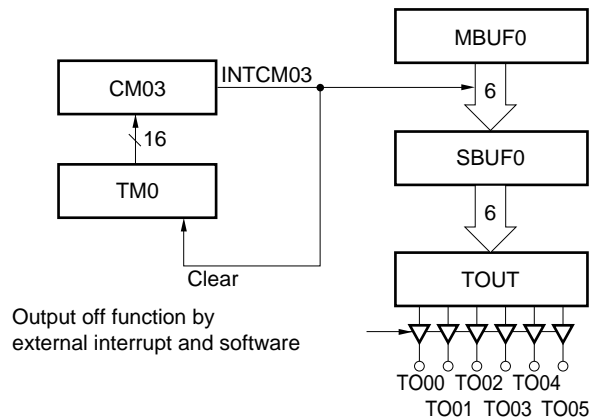
[Setting procedure]

- <1> Set pins P80 to P85 to the control mode (TO00 to TO05 outputs) (PMC80 to PMC85 = 1) by the port 8 mode control register (PMC8).
- <2> Set the TMOD02 bit of the TUM0 register to "0" to set to the general-purpose interval timer mode. Furthermore, put "1" in the ICME bit to enable the RTP output operation.
If it is desired to stop pins TO00 to TO05 output in the event of abnormality, also set bits TODIS1 and TODIS0.
- <3> Specify the TM0 count clock by bits PRM02 to PRM00 of the TMC0 register.
- <4> When the CE0 bit of the TMC0 register is set (1) TM0 starts counting and data is output from pins TO00 to TO05 as RTP.

[Operation]

The data placed in MBUF0 by INTCM03 is transferred to TOUT via SBUF0. The TOUT data is output to pins TO00 to TO05 unmodified. Therefore, the output pattern is set in MBUF0.

Figure 7-9. Block Diagram of RTP Output Function of TO00-TO05 Pins



Caution Setting of CM03 = 0000H is prohibited.

(3) PWM mode 0 ... Triangular wave modulation (symmetric waveform control)**[Setting procedure]**

- (a) Set pins P80 to P85 to the control mode (TO00 to TO05 outputs) by the port 8 mode control register (PMC8) (bits PMC80 to PMC85 = 1).
- (b) Set PWM mode 0 (symmetric triangular wave) by bits RMOD and TMOD00 to TMOD02 of the TUM0 register. Furthermore, set the active level of pins TO00 to TO05 by the ALVTO bit of the TUM0 register.
- (c) Set the count clock of TM0 by bits PRM02 to PRM00 of the TMC0 register, and set the manipulation for writing to CM00 to CM02 by the CMWE bit. Furthermore, set the transfer operation from BFCM03 to CM03 by the B3TR bit.
- (d) Set the initial values.
 - (i) Put the half-cycle width of the 1st PWM cycle in CM03.
 - $\text{PWM cycle} = \text{CM03 value} \times 2 \times \text{TM0 clock rate}$
(The clock rate of TM0 is set by the TMC0 register.)
 - (ii) Put the half-cycle width of the 2nd PWM cycle in BFCM03.
(Setting is not necessary because BFCM03 is not used when B3TR of the TMC0 register = 0)
 - (iii) Put the dead time width in DTIME.
 - $\text{Dead time width} = (\text{DTIME} + 1) \times T_{\text{CLK}}$
 T_{CLK} : System clock rate
 - (iv) Put the set/reset timing of the F/F used in the 1st cycle in CM00 to CM02.
 - Direct writing of CMWE of the TMC0 register = 0 to CM00 to CM02 is disabled.
If an instruction for writing to CM00 to CM02 is executed, data in BFCM00 to BFCM02 is transferred to CM00 to CM02. Use the following procedure for setting.
 - <1> Write values of CM00 to CM02 used in the 1st cycle to BFCM00 to BFCM02.
 - <2> If an instruction for writing to CM00 to CM02 is executed, the values of BFCM00 to BFCM02 set in <1> are transferred to CM00 to CM02.
 - Direct writing of CMWE of the TMC0 register = 1 to CM00 to CM02 is enabled.
 - (v) Set the set/reset timing of the F/F used in the 2nd cycle in BFCM00 to BFCM02.

- (e) Set (1) the CED bit of the TMC1 register to enable operation of the dead time timer. If it is desired not to take the dead time, set CED = 0.
- (f) Setting (1) the CE0 bit of the TMC0 register starts the counting of TM0, and a 6-channel PWM signal is output from pins TO00 to TO05.
If direct writing to CM00 to CM02 is not performed by an instruction during the operation, reset (0) the CMWE bit of the TMC0 register before starting the timer.

Caution Setting of CM03 = 0000H is prohibited.

[Operation]

In this mode, TM0 executes up/down count operation, and if TM0 = 0000H as a result of counting down, an underflow interrupt (INTTM0) occurs.

Switching from up count to down count is performed by a coincidence between TM0 and CM03 (INTCM03), while switching from down count to up count is performed by INTTM0.

The PWM cycle in this mode is (CM03 value $\times 2 \times$ TM0 clock rate). The data setting in CM03 varies as follows depending on the setting of the B3TR bit of the TMC0 register.

- B3TR = 0: No transfer from BFCM03 to CM03 is performed. Execute operations by the software processing started by INTTM0 and directly place the cycle data in CM03.
- B3TR = 1: The BFCM03 data is automatically transferred by the hardware to CM03 by INTTM0. Then, execute operations by the software processing started by INTTM0 and put the data in the next cycle in BFCM03.

Data settings in CM00 to CM02 that control the PWM duty width is explained below.

Concerning data settings in CM00 to CM02, when INTTM0 occurs, the hardware automatically transfers the values of BFCM00-BFCM02 to CM00-CM02. In addition, software processing is started and an arithmetic operation is executed, and the set/reset timing of F/F used for the next cycle is set to BFCM00-BFCM02. When CMWE of the TMC0 register = 0, direct writing to CM00 to CM02 is not possible. If CMWE = 1, direct writing is possible. However, data transfer from BFCM00 through BFCM02 to CM00 through CM02 by INTTM0 is performed irrespective of the setting of the CMWE bit.

The PWM cycle and PWM duty width are set in the above procedure.

The set/reset condition of the F/F that changes with a coincidence in CM00 to CM02 is as follows.

- Set : Coincidence of TM0 with CM00-CM02 when TM0 counts up
- Reset : Coincidence of TM0 with CM00-CM02 when TM0 counts down

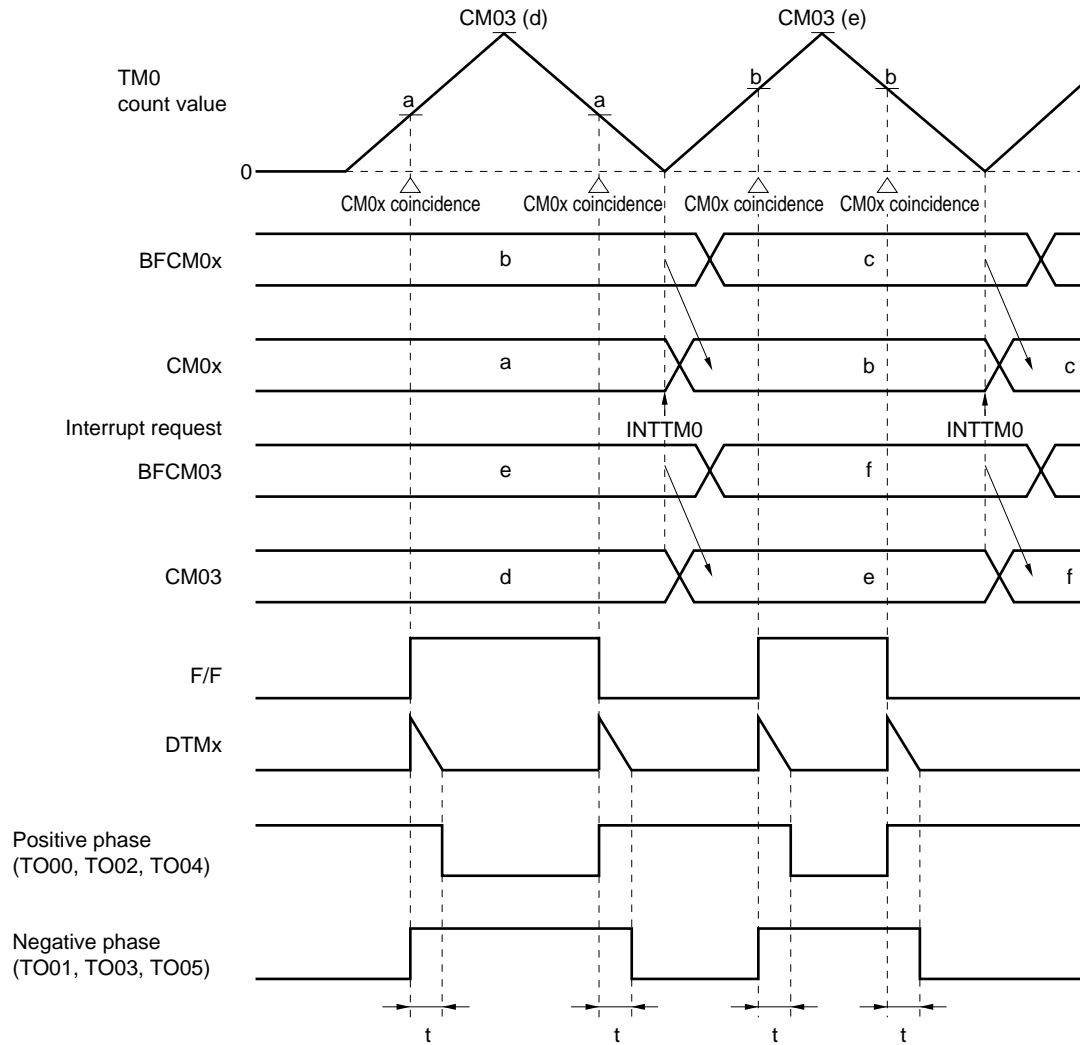
In this mode, F/F is set and reset at the same timing (symmetric control).

The value of DTIME is loaded to the corresponding dead time timer (DTM0-DTM2) in synchronization with the set/reset timing of F/F, and the dead time timer starts counting down. DTM0-DTM2 count down to 000H and stop when they count down further to 3FFH.

DTM0-DTM2 can automatically generate a width (dead time) at which the active levels of the positive (TO00, TO02, TO04) phase and negative phase (TO01, TO03, TO05) do not overlap.

In this way, the software processing is started by an interrupt (INTTM0) that occurs once during PWM cycle (one cycle) after initial setting has been performed, and by setting PWM cycle and PWM duty width used for the next cycle, the PWM waveform can be output to the TO00-TO05 pins with a dead time width taken into consideration.

Figure 7-10. Operation Timing in PWM Mode 0 (symmetric triangular wave)



Remarks 1. The above figure is the timing chart when the transfer operation from BFCM03 to CM03 is enabled with B3TR of the TMC0 register = 1. When B3TR = 0, no transfer is performed and the up/down cycle of TM0 is set by directly writing to CM03.

2. $x = 0-2$

3. $t: \text{dead time} = (\text{DTIME} + 1) \times T_{\text{CLK}}$
(T_{CLK} : system clock rate)

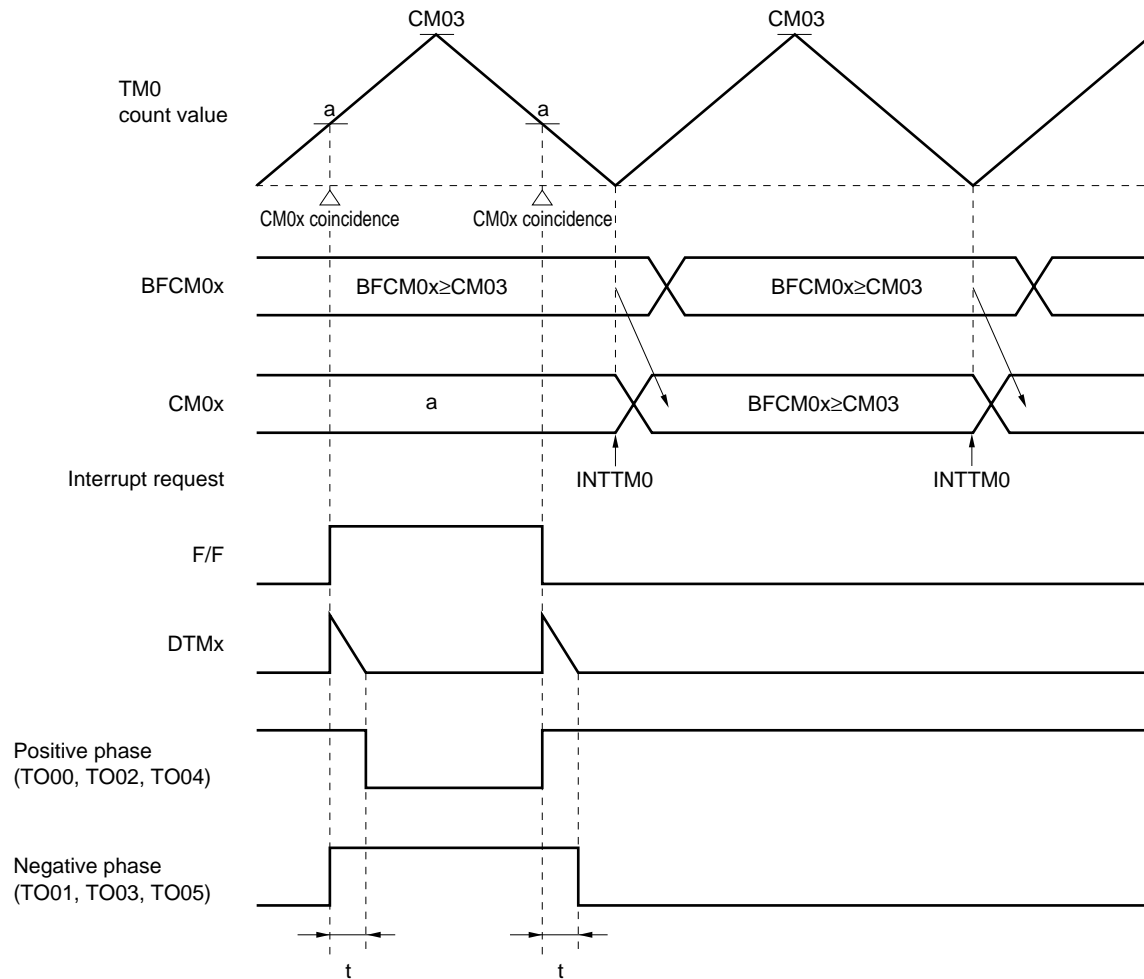
4. Not to take the dead time, clear the CED bit of the TMC1 register to 0.

5. The above figure shows an active low case.

An example of operation timing related to the set values of CM00-CM02 (BFCM00-BFCM02) is shown next.

(a) When CM0x (BFCM0x) \geq CM03 is set

Figure 7-11. Operation Timing in PWM Mode 0 (symmetric triangular wave, BFCM0x \geq CM03)



Remarks 1. x = 0-2

2. t: dead time = (DTIME + 1) \times T_{CLK}
(T_{CLK}: system clock rate)

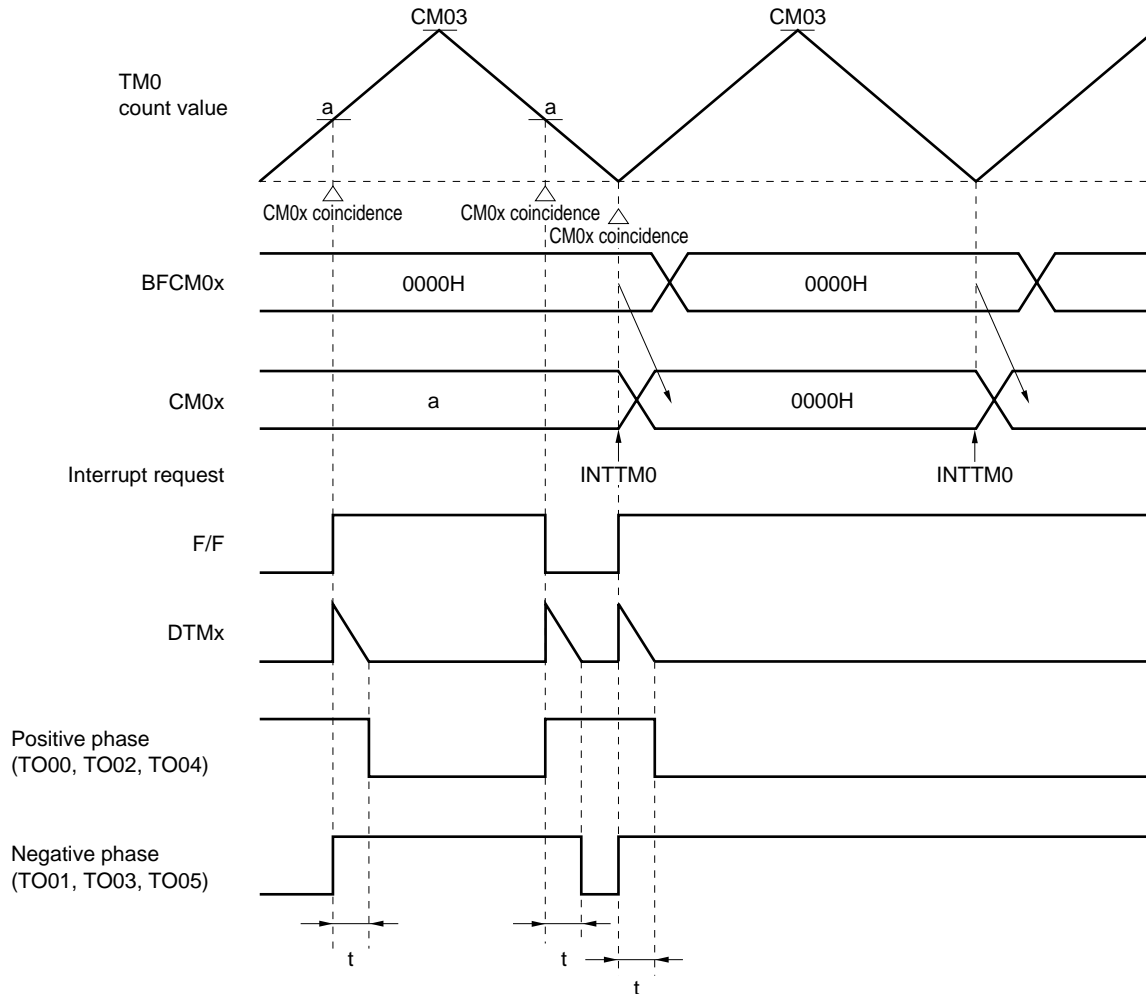
3. The above figure shows an active low case.

If a value greater than that of CM03 is set to BFCM0x, the positive-phase pins (TO00, TO02, TO04) continue outputting high level, and the negative-phase pins (TO01, TO03, TO05) continue outputting low level. This feature is effective for outputting a low-level or high-level width exceeding the PWM cycle in an application such as inverter control. When CM0x = CM03, coincidence between TM0 and CM0x is detected when TM0 counts down. Therefore, F/F remains reset and is not set.

The above explanation applies to an active low case. In an active high case, the levels of positive and negative phases are merely inverted and other operations remain the same.

(b) When CM0x (BFCM0x) = 0000H is set

Figure 7-12. Operation Timing in PWM Mode 0 (symmetric triangular wave, BFCM0x = 0000H)



Remarks 1. x = 0-2

2. t: dead time = (DTIME + 1) × T_{CLK}
(T_{CLK}: system clock rate)

3. The above figure shows an active low case.

Since coincidence of TM0 = CM0x = 0000H is detected when TM0 counts up, F/F is only set and is not reset. When the value setting is 0000H, the F/F is also changed in the cycle in which data is transferred from BFCM0x to CM0x in the same way as for value settings other than 0000H.

(4) PWM mode 0 ... triangular wave modulation (asymmetric wave control)**[Setting procedure]**

- (a) Set pins P80 to P85 to the control mode (TO00 to TO05 outputs) by the port 8 mode control register (PMC8) (bits PMC80 to PMC85 = 1).
- (b) Set PWM mode 0 (asymmetric triangular wave) by bits RMOD and TMOD00 to TMOD02 of the TUM0 register. Furthermore, set the active level of pins TO00 to TO05 by the ALVTO bit of the TUM0 register.
- (c) Set the count clock of TM0 by bits PRM02 to PRM00 of the TMC0 register, and set the manipulation for writing to CM00 to CM02 by the CMWE bit. Furthermore, set the transfer operation from BFCM03 to CM03 by the B3TR bit.
- (d) Set the initial values.
 - (i) Put the half-cycle width of the 1st PWM cycle in CM03.
 - PWM cycle = CM03 value $\times 2 \times$ TM0 clock rate
(The clock rate of TM0 is set by the TMC0 register.)
 - (ii) Put the half-cycle width of the 2nd PWM cycle in BFCM03.
(Setting is not necessary because BFCM03 is not used when B3TR of the TMC0 register = 0)
 - (iii) Put the dead time width in DTIME.
 - Dead time width = (DTIME + 1) \times T_{CLK}
T_{CLK}: System clock rate
 - (iv) Put the set timing of the F/F used in the 1st cycle in CM00 to CM02.
 - Direct writing of CMWE of the TMC0 register = 0 to CM00 to CM02 is disabled.
If an instruction for writing to CM00 to CM02 is executed, data in BFCM00 to BFCM02 is transferred to CM00 to CM02. Use the following procedure for setting.
 - <1> Write the values of CM00 to CM02 used in the 1st cycle to BFCM00 to BFCM02.
 - <2> If an instruction for writing to CM00 to CM02 is executed, the values of BFCM00 to BFCM02 set in <1> are transferred to CM00 to CM02.
 - Direct writing of CMWE of the TMC0 register to CM00 to CM02 is enabled.
 - (v) Put the reset timing of the F/F used in the 1st cycle in BFCM00 to BFCM02.

- (e) Set (1) the CED bit of the TMC1 register to enable the dead time timer operation. If it is desirable not to take the dead time, set CED = 0.
- (f) Setting (1) the CE0 bit of the TMC0 register starts the counting of TM0, and a 6-channel PWM signal is output from pins TO00 to TO05.
If direct writing to CM00 to CM02 is not performed by an instruction during the operation, reset (0) the CMWE bit of the TMC0 register before starting the timer.

Caution Setting of CM03 = 0000H is prohibited.

[Operation]

In this mode, TM0 executes up/down count operation, and if TM0 = 0000H as a result of counting down, an underflow interrupt (INTTM0) occurs.

Switching from up count to down count is performed by a coincidence between TM0 and CM03 (INTCM03), while switching from down count to up count is performed by INTTM0.

The PWM cycle in this mode is (CM03 value $\times 2 \times$ TM0 clock rate). Data setting in CM03 varies as follows depending on the setting of the B3TR bit of the TMC0 register.

- B3TR = 0: No transfer from BFCM03 to CM03 is performed. Execute operations by the software processing started by INTTM0 and directly set the cycle data in CM03.
- B3TR = 1: The BFCM03 data is automatically transferred by the hardware to CM03 by INTTM0. Then, execute operations by the software processing started by INTTM0 and put the data in the next cycle in BFCM03.

Data settings in CM00 to CM02, which control the PWM duty width, are explained below.

Concerning data settings in CM00 to CM02 when INTTM0 and INTCM03 (interrupt occurs when TM0 and CM03 coincide) occur, the hardware automatically transfers the values of BFCM00-BFCM02 to CM00-CM02. In addition, software processing is started and an arithmetic operation is executed, and the set/reset timing of F/F used after half a cycle is set to BFCM00-BFCM02.

When CMWE of the TMC0 register = 0, direct writing to CM00 to CM02 is not possible. If CMWE = 1, direct writing is possible. However, data transfer from BFCM00 through BFCM02 to CM00 through CM02 by INTTM0 and INTCM03 is performed irrespective of the setting of the CMWE bit.

The PWM cycle and PWM duty width are set in the above procedure.

The set/reset condition of the F/F that changes with a coincidence in CM00 to CM02 is as follows.

- Set : Coincidence of TM0 with CM00-CM02 when TM0 counts up
- Reset : Coincidence of TM0 with CM00-CM02 when TM0 counts down

The value of DTIME is loaded to the corresponding dead time timer (DTM0-DTM2) in synchronization with the set/reset timing of F/F, and the dead time timer starts counting down. DTM0-DTM2 count down to 000H and stop when they count down further to 3FFH.

DTM0-DTM2 can automatically generate a width (dead time) at which the active levels of the positive (TO00, TO02, TO04) phase and negative phase (TO01, TO03, TO05) do not overlap.

In this way, the software processing is started by two interrupts (INTTM0 and INTCM03) that occur during PWM cycle (one cycle) after initial setting has been performed, and by setting PWM cycle and PWM duty width used for half cycle later, the PWM waveform can be output to the TO00-TO05 pins with a dead time width taken into consideration.

The difference between the symmetric wave control mode and this asymmetric wave control mode is that in the symmetric wave control mode, INTTM0 (occurs once in PWM cycle) is used as an interrupt that transfers BFCM00-BFCM02 to CM00-CM02 and starts the software, while INTTM0 and INTCM03 (occur two times in PWM cycle = once during half a cycle) are used in this mode.

[Output waveform width in respect to set value]

- PWM cycle = $CM03 \times 2 \times T_{TM0}$
- Dead time width $T_{DTM} = (DTIME + 1) \times T_{CLK}$
- Active width of positive phase (TO00, TO02, TO04 pins)
= $\{(CM03 - CM0X_{up}) + (CM03 - CM0X_{down})\} \times T_{TM0} - T_{DTM}$
- Active width of negative phase (TO01, TO03, TO05 pins)
= $(CM0X_{down} + CM0X_{up}) \times T_{TM0} - T_{DTM}$

T_{CLK} : system clock rate

T_{TM0} : input clock rate of TM0

$CM0X_{up}$: set values of CM00-CM02 when TM0 counts up

$CM0X_{down}$: set values of CM00-CM02 when TM0 counts down

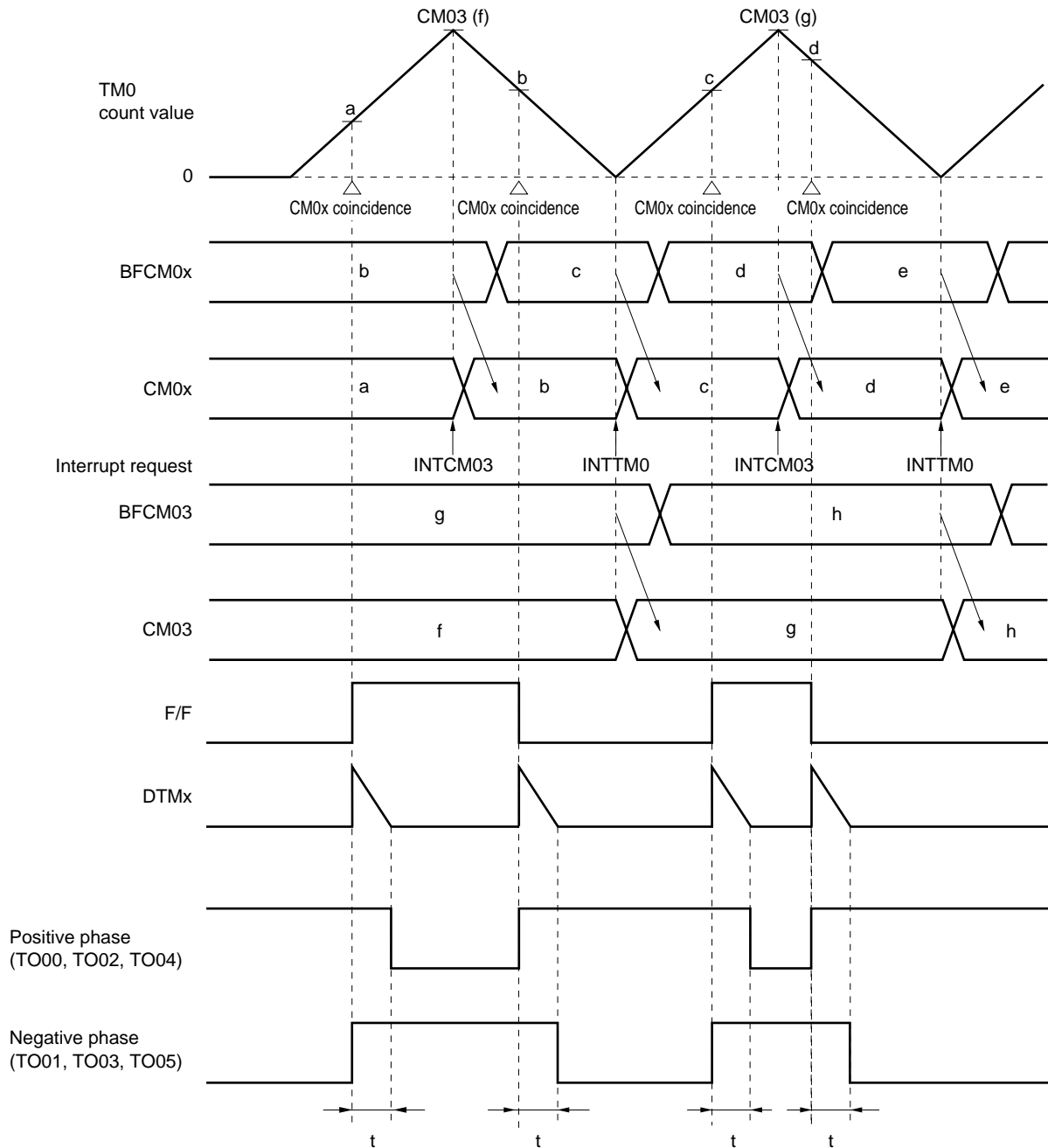
The TO00-TO05 pins are set in the input port mode and go into a high-impedance state on reset. When these pins are set in the control mode later, they output the following levels until TM0 is started:

- TO00, TO02, TO04 ... High level when they are low active
Low level when they are high active
- TO01, TO03, TO05 ... Low level when they are low active
High level when they are high active

The active level is set by the ALVTO bit of the TUM0 register. The default is low active.

Caution If values are set such that the active width of the positive phase or negative phase becomes “0” or “minus” according to the above expressions, pins TO00 to TO05 output a waveform with “0” active width and fixed at the inactive level.

Figure 7-13. Operation Timing in PWM Mode 0 (asymmetric triangular wave)



Remarks 1. The above figure is the timing chart when the transfer operation from BFCM03 to CM03 is enabled with B3TR of the TMC0 register = 1. When B3TR = 0, no transfer is performed and the up/down cycle of TM0 is set by directly writing to CM03.

2. $x = 0-2$

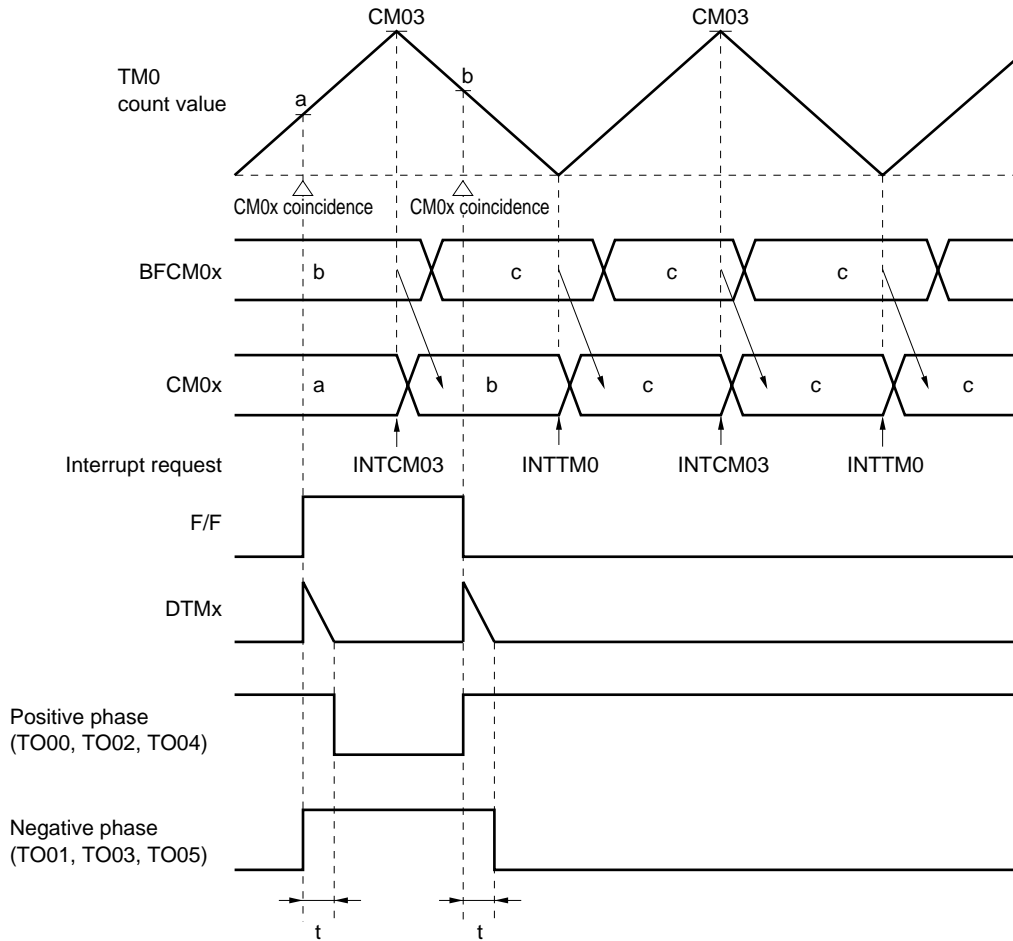
3. $t: \text{dead time} = (\text{DTIME} + 1) \times T_{\text{CLK}}$
(T_{CLK} : system clock rate)

4. Not to take the dead time, clear the CED bit of the TMC1 register to 0.

5. The above figure shows an active low case.

(a) When $BFCM0x \geq CM03$ is set via software started by $INTCM03$

Figure 7-14. Operation Timing in PWM Mode 0 (asymmetric triangular wave, $BFCM0x \geq CM03$)



Remarks 1. $x = 0-2$

2. $c \geq CM03$

3. t : dead time = $(DTIME + 1) \times T_{CLK}$ (T_{CLK} : system clock rate)

4. The above figure shows an active low case.

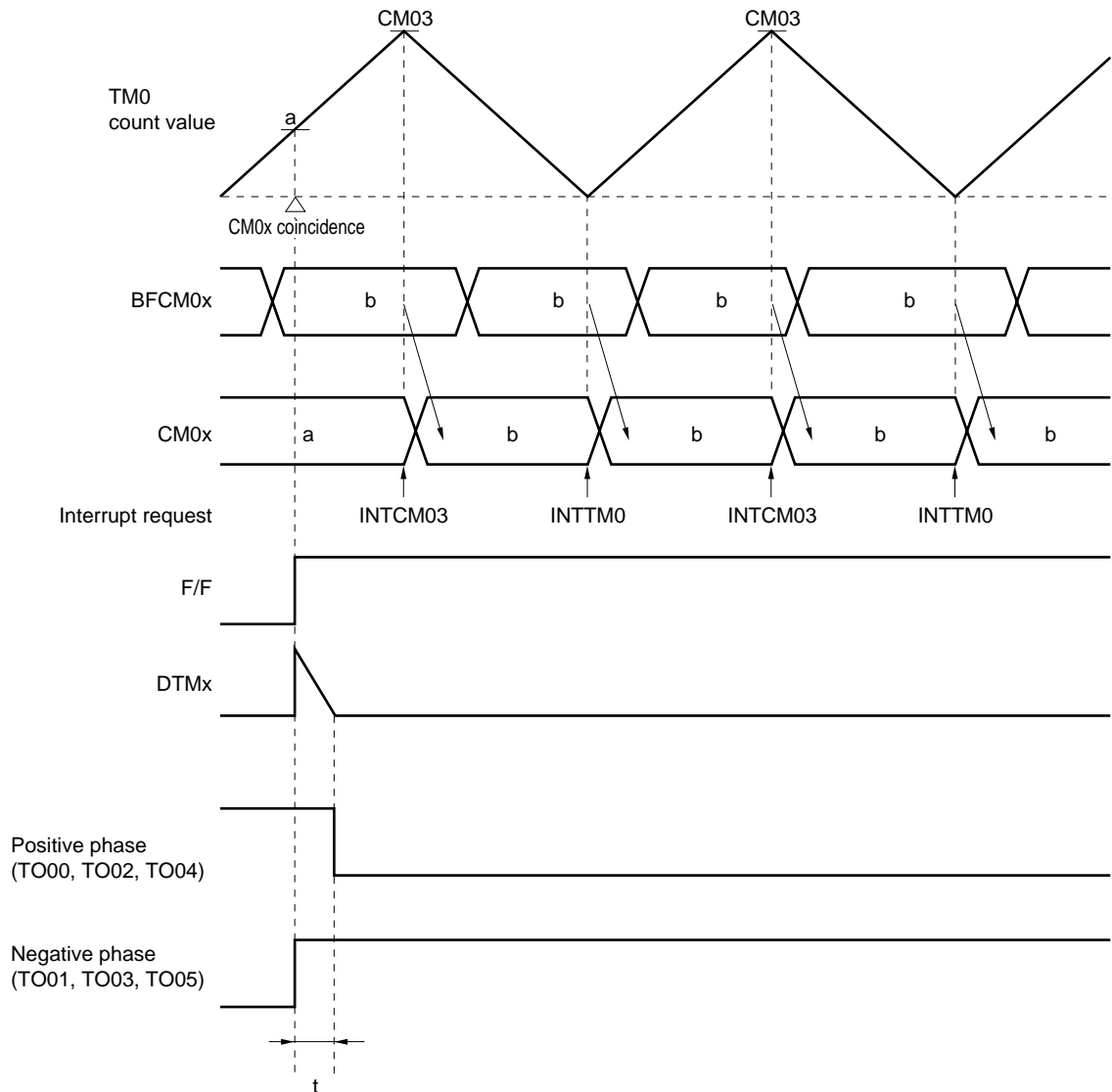
If a value greater than that of CM03 is set to BFCM0x, the positive-phase pins (TO00, TO02, TO04) continue outputting high level, and the negative-phase pins (TO01, TO03, TO05) continue outputting low level. This feature is effective for outputting a low-level or high-level width exceeding the PWM cycle in an application such as inverter control.

When $CM0x = CM03$, coincidence between TM0 and CM0x is detected when TM0 counts down. Therefore, F/F remains reset and is not set.

The above explanation applies to an active low case. In an active high case, the levels of positive and negative phases are merely inverted and other operations remain the same.

(b) When $BFCM0x > CM03$ is set via software processing started by $INTTM0$

Figure 7-15. Operation Timing in PWM Mode 0 (asymmetric triangular wave, $BFCM0x > CM03$)



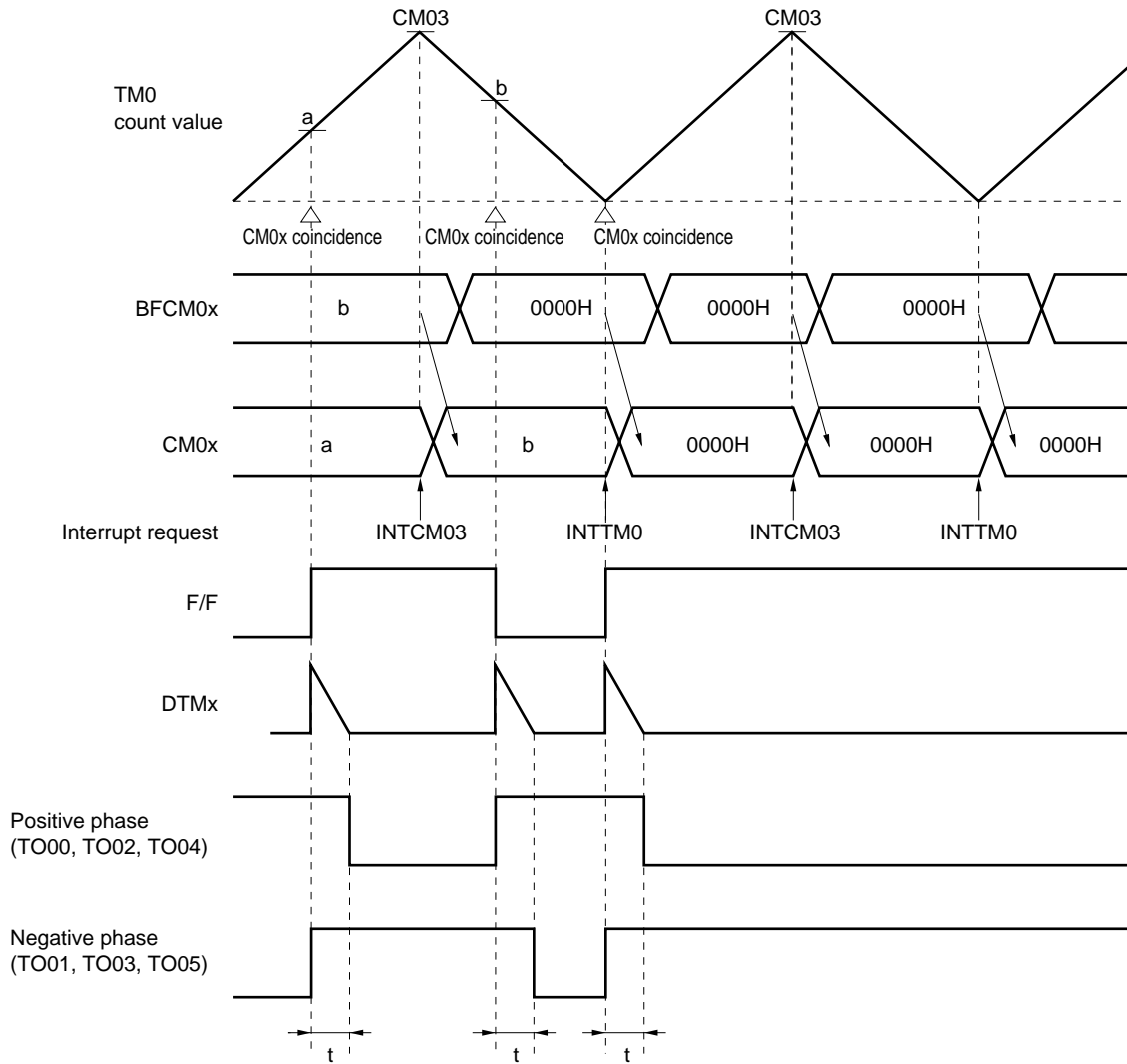
- Remarks**
1. $x = 0-2$
 2. $b > CM03$
 3. $t: \text{dead time} = (DTIME + 1) \times T_{CLK}$ (T_{CLK} : system clock rate)
 4. The above figure shows an active low case.

If a value greater than that of $CM03$ is set to $BFCM0x$, the positive-phase pins (TO00, TO02, TO04) continue outputting high level, and the negative-phase pins (TO01, TO03, TO05) continue outputting low level. This feature is effective for outputting a low-level or high-level width exceeding the PWM cycle in an application such as inverter control.

The above explanation applies to an active low case. In an active high case, the levels of positive and negative phases are merely inverted and other operations remain the same.

(c) When BFCM0x = 0000H is set via software processing started by INTCM03

Figure 7-16. Operation Timing in PWM Mode 0 (asymmetric triangular wave, BFCM0x = 0000H) (1)



Remarks 1. x = 0-2

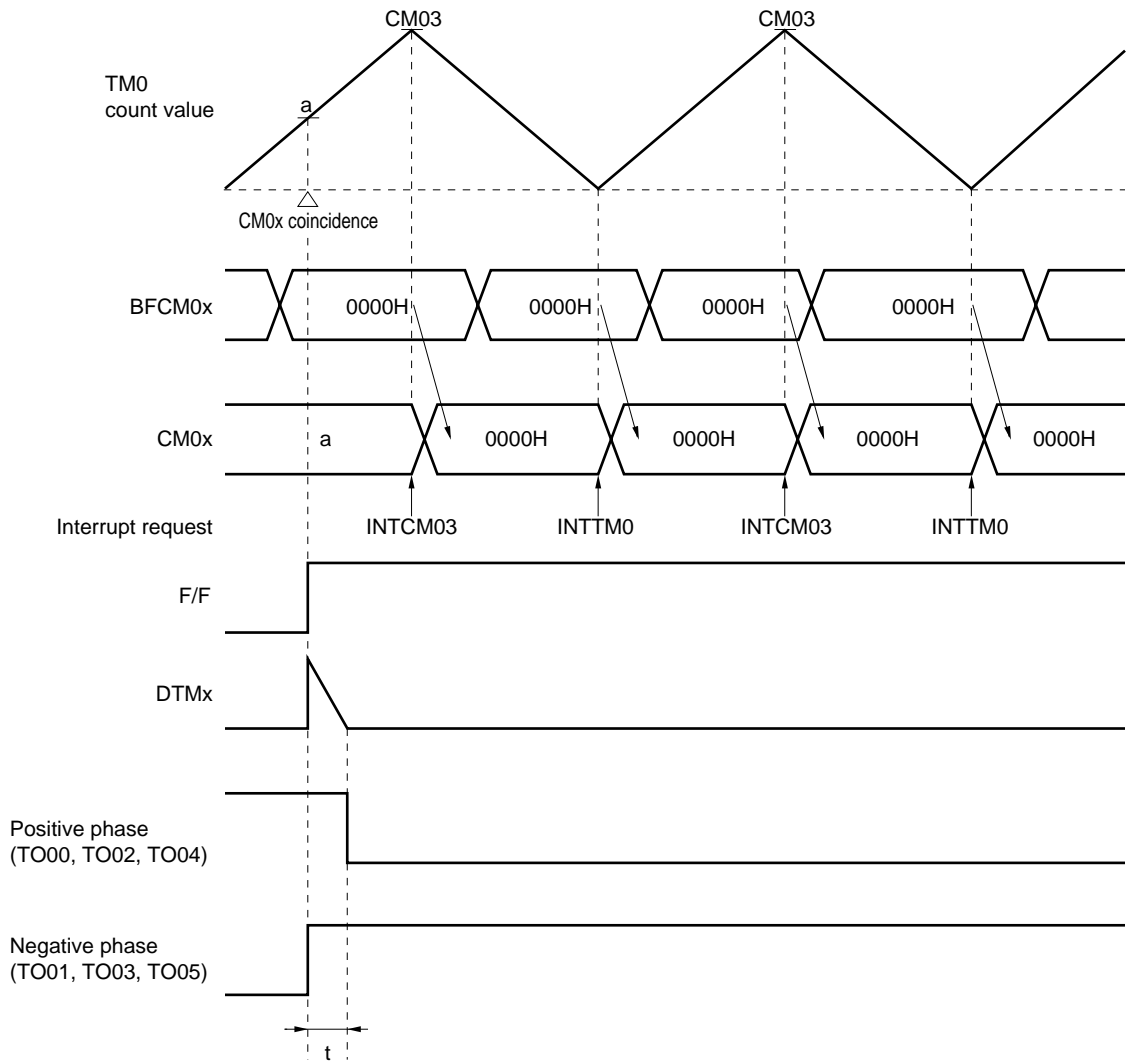
2. t: dead time = (DTIME + 1) × T_{CLK} (T_{CLK}: system clock rate)

3. The above figure shows an active low case.

Since coincidence of TM0 = CM0x = 0000H is detected when TM0 counts up, F/F is only set and is not reset. Furthermore, a coincidence is detected and the F/F is set in the cycle in which 0000H is transferred to CM0x by INTTM0.

(d) When $BFCM0x = 0000H$ is set via software processing started by $INTTM0$

Figure 7-17. Operation Timing in PWM Mode 0 (asymmetric triangular wave, $BFCM0x = 0000H$) (2)



Remarks 1. $x = 0-2$

2. $t: \text{dead time} = (DTIME + 1) \times T_{CLK}$ (T_{CLK} : system clock rate)

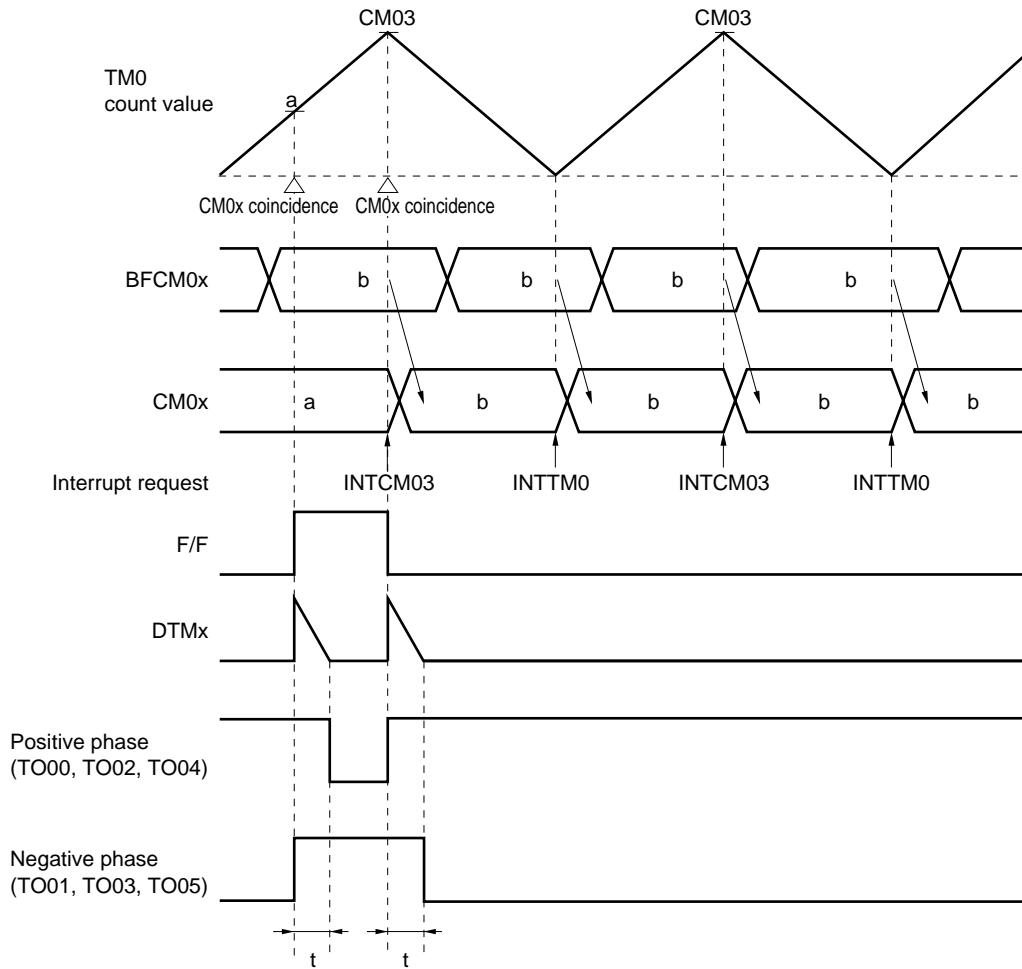
3. The above figure shows an active low case.

Since coincidence of $TM0 = CM0x = 0000H$ is detected when $TM0$ counts up, F/F is only set and is not reset. Therefore, the positive-phase pins (TO00, TO02, TO04) output low level, and the negative-phase pins (TO01, TO03, TO05) continue outputting high level.

The above explanation applies to an active low case. In an active high case, the levels of positive and negative phases are merely inverted and other operations remain the same.

(e) When $BFCM0x = CM03$ is set via software processing started by $INTTM0$

Figure 7-18. Operation Timing in PWM Mode 0 (asymmetric triangular wave, $BFCM0x = CM03$)



Remarks 1. $x = 0-2$

2. $b = CM03$

3. t : dead time = $(DTIME + 1) \times T_{CLK}$ (T_{CLK} : system clock rate)

4. The above figure shows an active low case.

When $BFCM0x = CM03$, coincidence between TM0 and CM0x is detected when TM0 counts down. Therefore, F/F remains reset and is not set. Consequently, the positive-phase pins (TO00, TO02, TO04) output high level, and the negative-phase pins (TO01, TO03, TO05) continue outputting low level. The coincidence with TM0 when CM0x = CM03 is performed in the cycle in which data is transferred from BFCM0x to CM0x by $INTCM03$.

The above explanation applies to an active low case. In an active high case, the levels of positive and negative phases are merely inverted and other operations remain the same.

(5) PWM mode 0 ... toothed wave modulation**[Setting procedure]**

- (a) Set pins P80 to P85 to the control mode (TO00 to TO05 outputs) by the port 8 mode control register (PMC8) (bits PMC80 to PMC85 = 1).
- (b) Set PWM mode 0 (toothed wave) by bits RMOD and TMOD00 to TMOD02 of the TUM0 register. Furthermore, set the active level of pins TO00 to TO05 by the ALVTO bit of the TUM0 register.
- (c) Set the count clock of TM0 by bits PRM02 to PRM00 of the TMC0 register, and set the manipulation for writing to CM00 to CM02 by the CMWE bit. Furthermore, set the transfer operation from BFCM03 to CM03 by the B3TR bit.
- (d) Set the initial values.
 - (i) Put the half-cycle width of the 1st PWM cycle in CM03.
 - PWM cycle = CM03 value $\times 2 \times$ TM0 clock rate (The clock rate of TM0 is set by the TMC0 register.)
 - (ii) Put the half-cycle width of the 2nd PWM cycle in BFCM03. (Setting is not necessary because BFCM03 is not used when B3TR of the TMC0 register = 0)
 - (iii) Put the dead time width in DTIME.
 - Dead time width = (DTIME + 1) \times T_{CLK}
T_{CLK}: System clock rate
 - (iv) Put the set/reset timing of the F/F used in the 2nd cycle in BFCM00 to BFCM02.

Caution The status of F/F does not change during the first cycle even if CM00-CM02 coincide. Therefore, the positive-phase pins (TO00, TO02, TO04) remain inactive level and the negative-phase pins (TO01, TO03, TO05) remain active level. The PWM wave can be output from the second cycle. The active level is set by the ALVTO bit of the TUM0 register.

- (e) Set (1) the CED bit of the TMC1 register to enable operation of the dead time timer. If it is desirable not to take the dead time, set CED = 0.
- (f) Setting (1) the CE0 bit of the TMC0 register starts the counting of TM0, and a 6-channel PWM signal is also output from pins TO00 to TO05.
If direct writing to CM00 to CM02 is not performed by an instruction during the operation, reset (0) the CMWE bit of the TMC0 register before starting the timer.

Caution Setting of CM03 = 0000H is prohibited.

[Operation]

In this mode, TM0 performs up count operation and if it coincides with CM03, it generates coincidence interrupt INTCM03 and clears TM0.

The PWM cycle in this mode is $(\text{CM03 value} + 1) \times \text{TM0 clock rate}$. Data setting in CM03 varies as follows depending on the setting of the B3TR bit of the TMC0 register.

- B3TR = 0 : No transfer from BFCM03 to CM03 is performed. Execute operations by the software processing started by INTCM03 and directly set the cycle data in CM03.
- B3TR = 1 : The BFCM03 data is automatically transferred by the hardware to CM03 by INTCM03. Then, execute operations by the software processing started by INTCM03 and set the data in the next cycle in BFCM03.

Data settings in CM00 to CM02, which control the PWM duty width, are explained below.

Concerning data settings in CM00 to CM02, the values of BFCM00 to BFCM02 are automatically transferred by the hardware to CM00 to CM02 by INTCM03. Then, execute operations by the software processing started by INTCM03 and put the reset timing of the F/F in the next cycle in BFCM00 to BFCM02.

When CMWE of the TMC0 register = 0, direct writing to CM00 to CM02 is not possible. If CMWE = 1, direct writing is possible. However, data transfer from BFCM00 through BFCM02 to CM00 through CM02 by INTCM03 is performed irrespective of the setting of the CMWE bit.

The PWM cycle and PWM duty width are set in the above procedure.

The set/reset condition of the F/F that changes with a coincidence in CM00 to CM02 is as follows.

- Set : Detection of coincidence between TM0 and CM03
- Reset : Detection of coincidence between TM0 and CM00 to CM02

The value of DTIME is loaded to the corresponding dead time timer (DTM0-DTM2) in synchronization with the set/reset timing of F/F, and the dead time timer starts counting down. DTM0-DTM2 count down to 000H and stop when they count down further to 3FFH. DTM0-DTM2 can automatically generate a width (dead time) at which the active levels of the positive (TO00, TO02, TO04) phase and negative phase (TO01, TO03, TO05) do not overlap.

In this way, the software processing is started by an interrupt (INTCM03) that occurs once during PWM cycle (one cycle) after initial setting has been performed, and by setting the PWM cycle and PWM duty width used for the next cycle later to the PWM waveform can be output to the TO00-TO05 pins with a dead time width taken into consideration.

[Output waveform width in respect to set value]

- PWM cycle = $(CM03+1) \times T_{TM0}$
- Dead time width $T_{DTM} = (DTIME + 1) \times T_{CLK}$
- Active width of positive phase (TO00, TO02, TO04 pins) = $(CM0X + 1) \times T_{TM0} - T_{DTM}$
- Active width of negative phase (TO01, TO03, TO05 pins) = $(CM03 - CM0X) \times T_{TM0} - T_{DTM}$

T_{CLK} : system clock rate

T_{TM0} : input clock rate of TM0

CM0X : set values of CM00-CM02

The TO00-TO05 pins are set in the input port mode and go into a high-impedance state on reset.

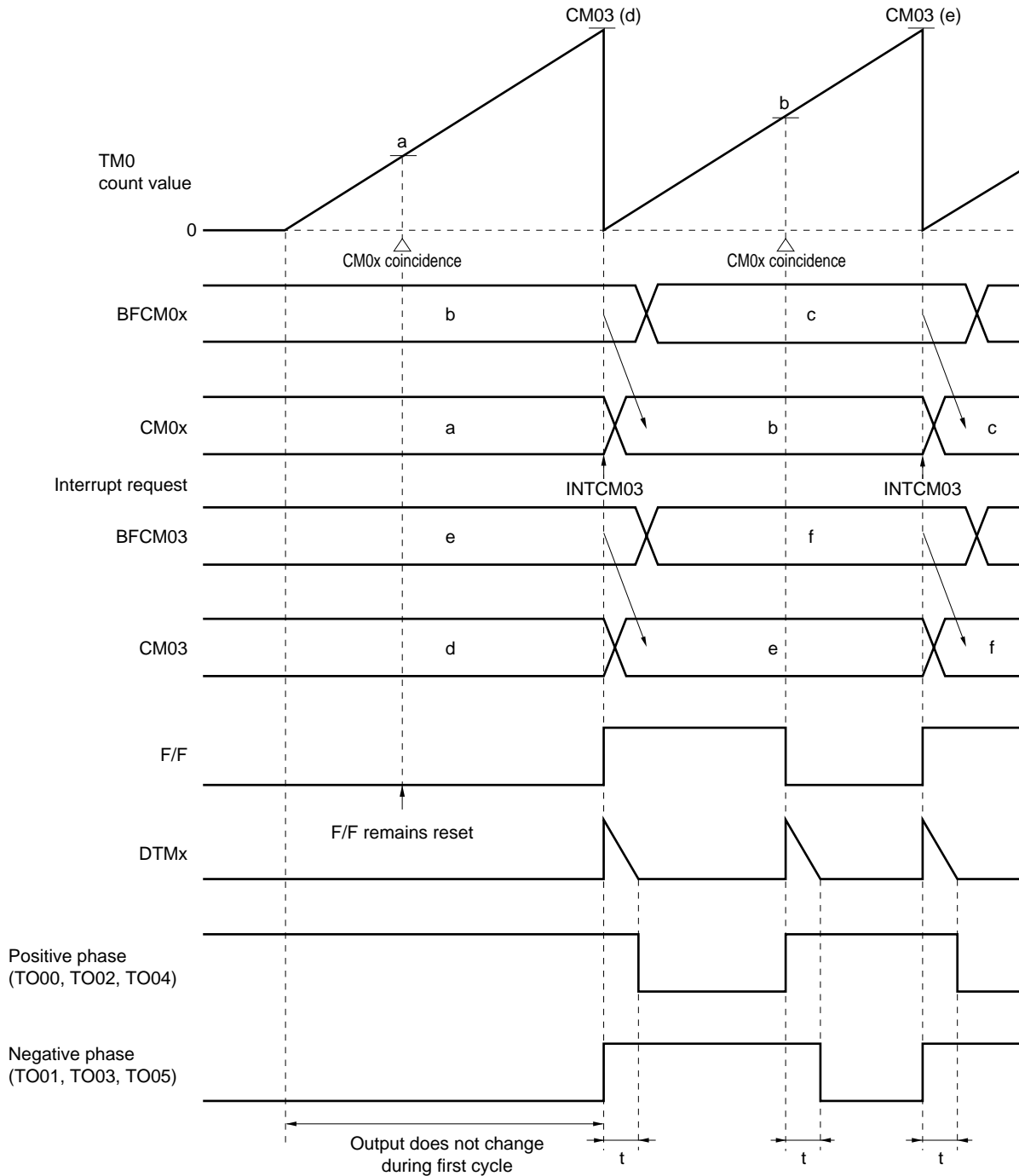
When these pins are set in the control mode later, they output the following levels until TM0 is started:

- TO00, TO02, TO04 ... High level when they are low active
Low level when they are high active
- TO01, TO03, TO05 ... Low level when they are low active
High level when they are high active

The active level is set by the ALVTO bit of the TUM0 register. The default is low active.

Caution If values are set such that the active width of the positive phase or negative phase becomes “0” or “minus” according to the above expressions, pins TO00 to TO05 output a waveform with “0” active width and fixed at the inactive level.

Figure 7-19. Operation Timing in PWM Mode 0 (toothed wave)

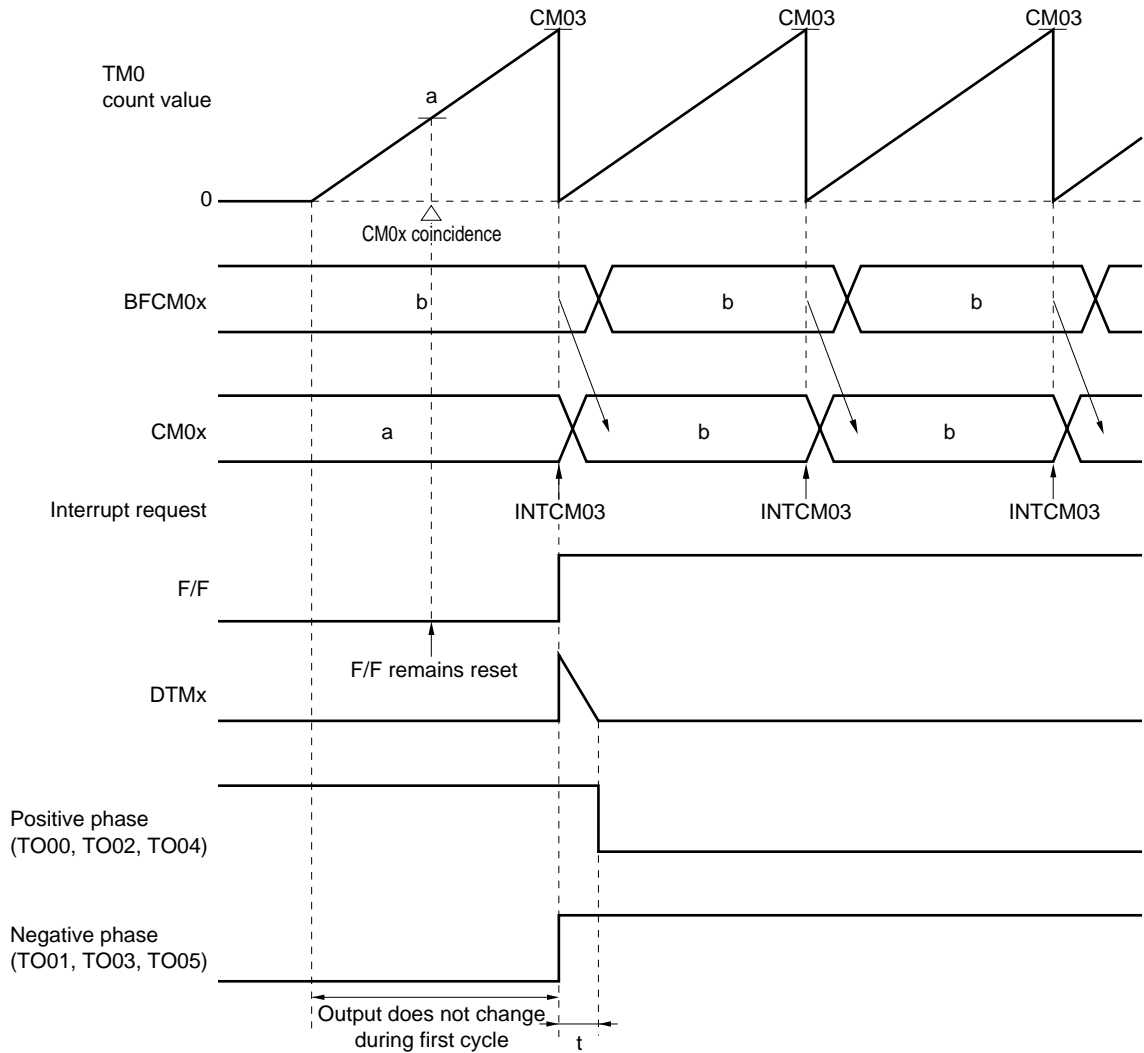
**Remarks 1.** $x = 0-2$

2. t : dead time = $(DTIME + 1) \times T_{CLK}$ (T_{CLK} : system clock rate)
3. Not to take the dead time, clear the CED bit of the TMC1 register to 0.
4. The above figure shows an active low case.

Since F/F is reset when TM0 coincides with CM0x, the output level does not change during the first cycle of TM0.

(a) When BFCM0x > CM03 is set

Figure 7-20. Operation Timing in PWM Mode 0 (toothed wave, BFCM0x > CM03)



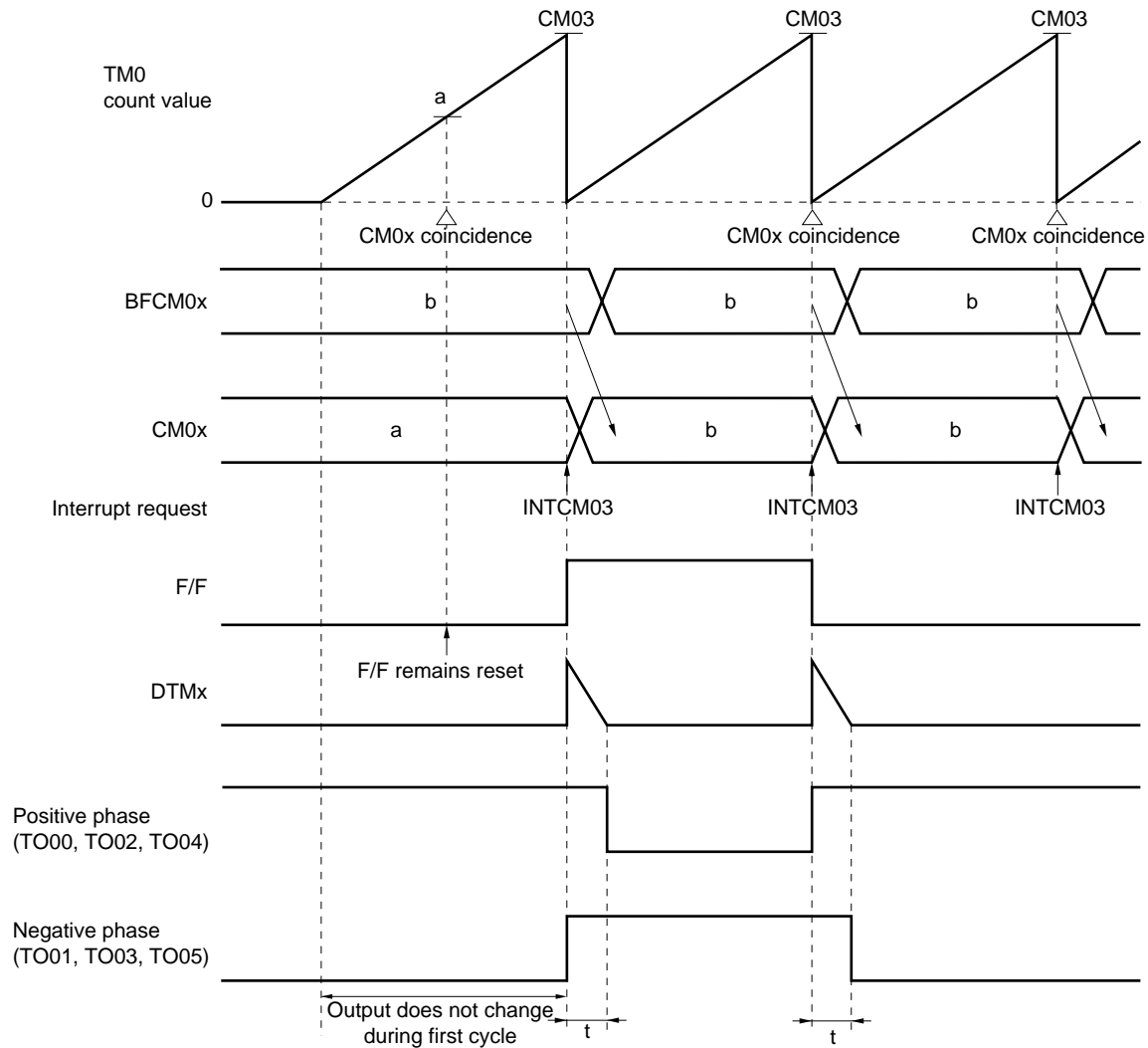
- Remarks**
1. x = 0-2
 2. b > CM03
 3. t: dead time = (DTIME + 1) × T_{CLK} (T_{CLK}: system clock rate)
 4. The above figure shows an active low case.

If a value greater than that of CM03 is set to BFCM0x, the positive-phase pins (TO00, TO02, TO04) continue outputting low level, and the negative-phase pins (TO01, TO03, TO05) continue outputting high level. Because TM0 and CM0x do not coincide, F/F is not reset. This feature is effective for outputting a low-level or high-level width exceeding the PWM cycle in an application such as inverter control.

The above explanation applies to an active low case. In an active high case, the levels of positive and negative phases are merely inverted and other operations remain the same.

(b) When BFCM0x = CM03 is set

Figure 7-21. Operation Timing in PWM Mode 0 (toothed wave, BFCM0x = CM03)

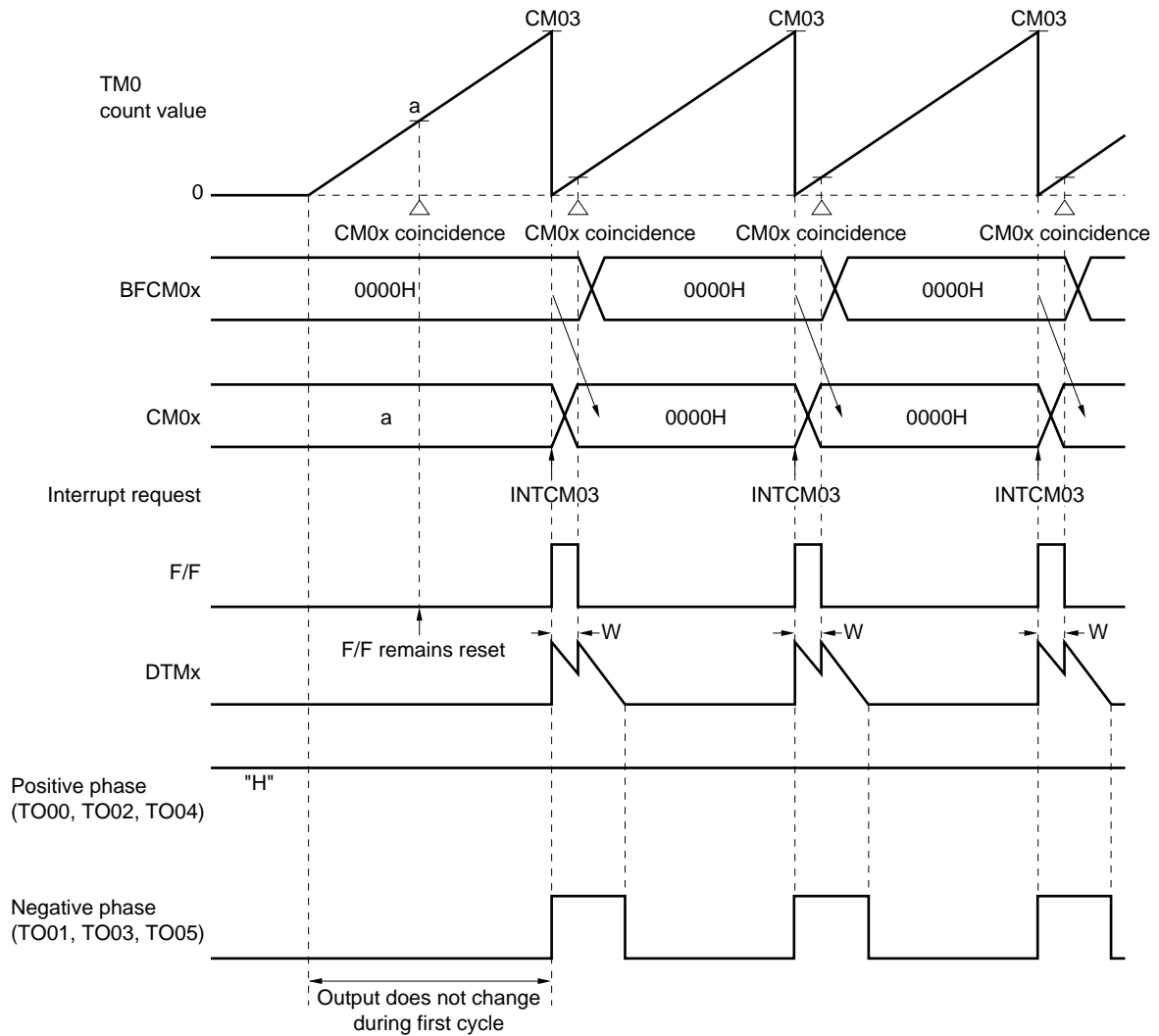
**Remarks 1.** $x = 0-2$ 2. $b > CM03$ 3. $t: \text{dead time} = (DTIME + 1) \times T_{CLK}$ (T_{CLK} : system clock rate)

4. The above figure shows an active low case.

If INTCM03, coincidence signal between TM0 and CM03, contends with coincidence between TM0 and CM0x, resetting F/F takes precedence. Therefore, F/F is not set after CM0x (= CM03) has coincided with TM0.

(c) When BFCM0x = 0000H is set

Figure 7-22. Operation Timing in PWM Mode 0 (toothed wave, BFCM0x = 0000H)



Remarks 1. x = 0 to 2

2. The above figure shows an active low case.

3. W: Width from CM03 coincidence to CM0x coincidence (timer count clock)

When CM0x = 0000H is set, the output waveform varies depending on the TM0 count clock rate and the DTIME set value. If the result of calculating both according to the expression for the active width is "0" or minus, the output will be a waveform without the active width (on the positive phase side) as shown in the figure above. If the result of the active width calculation is positive, the active width is output according to the result.

(6) PWM mode 1 (buffer mode)**[Setting procedure]**

- (a) Set pins P80 to P85 to the control mode (TO00 to TO05 outputs) by the port 8 mode control register (PMC8) (bits PMC80 to PMC85 = 1).
- (b) Set PWM mode 1 by bits RMOD and TMOD00 to TMOD02 of the TUM0 register. Furthermore, setting of the active level by the ALVTO bit of the TUM0 register is not necessary because this setting has no effect in PWM mode 1.
- (c) Set the count clock of TM0 by bits PRM02 to PRM00 of the TMC0 register, and set the manipulation for writing to CM00 to CM02 by the CMWE bit. Furthermore, set the transfer operation from BFCM03 to CM03 by the B3TR bit.
- (d) Set the initial values.
 - (i) Put the half-cycle width of the 1st PWM cycle in CM03.
 - PWM cycle = CM03 value $\times 2 \times$ TM0 clock rate (The clock rate of TM0 is set by the TMC0 register.)
 - (ii) Put the half-cycle width of the 2nd PWM cycle in BFCM03.
(Setting is not necessary because BFCM03 is not used when B3TR of the TMC0 register = 0)
 - (iii) Put the dead time width in DTIME.
 - Dead time width = (DTIME + 1) \times T_{CLK}
T_{CLK}: System clock rate
 - (iv) Put the set timing of the F/F used in the 1st cycle in CM00 to CM02.
 - Direct writing of CMWE of the TMC0 register = 0 to CM00 to CM02 is disabled.
If an instruction for writing to CM00 to CM02 is executed, data in BFCM00 to BFCM02 is transferred to CM00 to CM02. Use the following procedure for setting.

<1> Write the values of CM00 to CM02 used in the 1st cycle to BFCM00 to BFCM02.

<2> If an instruction for writing to CM00 to CM02 is executed, the values of BFCM00 to BFCM02 set in <1> are transferred to CM00 to CM02.
 - Direct writing of CMWE of the TMC0 register = 1 to CM00 to CM02 is enabled.
 - (v) Put the reset timing of the F/F used in the 2nd cycle in BFCM00 to BFCM02.
 - (vi) Put the output data of pins TO00 to TO05 used in the 1st cycle in SBUF0 to SBUF5.
 - (vii) Put the output data of pins TO00 to TO05 used in the 2nd cycle in MBUF0 to MBUF5.

- (e) Set (1) the CED bit of the TMC1 register to enable the operation of dead time timer. If it is desired not to take the dead time, set CED = 0.
- (f) Setting (1) the CE0 bit of the TMC0 register starts the counting of TM0, and a 6-channel PWM signal is output from pins TO00 to TO05.
If direct writing to CM00 to CM02 is not performed by an instruction during the operation, reset (0) the CMWE bit of the TMC0 register before starting the timer.

Cautions 1. Setting of CM03 = 0000H is prohibited.

- 2. SBUF0 to SBUF5 and MBUF0 to MBUF5 are 8-bit access registers. However, only the low-order 6 bits are output to TO00 to TO05. The high-order 2 bits are ignored (fixed at "0" by hardware).

[Operation]

In this mode, TM0 up counts and if it coincides with CM03, it generates coincidence interrupt INTCM03 and clears TM0.

Data setting in CM03, which controls the TM0 cycle, varies as follows depending on the setting of the B3TR bit of the TMC0 register.

- B3TR = 0 : No transfer from BFCM03 to CM03 is performed. Execute operations by the software processing started by INTCM03 and directly set the cycle data in CM03.
- B3TR = 1 : The BFCM03 data is automatically transferred by the hardware to CM03 by INTCM03. Then, execute operations by the software processing started by INTCM03 and set the data in the next cycle in BFCM03.

Setting of the PWM output timing (CM00 to CM02) and the output data (SBUF0 to SBUF5) is explained below. Concerning data settings in CM00 to CM02, the values of BFCM00 to BFCM02 are automatically transferred by the hardware to CM00 to CM02 by INTCM03 and the values of MBUF0 to MBUF5 are transferred to SBUF0 to SBUF5. Then, perform the software processing started by INTCM03 to execute operations and set the set/reset timing of the F/F in the next cycle in BFCM00 to BFCM02, and put the output data in the next cycle in MBUF0 to MBUF5.

When CMWE of the TMC0 register = 0, direct writing to CM00 to CM02 is not possible. If CMWE = 1, direct writing is possible. However, data transfer from BFCM00 through BFCM02 to CM00 through CM02 by INTCM03 is performed irrespective of the setting of the CMWE bit.

The PWM cycle, PWM output timing and output data are set in the above procedure.

The set/reset condition of the F/F is detection of a coincidence between TM0 and CM00 to CM02.

The value of DTIME is loaded to the corresponding dead time timer (DTM0-DTM2) in synchronization with the set/reset timing of F/F, and the dead time timer starts counting down. DTM0-DTM2 count down to 000H and stop when they count down further to 3FFH. DTM0-DTM2 can automatically generate a width (dead time) at which the active levels of the positive (TO00, TO02, TO04) phase and negative phase (TO01, TO03, TO05) do not overlap.

The data of SBUF0-SBUF5 are automatically transferred to the output register (TOUT) in synchronization with the operations of DTM0-DTM2. Data in TOUT is output to TO00 to TO05 pins as is.

- DTM0 starts ... SBUF0 is transferred to TOUT
- DTM0 underflows ... SBUF1 is transferred to TOUT
- DTM1 starts ... SBUF2 is transferred to TOUT
- DTM1 underflows ... SBUF3 is transferred to TOUT
- DTM2 starts ... SBUF4 is transferred to TOUT
- DTM2 underflows ... SBUF5 is transferred to TOUT

Furthermore, TOUT is a write-only register and it is possible to directly write to it to change the output data. In this way, the software processing is started by an interrupt (INTCM03) that occurs once during PWM cycle (one cycle) after initial setting has been performed, and by setting the TM0 cycle to be used next, PWM output timing, and output data, the PWM waveform can be output to the TO00-TO05 pins with a dead time width taken into consideration.

[Output waveform width in respect to set value]

To prevent contention among several data when the buffer contents are output and when coincidence is detected simultaneously by two or more compare registers, the following priority is set by the hardware. If data contention occurs, the data with the lower priority is invalid and not output.

SBUF1 > SBUF0 > SBUF3 > SBUF2 > SBUF5 > SBUF4

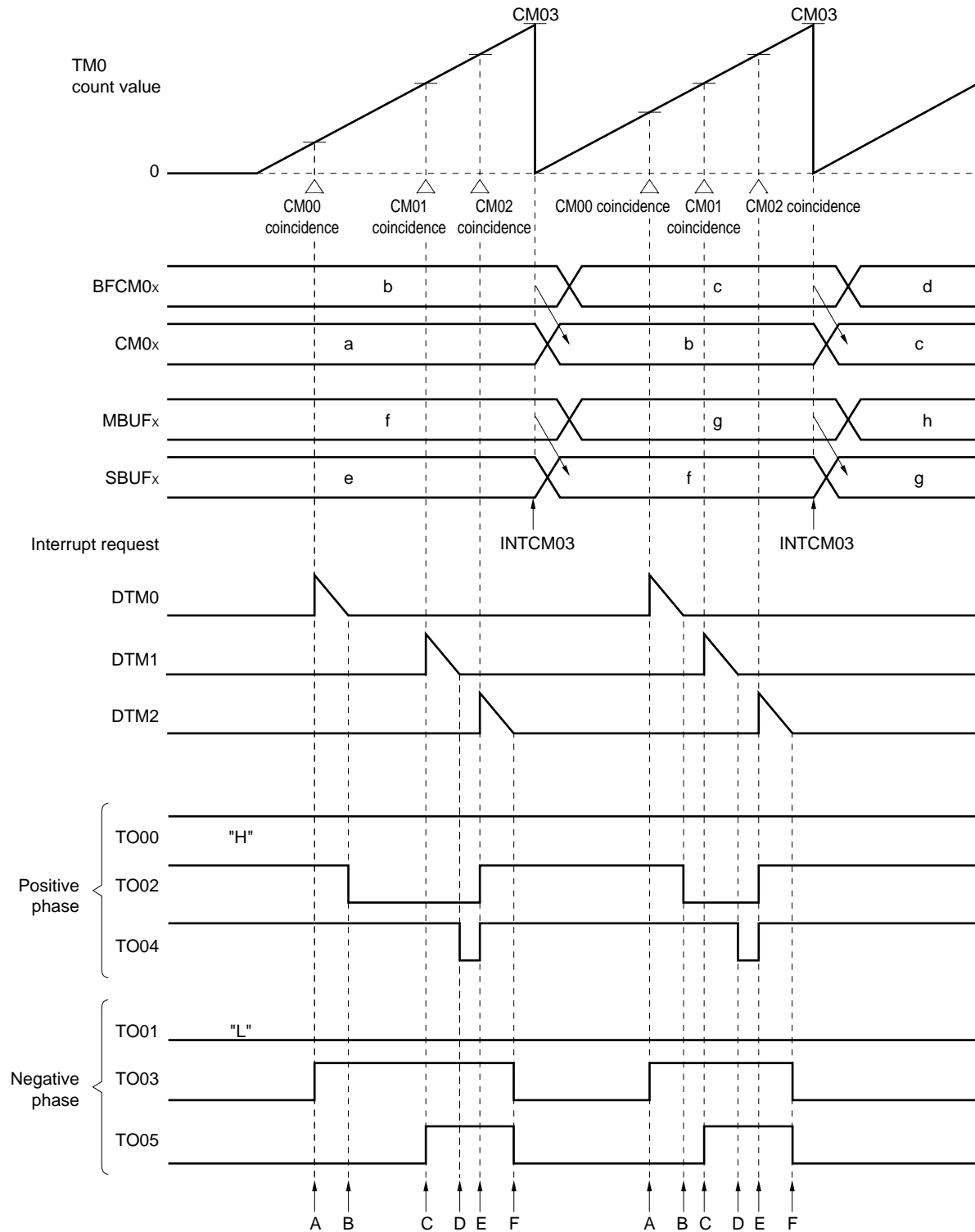
The TO00-TO05 pins are set in the input port mode and go into a high-impedance state on reset.

When these pins are set in the control mode later, they output the following levels until TM0 is started:

- TO00, TO02, TO04 ... high level
- TO01, TO03, TO05 ... low level

Then if data is written in TOUT before starting TM0, the written data is output (TOUT can also be written during the operation of TM0 and can directly change the output data).

Figure 7-23. Operation Timing in PWM Mode 1



Remarks 1. A: SBUF0 B: SBUF1 C: SBUF2

D: SBUF3 E: SBUF4 F: SBUF5

2. The above figure shows an active low case.

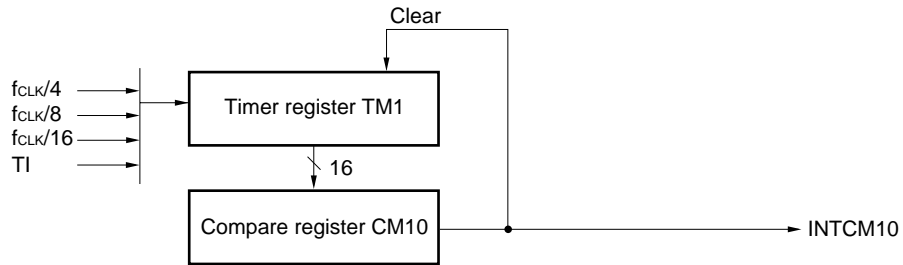
7.3 Timer 1

7.3.1 Configuration

Timer 1 consists of a 16-bit timer 1 (TM1) and a 16-bit compare register (CM10).

Figure 7-24 shows the block diagram of timer 1.

Figure 7-24. Block Diagram of Timer 1



Remark fCLK: internal system clock

(1) 16-bit timer 1 (TM1)

TM1 is a 16-bit interval timer and counts the internal clock and the external events input from the TI pin.

TM1 is cleared by the next count clock which coincides with the value of the compare register (CM10).

All the bits of TM1 are cleared to 0 by the $\overline{\text{RESET}}$ input.

(2) 16-bit compare register 10 (CM10)

CM10 is a 16-bit register that always compares its value with the value of TM1. When the two values coincide, CM10 generates an interrupt signal (INTCM10). When the value of CM10 and the count value of TM1 coincide, TM1 is cleared by the next count clock.

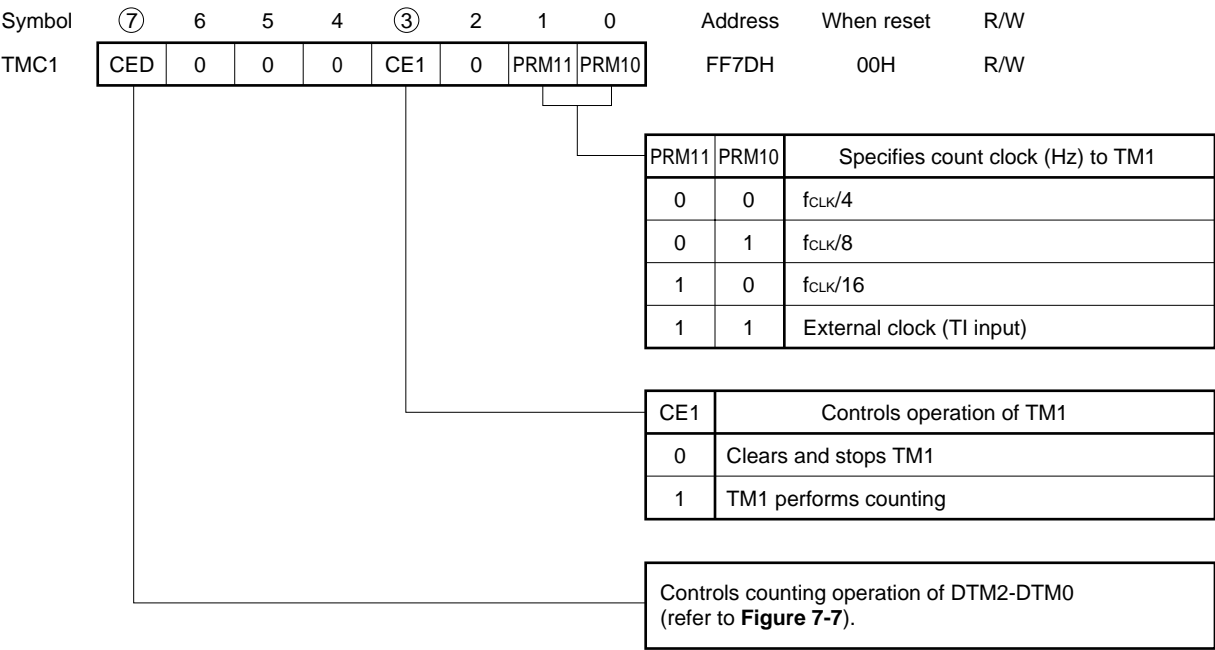
The value of CM10 becomes undefined when the $\overline{\text{RESET}}$ signal is input.

7.3.2 Control registers

(1) Timer control register 1 (TMC1)

Timer control register 1 (TMC1) is an 8-bit register that controls the operations of TM1 and DTM0-DTM2. This register can be read or written by a bit manipulation instruction or an 8-bit manipulation instruction. TMC1 is cleared to 00H by the RESET input.

Figure 7-25. Format of Timer Control Register 1



- Cautions**
1. Bits 6 to 4 and 2 of the TMC1 register are fixed to “0” by hardware. Even if “1” is written, they remain “0”.
 2. It is prohibited to change the PRM11 and PRM10 bits during the operation of TM1 (CE1 = 1).

Remark f_{CLK}: internal system clock

7.3.3 Operation

(1) Basic operation

Timer 1 (TM1) is a 16-bit interval timer/counter.

All the bits of TM1 are cleared to 0 by the $\overline{\text{RESET}}$ input and the timer stops counting.

Counting is enabled or disabled by the CE1 bit of timer control register 1 (TMC1). When the CE1 bit is set to 1 through software, the timer starts counting; when the bit is reset to 0, TM1 is cleared and stops counting. When the value set in advance to the compare register (CM10) coincides with the current count value of TM1, a coincidence interrupt (INTCM10) occurs, and TM1 is cleared.

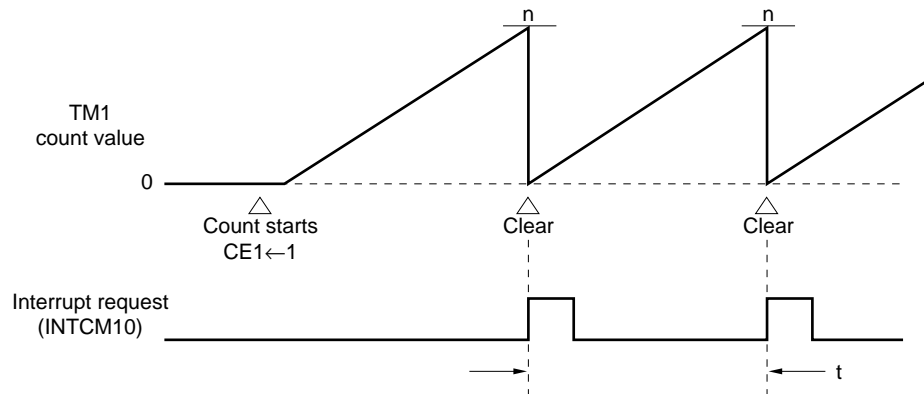
Three internal clocks or external clock input (TI) can be selected by the TMC1 register as the count clock to TM1.

(2) Compare operation

A compare operation is performed that always compares the value of the compare register (CM10) with the count value of timer 1 (TM1).

If the value set in advance to CM10 coincides with the count value of TM1, the interrupt signal INTCM10 is generated, and TM1 is cleared to 0 by the next count clock input. With this feature, TM1 can be used as an interval timer that counts the count clock cycle of the value set to CM10.

Figure 7-26. Example of Compare Operation (TM1, interval timer mode)



Remark n: value of CM10 register

t : interval time = $(n + 1) \times \text{count clock cycle}$

7.4 Timer 2

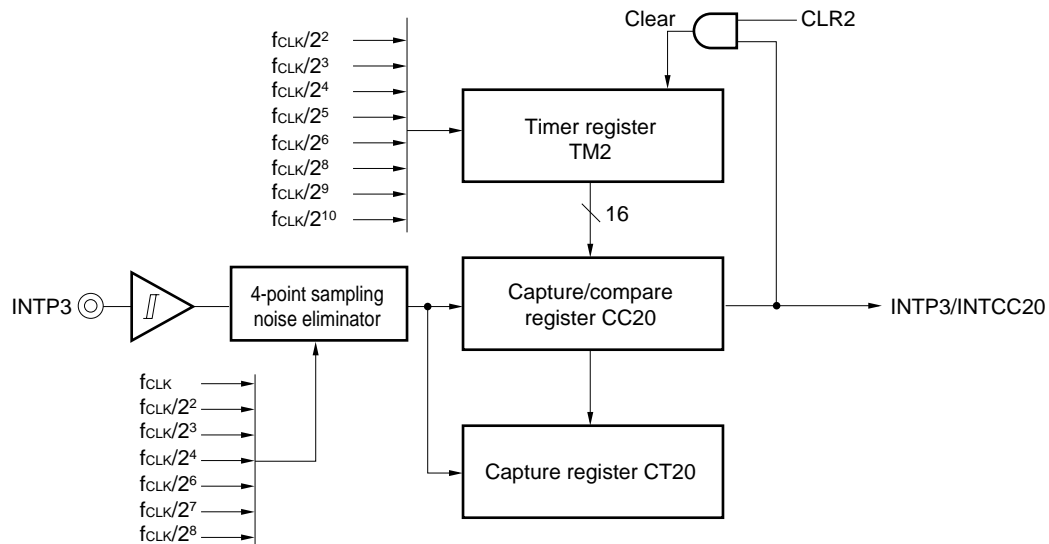
7.4.1 Configuration

Timer 2 consists of a 16-bit timer 2 (TM2), a 16-bit capture /compare register (CC20), and a 16-bit capture register (CT20).

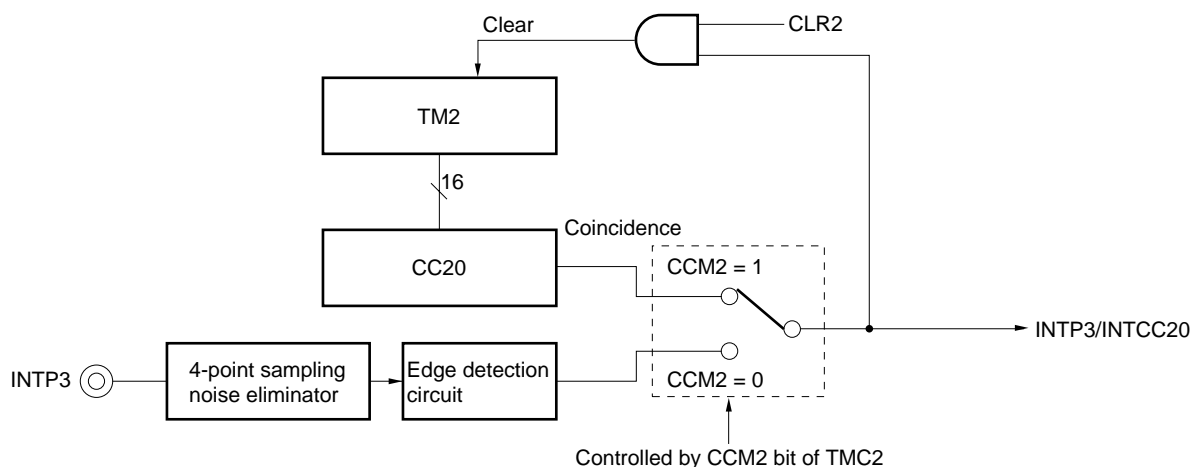
Furthermore, a programmable input sampling circuit is added to the interrupt pin (INTP3) and it can eliminate interrupt signal noise.

Figure 7-27 shows the block diagram of timer 2.

Figure 7-27. Block Diagram of Timer 2



Remark f_{CLK} : internal system clock

Figure 7-28. Block Diagram of INTP3/INTCC20 Generation

Remark When CC20 is used as a compare register (CCM2 = 1), it is not possible to use INTP3 as an external interrupt pin.

(1) 16-bit timer 2 (TM2)

TM2 functions as a 16-bit free running timer or interval timer, and counts the internal clock.

When TM2 serves as an interval timer, TM2 is cleared when the capture/compare register (CC20) generates an interrupt (INTP3/INTCC20) as a result of coincidence of its value with the value of the timer.

All the bits of TM2 are cleared to 0 by the $\overline{\text{RESET}}$ input.

(2) 16-bit capture/compare register 20 (CC20)

CC20 is a 16-bit register that latches the value of TM2 when it detects the valid edge of the corresponding interrupt request signal (INTP3) (capture operation).

This register also operates as a compare register that always compares its value with the value of TM2 (compare operation). When the two values coincide, CC20 generates an interrupt (INTCC20), and TM2 is cleared.

The value of CC20 becomes undefined when the $\overline{\text{RESET}}$ signal is input.

(3) 16-bit capture register 20 (CT20)

CT20 is a 16-bit register that latches the value of TM2 when it detects the valid edge of the corresponding interrupt request signal (INTP3) (capture operation).

The value of CT20 becomes undefined when the $\overline{\text{RESET}}$ signal is input.

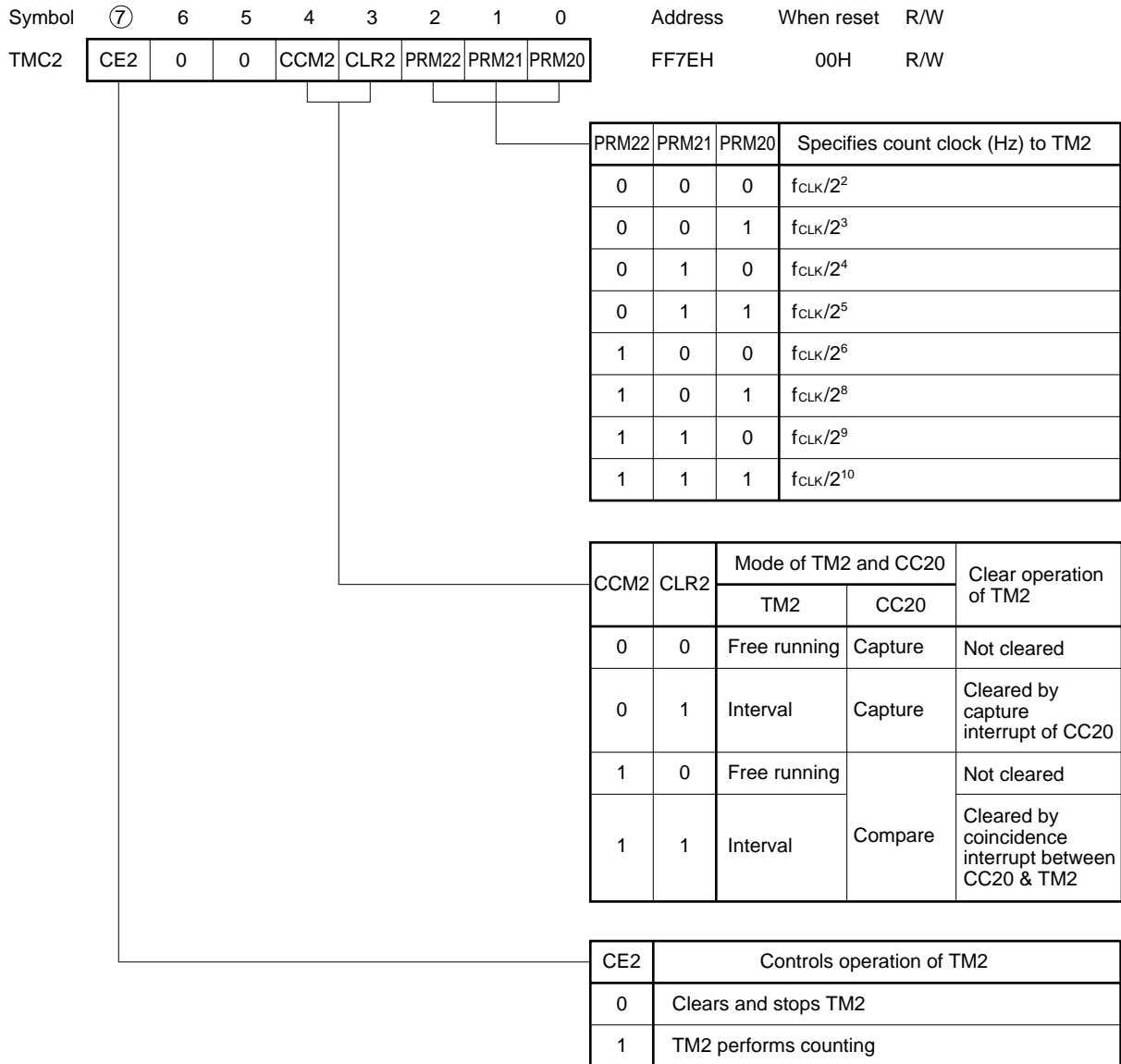
7.4.2 Control registers

(1) Timer control register 2 (TMC2)

Timer control register 2 (TMC2) is an 8-bit timer that controls the operation of TM2 and CC20.

This register can be read or written by a bit manipulation instruction or an 8-bit manipulation instruction.

TMC2 is cleared to 00H by the $\overline{\text{RESET}}$ input.

Figure 7-29. Format of Timer Control Register 2

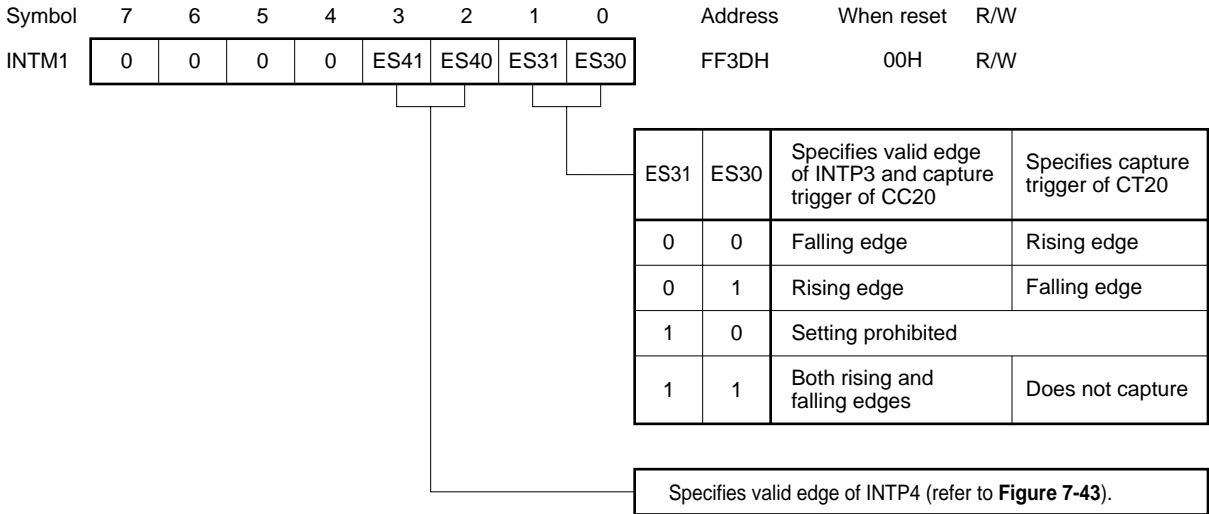
- Cautions**
1. INTP3 cannot be used as an external interrupt pin when CC20 is used as a compare register (CCM2 = 1).
 2. On hardware, the CC20 incorporates the compare register and capture register separately. Either of the two can be selected by setting the TMC2 register. Writing is only possible to the compare register. Reading is possible from the register which is selected.
 3. Bits 6 and 5 of the TMC2 register are fixed to "0" by hardware. Even if "1" is written, they remain "0".
 4. It is prohibited to change the CCM2, CLR2 and PRM22 to PRM20 bits during the operation of TM2 (CE2 = 1).

Remark f_{CLK} : internal system clock

(2) External interrupt mode register 1 (INTM1)

External interrupt mode register 1 (INTM1) is an 8-bit register that specifies the valid edges of INTP3 and INTP4, and the capture triggers of CC20 and CT20. This register can be read or written by a bit manipulation instruction or an 8-bit manipulation instruction. INTM1 is cleared to 00H by the RESET input.

Figure 7-30. Format of External Interrupt Mode Register 1



Caution Bits 7 to 4 of the INTM1 register are fixed to “0” by hardware. Even if “1” is written, they remain “0”.

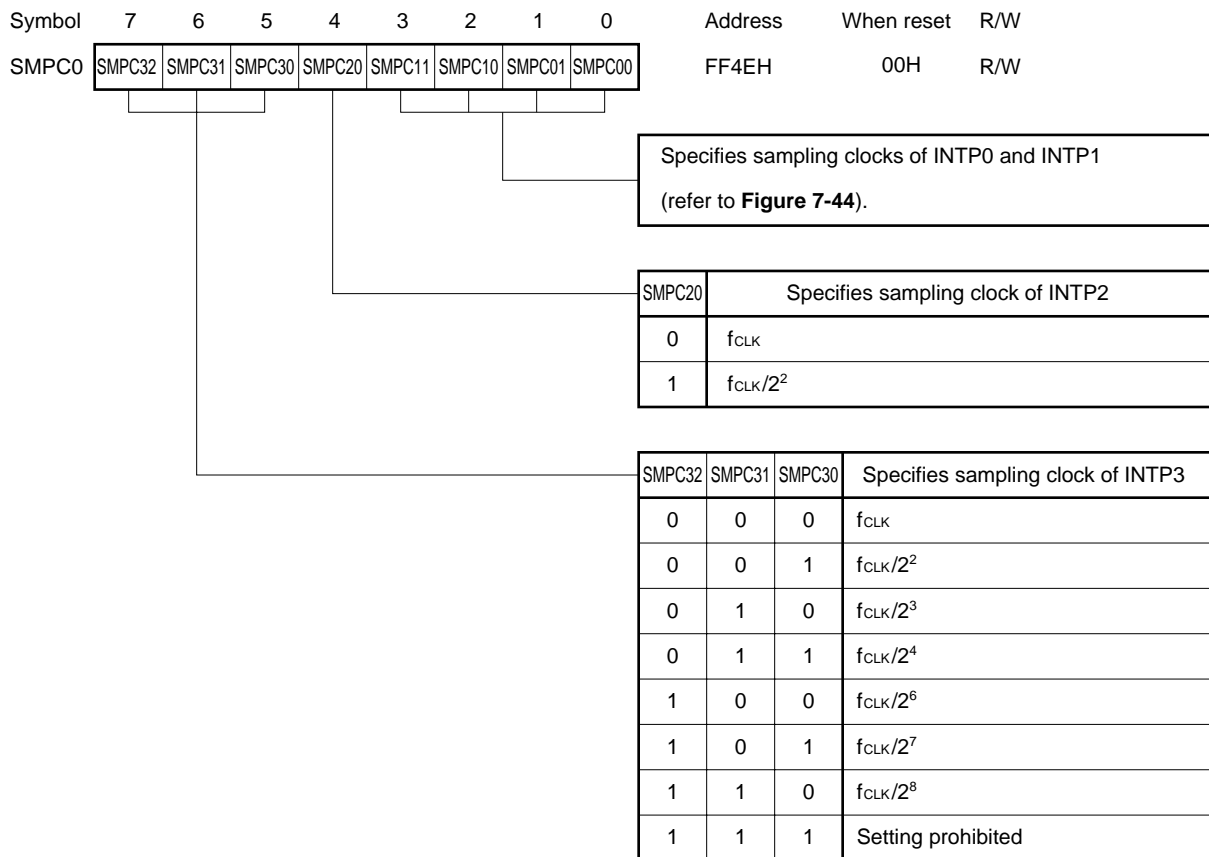
(3) Sampling control register 0 (SMPC0)

Sampling control register 0 (SMPC0) is an 8-bit register that specifies the sampling clocks of INTP0, INTP1, and INTP3.

The sampling circuit performs 4-point sampling with the specified sampling clock, and if the results at the 4 points are at the same level, that level is fetched.

This register can be read or written by a bit manipulation instruction or an 8-bit manipulation instruction.

SMPC0 is cleared to 00H by the $\overline{\text{RESET}}$ input.

Figure 7-31. Format of Sampling Control Register 0

Remark f_{CLK}: internal system clock

7.4.3 Operation**(1) Basic operation**

Timer 2 (TM2) operates as a 16-bit free-running timer or interval timer.

Enable/disable of the count operation is controlled by the CE2 bit of timer control register 2 (TMC2). The count clock for TM2 can be selected from among 8 kinds of internal clock by the TMC2 register.

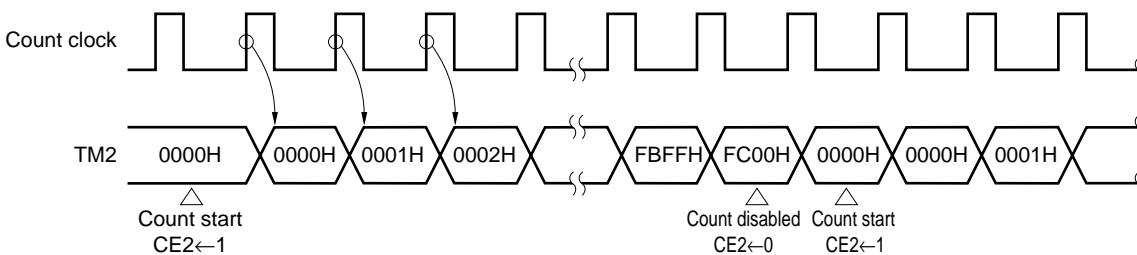
After the CE2 bit is set (1), TM2 becomes 0000H → 0000H by the initial count clock input and becomes 0001H by the 2nd count clock input.

Even if CE2 is set (1) again with CE2 = 1 during the operation of TM2, the count operation continues and the timer is not cleared.

If CE2 = 0 is set, the count operation stops with TM2 = 0000H.

$\overline{\text{RESET}}$ input clears (0) all bits of TM2 and stops the count operation.

Figure 7-32. Basic Operation of Timer 2 (TM2)



(2) Interval operation

If TM2 is operated as an interval timer, set it by the CCM2 and CLR2 bits of the TMC2 register.

There are two triggers that clear TM2 as follows and they are differentiated by the setting of the CCM2 and CLR2 bits.

(a) INTCC20 (CCM2 = 1, CLR2 = 1)

INTCC20 is an interrupt generated when CC20 is in the compare mode. TM2 and CC20 always perform compare operations and if they detect a coincidence they generate interrupt signal INTCC20. The result of a compare coincidence is retained by hardware and TM2 is cleared (0000H) with the next count clock after the coincidence. Furthermore, when the next count clock is input TM2 is counted up to 0001H.

$$\text{Interval cycle} = (\text{CC20 value} + 1) \times \text{TM2 count clock rate}$$

(b) INTP3 (CCM2 = 0, CLR2 = 1)

If TM2 is set to interval operation while CC20 is in the capture mode, TM2 is cleared (0000H) by the valid edge of external interrupt pin INTP3.

The clear timing is the point when valid edge of INTP3 is detected. TM2 counts up to 0001H with the next count clock after the coincidence.

Caution INTCC20 and INTP3 share the interrupt vector table.

(3) Free-running operation (CCM2 = 0/1, CLR2 = 0)

TM2 performs a full count from 0000H to FFFFH. After overflow, TM2 is cleared (0000H) with the next count clock and continues counting thereafter.

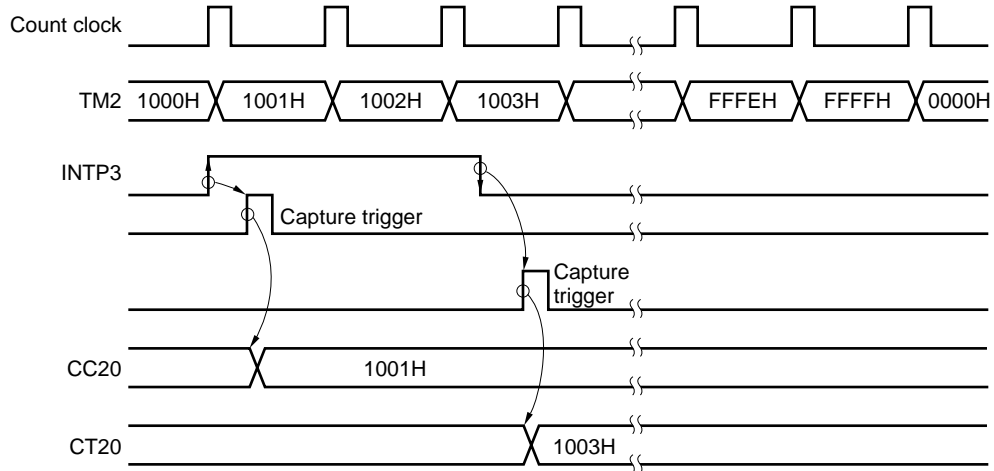
$$\text{Free-running cycle} = 65536 \times \text{TM2 count clock rate}$$

(4) Summary of capture/compare operation

Capture/compare operation is controlled by the CCM2 and CLR2 bits of the TMC2 register.

<1> CCM2 = 0, CLR2 = 0 (TM2 ... free-running timer, CC20 ... capture)

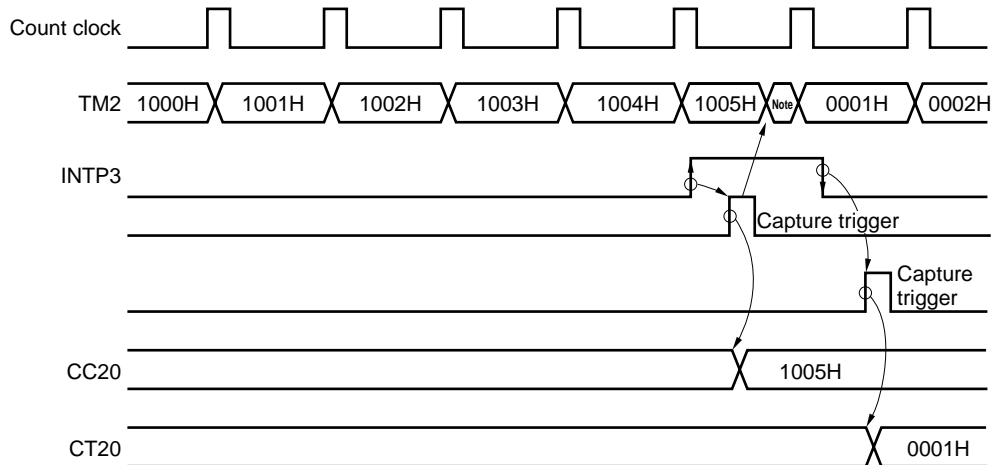
Figure 7-33. Example of TM2 Capture Operation (free-running operation)



Remark The INTP3 (CC20) valid edge is set as the rising edge.

<2> CCM2 = 0, CLR2 = 1 (TM2 ... interval timer, CC20 ... capture)

Figure 7-34. Example of TM2 Capture Operation (interval operation)

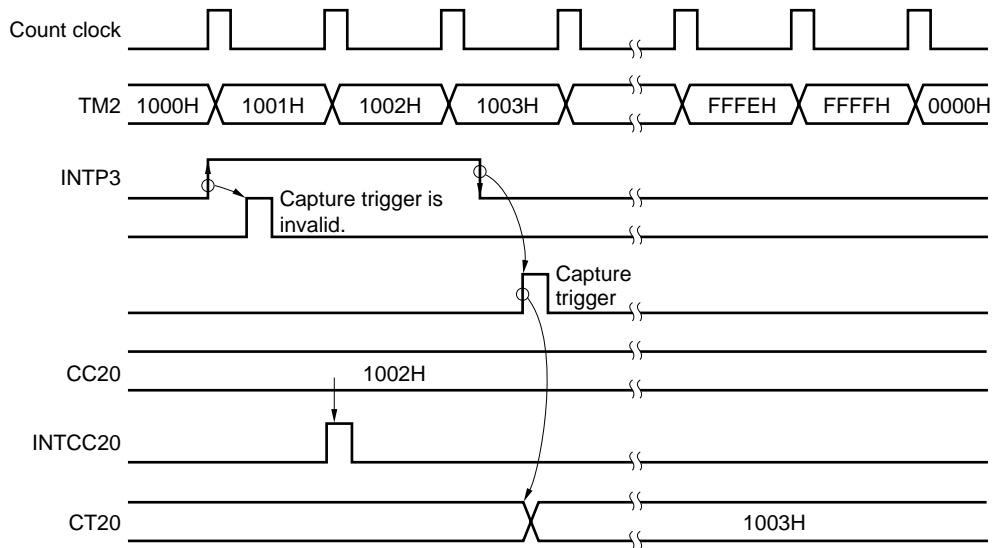


Note 0000H

Remark The INTP3 (CC20) valid edge is set as the rising edge.

<3> CCM2 = 1, CLR2 = 0 (TM2 ... free-running timer, CC20 ... compare)

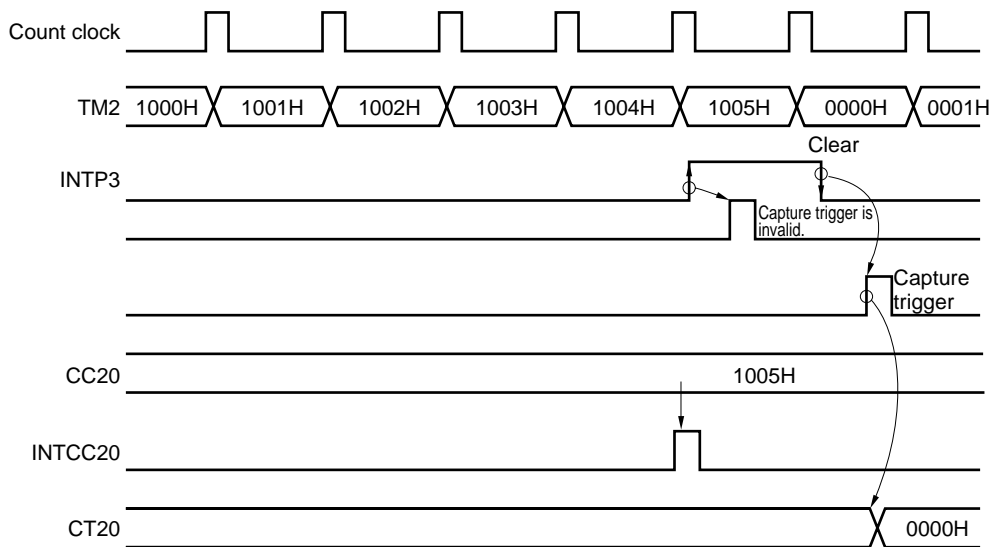
Figure 7-35. Example of TM2 Compare Operation (free-running operation)



Remark The INTP3 (CC20) valid edge is set as the rising edge.

<4> CCM2 = 1, CLR2 = 1 (TM2 ... interval timer, CC20 ... compare)

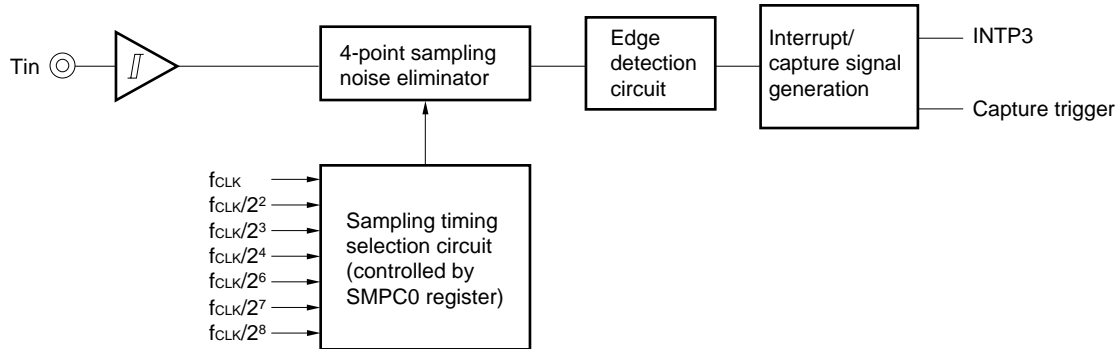
Figure 7-36. Example of TM2 Compare Operation (interval operation)



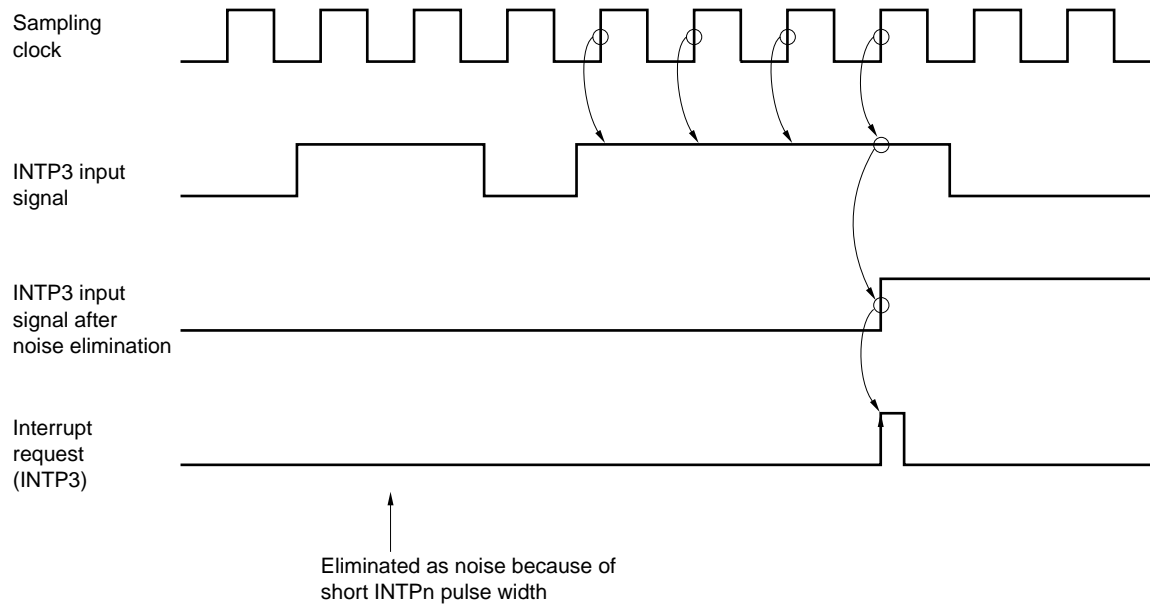
Remark The INTP3 (CC20) valid edge is set as the rising edge.

(5) Operation of sampling circuit

The sampling circuit of the μ PD78366A performs 4-point sampling with the timing specified by the SMPC0 register. If the results are at the same level four consecutive times, that level is fetched inside.

Figure 7-37. Block Diagram of Sampling Circuit (TM2)

Remark f_{CLK} : Internal system clock

Figure 7-38. Sampling Timing Chart (TM2)

Judgment of valid signals by the 4-point sampling is carried out with the following timing.

- <1> $T_{in} \leq (3 \times T_{smp})$... Eliminated as noise
- <2> $(3 \times T_{smp}) < T_{in} < (4 \times T_{smp})$... Eliminated as noise or passed as valid signals depending on the timing
- <3> $T_{in} \geq (4 \times T_{smp})$... Passed as valid signals

T_{in} : Input signal width of INTP3 pin

T_{smp} : Sampling timing

Therefore, it is necessary to input a signal of $4 \times T_{smp}$ in width for the input signal to pass as a valid signal.

7.5 Timer 3

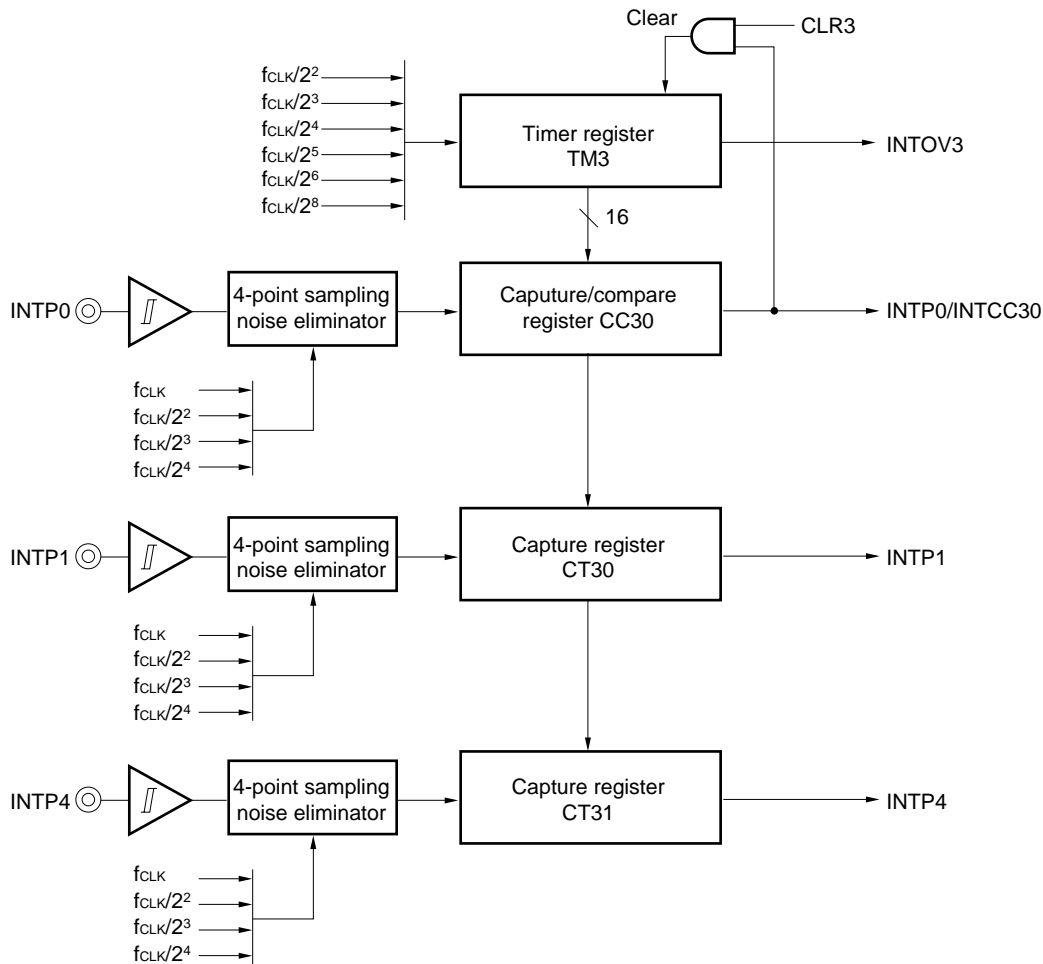
7.5.1 Configuration

Timer 3 consists of a 16-bit timer 3 (TM3), a 16-bit capture /compare register (CC30), and two 16-bit capture registers (CT30 and CT31).

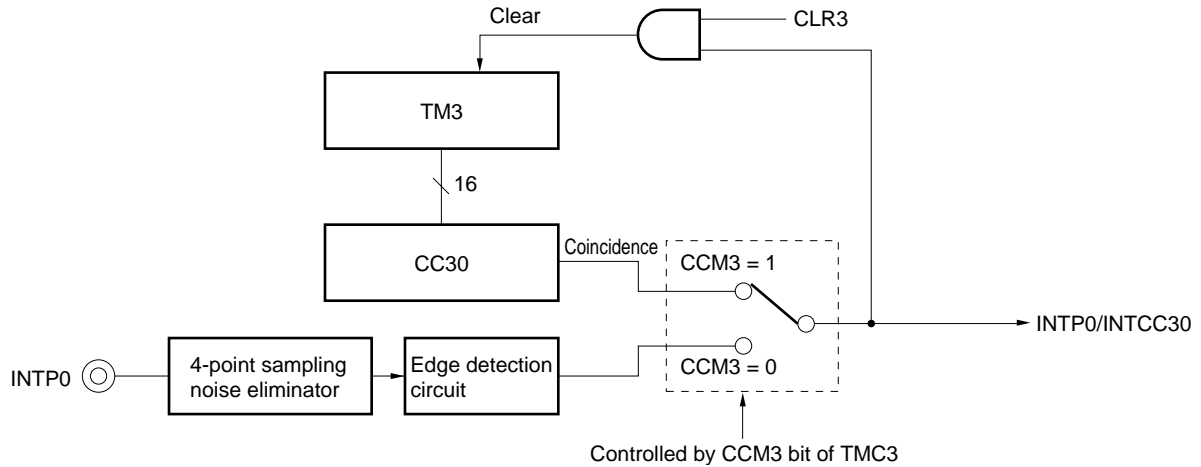
Each interrupt pin (INTP0, INTP1 and INTP4) is provided with a programmable input sampling circuit, and eliminates interrupt signal noise.

Figure 7-39 shows the block diagram of timer 3.

Figure 7-39. Block Diagram of Timer 3



Remark f_{CLK} : internal system clock

Figure 7-40. Block Diagram of INTP0/INTCC30 Generation

Remark When CC30 is used as a compare register (CCM3 = 1), it is not possible to use INTP0 as an external interrupt pin.

(1) 16-bit timer 3 (TM3)

TM3 functions as a 16-bit free running timer or interval timer, and counts the internal clock.

When TM3 serves as an interval timer, TM3 is cleared when the capture/compare register (CC30) generates an interrupt (INTP0/INTCC30) as a result of coincidence of its value with the value of the timer.

All the bits of TM3 are cleared to 0 by the $\overline{\text{RESET}}$ input.

(2) 16-bit capture/compare register 30 (CC30)

CC30 is a 16-bit register that latches the value of TM3 when it detects the valid edge of the corresponding interrupt request signal (INTP0) (capture operation).

This register also operates as a compare register that always compares its value with the value of TM3 (compare operation). When the two values coincide, CC30 generates an interrupt (INTCC30), and TM3 is cleared.

The value of CC30 becomes undefined when the $\overline{\text{RESET}}$ signal is input.

(3) 16-bit capture registers 30 and 31 (CT30 and CT31)

CT30 and CT31 are 16-bit registers that latch the value of TM3 when they detect the valid edge of the corresponding interrupt request signal (INTP1, INTP4) (capture operation).

The values of CT30 and CT31 become undefined when the $\overline{\text{RESET}}$ signal is input.

7.5.2 Control registers

(1) Timer control register 3 (TMC3)

Timer control register 3 (TMC3) is an 8-bit register that controls the operation of TM3 and CC30.

This register can be read or written by a bit manipulation instruction or an 8-bit manipulation instruction.

TMC3 is cleared to 00H by the $\overline{\text{RESET}}$ input.

Figure 7-41. Format of Timer Control Register 3 (TMC3)

Symbol	⑦	6	5	4	3	2	1	0	Address	When reset	R/W
TMC3	CE3	0	0	CCM3	CLR3	PRM32	PRM31	PRM30	FF7FH	00H	R/W

PRM32	PRM31	PRM30	Specifies count clock (Hz) to TM3
0	0	0	$f_{CLK}/2^2$
0	0	1	$f_{CLK}/2^3$
0	1	0	$f_{CLK}/2^4$
0	1	1	$f_{CLK}/2^5$
1	0	0	$f_{CLK}/2^6$
1	0	1	$f_{CLK}/2^8$
1	1	0	Setting prohibited
1	1	1	

CCM3	CLR3	Mode of TM2 CC30		Clear operation of TM3
		TM3	CC30	
0	0	Free running	Capture	Not cleared
0	1	Interval	Capture	Cleared by capture interrupt of CC30
1	0	Free running	Compare	Not cleared
1	1	Interval		Cleared by coincidence interrupt between CC30 & TM3

CE3	Controls operation of TM3
0	Clears and stops TM3
1	TM3 performs counting

- Cautions**
1. INTP0 cannot be used as an external interrupt pin when CC30 is used as a compare register (CCM3 = 1).
 2. In the hardware, the CC30 incorporates the compare register and capture register separately. Either of the two can be selected by the setting of the TMC3 register. Writing is only possible to the compare register. Reading is possible from the register which is selected.
 3. Bits 6 and 5 of the TMC3 register are fixed to “0” by hardware. Even if “1” is written, they remain “0”.
 4. It is prohibited to change the CCM3, CLR3 and PRM32 to PRM30 bits during the operation of TM3 (CE3 = 1).

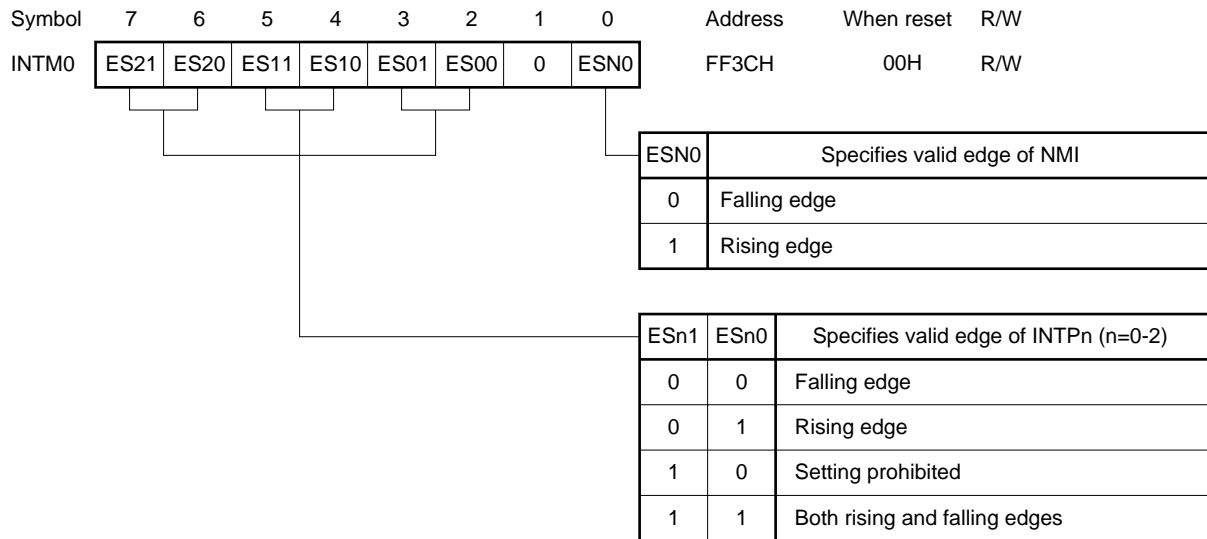
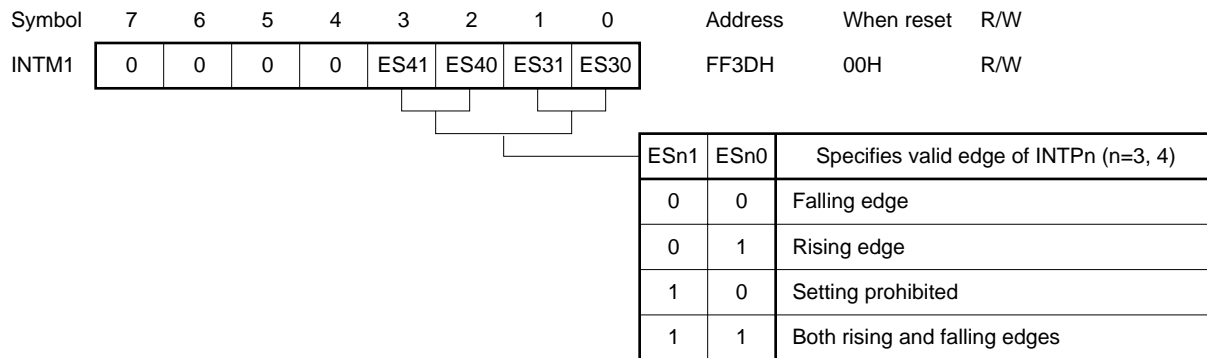
Remark f_{CLK} : internal system clock

(2) External interrupt mode registers (INTM0, INTM1)

External interrupt mode register 0 (INTM0) is an 8-bit register that specifies the valid edges of NMI and INTP0-INTP2.

External interrupt mode register 1 (INTM1) is an 8-bit register that specifies the valid edges of INTP3 and INTP4.

These registers can be read or written by a bit manipulation instruction or an 8-bit manipulation instruction. When the RESET signal is input, INTM0 and INTM1 are cleared to 00H.

Figure 7-42. Format of External Interrupt Mode Register 0**Figure 7-43. Format of External Interrupt Mode Register 1**

Caution Bit 1 of the INTM0 register and bits 7 to 4 of the INTM1 register are fixed to “0” by hardware. Even if “1” is written, they remain “0”.

(3) Sampling control registers (SMPC0 and SMPC1)

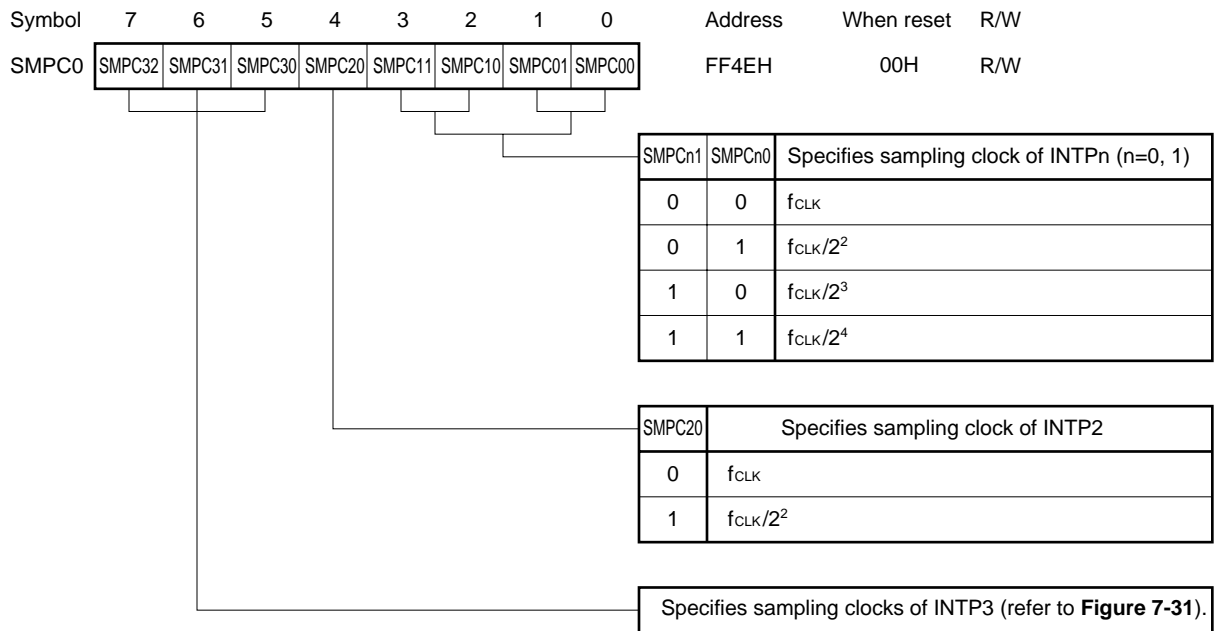
Sampling control register 0 (SMPC0) is an 8-bit register that specifies the sampling clocks of INTP0-INTP3.

Sampling control register 1 (SMPC1) is an 8-bit register that specifies the sampling clock of INTP4.

The sampling circuit performs 4-point sampling with the specified sampling clock and if the results are at the same level four times, that level is fetched.

These registers can be read or written by a bit manipulation instruction or an 8-bit manipulation instruction.

SMPC0 and SMPC1 are cleared to 00H by the $\overline{\text{RESET}}$ input.

Figure 7-44. Format of Sampling Control Register 0

Remark f_{CLK} : internal system clock

Figure 7-45. Format of Sampling Control Register 1

Symbol	7	6	5	4	3	2	1	0	Address	When reset	R/W
SMPC1	0	0	0	0	0	0	SMPC41	SMPC40	FF4FH	00H	R/W

SMPC41	SMPC40	Specifies sampling clock of INTP4
0	0	f _{CLK}
0	1	f _{CLK} /2 ²
1	0	f _{CLK} /2 ³
1	1	f _{CLK} /2 ⁴

Caution Bits 7 to 2 of the SMPC1 register are fixed to “0”. Even if “1” is written, they remain “0”.

Remark f_{CLK}: internal system clock

7.5.3 Operation

(1) Basic operation

Timer 3 (TM3) operates as a 16-bit free-running timer or interval timer.

Counting is enabled or disabled by the CE3 bit of timer control register 3 (TMC3). Six internal clocks can be selected by the TMC3 register as the count clock to TM3.

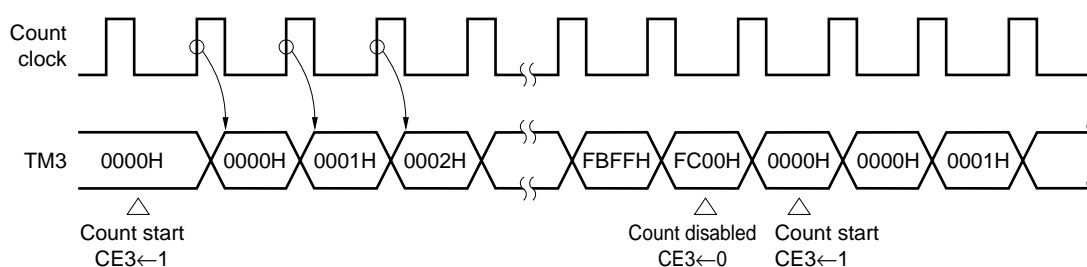
After setting (1) the CE3 bit, TM3 becomes 0000H to 0000H on the first count clock input, and to 0001H on the second count clock input.

If the CE3 is set (1) again with CE3 = 1, during TM3 operation, TM3 continues counting and is not cleared.

When CE3 = 0, TM3 = 0000H stops count operation.

All the bits of TM3 are cleared to 0 by the $\overline{\text{RESET}}$ input and the timer stops counting.

Figure 7-46. Basic Operation of Timer 3 (TM3)



(2) Interval operation

If TM3 is operated as an interval timer, set it by the CCM3 and CLR3 bits of the TMC3 register.

There are two triggers that clear TM3 as follows and they are differentiated by the setting of the CCM3 and CLR3 bits.

(a) INTCC30 (CCM3 = 1, CLR3 = 1)

INTCC30 is an interrupt generated when CC30 is in the compare mode. TM3 and CC30 always perform compare operations and if they detect a coincidence they generate interrupt signal INTCC30. The result of a compare coincidence is retained by hardware and TM3 is cleared (0000H) with the next count clock after the coincidence. Furthermore, when the next count clock is input TM3 is counted up to 0001H.

$$\text{Interval cycle} = (\text{CC30 value} + 1) \times \text{TM3 count clock rate}$$

(b) INTP0 (CCM3 = 0, CLR3 = 1)

If TM3 is set to interval operation while CC30 is in the capture mode, TM3 is cleared (0000H) by the valid edge of external interrupt pin INTP0.

The clear timing is the point when the valid edge of INTP0 is detected. TM3 counts up to 0001H with the next count clock after the coincidence.

Caution INTCC30 and INTP0 share the interrupt vector table.

(3) Free-running operation (CCM3 = 0/1, CLR3 = 0)

TM3 performs a full count from 0000H to FFFFH, and generate an overflow interrupt INTOV3. After overflow, TM3 is cleared (0000H) with the next count clock and continues counting thereafter.

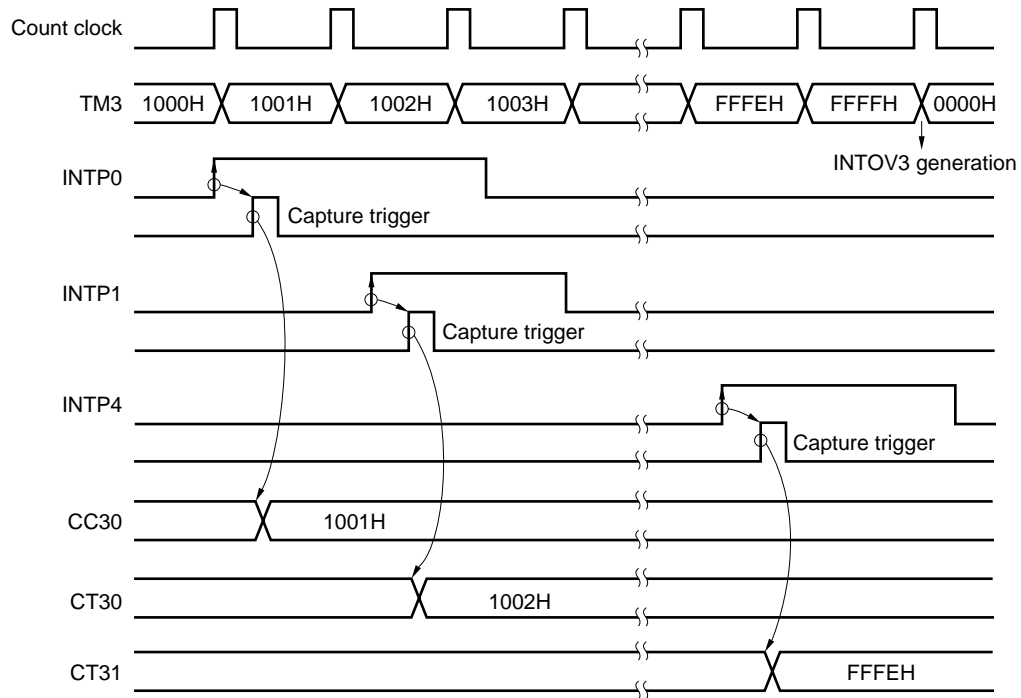
$$\text{Free-running cycle} = 65536 \times \text{TM3 count clock rate}$$

(4) Summary of capture/compare operation

Capture/compare operation is controlled by the CCM3 and CLR3 bits of the TMC3 register.

<1> CCM3 = 0, CLR3 = 0 (TM3 ... free-running timer, CC30 ... capture)

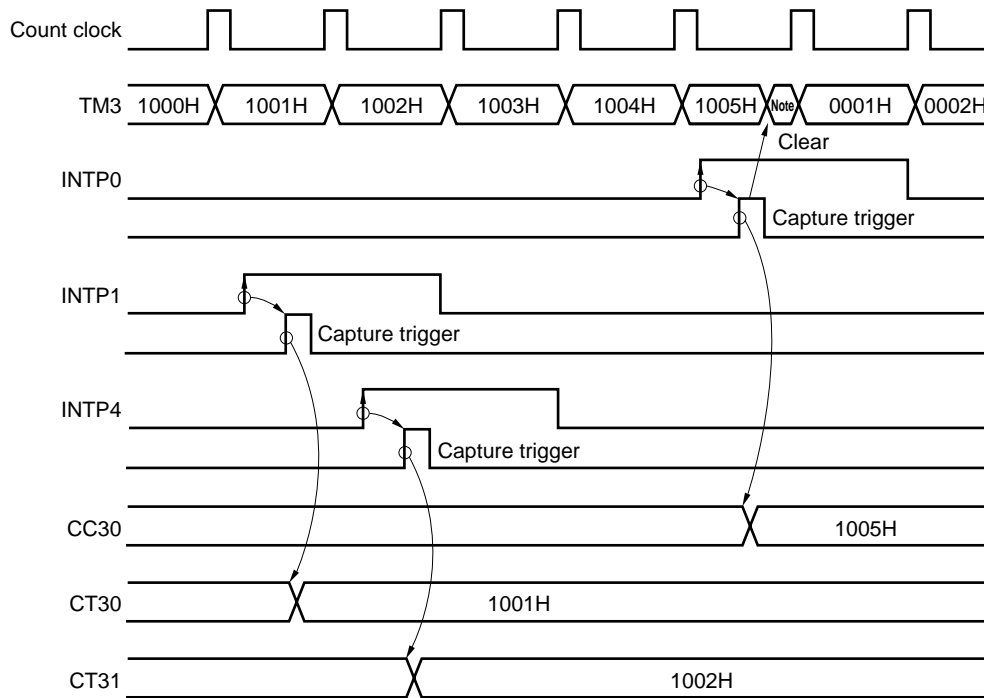
Figure 7-47. Example of TM3 Capture Operation (free-running operation)



Remark The INTP0, INTP1 and INTP4 valid edges are set as the rising edges.

<2> CCM3 = 0, CLR3 = 1 (TM3 ... interval timer, CC30 ... capture)

Figure 7-48. Example of TM3 Capture Operation (interval operation)

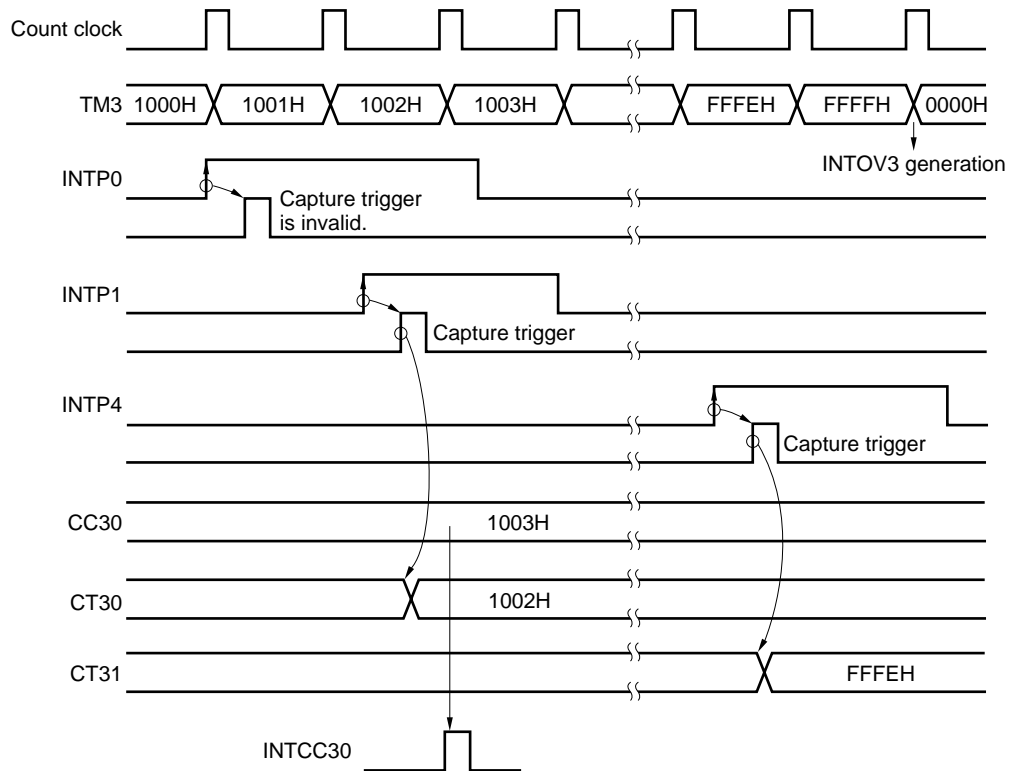


Note 0000H

Remark The INTP0, INTP1 and INTP4 valid edges are set as the rising edges.

<3> CCM3 = 1, CLR3 = 0 (TM3 ... free-running timer, CC30 ... compare)

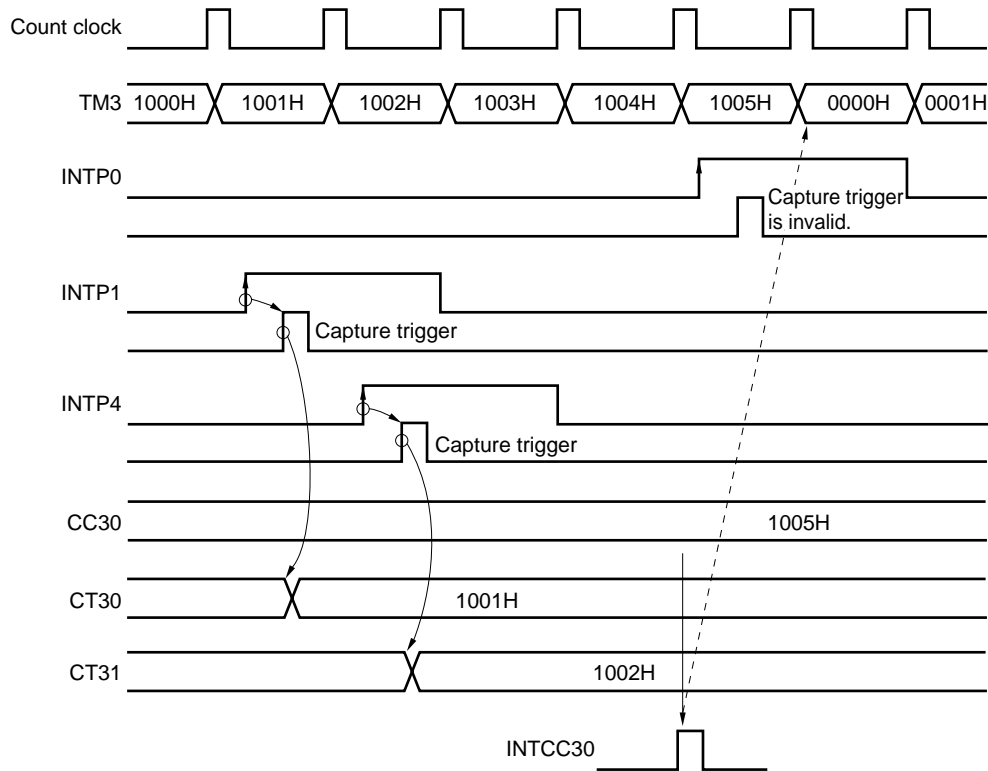
Figure 7-49. Example of TM3 Compare Operation (free-running operation)



Remark The INTP0, INTP1 and INTP4 valid edges are set as the rising edges.

<4> CCM3 = 1, CLR3 = 1 (TM3 ... interval timer, CC30 ... compare)

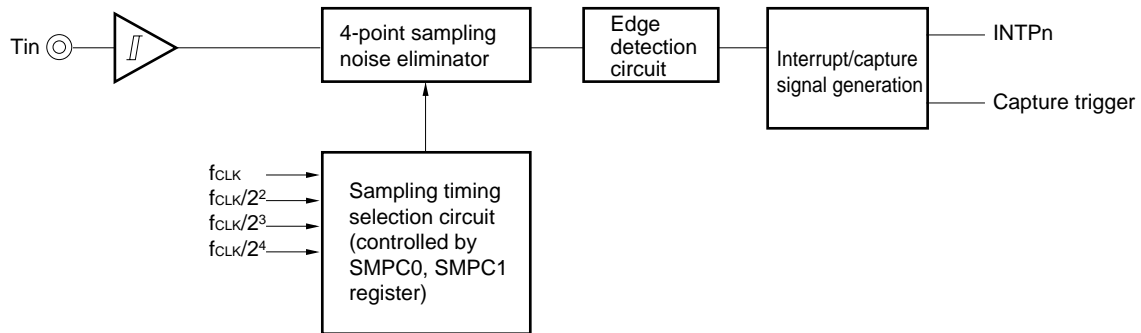
Figure 7-50. Example of TM3 Compare Operation (interval operation)



Remark The INTP0, INTP1 and INTP4 valid edges are set as the rising edges.

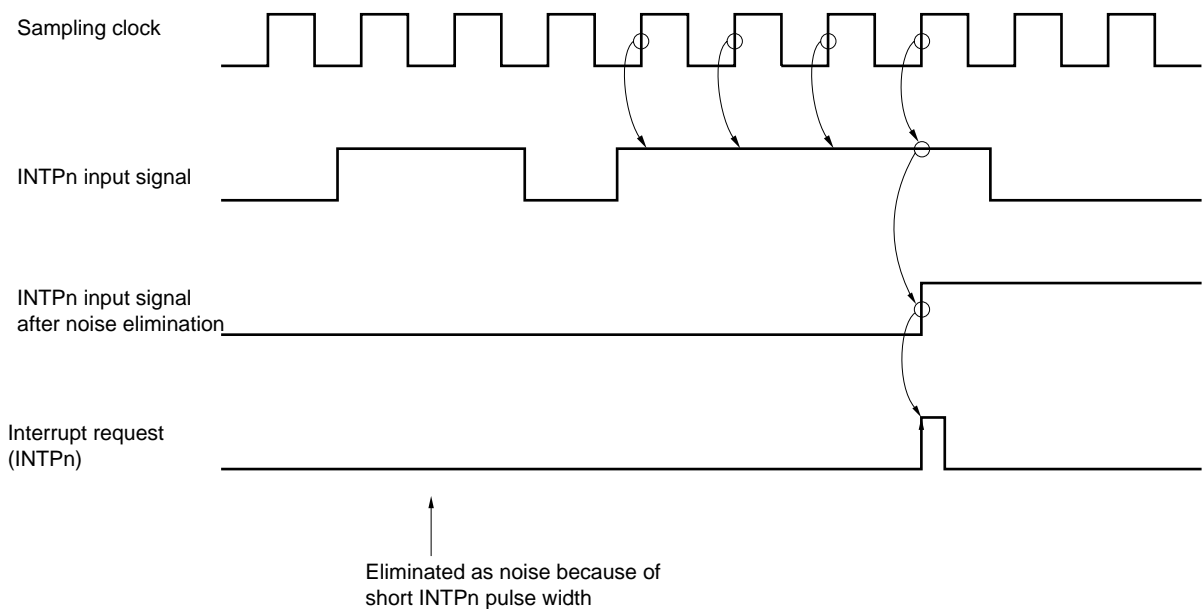
(5) Operation of sampling circuit

The sampling circuit of the μ PD78366A performs 4-point sampling with the timing specified by the SMPC0 and SMPC1 registers. If the results are at the same level four consecutive times, that level is fetched inside.

Figure 7-51. Block Diagram of Sampling Circuit (TM3)

Remarks 1. f_{CLK} : Internal system clock

2. $INTP_n$ ($n = 0, 1, 4$)

Figure 7-52. Sampling Timing Chart (TM3)

Remark $INTP_n$ ($n = 0, 1, 4$)

Judgment of valid signals by the 4-point sampling is carried out with the following timing.

- <1> $T_{in} \leq (3 \times T_{smp})$... Eliminated as noise
- <2> $(3 \times T_{smp}) < T_{in} < (4 \times T_{smp})$... Eliminated as noise or passed as valid signals depending on the timing
- <3> $T_{in} \geq (4 \times T_{smp})$... Passed as valid signals

T_{in} : Input signal width of INTP_n pin ($n = 0, 1, 4$)

T_{smp} : Sampling timing

Therefore, it is necessary to input a signal of $4 \times T_{smp}$ in width for the input signal to pass as a valid signal.

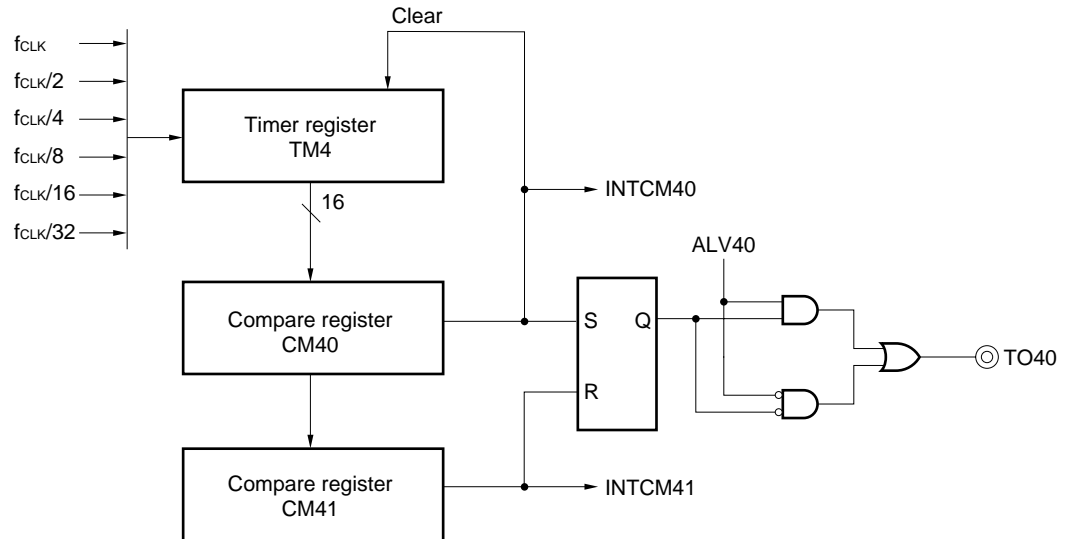
7.6 Timer 4

7.6.1 Configuration

Timer 4 consists of a 16-bit timer 4 (TM4) and two 16-bit compare registers (CM40 and CM41).

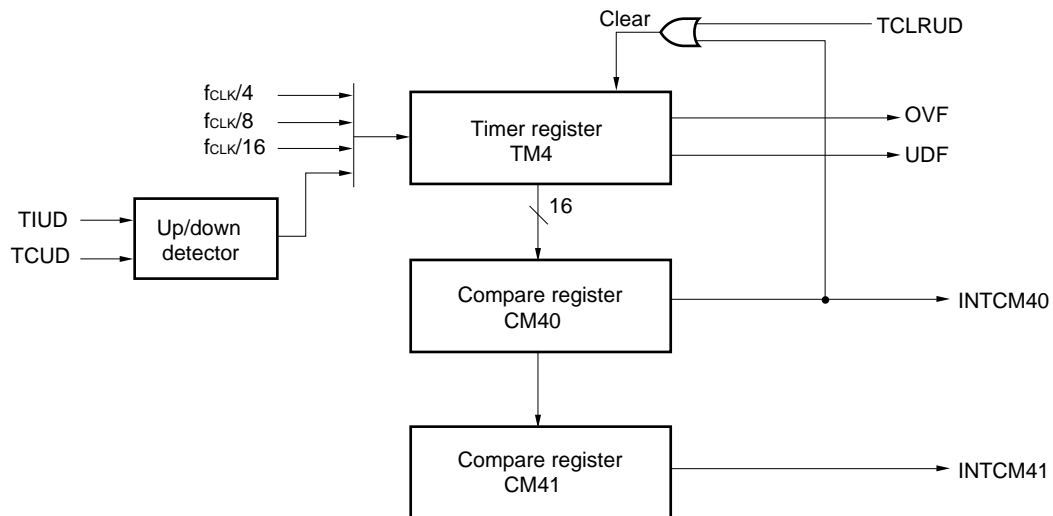
Figures 7-53 and 7-54 show the block diagrams of timer 4.

Figure 7-53. Block Diagram of Timer 4 (general-purpose timer mode)



Remark fCLK: internal system clock

Figure 7-54. Block Diagram of Timer 4 (UDC mode)



Remark fCLK: internal system clock

(1) 16-bit timer 4 (TM4)

TM4 has the following two operation modes:

(a) General-purpose timer mode

In this mode, timer 4 operates as a 16-bit interval timer, free running timer, or PWM output timer, and counts the internal clock.

When TM4 operates as an interval timer, TM4 is cleared on the next count clock after the coincidence between the timer value and the value of a compare register (CM40).

All the bits of TM4 are cleared to 0 when the $\overline{\text{RESET}}$ signal is input.

(b) Up/down counter mode (UDC mode)

In this mode, timer 4 operates as a 16-bit up/down counter, and counts the internal or external clock.

TM4 is cleared on the next count clock after the coincidence between the timer value and the value of a compare register (CM40), or when an external clear signal (TCLRUD) is input.

All the bits of TM4 are cleared to 0 when the $\overline{\text{RESET}}$ signal is input.

(2) 16-bit compare registers (CM40 and CM41)

CM40 and CM41 are 16-bit registers that always compare their values with the value of TM4. When the value of a compare register coincides with the value of TM4, a coincidence interrupt signal (INTCM40 or INTCM41) is generated.

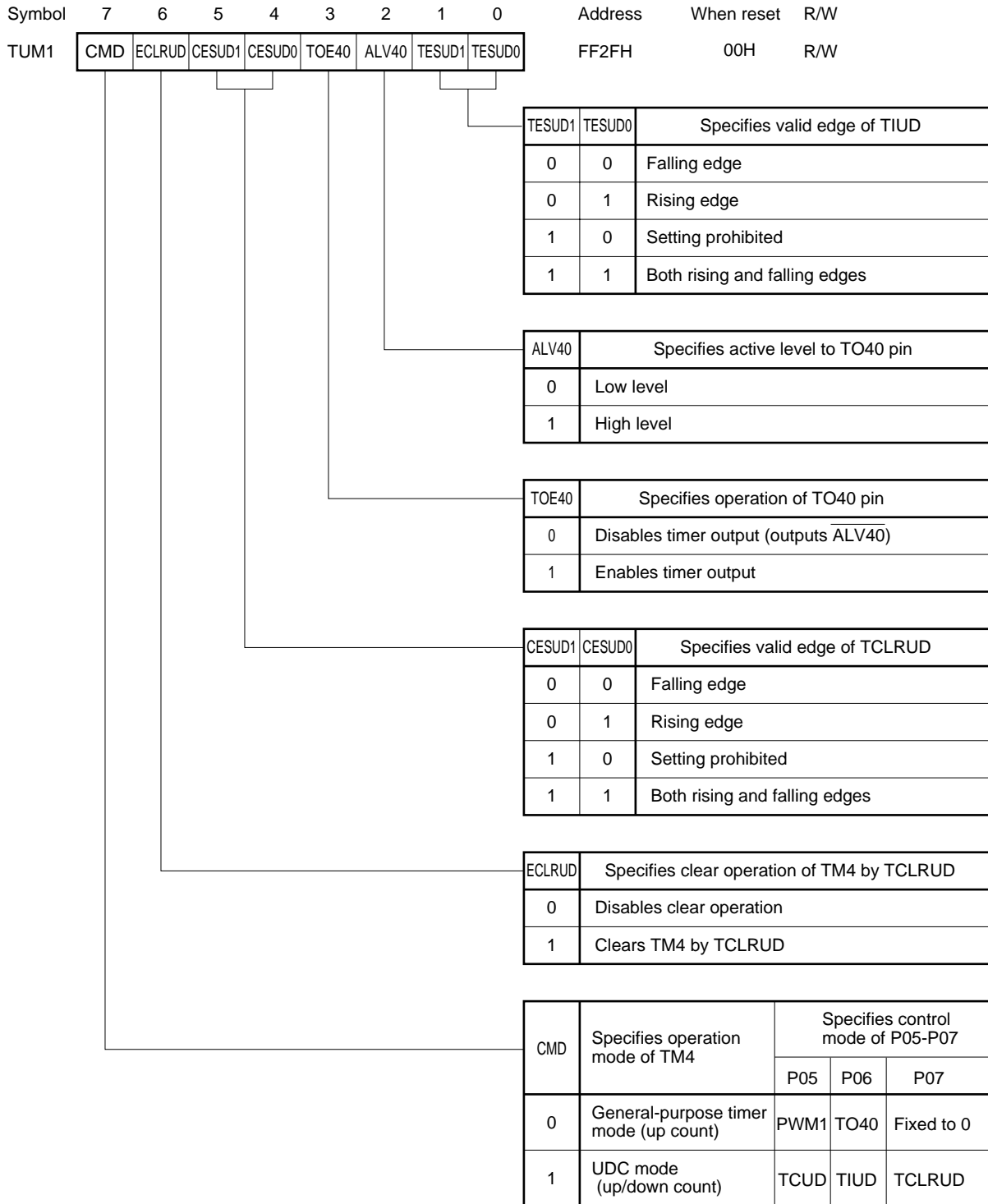
The values of CM40 and CM41 become undefined when the $\overline{\text{RESET}}$ signal is input.

7.6.2 Control registers

(1) Timer unit mode register 1 (TUM1)

Timer unit mode register 1 (TUM1) is an 8-bit register that specifies the operation mode of TM4, controls the functions of the P05-P07 pins when they are in the control mode, controls the operation of the timer output pin (TO40), and controls the external triggers (TCLRUD and TIUD).

This register can be read or written by a bit manipulation instruction or an 8-bit manipulation instruction. It is cleared to 00H when the RESET signal is input.

Figure 7-55. Format of Timer Unit Mode Register 1

- Cautions**
1. Bits 0, 1, and 4-6 are valid only when CMD = 1. Bits 2 and 3 are valid only when CMD = 0.
 2. If TM4 is set in mode 4 (specified by TMC4 register), specifying the valid edge of the TIUD pin is invalid (TESUD0 and TESUD1 bits).
 3. It is prohibited to change any bit of the TUM1 register during the operation of TM4 (CE4 = 1).

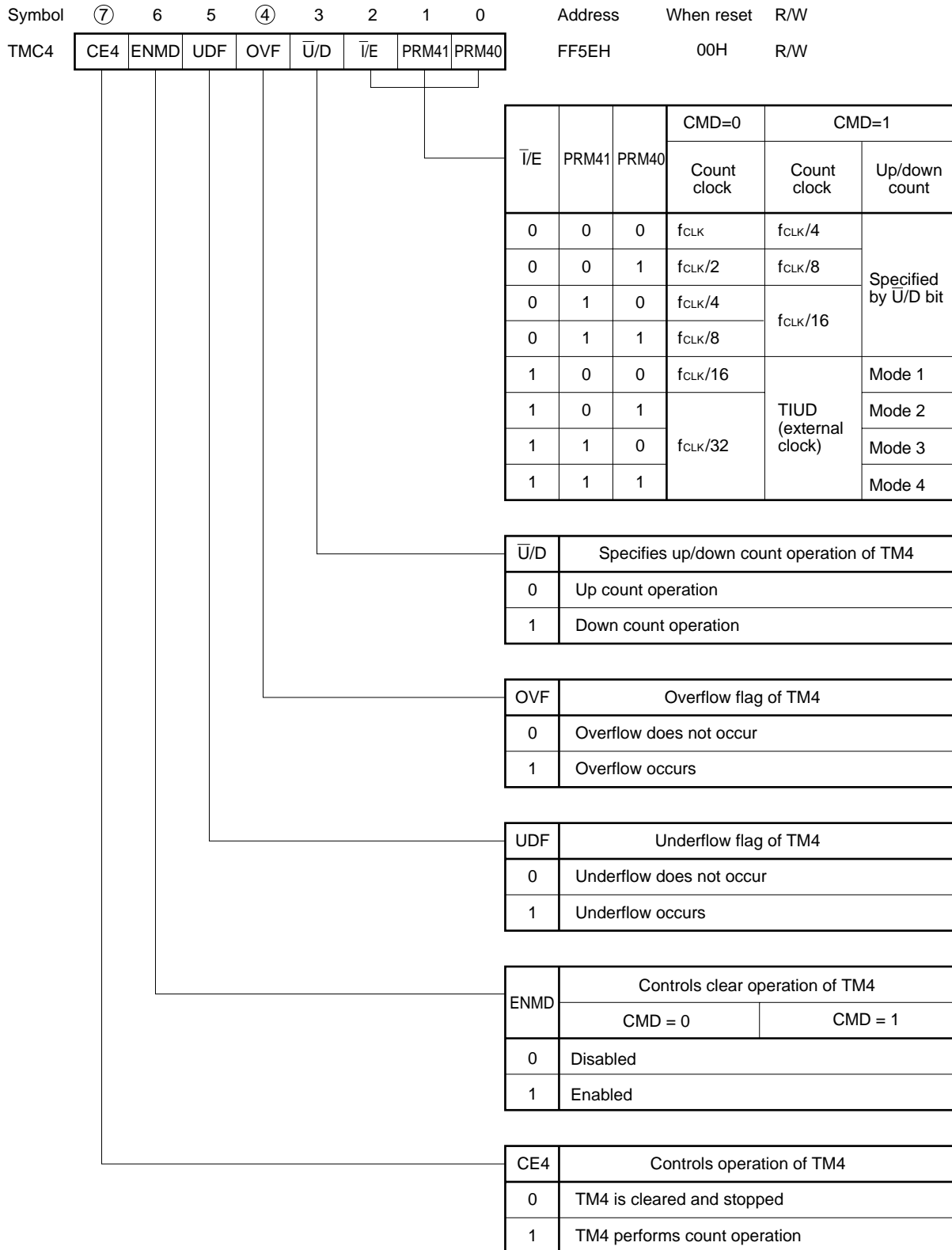
(2) Timer control register 4 (TMC4)

Timer control register 4 (TMC4) is an 8-bit register that controls the operation of TM4.

This register can be read or written by a bit manipulation instruction or an 8-bit manipulation instruction.

TMC4 is cleared to 00H by the RESET input.

Figure 7-56. Format of Timer Control Register 4



Cautions 1. The U/D bit is valid only when CMD = 1.

2. It is prohibited to change any bit of ENMD, U/D, I/E, PRM41 and PRM40 during the operation of TM4 (CE4 = 1).

Remark f_{CLK}: internal system clock

Each bit of the TMC4 register is described below.

(a) CE4 bit (bit 7)

This bit controls the operation of TM4 as follows:

- When CE4 \leftarrow 0: TM4 stops and does not operate.
- When CE4 \leftarrow 1: TM4 performs count operation.

(b) ENMD bit (bit 6)

This bit controls the clear operation of TM4.

- When ENMD \leftarrow 0: Disabled
TM4 does not perform clear operation.
When CMD = 0 and ENMD = 0, TM4 performs free running operation.
- When ENMD \leftarrow 1: Enabled

In General-Purpose Timer Mode (CMD = 0)	In UDC Mode (CMD = 1)
TM4 is cleared when TM4 coincides with CM40 (interval operation).	Setting prohibited

(c) UDF bit (bit 5)

This is the underflow flag of TM4, which is cleared on system reset or software reset.

- When UDF \leftarrow 0: TM4 does not underflow.
- When UDF \leftarrow 1: TM4 underflows.

(d) OVF bit (bit 4)

This is the overflow flag of TM4, which is cleared on system reset or software reset.

- When OVF \leftarrow 0: TM4 does not overflow.
- When OVF \leftarrow 1: TM4 overflows.

(e) \bar{U}/D bit (bit 3)

This bit specifies the up or down operation of TM4 when TM4 counts the internal clock.

When TM4 counts the external clock, the up or down count operation of TM4 can be read through this bit.

This bit is cleared by combination of system reset, resetting of the CE4 bit, software reset, and TIUD and TCUD when the external clock is counted.

- When $\bar{U}/D \leftarrow 0$: TM4 counts up.
- When $\bar{U}/D \leftarrow 1$: TM4 counts down.

Cautions 1. When $CMD = 0$,

→ The \bar{U}/D bit is fixed to “0” by hardware. Even if “1” is written, it remains “0”. If this bit is read, the value read is always “0”.

When $CMD = 1$ and the internal clock is selected,

→ The \bar{U}/D bit is write-enabled. If a read operation is applied to it, the written value is read.

When $CMD = 1$ and the external clock is selected,

→ Writing to the \bar{U}/D bit is not possible by hardware. If a read operation is applied to it, the up/down status of TM4 is read.

2. While TM4 is stopped ($CE4 = 0$), even if “1” is written to the \bar{U}/D bit, it remains “0” and it is not possible to rewrite it (see 7.6.3 (3) (b) Internal clock operation in UDC mode).

(f) \bar{I}/E bit (bit 2)

This bit specifies the count clock of TM4.

<1> In general-purpose timer mode ($CMD = 0$)

The count clock is fixed to the internal clock in this mode. The clock rate of TM4 is specified by the \bar{I}/E , PRM41, and PRM40 bits.

<2> In UDC mode ($CMD = 1$)

- When $\bar{I}/E \leftarrow 0$: internal clock
- When $\bar{I}/E \leftarrow 1$: external clock (TIUD)

(g) PRM41 and PRM40 bits (bits 1 and 0)

These bits specify the up or down count operation mode when the clock rate of the internal clock or the external clock is input.

\bar{I}/E	PRM41	PRM40	CMD = 0	CMD = 1	
			Count Clock	Count Clock	Up/Down Count
0	0	0	f _{CLK}	f _{CLK} /4	Specified by \bar{U}/D bit
0	0	1	f _{CLK} /2	f _{CLK} /8	
0	1	0	f _{CLK} /4	f _{CLK} /16	
0	1	1	f _{CLK} /8		
1	0	0	f _{CLK} /16	TIUD (external clock)	Mode 1
1	0	1	f _{CLK} /32		Mode 2
1	1	0			Mode 3
1	1	1			Mode 4

Operation modes of TM4 when external clock is input

Operation Mode	Operation of TM4
Mode 1	Counts down when TCUD = H. Counts up when TCUD = L.
Mode 2	Counts up when valid edge of TIUD input is detected. Counts down when rising edge of TCUD input is detected.
Mode 3	Automatic selection of TCUD input level when valid edge of TIUD input is detected.
Mode 4	Automatic selection when both edges of TIUD input and both edges of TCUD input are detected.

Caution When TM4 is set in mode 4, specification of the valid edge of the TIUD pin (specified by timer unit mode register 1 (TUM1)) is invalid.

7.6.3 Operation

(1) Basic operation

Timer 4 (TM4) has the following two operation modes:

(a) General-purpose timer mode

In this mode, timer 4 operates as a 16-bit interval timer, free running timer, or PWM output timer, and counts the internal clock.

Six internal clocks can be selected as the count clock to TM4 by TMC4 register.

(b) Up/down counter mode (UDC mode)

In this mode, timer 4 operates as a 16-bit up/down counter.

As the count clock to TM4, three internal clocks or external clock input (TIUD) can be selected by TMC4 register.

TM4 can be cleared by external clear input (TCLRUD).

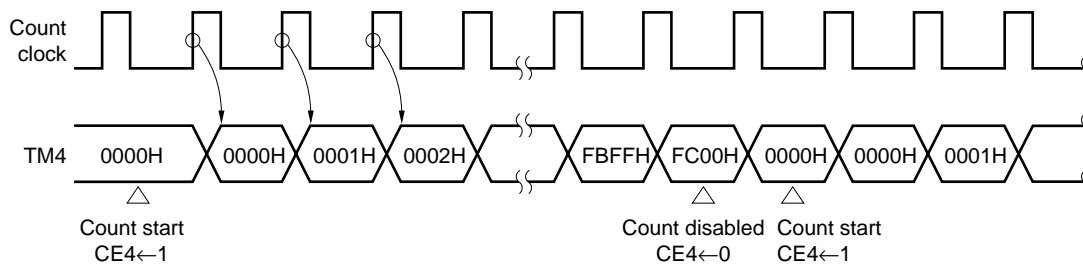
After the CE4 bit is set (1), TM4 becomes 0000H → 0000H by the initial count clock input and becomes 0001H by the 2nd count clock input. If TM4 is started to down count (UDC mode only), TM4 becomes 0000H → 0000H by the 1st count clock input and FFFFH by the 2nd count clock input.

Even if CE4 is set (1) again with CE4 = 1 during the operation of TM4, the count operation continues and the timer is not cleared.

If CE4 = 0 is set, the count operation stops with TM4 = 0000H.

RESET input clears (0) all bits of TM4 and stops the count operation.

Figure 7-57. Basic Operation of Timer 4 (TM4)



(2) General-purpose timer mode

If TM4 is to be operated in the general-purpose timer mode, set the CMD bit of the TUM1 register = 0. In this mode, TM4 operates as a 16-bit interval timer, free-running timer and PWM output timer.

The count clock for TM4 can be selected from among 6 kinds of internal clock by the TMC4 register.

(a) Interval operation

TM4 and CC40 always perform compare operations and if they detect a coincidence they generate interrupt signal INTCM40. The result of a compare coincidence is retained by the hardware and TM4 is cleared (0000H) with the next count clock after the coincidence. Furthermore, when the next count clock is input TM4 counts up to 0001H.

$$\text{Interval cycle} = (\text{CC40 value} + 1) \times \text{TM4 count clock rate}$$

(b) Free-running operation

TM4 performs a full count from 0000H to FFFFH. After TM4 sets an overflow flag (OVF) and clears the timer, it continues counting.

$$\text{Free-running cycle} = 65536 \times \text{TM4 count clock rate}$$

(3) UDC mode**(a) Basic operation of UDC mode****[Setting procedure]**

- <1> When TM4 is to be operated with an external clock (modes 1 to 4), set pins P05 to P07 to the control mode by the port 0 mode control register (PMC0).
Also in the case of internal clock operation, if external clear (TCLRUD) is to be used, set only P07 to the control mode.
- <2> Set (1) the CMD bit of the TUM1 register to set TM4 to the UDC mode. With this setting, the control pins specified in <1> become the function pins (TCUD, TIUD and TCLRUD) used in the UDC mode.
- <3> Specify the following using the TUM1 register.
- ECLRUD bit : Enable/disable of TM4 clear by valid edge of TCLRUD pin
 - CESUD1 and 0 bits : Specification of valid edge of TCLRUD pin
 - TESUD1 and 0 bits : Specification of valid edge of TIUD pin
- <4> Specify the following using the TMC4 register.
- ENMD bit : Be sure to set "0".
 - \bar{U}/D bit : Specifies TM4 operation only when timer is operated with internal clock in UDC mode.
 - \bar{I}/E bit : Selects either internal or external count clock of TM4.
 - PRM41-40 bits : Specifies TM4 count clock and operating mode.
- <5> Finally, when the CE4 bit of the TMC4 register is set (1), TM4 starts operating.

[Operation by mode]

When TM4 is to be operated in the UDC mode, set CMD of the TUM1 register = 1.

UDC mode operations can be summarized as follows.

Table 7-5. List of UDC Mode Operations

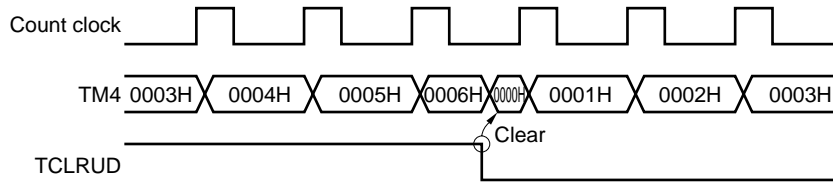
TMC4 Register			Operating Mode	TM4 Operation	TM4 Clear Operation	
T/E	PRM41	PRM40			CM40 Coincidence	TCLRUD
0	×	×	Internal clock	Up/down setting by \overline{U}/D bit	Disabled: ENMD = 0	Enabled: ECLRUD = 1
1	0	0	Mode 1	When TCUD = H, down count When TCUD = L, up count		Disabled: ECLRUD = 0
1	0	1	Mode 2	Up count by detection of valid edge of TIUD input Down count by detection of TCUD input rising edge		
1	1	0	Mode 3	Automatic judgment by TCUD input level when valid edge of TIUD input is detected		
1	1	1	Mode 4	Automatic judgment by detection of both edges of TIUD input and both edges of TCUD input		

Caution Setting ENMD = 1 is prohibited in the UDC mode.

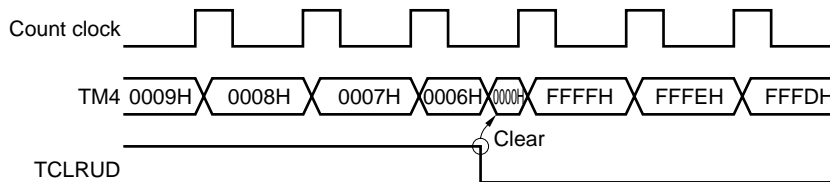
Remarks 1. ENMD bit : Bit 6 of TMC4 register
2. ECLRUD bit: Bit 6 of TUM1 register

[External clear (TCLRUD) operation]

The TCLRUD signal is a timer clear signal that is valid only when TM4 is in the UDC mode (with internal clock and external clock). The valid edge of the TCLRUD pin is specified by the CESUD1 and CESUD0 bits of the TUM1 register, and when the valid edge is detected TM4 is cleared.

Figure 7-58. TM4 Clear Operation (during up count in UDC mode)

Remark The valid edge of TCLRUD is set as the falling edge.

Figure 7-59. TM4 Clear Operation (during down count in UDC mode)

Remark The valid edge of TCLRUD is set as the falling edge.

(b) Internal clock operation in UDC mode

When TM4 is to be operated with an internal clock, set it by the \bar{I}/E , PRM41 and PRM40 bits of the TMC4 register.

Table 7-6. Setting of TMC4 Register (during internal clock operation in UDC mode)

\bar{I}/E	PRM41	PRM40	Count Clock
0	0	0	$f_{CLK}/4$
0	0	1	$f_{CLK}/8$
0	1	0	$f_{CLK}/16$
0	1	1	

If internal clock operation in the UDC mode is specified, up/down operation is specified by the \bar{U}/D bit of the TMC4 register. Resetting (0) the \bar{U}/D bit starts up count and setting (1) it starts down count.

Caution While TM4 is halted ($CE4 = 0$), even if “1” is written in the \bar{U}/D bit it remains “0” and it is not possible to rewrite it. If it is desired to let TM4 perform a down operation from the beginning, set (1) the $CE4$ bit and \bar{U}/D bit in the TMC4 register by the MOV instruction simultaneously.

<Example of correct program>

```
MOV TUM1, #80H ; Sets TM4 to UDC mode.
MOV TMC4, #88H ;  $CE4 = 1$ ,  $\bar{I}/E = 0$ ,  $\bar{U}/D = 1$ 
```

In the following program, TM4 performs an up operation from the beginning.

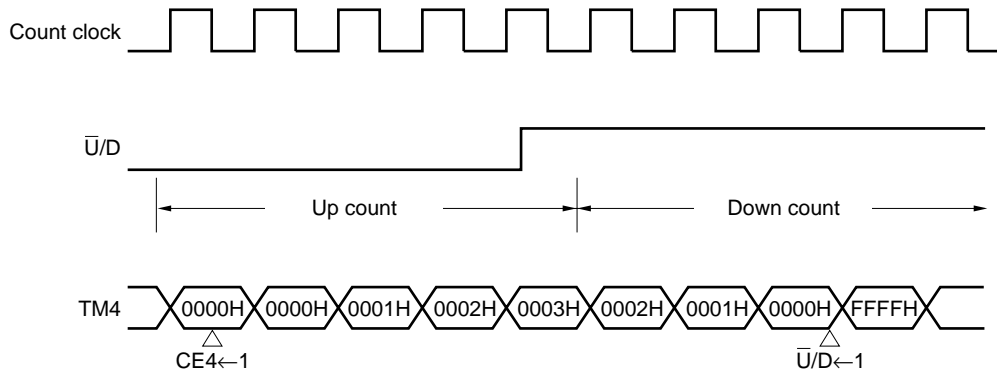
<Example of wrong program>

```
MOV TUM1, #80H ; Sets TM4 to UDC mode.
MOV TMC4, #08H ;  $\bar{I}/E = 0$ ,  $\bar{U}/D = 1$  (TM4 performs down operation with an internal clock.) ← <1>
SET1 TMC4.7 ;  $CE4 = 1$  (TM4 start) ← <2>
```

<1> Because $CE4 = 0$ at this time, $\bar{U}/D = 0$ remains unchanged and can not be rewritten.

<2> With $\bar{U}/D = 0$ TM4 starts an up operation.

Figure 7-60. Internal Clock Operation in UDC Mode

**(c) External clock operation in UDC mode**

When the external clock is used as the count clock to timer 4 (TM4), the up/down count operation of TM4 can be executed in a mode specified by an external count clock input pin (TIUD) and count operation select signal input pin (TCUD). The following four modes can be selected by the PRM41 and PRM40 bits of timer control register 4 (TMC4):

Table 7-7. Up/Down Count Operation Modes

TMC4 Register			Mode	Operation of TM4
I/E	PRM41	PRM40		
1	0	0	1	Counts down when TCUD = H. Counts up when TCUD = L.
1	0	1	2	Counts up when valid edge of TIUD input is detected. Counts down when rising edge of TCUD input is detected.
1	1	0	3	Automatic selection of TCUD input level when valid edge of TIUD input is detected.
1	1	1	4	Automatic selection when both edges of TIUD input and both edges of TCUD input are detected.

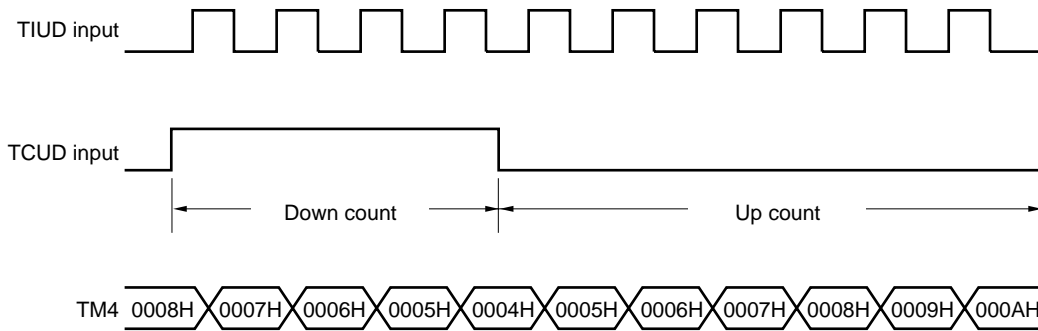
Caution When TM4 is set in mode 4, specification of the valid edge of the TIUD pin (specified by timer unit mode register 1 (TUM1)) is invalid.

(i) Mode 1 (PRM41 = 0, PRM40 = 0)

In this mode, TM4 counts down the external count clock input (TIUD pin) while the TCUD pin is high, and counts up while the TCUD pin is low.

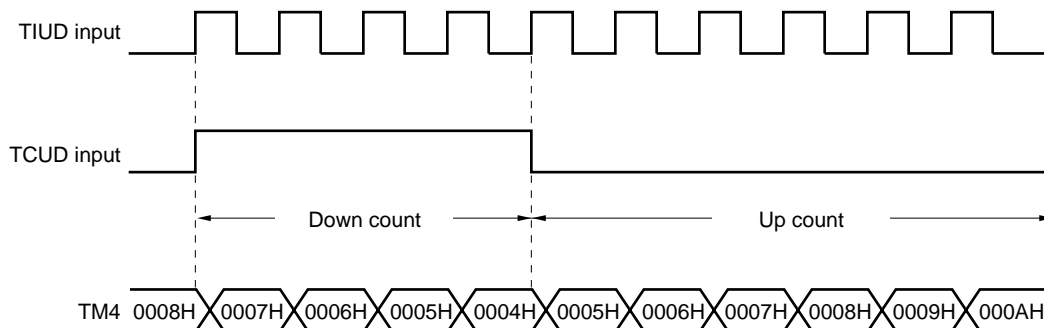
When the rising edge of the TIUD pin is specified as the valid edge (by the TUM1 register), the operation of TM4 is as shown in Figure 7-61.

Figure 7-61. Example of Operation in Mode 1 (when valid edge of TIUD pin is rising edge)



If TIUD and TCUD change simultaneously, the operation is as follows.

Figure 7-62. Example of Operation in Mode 1 (when valid edge of TIUD pin is rising edge)



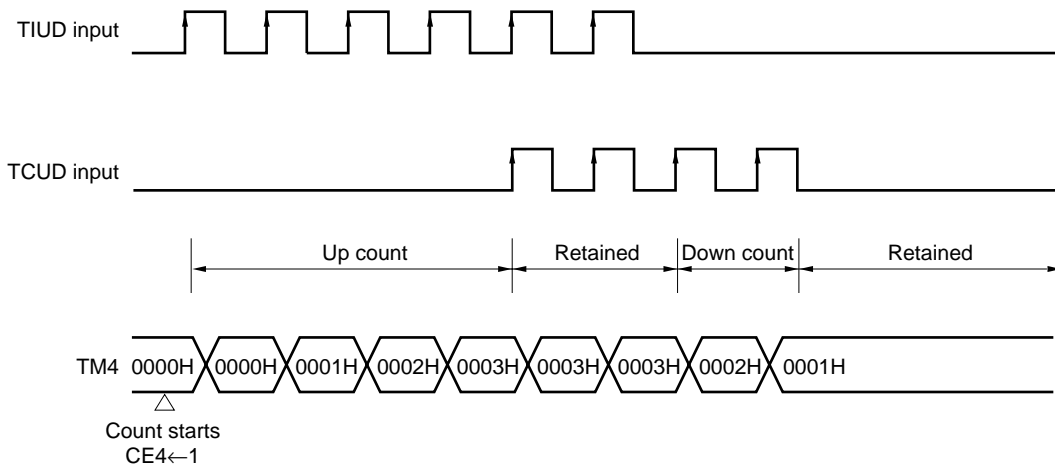
(ii) Mode 2 (PRM41 = 0, PRM40 = 1)

In mode 2, TM4 counts up the valid edge input to the TIUD pin (specified by the timer unit mode register 1 (TUM1)) and counts down the rising edge of the TCUD pin.

If the count clock is simultaneously input to the TIUD and TCUD pins, TM4 does not perform counting, but retains the value immediately before.

Figure 7-63. shows the operation of timer 4 (TM4) when the rising edge of the TIUD pin is specified as the valid edge.

Figure 7-63. Example of Operation in Mode 2 (when valid edge of TIUD pin is rising edge)



(iii) Mode 3 (PRM41 = 1, PRM40 = 0)

Mode 3 is the most effective when two signals 90 degrees out of phase, such as those output by the shaft encoder of a servo motor, are input to the TIUD and TCUD pins as the count clocks. Timer 4 (TM4) detects the relative leading or lagging between these two signals, and automatically selects the up or down count operation.

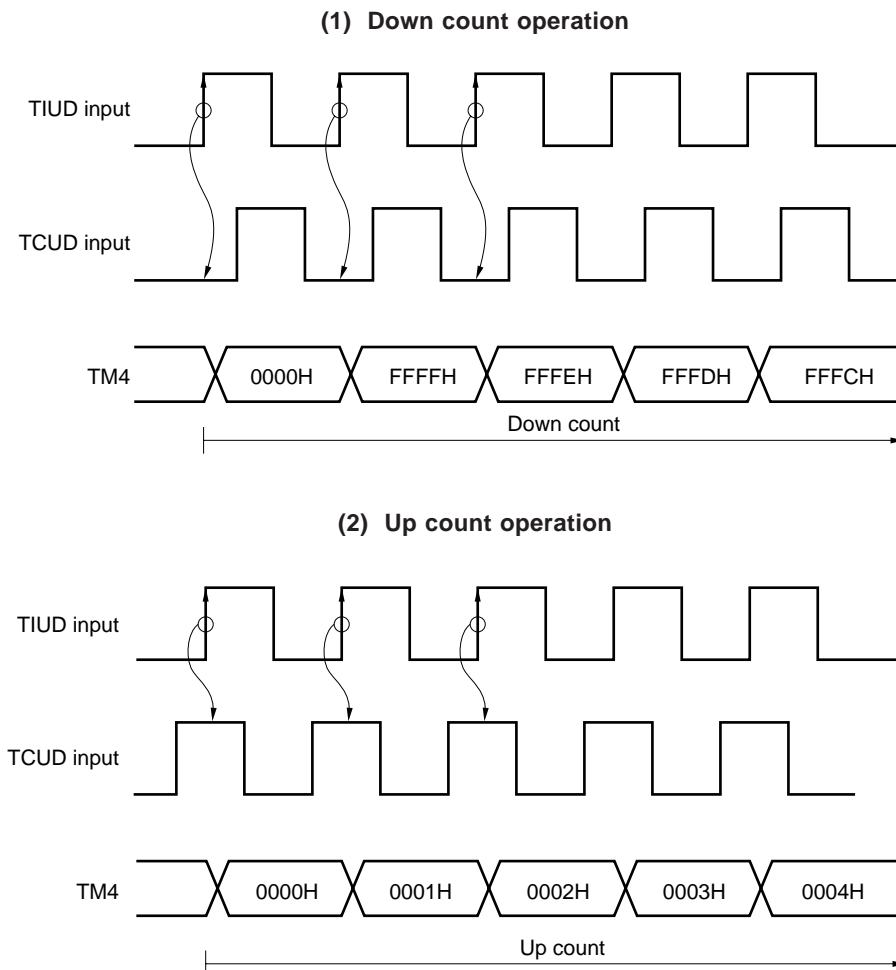
When the two signals 90 degrees out of phase are input to the TIUD and TCUD pins, the level of the TCUD pin is sampled when the valid edge is input to the TIUD pin.

If the TCUD pin level sampled at the valid edge input to the TIUD pin is low, TM4 counts down when the valid edge is input to the TIUD pin.

If the TCUD pin level sampled at the valid edge input to the TIUD pin is high, TM4 counts up when the valid edge is input to the TIUD pin.

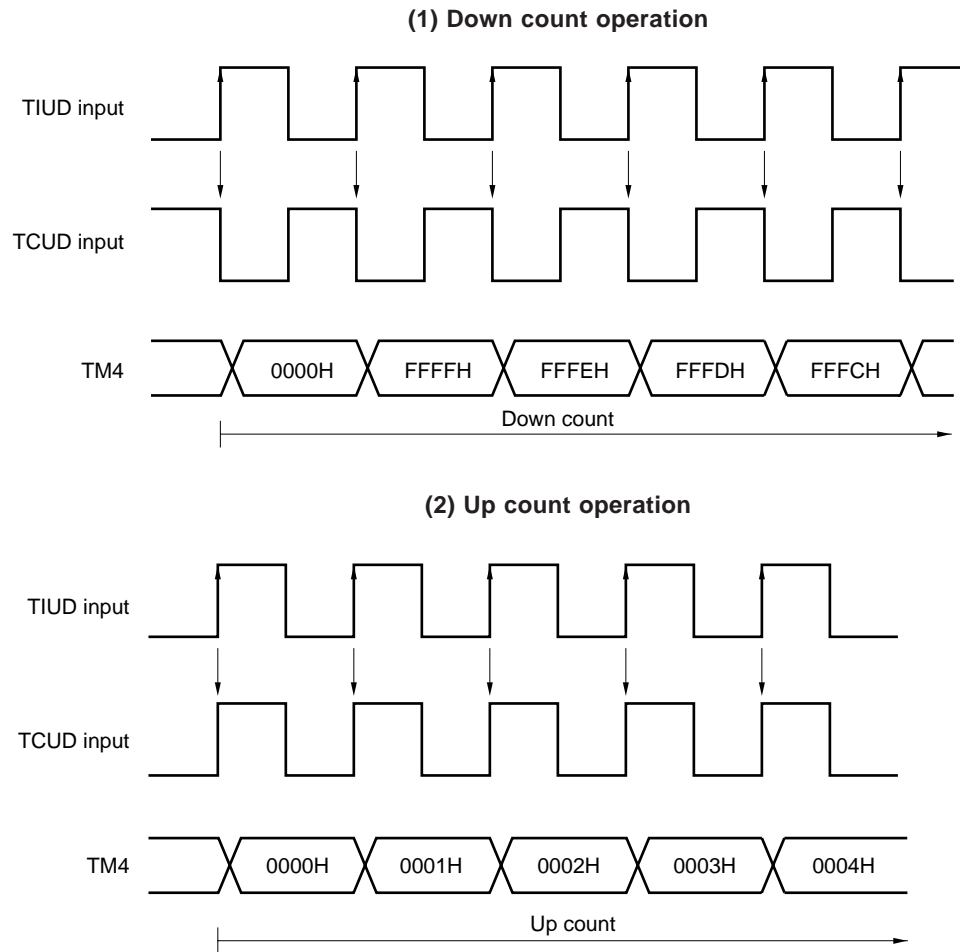
Figure 7-64 shows the operation of TM4 when the rising edge of the TIUD pin is specified as the valid edge.

Figure 7-64. Example of Operation in Mode 3 (when valid edge of TIUD pin is rising edge)



If TIUD and TCUD change simultaneously, the operation is as follows.

Figure 7-65. Example of Operation in Mode 3 (when valid edge of TIUD pin is rising edge)



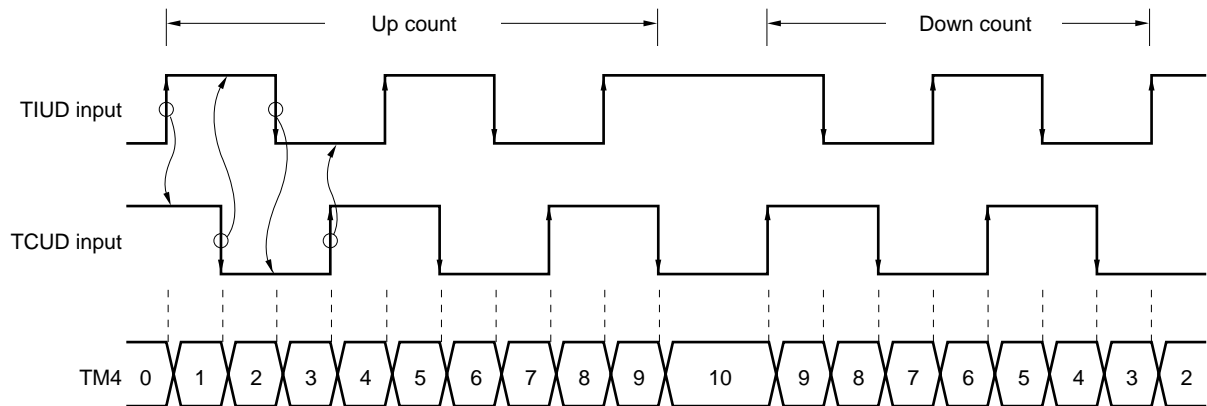
(iv) Mode 4 (PRM41 = 1, PRM40 = 1)

Mode 4, like mode 3, is effective when two signals 90 degrees out of phase, such as those output by the shaft encoder of a servo motor, are input to the TIUD and TCUD pins as the count clocks. When the two signals 90 degrees out of phase are input to the TIUD and TCUD pins, TM4 automatically selects up or down operation in the timing as shown in Figure 7-66, and executes counting.

In mode 4, counting is executed at both the rising and falling edges of the two signals input to the TIUD and TCUD pins. Therefore, timer 4 (TM4) counts four times per cycle of an input signal (x4 count).

- Cautions**
1. The valid edge of the TIUD pin specified by timer unit mode register 1 (TUM1) is invalid when TM4 is in mode 4.
 2. If the TIUD pin edge and TCUD pin edge are input simultaneously in mode 4, TM4 continues counting while keeping the same up/down operation immediately before the input.

Figure 7-66. Example of Operation in Mode 4



Remark The count value of TM4 is represented in decimal.

(4) PWM output operation

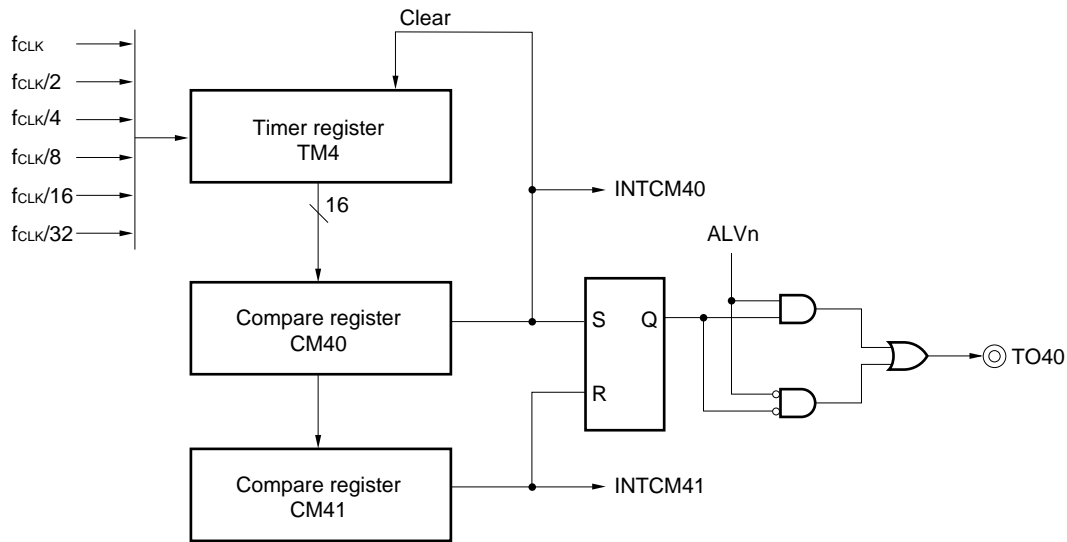
The real-time pulse unit (RPU) is provided with a 16-bit cycle programmable PWM output function that can change the duty factor of an active level in a specific cycle.

When TM4 is set in the general-purpose timer mode by timer unit mode register 1 (TUM1), the TO40 pin can be used for the PWM output operation.

The resolution is 16 bits, and as many as six internal clocks can be selected as the count clock.

Figure 7-67 shows the block diagram of timer 4 when it performs the PWM output operation.

Figure 7-67. Block Diagram of Timer 4 (PWM output operation)



(a) Setting procedure

The PWM output operation is executed in the following sequence:

- <1> Set the P06 pin in the control mode by using the port 0 mode control register (PMC0) (PMC06 = 1).
- <2> Clear the CMD bit of the TUM1 register to "0" (TM4 is set in the general-purpose timer mode, and the P06 pin serves as the TO40 output pin).
- <3> Set the TOE40 bit of the TUM1 register to 1 to enable the output operation of TO40. Also set the active level of TO40 by using the ALV40 bit.
- <4> Set the operation of TM4 by using the ENMD bit of the timer control register 4 (TMC4) (ENMD = 1), and set the count clock of TM4 with the I/E, PRM41, and PRM40 bits.
- <5> When the CE4 bit of the TMC4 register is set to 1, TM4 starts counting, and the PWM signal is output from the TO40 pin.

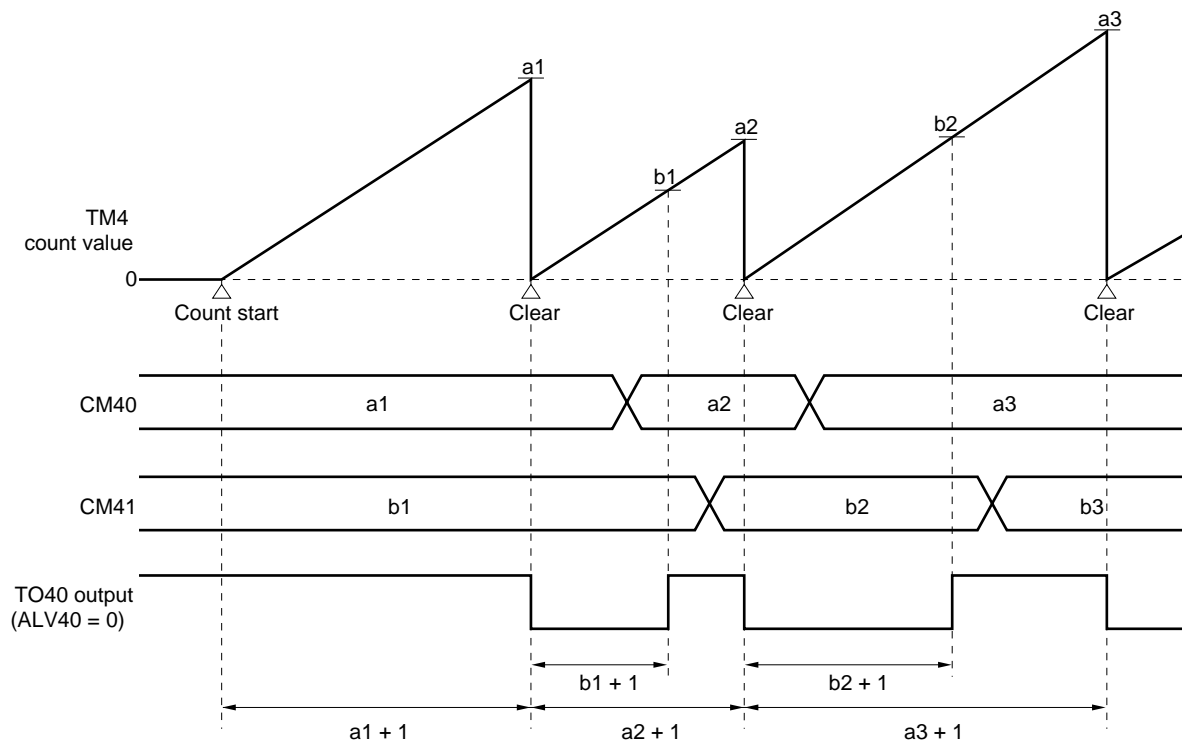
(b) Operation

CM40 is a compare register that sets the cycle of TO40. If CM40 coincides with TM4, it generates interrupt signal INTCM40. The result of the compare coincidence is retained by the hardware and TM4 is cleared with the next count clock after the coincidence.

CM41 is a compare register that sets the duty factor of the TO40 output. Set the duty factor necessary for each cycle.

When software processing is started by cycle interrupt INTCM40 or interrupt INTCM41 that is generated from CM41, and when the values of CM40 and CM41 are set through software, the PWM output waveform can be efficiently generated from TO40.

Figure 7-68. Example of PWM Output Operation of TO40



Remark If CM40 = CM41 is set, a reset takes precedence, and therefore TO40 outputs the inactive level.

7.7 Real-Time Output Function

The real-time output function is used to output the contents of the real-time output port register (RTP) to P00-P03 in units of 4 bits, in synchronization with the trigger signal from the real-time pulse unit (RPU). With this function, pulses can be synchronously output from two or more channels.

In addition, the output of P00 to P03 can be modulated for PWM.

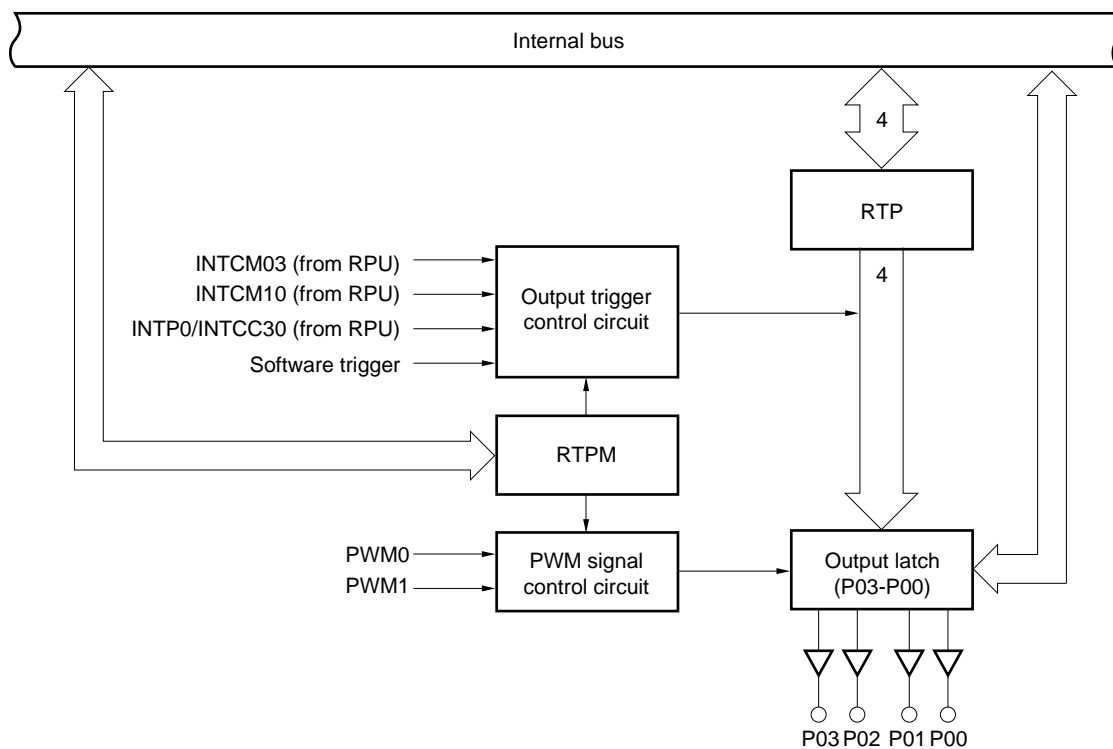
7.7.1 Configuration

The real-time output port is multiplexed with port 0 (P0). To control the output status, the following two registers are provided:

- Real-time output port mode register (RTPM)
- Real-time output port register (RTP)

The real-time output is changed by the trigger signal specified by the RTRG1 and RTRG0 bits of the RTPM register. As the trigger signal, the signal output by RPU or a software trigger can be selected.

Figure 7-69. Block Diagram of Real-Time Output Port



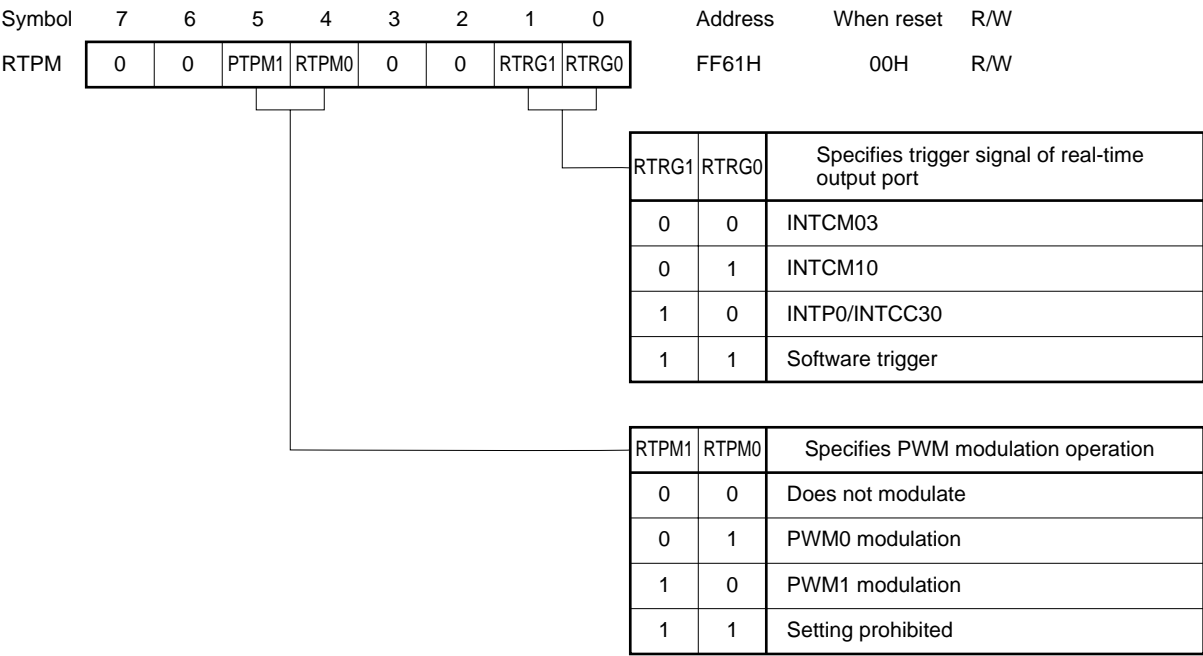
Caution The RTP register is an 8-bit access register. When the RTP register is read, the data set in the RTP register is placed in the lower 4 bits. The higher 4 bits are fixed to “0” by hardware, and therefore “0” is read.

7.7.2 Control registers

(1) Real-time output port mode register (RTPM)

This is an 8-bit register that specifies the operation mode of the real-time output port.
This register can be read or written by a bit manipulation instruction or an 8-bit manipulation instruction.
When the RESET signal is input, RTPM is cleared to 00H.

Figure 7-70. Format of Real-Time Output Port Mode Register



Caution Bits 7, 6, 3 and 2 of the RTPM register are fixed to “0” by hardware. Even if “1” is written to them, they remain “0”.

(2) Real-time output port register (RTP)

This is an 8-bit register that stores the data output from the real-time output port.

The data written to this register is output to the pin connected to it according to the specification of the real-time output port mode register (RTPM).

If the signal output by the real-time pulse unit (RPU) is specified as the output trigger of the real-time output port, the data written to the RTP register is output to the pin in synchronization with a trigger signal. If the software trigger is specified as the trigger signal, the data written to the RTP register is output to the pin as is.

The RTP register can be read or written by a bit manipulation or 8-bit manipulation instruction.

The contents of this register become undefined when the $\overline{\text{RESET}}$ signal is input.

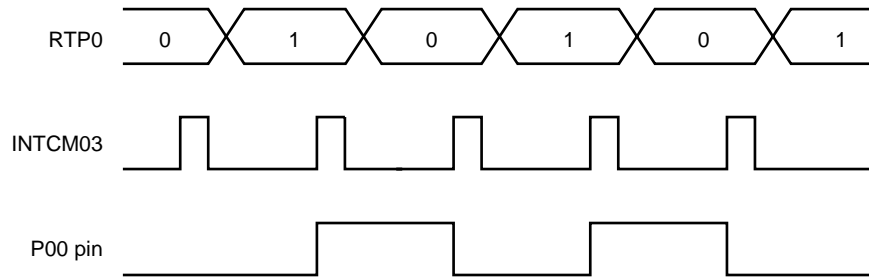
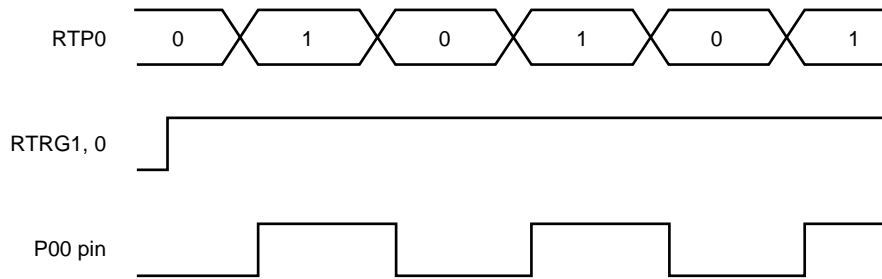
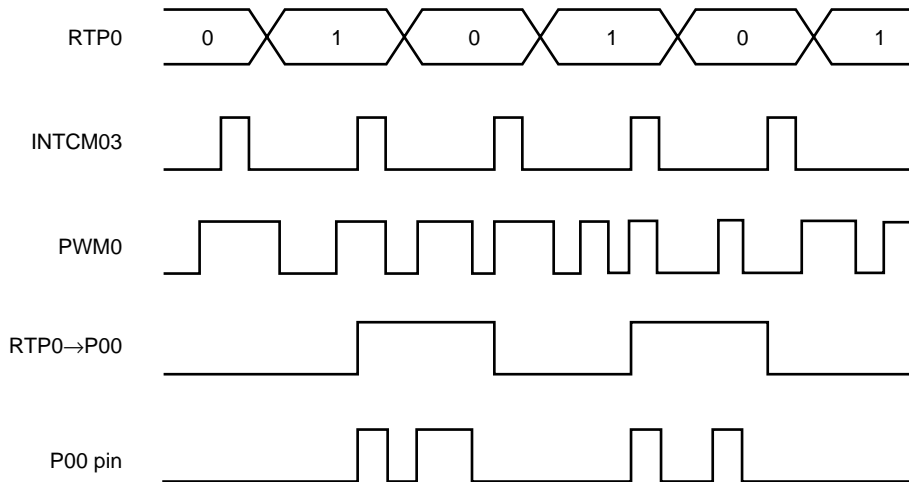
Caution The higher 4 bits of the RTP register have no latch circuit and are fixed to “0”. Therefore, even if “1” is written to the higher 4 bits, they remain “0”.

7.7.3 Operation

The value of the real-time output port register (RTP) is set by software. The trigger signal that is used to output the contents of the RTP register to P00-P03 can be selected from three types of interrupts by the real-time output port mode register (RTPM).

To change the pin level directly, write “1” to the RTRG1 and RTRG0 bits of the RTPM register through software. The contents of the RTP register will be directly output to P00-P03.

When P00-P03 are in the control mode (when the real-time output function is specified), the output pattern of each pin can be modulated for PWM. When PWM modulation is performed, the signal transferred from the RTP register to the output latch is ANDed with the PWM signal (PWM0 or PWM1), and the result of this ANDing is output to the P00-P03 pins.

Figure 7-71. Example of Real-Time Output Function Operation (P00 pin)**(a) $RTRG1 = RTRG0 = 0$ (trigger signal: INTCM03), $RTPM1 = RTPM0 = 0$ (no PWM modulation)****(b) $RTRG1 = RTRG0 = 1$ (software trigger), $RTPM1 = RTPM0 = 0$ (no modulation)****(c) $RTRG1 = RTRG0 = 0$ (trigger signal: INTCM03), $RTPM1 = 0$ $RTPM0 = 1$ (PWM0 modulation)****Remark** All the P00-P03 pins perform the same operation.

CHAPTER 8 A/D CONVERTER

The μ PD78366A is provided with a high-speed, high-resolution 10-bit analog-to-digital (A/D) converter, which has eight analog input pins (ANI0-ANI7) through which analog signals are input, and a 10-bit A/D conversion result register (ADCR) which stores the conversion results.

This A/D converter is a successive approximation type, and operates in the following three modes. The operation modes are selected by using the A/D converter mode register (ADM) through software, so that A/D conversion suitable for the application system can be carried out.

- **Select mode**

In this mode, the A/D converter converts data input from one analog input line.

- **Scan mode**

In this mode, two or more sets of analog input data are sequentially converted.

- **Mixed mode**

Processing in the select and scan modes is executed in combination.

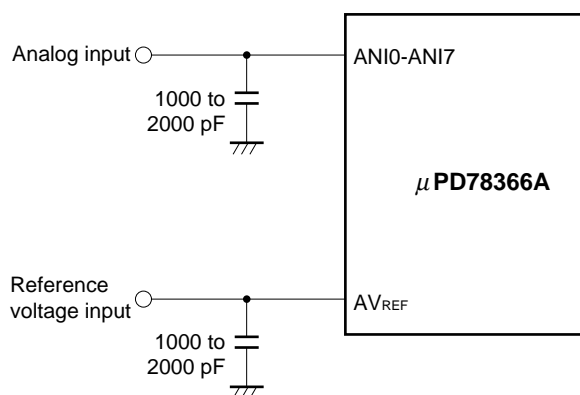
In each mode, the conversion results are stored in the ADCR register each time A/D conversion is executed. When conversion has been completed, an A/D conversion end interrupt (INTAD) occurs, which starts a macro service that executes automatic data transfer, etc.

The configuration of the A/D converter is as shown in Figure 8-1.

The diagram illustrates the internal architecture of the ADC module. It features an **Input circuit** with eight channels (ANI0 to ANI7). A **Sample & hold circuit** (dashed box) is connected to the input circuit and a **Comparator**. The **Comparator** is also connected to a **Resistor string**, which is powered by AV_{DD} and AV_{SS} and has a reference input AV_{REF} . The **Resistor string** provides a 10-bit digital output (bits 9 to 0) to the **SAR (10)** (Successive Approximation Register). The **SAR** is connected to the **Controller** and the **ADCR** (ADC Register) block. The **Controller** is also connected to the **ADM (8)** (ADC Module) and the **Internal bus**. The **ADM** is connected to the **Internal bus** via an 8-bit bus. The **ADCR** block contains eight registers (ADCR0 to ADCR7) and provides a 10-bit digital output (bits 9 to 0) to the **Internal bus**. The **Internal bus** is shown at the bottom of the diagram.

Cautions 1. Connect capacitors to the analog input pins (ANI0-ANI7) and reference voltage input pin (AVREF) to prevent malfunctioning due to noise.

Figure 8-2. Example of Connecting Capacitor to A/D Converter Pin



2. Do not apply a voltage exceeding the range of AVSS to AVDD to the pin used as the A/D converter input pin.

(1) Input circuit

The input circuit selects an analog input line in accordance with an operation mode specified by the A/D converter mode register (ADM) and sends the signal input from the selected analog line to the sample and hold function.

(2) Sample and hold circuit

The sample and hold circuit samples the analog input signals sent from the input circuit one by one, and sends them to the comparator. If A/D conversion is in progress at this time, the analog input signal is retained.

(3) Comparator

The comparator compares the voltage of the analog input signal with the output voltage of the D/A converter.

(4) Resistor string

The resistor string generates a voltage that matches the voltage of the input analog signal. The output voltage of the resistor string is controlled by the SAR register.

(5) SAR (Successive Approximation Register)

The SAR register is a 10-bit register that controls the output value of the resistor string to compare the voltage of the input analog signal.

When A/D conversion has been completed, the contents of the SAR register at that time (conversion result) are stored in the A/D conversion result register (ADCR). After specified A/D conversion has been completed, an A/D conversion end interrupt (INTAD) occurs.

(6) Controller

The controller selects an analog input signal in accordance with the mode specified by the ADM register, generates the operation timing of the sample and hold function, and controls the conversion trigger.

(7) ANI0-ANI7 pins

These pins are eight channels of analog input pins of the A/D converter which input analog signals to be converted into digital signals.

Caution Do not apply a voltage exceeding that specified to the ANI0-ANI7 pins. If a voltage higher than V_{DD} or lower than V_{SS} (even if the voltage is within the range of the absolute maximum ratings) is input, the conversion value of that channel is undefined and may influence the conversion values of other channels.

(8) AV_{REF} pin

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

Based on the voltage applied across the AV_{REF} and AV_{SS} pins, the signal input to ANI0-ANI7 is converted into a digital signal.

(9) AV_{SS} pin

This is the ground pin of the A/D converter. Input the V_{SS} level to this pin.

(10) AV_{DD} pin

Input the V_{DD} level to this pin.

8.2 A/D Converter Mode Register (ADM)

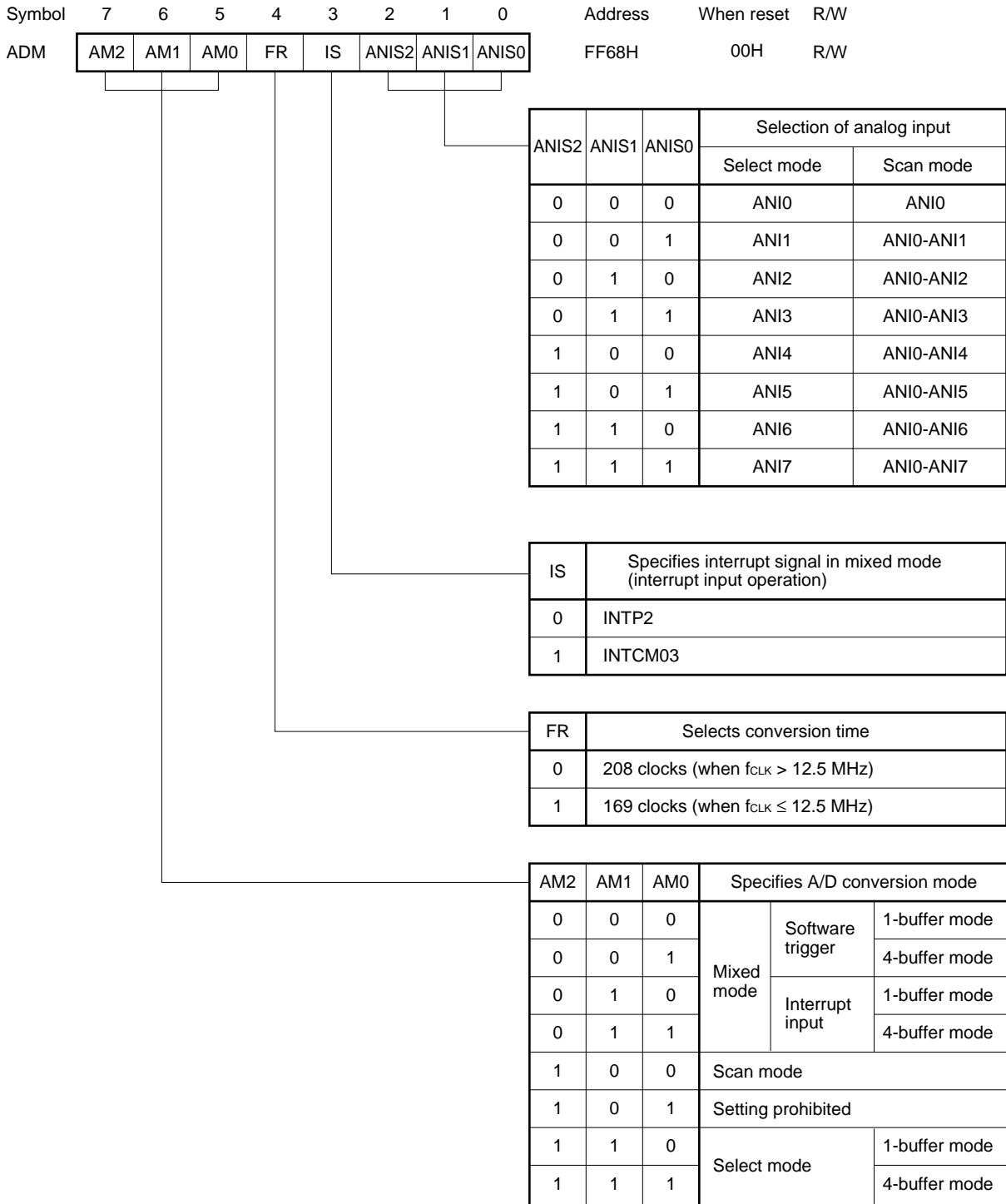
The A/D converter mode register (ADM) is an 8-bit register that controls the operation of the A/D converter.

This register can be read or written by a bit manipulation instruction or an 8-bit manipulation instruction. If data is written to the ADM register while conversion is in progress, conversion is initialized and started from the beginning.

The ADM register is initialized to 00H when the $\overline{\text{RESET}}$ signal is input.

Figure 8-3 shows the format of the ADM register.

Figure 8-3. Format of A/D Converter Mode Register



Remarks 1. f_{CLK} : internal system clock

2. Software trigger is to start A/D conversion by using data written to the ADM register as a trigger.
3. In the select and scan modes, A/D conversion is started only by the software trigger.

Each bit of the ADM register has the following function:

(1) ANIS2-ANIS0 bits (bits 2-0)

These bits select an analog input signal (ANI0-ANI7) to be converted into a digital signal.

(2) IS bit (bit 3)

This bit selects an A/D conversion start trigger (INTP2 or INTCM03) when the mixed mode (interrupt input operation) is specified by the AM2-AM0 bits.

The software trigger operation in the mixed mode, and the operation of the scan and select modes are not influenced by the IS bit.

(3) FR bit (bit 4)

This bit controls the A/D conversion time.

The FR bit is set (to 1) by an instruction when the internal system clock $f_{CLK} \leq 12.5$ MHz. The FR bit performs control so that the A/D conversion time is not significantly changed even if the operating frequency of the μ PD78366A is changed.

The time required for one conversion, which is determined by the internal system clock (f_{CLK}) and the FR bit, is expressed as follows:

- When FR = 0, conversion time = $\frac{208}{f_{CLK}}$ (μ s)
- When FR = 1, conversion time = $\frac{169}{f_{CLK}}$ (μ s)

Table 8-1. Example of Conversion Time Set by FR Bit

Internal system clock f_{CLK} (MHz)	16	12.5
Oscillation frequency f_{XX} , external clock f_X (MHz)	8	6.25
FR bit	0	1
Conversion time (μ s)	13.0	13.5

★

(4) AM2-AM0 bits (bits 7-5)

These bits select an operation mode of the A/D converter.

The A/D converter of the μ PD78366A can operate in three modes. In addition, the result of the A/D conversion can be buffered in two modes: 1-buffer and 4-buffer modes. For details of the A/D conversion modes, refer to **8.4 Operation**.

Table 8-2. A/D Conversion Modes

Conversion Mode	Buffer Mode	Conversion Start Trigger
Mixed mode	1-buffer mode	Software trigger/interrupt input (INTP2/INTCM03)
	4-buffer mode	
Scan mode	1-buffer mode	Software trigger
Select mode	1-buffer mode	
	4-buffer mode	

Caution The A/D converter of the μ PD78366A cannot stop conversion. When data is written to the ADM register, therefore, the converter continues to operate in the specified mode, until the contents of the ADM register are changed.

Remark The start trigger of the A/D conversion is a signal that makes the operation mode set by the ADM register valid, and starts A/D conversion.
For details, refer to **8.4 Operation**.

8.3 A/D Conversion Result Register (ADCR)

The μ PD78366A is provided with eight 10-bit A/D conversion result registers (ADCRs) to store the results of A/D conversion.

Each ADCR register can be read independently of the others by a 16-bit manipulation instruction or 8-bit manipulation instruction.

The result of conversion can be read from an ADCR register in the following two ways:

(1) Word access (execution of 16-bit manipulation instruction)

The lower 10 bits of the read word data are valid.

The higher 6 bits are always “0” when they are read.

Figure 8-4 illustrates word access to an ADCR register.

Figure 8-4. Word Access to ADCR 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)

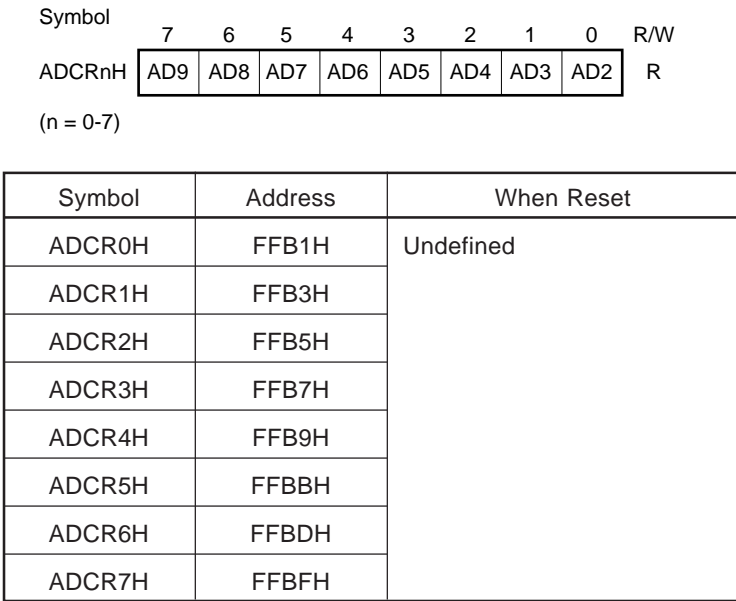
Symbol	Address	When Reset
ADCR0	FFB0H	Undefined
ADCR1	FFB2H	
ADCR2	FFB4H	
ADCR3	FFB6H	
ADCR4	FFB8H	
ADCR5	FFBAH	
ADCR6	FFBCH	
ADCR7	FFBEH	

Remark AD0-AD9: A/D conversion result

(2) Byte access (execution of 8-bit manipulation instruction)

The higher 8 bits of the A/D conversion result, which is 10-bit data, are read.
Figure 8-5 illustrates byte access to an ADCR register.

Figure 8-5. Byte Access to ADCR Register



Remark AD2-AD9: A/D conversion result (higher 8 bits of 10 bits)

8.4 Operation

8.4.1 Basic operation of A/D converter

A/D conversion is carried out in the following sequence:

- (1) Analog input signal and operation mode are selected by the A/D converter mode register (ADM). If a software trigger is specified and when data is written to the ADM register, A/D conversion is started.
- (2) The voltage generated by the resistor string is compared with the analog input voltage by the comparator for each 1 bit of the SAR register.
- (3) When all 10 bits have been compared, the A/D conversion result remains in the SAR register. This result is transferred to the A/D conversion result register (ACDR) and is latched.

At the same time, an A/D conversion end interrupt (INTAD) occurs (refer to **Figure 8-6**).

Caution Do not apply a voltage exceeding the range of AV_{SS} to AV_{DD} to the pin used as the input pin of the A/D converter.

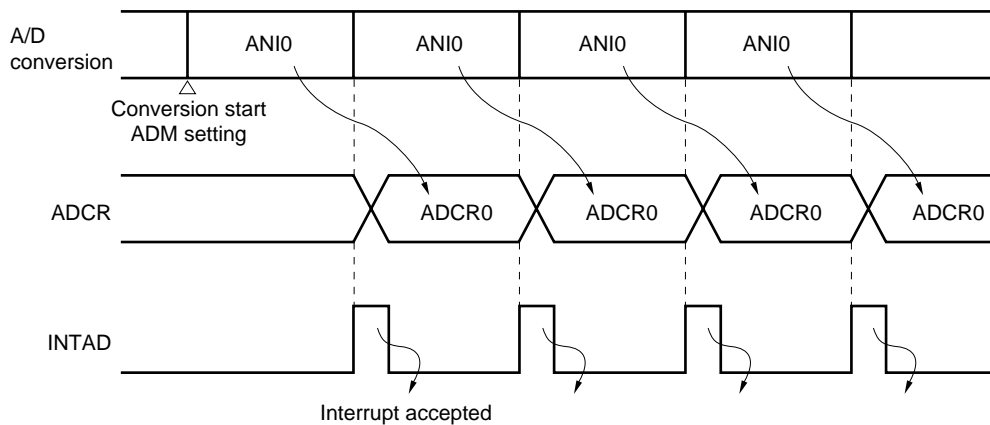
Operation of the A/D converter is started by one of the three conversion start triggers (software trigger, INTP2, or INTCM03) (the conversion mode set by the ADM register is made valid).

(a) Software trigger

When data is written to the ADM register, A/D conversion is started. A/D conversion can be synchronized by this trigger through software.

This trigger can be used in all the A/D conversion modes (select, scan, and mixed modes).

Figure 8-6. A/D Conversion Basic Operation (in select mode, with software trigger)



(b) External trigger (INTP2, INTCM03)

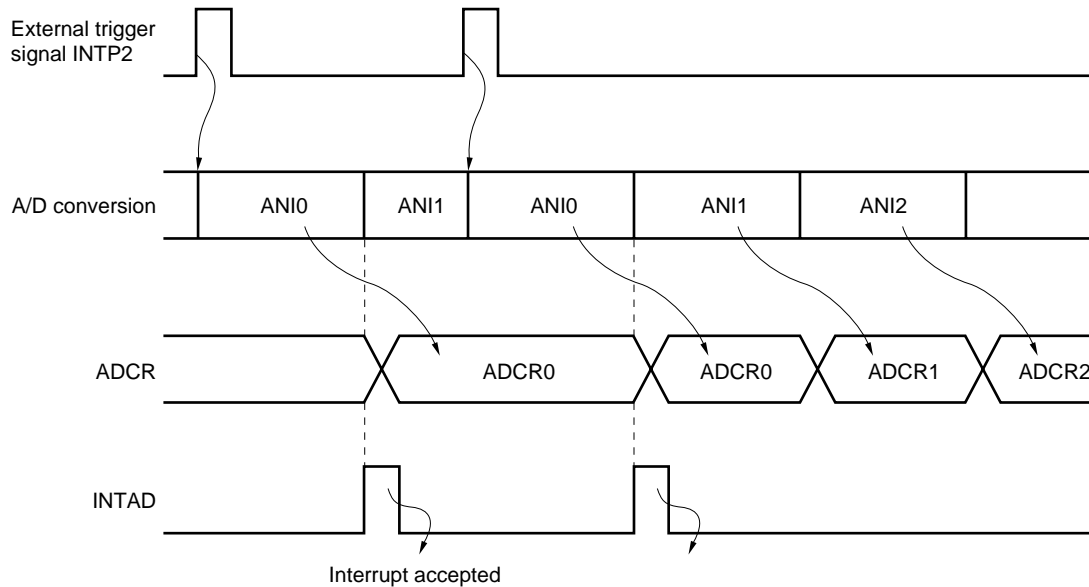
When the valid edge is input to external interrupt pin INTP2, or when INTCM03 that indicates coincidence between TM0 and CM03 of the real-time pulse unit (RPU) occurs, A/D conversion is started.

This trigger can only be used in the mixed mode.

Each time the valid edge is input to INTP2 or INTCM03 occurs, the SAR register is initialized, and A/D conversion is started.

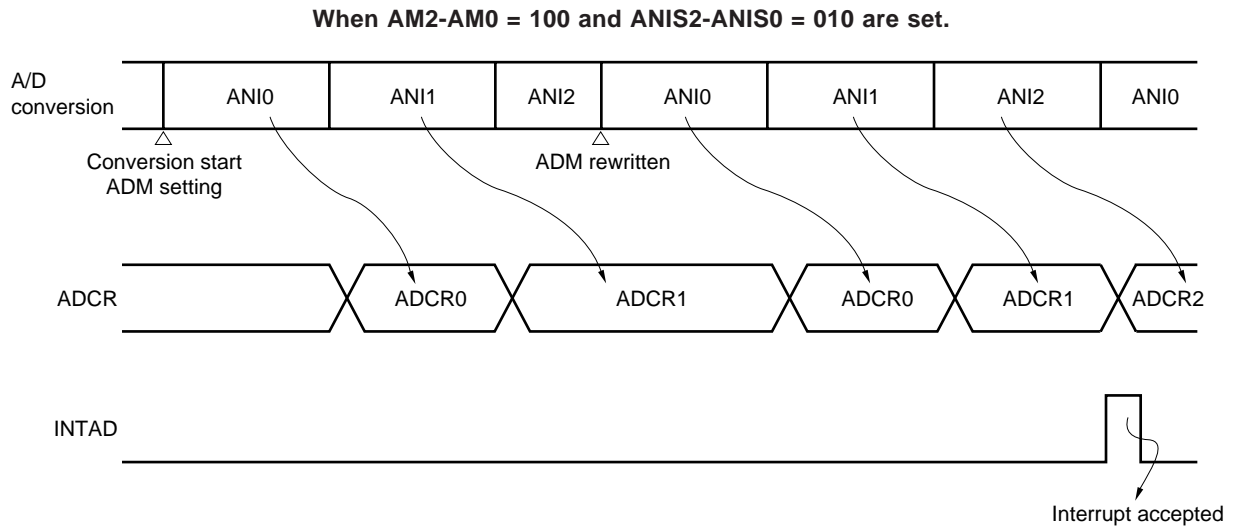
Which of INTP2 or INTCM03 is to be made valid can be specified by the IS bit of the ADM register.

Figure 8-7. A/D Conversion Basic Operation (in mixed mode, with external trigger)



If data is written to the ADM register while A/D conversion is in progress, conversion is initialized and started from the beginning.

Figure 8-8. Rewriting ADM during A/D Conversion (in scan mode, with software trigger)



Caution When performing branch processing directly using the values resulting from the A/D conversion, if a program is created that branches only when the conversion result reaches a specific value, the conversion result may not reach that specific value due to the effect of a conversion error and the program may not be able to branch to the prescribed routine. Therefore, create a program that will branch when the conversion result is within the overall error range.

The following is an example of a program that performs specific processing when an analog input voltage input to the ANI0 pin is $1/2 AV_{REF}$.

<Bad example>

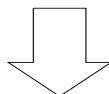
```

CMPW   ADCR0, #01FFH
BNE     $UNMATCH
        .
        .
        .
        } <1> Processing when A/D conversion result is 1FFH

UNMATCH:
        .
        .
        .
        } <2> Processing when A/D conversion result is other than 1FFH

```

Program cannot branch to perform processing <1> if conversion result does not become 1FFH due to conversion error.



<Good example>

```

CMPW   ADCR0, #0201H
BGT     $UNMATCH
CMPW   ADCR0, #01FCH
BLT     $UNMATCH
        .
        .
        .
        } <1> Processing when A/D conversion result is 1FCH-201H

UNMATCH:
        .
        .
        .
        } <2> Processing when A/D conversion result is outside the above-mentioned
           range

```

If the conversion result is within the range of 1FCH-201H, processing continues, assuming that an analog voltage of $1/2 AV_{REF}$ has been input.

8.4.2 Operation mode of A/D converter

The following three modes can be selected as the operation modes of the A/D converter. They are selected by the A/D converter mode register (ADM). A/D conversion is started when data is written to the ADM register (software trigger), and continues until the contents of the ADM register are changed.

- Select mode
- Scan mode
- Mixed mode

(1) Select mode

In this mode, one analog input signal (ANIn: $n = 0-7$) specified by the ADM register is converted into a digital signal. The result of the conversion is stored in the A/D conversion result register (ADCRn: $n = 0-7$) corresponding to the analog input signal.

The following two buffer modes can be selected depending on how the conversion result is stored:

- 1-buffer mode
- 4-buffer mode

(a) 1-buffer mode

In this mode, one analog input signal is converted once into a digital signal, which is then stored in an A/D conversion result register (refer to **Table 8-3**). The analog input signal and the A/D conversion result register correspond to each other on a one-to-one basis.

An A/D conversion end interrupt (INTAD) occurs each time conversion has been completed, to indicate the end of the conversion. In this mode, A/D conversion continues even after that.

Figure 8-9. A/D Conversion in Select Mode (1-buffer mode)

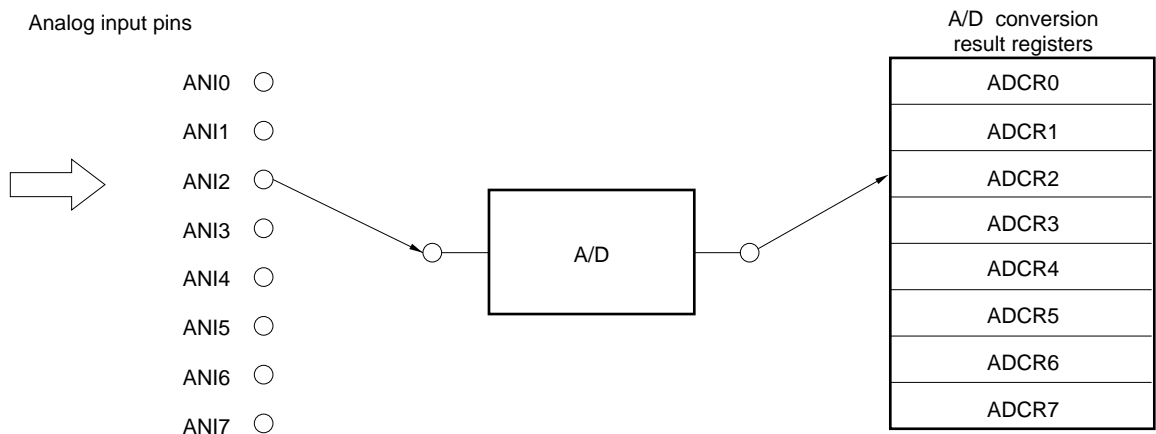
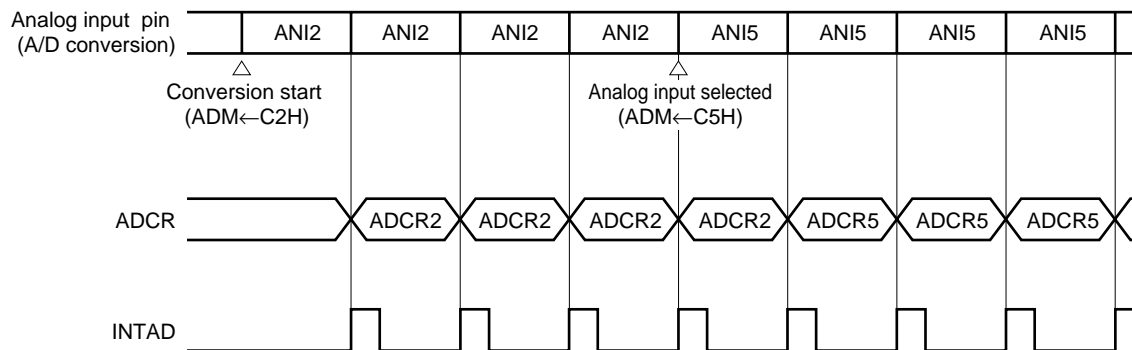


Table 8-3. Correspondence Between Analog Input and A/D Conversion Result Registers (ADCRs)
(select mode, 1-buffer mode)

Analog Input	ADCR
ANI0	ADCR0
ANI1	ADCR1
ANI2	ADCR2
ANI3	ADCR3
ANI4	ADCR4
ANI5	ADCR5
ANI6	ADCR6
ANI7	ADCR7

Figure 8-10. Example of Operation Timing in Select Mode (1-buffer mode)



(b) 4-buffer mode

In this mode, one analog input signal is converted four times to a digital signal. The resultant four digital signals are stored in four A/D conversion result registers (ADCR0-ADCR3 or ADCR4-ADCR7) (refer to **Table 8-4**).

When the conversion has been executed four times, an A/D conversion end interrupt (INTAD) occurs, to indicate the end of the conversion. In this mode, A/D conversion continues even after that.

This mode is suitable for applications where the average of the A/D conversion results needs to be obtained.

Figure 8-11. A/D Conversion in Select Mode (4-buffer mode)

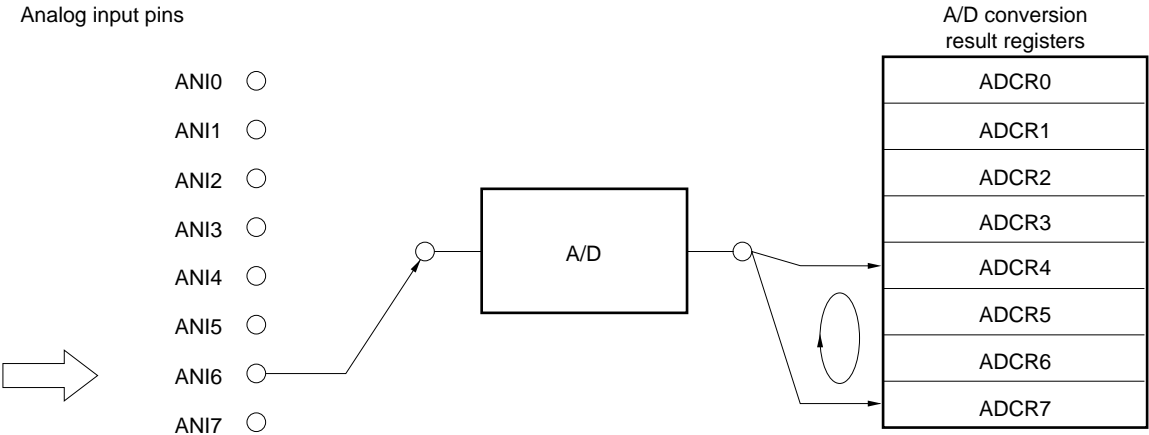
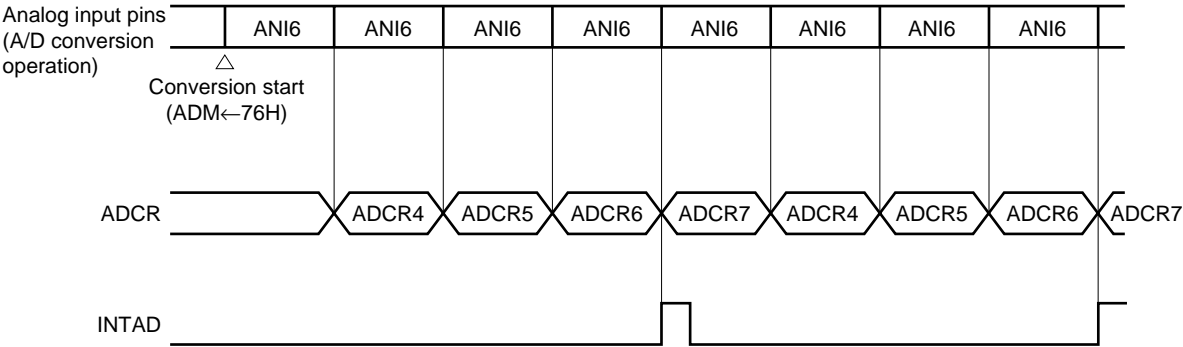


Table 8-4. Correspondence Between Analog Input and A/D Conversion Result Registers (ADCRs) (select mode, 4-buffer mode)

Analog Input	ADCR
ANI0	ADCR0 to ADCR3
ANI1	
ANI2	
ANI3	
ANI4	ADCR4 to ADCR7
ANI5	
ANI6	
ANI7	

Figure 8-12. Example of Operation Timing in Select Mode (in 4-buffer mode)



(2) Scan mode

In this mode, two or more analog input signals specified by the ADM register are sequentially converted into digital signals. Each digital signal is stored in an A/D conversion result register corresponding to the analog input signal (refer to **Table 8-5**). Conversion is started when data is written to the ADM register (software trigger). The end of the conversion can be detected by an A/D conversion end interrupt (INTAD) that occurs when the specified analog input signals have been completely converted (i.e., scanned). In this mode, A/D conversion is repeated by scan operation.

This mode is suitable for applications where two or more analog input signals always need to be monitored.

Figure 8-13. A/D Conversion in Scan Mode

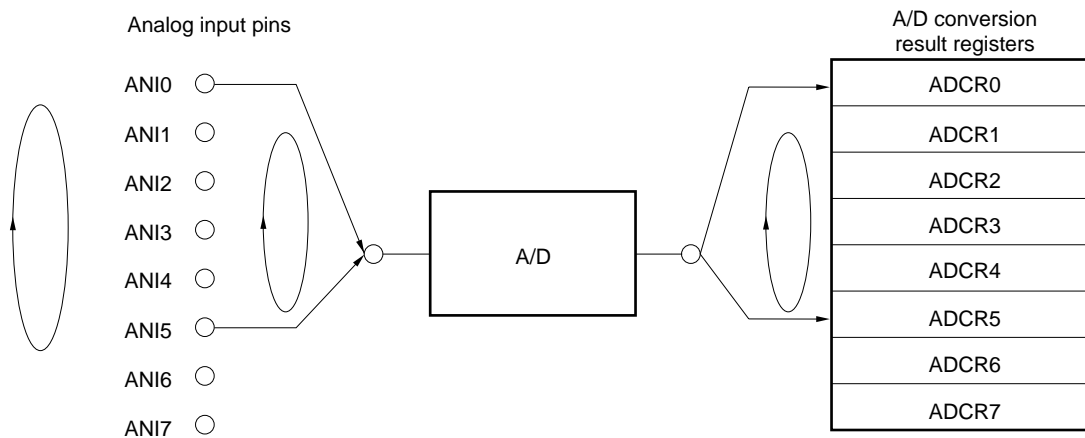
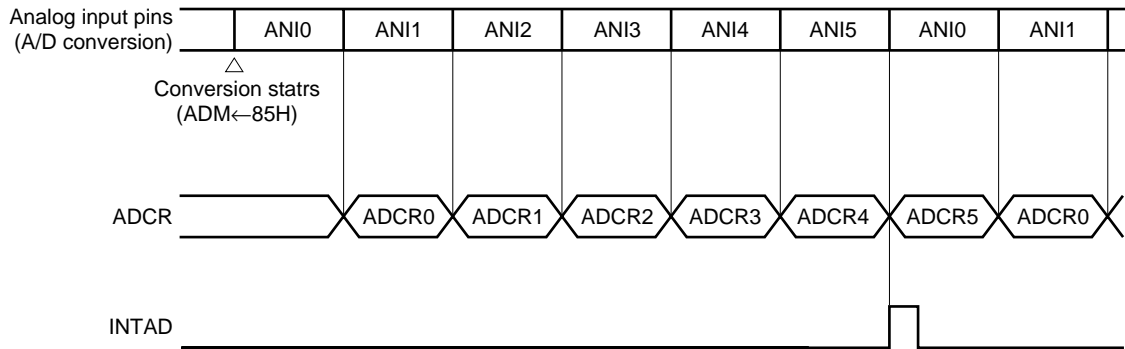


Table 8-5. Correspondence Between Analog Input Pins and A/D Conversion Result Registers (ADCRs) (scan mode)

Analog Input	ADCR
ANI0	ADCR0
ANI1	ADCR1
ANI2	ADCR2
ANI3	ADCR3
ANI4	ADCR4
ANI5	ADCR5
ANI6	ADCR6
ANI7	ADCR7

Figure 8-14. Example of Operation Timing in Scan Mode**(3) Mixed mode**

This mode is a combination of the select and scan modes.

A/D conversion in the mixed mode can be started by the following three triggers. These triggers are used to start A/D conversion in the select mode (select processing). A/D conversion in the scan mode (scan processing) is then executed.

- Software trigger : Writing data to ADM register
- External trigger : Input of valid edge to INTP2 pin (valid edge is specified by INTM0 register)
- Interrupt trigger from RPU : Interrupt INTCM03 that occurs when TM0 and CM03 coincide

To use the software trigger, select processing is first executed when data is written to the ADM register, followed by scan processing.

When the external trigger or interrupt trigger is used, scan processing is executed first when data is written to the ADM register, and select processing is executed when the external trigger or interrupt trigger is input. When select processing is completed, scan processing is continued.

An A/D conversion end interrupt (INTAD) occurs when select processing has been completed. A/D conversion by scan processing continues until a new trigger is input. If a new trigger is input, select processing is executed again, and when it is completed, scan processing executed before the new trigger was input is resumed. A/D conversion in the mixed mode can be executed in two buffer modes, depending on how the result of select processing is stored: 1-buffer mode or 4-buffer mode.

The analog input signals to be scanned are determined by the buffer mode and the analog input signal selected for processing.

Table 8-6. Analog Input in Mixed Mode

Select Processing		Scan Processing
Buffer Mode	Selection of Analog Input	
1-buffer mode	One of ANI0-ANI7 selected	ANI0-ANI7 scanned
4-buffer mode	One of ANI0-ANI3 selected	ANI4-ANI7 scanned
	One of ANI4-ANI7 selected	ANI0-ANI3 scanned

(a) Select processing in 1-buffer mode

In the select mode, one analog input signal is specified and converted. In the scan mode, all the analog input signals are sequentially converted. After that, A/D conversion (scan processing) is repeatedly executed.

The result of the conversion is stored in the A/D conversion result register corresponding to the analog input signal, as shown in Table 8-7.

Figure 8-15. A/D Conversion in Mixed Mode (select processing in 1-buffer mode)

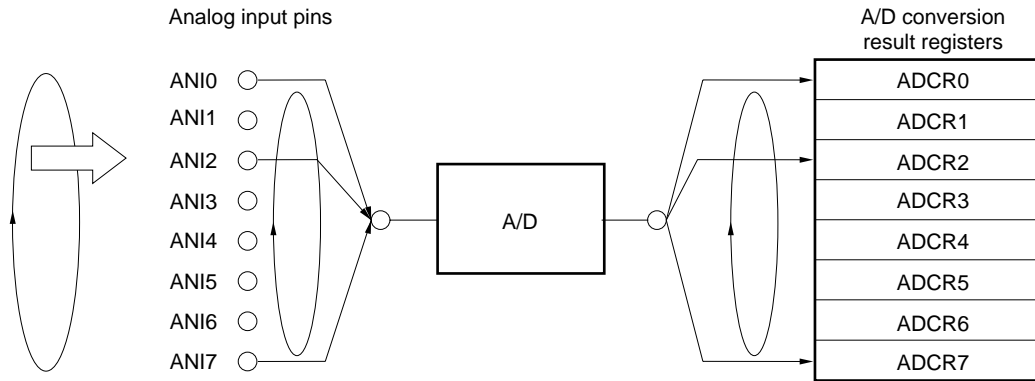
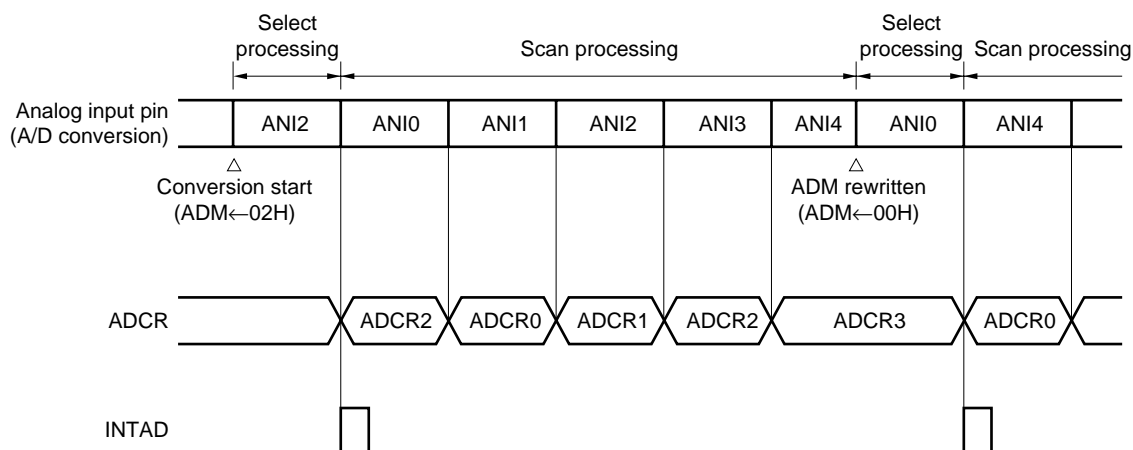
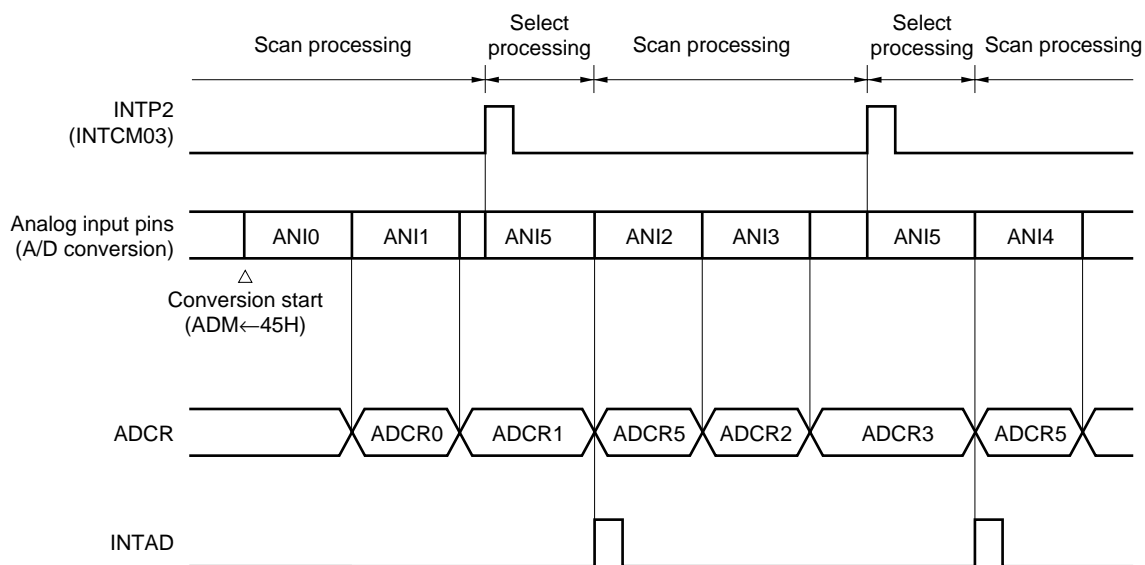


Table 8-7. Correspondence Between Analog Input and A/D Conversion Result Registers (ADCRs)
(mixed mode: select processing in 1-buffer mode)

Analog Input	ADCR	
	Select Mode	Scan Mode
ANI0	ADCR0	Stores result of conversion of all analog inputs (ANI0-ANI7) to ADCR0-ADCR7
ANI1	ADCR1	
ANI2	ADCR2	
ANI3	ADCR3	
ANI4	ADCR4	
ANI5	ADCR5	
ANI6	ADCR6	
ANI7	ADCR7	

Figure 8-16. Example of Operation Timing in Mixed Mode (select processing in 1-buffer mode) (software trigger)

Caution To execute select processing by software trigger and start scan processing again, A/D conversion before software trigger was input is resumed.

Figure 8-17. Example of Operation Timing in Mixed Mode (select processing in 1-buffer mode) (external trigger or interrupt trigger)

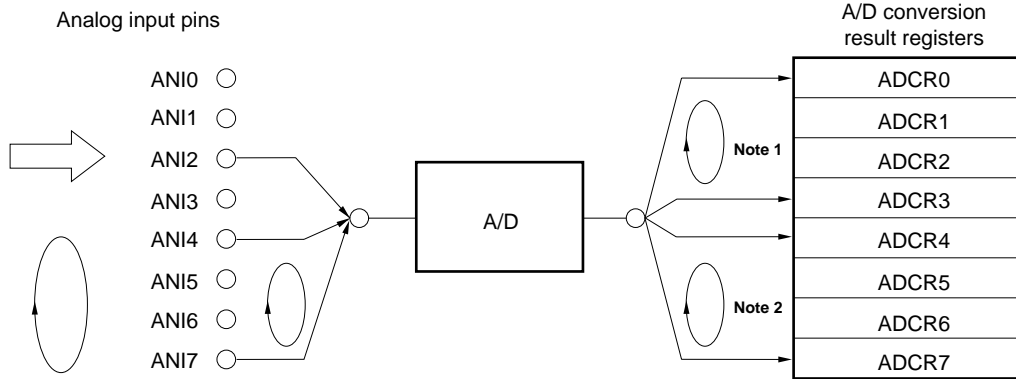
Caution To execute select processing by external trigger or interrupt trigger and to start scan processing again, A/D conversion before external trigger or interrupt trigger was input is resumed.

(b) Select processing in 4-buffer mode

In the select mode, one analog input signal is specified and converted to a digital signal. The resultant digital signal is stored in four A/D conversion result registers (ADCRs) corresponding to the analog input signal. In the scan mode, four analog input signals excluding the analog signal processed in the select mode (e.g., when ANI2 is selected for select processing, ANI4-ANI7 are processed in the scan mode) are sequentially converted. After that, scan processing is repeated.

The conversion results are stored in the ADCR registers corresponding to the analog input signals, as shown in Table 8-8.

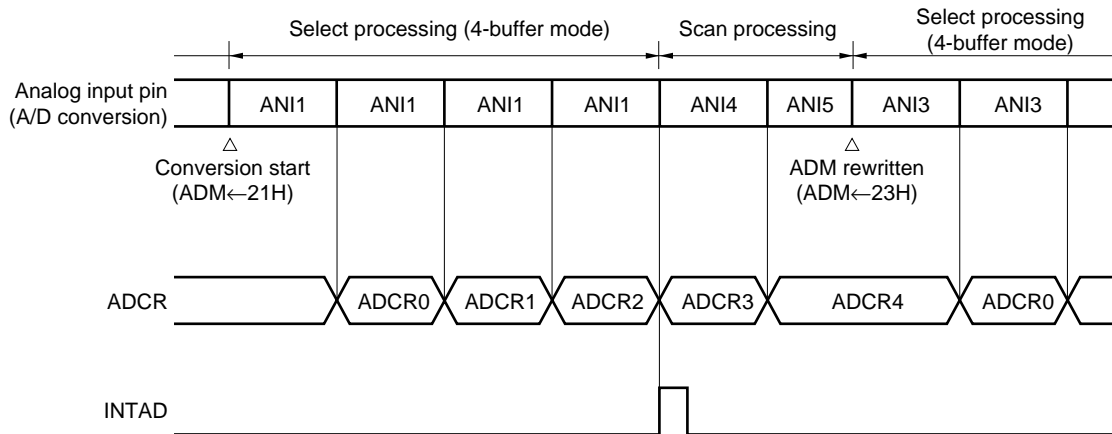
Figure 8-18. A/D Conversion Timing in Mixed Mode (select processing in 4-buffer mode)



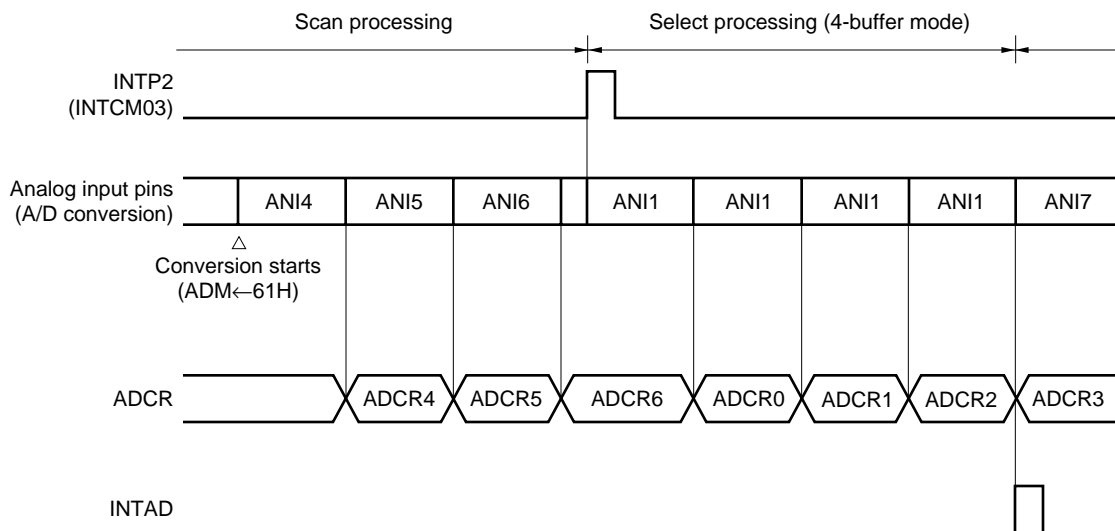
- Notes**
1. Select processing (4-buffer mode)
 2. Scan processing

Table 8-8. Correspondence Between Analog Input and A/D Conversion Result Registers (ADCRs) (in mixed mode: select processing in 4-buffer mode)

Analog Input	ADCR	
	Select Processing	Scan Processing
ANI0	Stores conversion result of specified analog input to ADCR0-ADCR3	ANIn → ADCRn (n = 4-7)
ANI1		
ANI2		
ANI3		
ANI4	Stores conversion result of specified analog input to ADCR4-ADCR7	ANIn → ADCRn (n = 0-3)
ANI5		
ANI6		
ANI7		

Figure 8-19. Example of Operation Timing in Mixed Mode (select processing in 4-buffer mode) (software trigger)

Caution To execute select processing by software trigger and start scan processing again, A/D conversion before software trigger was input is resumed.

Figure 8-20. Example of Operation Timing in Mixed Mode (select processing in 4-buffer mode) (external trigger or interrupt trigger)

Caution To execute select processing by external trigger or interrupt trigger and to start scan processing again, A/D conversion before external trigger or interrupt trigger was input is resumed.

8.5 How to Read A/D Converter Characteristic Tables

This section describes the technical terms peculiar to the A/D converter.

(1) Resolution

Minimum analog input voltage that can be identified. The ratio of 1 digital output bit to an analog input voltage is said to be 1LSB (Least Significant Bit). The ratio of the full scale to 1LSB is expressed in %FSR (Full Scale Range).

Where the resolution is 10 bits,

$$\begin{aligned} 1\text{LSB} &= 1/2^{10} = 1/1024 \\ &= 0.098\%\text{FSR} \end{aligned}$$

The accuracy is independent of the resolution and is determined by the total error.

(2) Total error

Maximum difference between actually measured and theoretical values.

Total error of a zero scale error, full scale error, non-linearity error, and combination of these errors.

Note that the total error set forth in the characteristic table does not include the quantized error.

(3) Quantized error

An error of $\pm 1/2\text{LSB}$ that inevitably occurs when an analog value is converted into a digital value. Since an A/D converter converts an analog voltage in the range of $\pm 1/2\text{LSB}$ into the same digital code, a quantized error is unavoidable.

This error is not included in the total error, zero scale error, full scale error, and non-linearity error set forth in the characteristic table.

Figure 8-21. Total Error

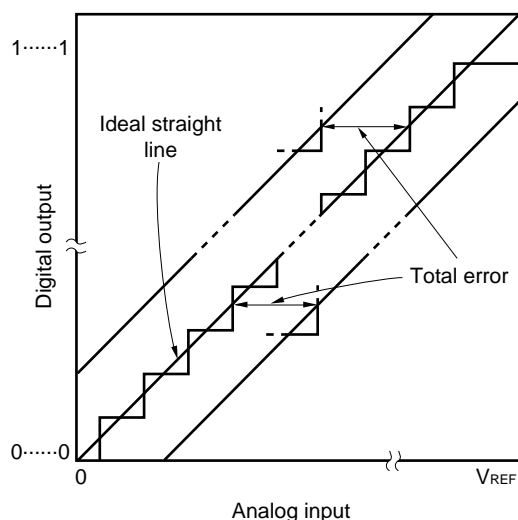
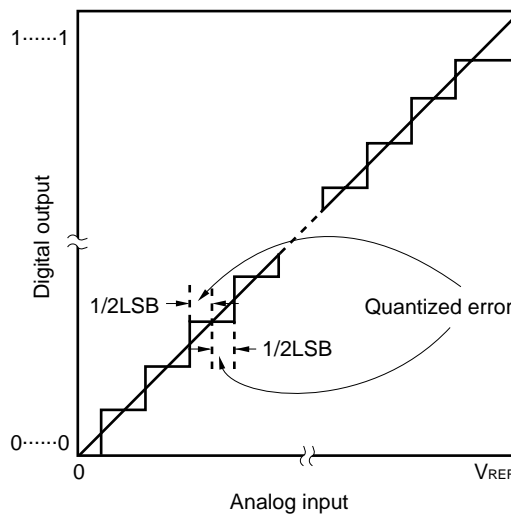


Figure 8-22. Quantized Error

**(4) Zero scale error**

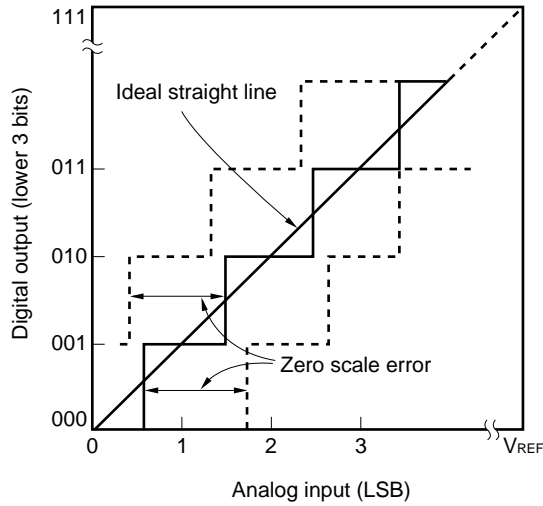
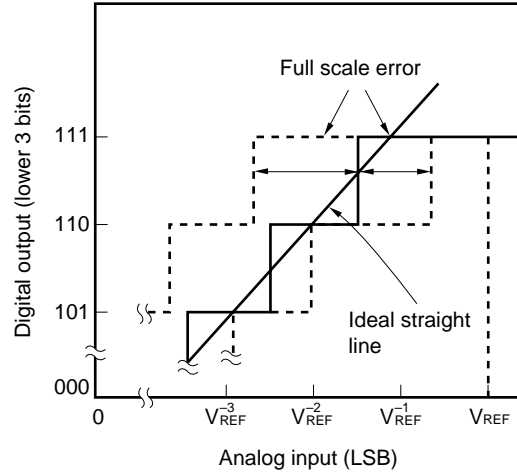
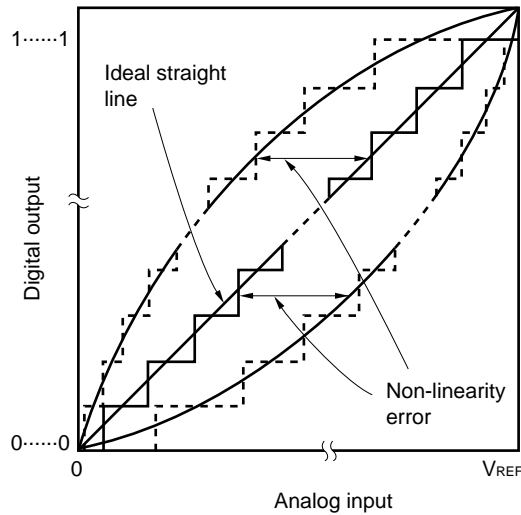
This is the difference between the actually measured value of an analog input voltage and the theoretical value ($1/2\text{LSB}$) when the digital output changes from 0...000 to 0...001. If the measured value is greater than the theoretical value, it is the difference between the actually measured value of the analog input voltage and the theoretical value ($3/2\text{LSB}$) when the digital output changes from 0...001 to 0...010.

(5) Full scale error

This is the difference between the actually measured value of an analog input voltage and the theoretical value (full scale - $3/2\text{LSB}$) when the digital output changes from 1 ... 110 to 1 ... 111.

(6) Non-linearity error

This is the degree to which the conversion characteristics shift from the ideal straight line. It indicates the maximum difference between the measured value and the ideal straight line where the zero scale error and full scale error are 0.

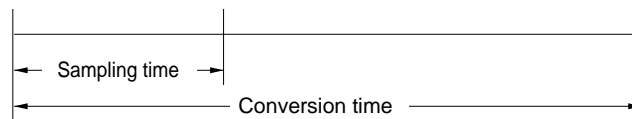
Figure 8-23. Zero Scale Error**Figure 8-24. Full Scale Error****Figure 8-25. Non-Linearity Error****(7) Conversion time**

Time required from when an analog input voltage is given until the digital output is obtained.

The conversion time set forth in the characteristics table includes the sampling time.

(8) Sampling time

Time during which an analog switch is ON to load an analog voltage to the sample and hold function.



CHAPTER 9 ASYNCHRONOUS SERIAL INTERFACE

The μ PD78366A is provided with a UART (Universal Asynchronous Receiver Transmitter) asynchronous serial interface. This interface transmits or receives 1-byte serial data following a start bit and can perform full duplex operation. A baud rate generator is also provided, so that communication can be established in a wide range of baud rates.

With the μ PD78366A, two sets of transmission/reception pins can be selected by software.

The asynchronous serial interface is independent of the clocked serial interface.

Remark Unless otherwise specified, this chapter describes the TxD0 and TxD1, and RxD0 and RxD1 pins as TxD and RxD pins, respectively.

9.1 Asynchronous Serial Interface Configuration

The asynchronous serial interface is controlled by the asynchronous serial interface mode register (ASIM) and asynchronous serial interface status register (ASIS). Receive data is stored in the receive buffer (RXB), and transmit data is written to the transmit shift register (TXS).

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

(1) Asynchronous serial interface mode register (ASIM)

The ASIM register is an 8-bit register that specifies the operation of the asynchronous serial interface. This register can be read or written by an 8-bit manipulation instruction or a bit manipulation instruction. It is initialized to 80H when the $\overline{\text{RESET}}$ signal is input.

(2) Asynchronous serial interface status register (ASIS)

The ASIS register is a collection of flags that identify the nature of an error in case a reception error occurs. Each flag is set to 1 when a reception error occurs, and is reset to 0 when data is read from the receive buffer (RXB) or when a new, subsequent error is received (if an error occurs in the next data, the corresponding error flag is set).

This register can only be read by an 8-bit manipulation instruction. It is initialized to 00H when the $\overline{\text{RESET}}$ signal is input.

(3) Reception control parity checking unit

Reception is controlled in accordance with the contents written to the ASIM register. Whether an error, such as a parity error, has occurred is also checked during reception. If an error is detected, a value corresponding to the error is written to the ASIS register.

(4) Receive shift register

This register converts the serial data input to the RxD pin into parallel data. When 1 byte of data is received, it is transferred to the receive buffer.

This register cannot be manipulated directly from the CPU.

(5) Receive buffer (RXB)

This register holds the receive data. Each time 1 byte of data is received, it is transferred from the shift register. When the data length is set to 7 bits, the receive data is transferred to bits 0 to 6 of the RXB, and the MSB of the RXB is always "0".

This register can only be read by an 8-bit manipulation instruction. Its contents are undefined when the $\overline{\text{RESET}}$ signal is input.

(6) Transmit shift register (TXS)

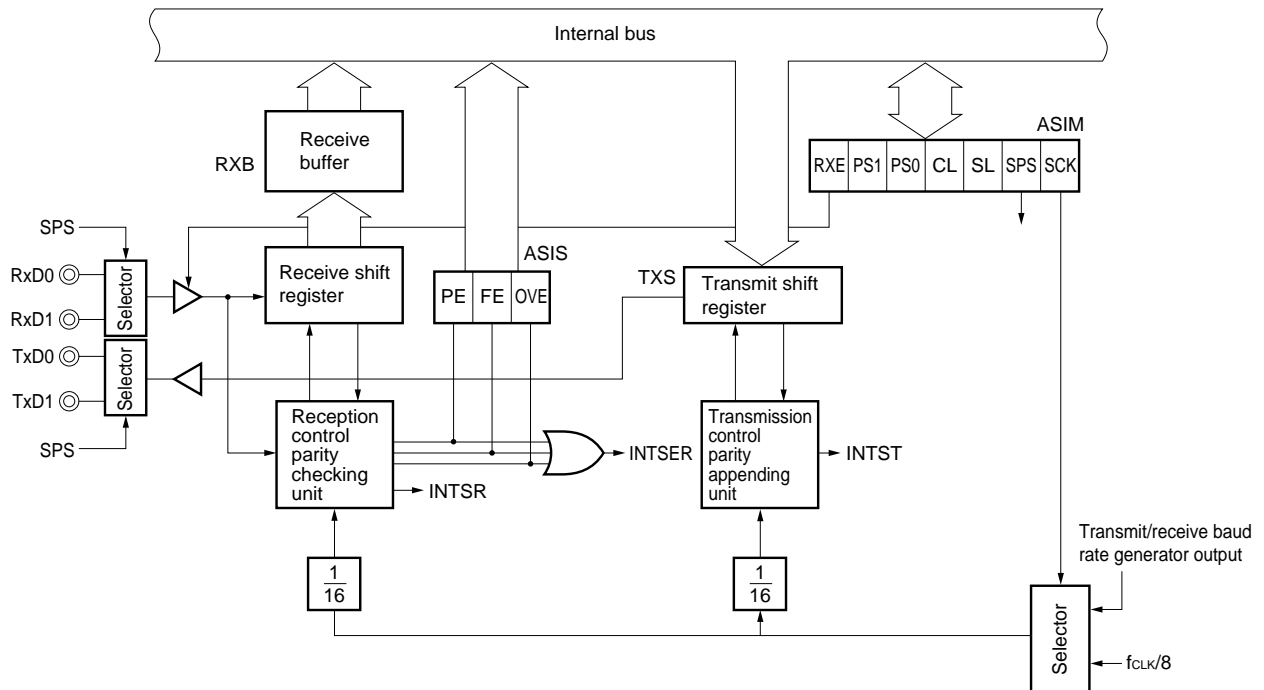
This register sets the data to be transmitted. Data written to the TXS register is transmitted as serial data. When the data length is set to 7 bits, bits 0 to 6 of the data written to the TXS are treated as transmit data. When data is written to the TXS register, transmission is started. Do not write to the TXS register while transmission is in progress.

Data can only be written to this register by an 8-bit manipulation instruction. The contents of the TXS register are undefined when the $\overline{\text{RESET}}$ signal is input.

(7) Transmission control parity appending unit

Transmission is controlled by automatically appending a start bit, parity bit and stop bit to the data written to the TXS register, in accordance with the contents written to the ASIM register.

Figure 9-1. Block Diagram of Asynchronous Serial Interface



9.2 Selecting Pins for Serial Communication

The asynchronous serial interface selects the pins used for serial communication through software.

- **Selecting pins for serial communication**

The asynchronous serial interface selects the pins used for serial communication (TxD0 and RxD0, or TxD1 and RxD1 pins) through software. Selection is made by using the SPS bit of the asynchronous serial interface mode register (ASIM), and the two sets of pins can be used by means of time-division multiplexing.

- (a) **To use TxD0 and RxD0 pins**

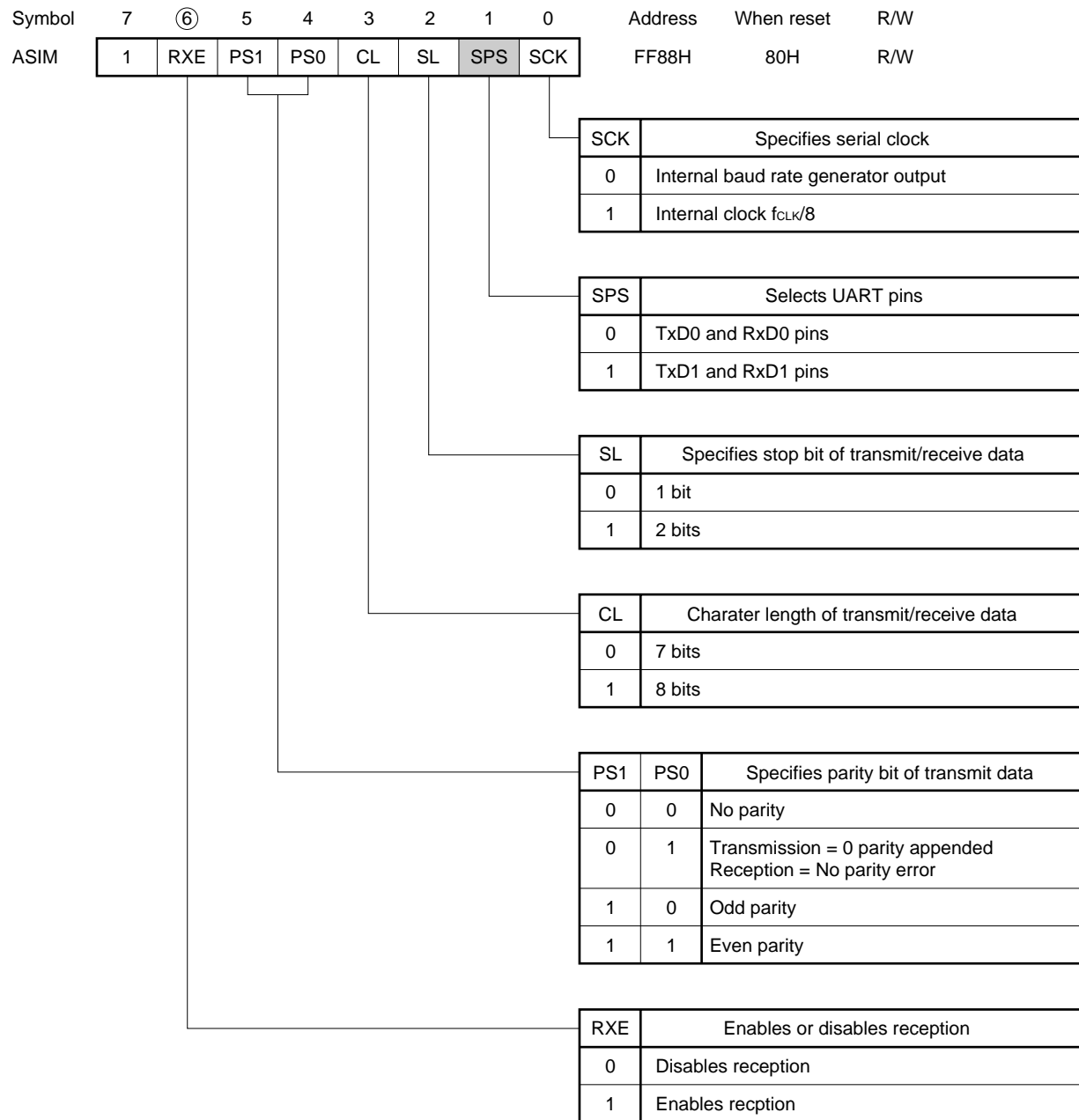
To select the TxD0 and RxD0 pins for serial communication, reset the SPS bit of the ASIM register to 0. At this time, the levels of the TxD1 and RxD1 pins not used for serial communication are as follows:

- TxD1 ... High level
- RxD1 ... High impedance

- (b) **To use TxD1 and RxD1 pins**

To select the TxD1 and RxD1 pins for serial communication, set the SPS bit of the ASIM register to 1. At this time, the levels of the TxD0 and RxD0 pins not used for serial communication are as follows:

- TxD0 ... High level
- RxD0 ... High impedance

Figure 9-2. Setting of ASIM Register (pin selection)

Caution Bit 7 of the ASIM register is fixed to “1” by hardware. Even if “0” is written, they remain “1”.

Remark f_{CLK} : internal system clock

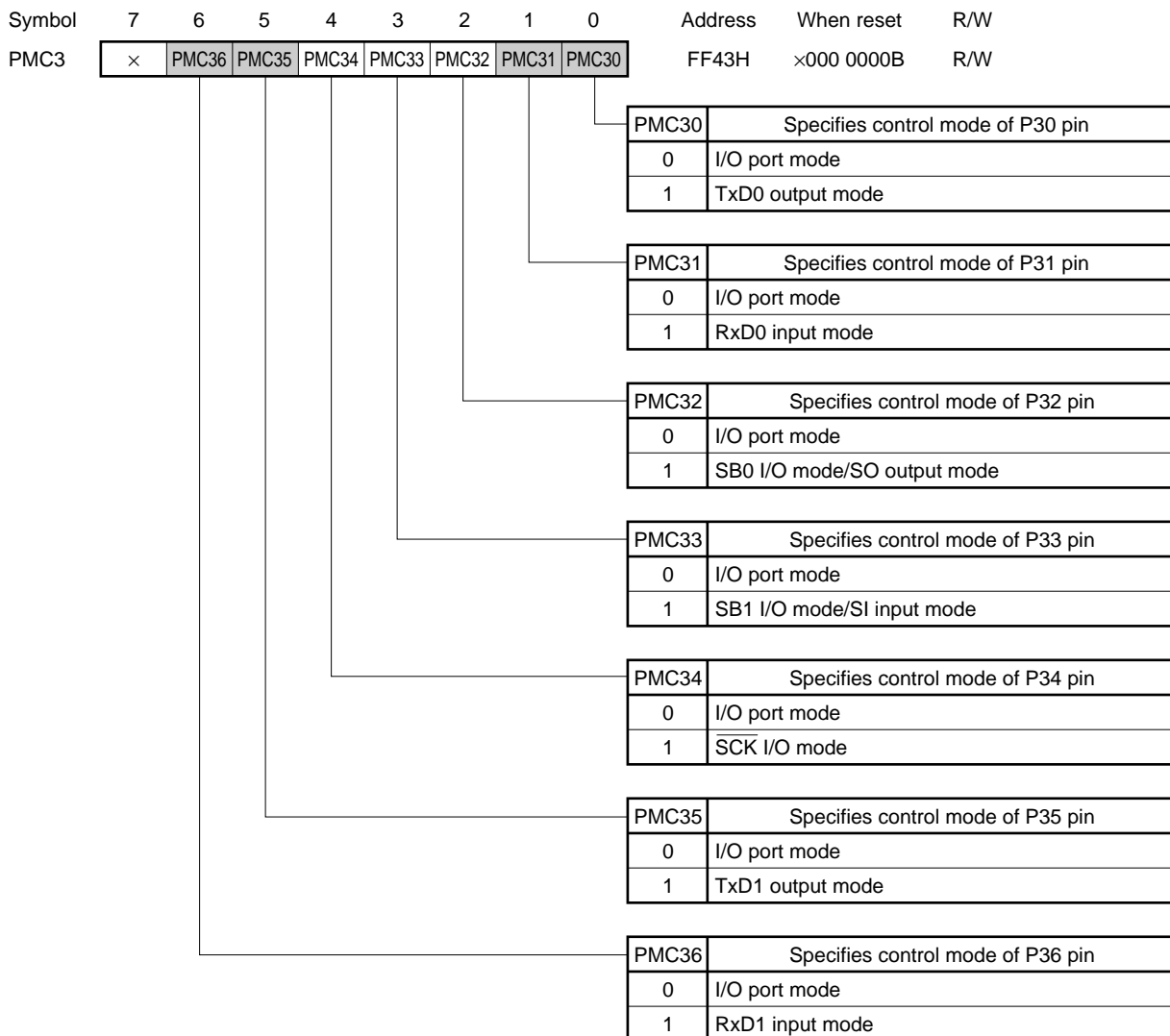
9.3 Setting Pins for Serial Communication

Since the RxD and TxD pins are multiplexed with general-purpose port pins, they must be set in the control mode before communication is started.

(1) Setting pins for serial communication

The asynchronous serial interface uses the TxD0 (or TxD1) pin for transmission and the RxD0 (or RxD1) pin for reception. These pins, however, are multiplexed with general-purpose port pins P30 (P35) and P31 (P36), respectively. Therefore, before starting communication, each pin must be set in the control mode by using the port 3 mode control register (PMC3).

Figure 9-3. Format of Port 3 Mode Control Register



Remark ×: don't care

(2) Reading pin level

When port 3 (P3) is set in the control mode by the port 3 mode control register (PMC3), the following statuses can be read when an instruction that reads port 3 (P3) is executed:

(a) TxD0/P30 pin (or TxD1/P35 pin)

- When bit 0 (or bit 5) of the port 3 mode register (PM3) is set to 1, the level of the TxD pin can be read.
- When bit 0 (or bit 5) of the port 3 mode register (PM3) is reset to 0, the level of the internal transmit data can be read.

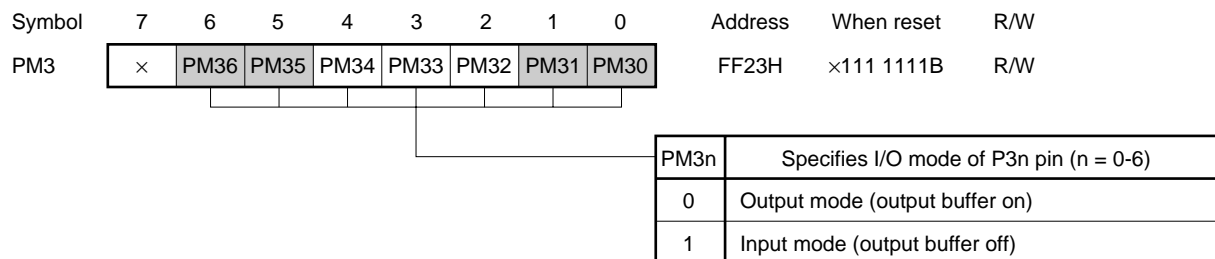
(b) RxD0/P31 pin (or RxD1/P36 pin)

- When bit 1 (or bit 6) of the port 3 mode register (PM3) is set to 1, the level of the RxD pin can be read.

By reading the pin level, contention of the TxD pin can be checked.

The levels of the TxD and RxD pins do not change even if data is written to port 3 (P3) (data is written to the output buffer of port 3).

Figure 9-4. Format of Port 3 Mode Register



Remark ×: don't care

9.4 Data Format Setting

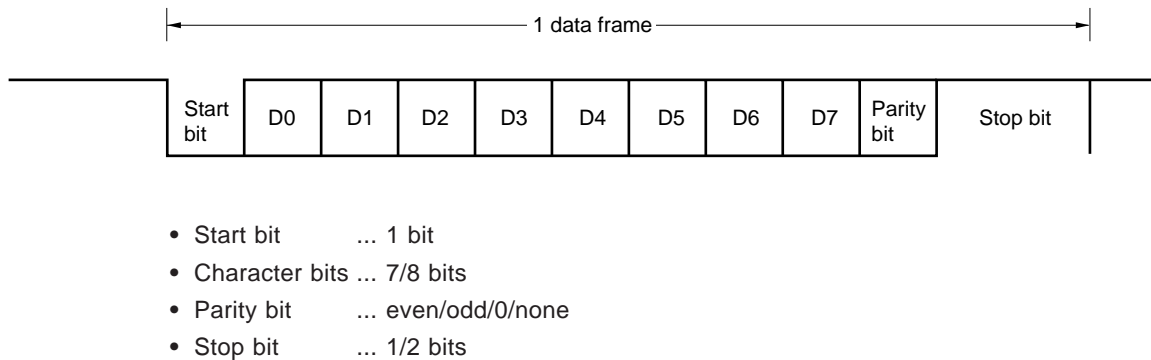
The character bit length, parity, and stop bit length are specified by the asynchronous serial interface mode register (ASIM).

(1) Setting data format

Figure 9-5 shows the format of the transmit/receive data. One data frame consists of a start bit, character bits, parity bit, and stop bit(s).

The character bit length, parity, and stop bit length of one data frame are specified by the asynchronous serial interface mode register (ASIM) (refer to **Figure 9-6**).

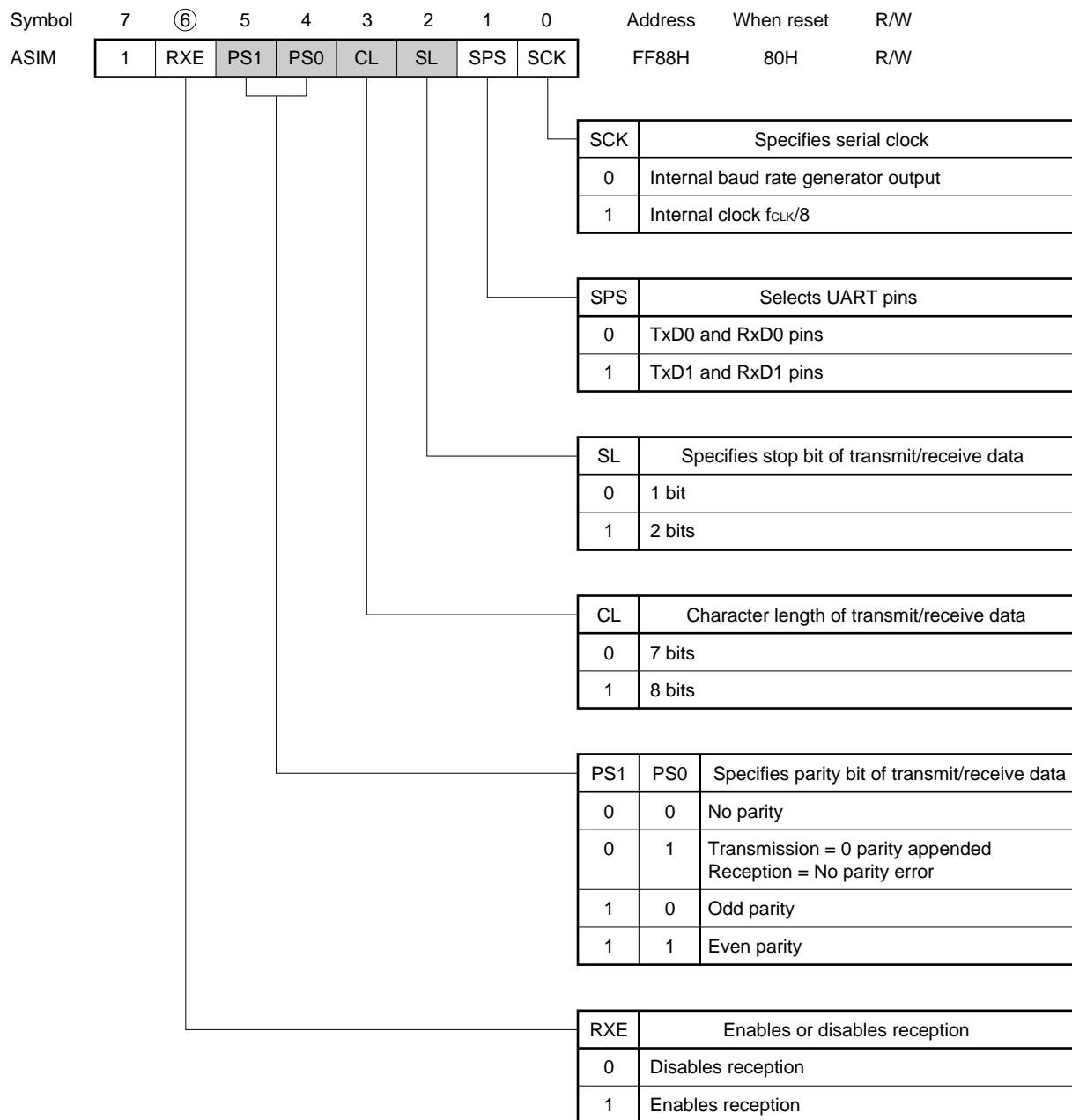
Figure 9-5. Format of Transmit/Receive Data of Asynchronous Serial Interface



(2) 0 parity is effective for starting up system

The asynchronous serial interface of the μ PD78366A is provided with a “0 parity” function as a special transmit/receive data format.

When “0 parity” is selected, “0” is unconditionally appended to the transmit data when serial data is transmitted. Serial data is received regardless of the status of the parity bit appended to it, and a parity error does not occur. The “0 parity” function is useful for executing serial communication when the data format is not determined such as when power is applied to the system.

Figure 9-6. Setting of ASIM Register (data format)

Caution Bit 7 of the ASIM register is fixed to “1” by hardware. Even if “0” is written, they remain “1”.

Remark f_{CLK} : internal system clock

9.5 Baud Rate Setting

The output of the baud rate generator or internal clock $f_{CLK}/8$ can be selected as the serial clock. When the baud rate generator output is selected, communication can be carried out at the specified baud rate, regardless of the operating frequency.

(1) Baud rate = serial clock/16

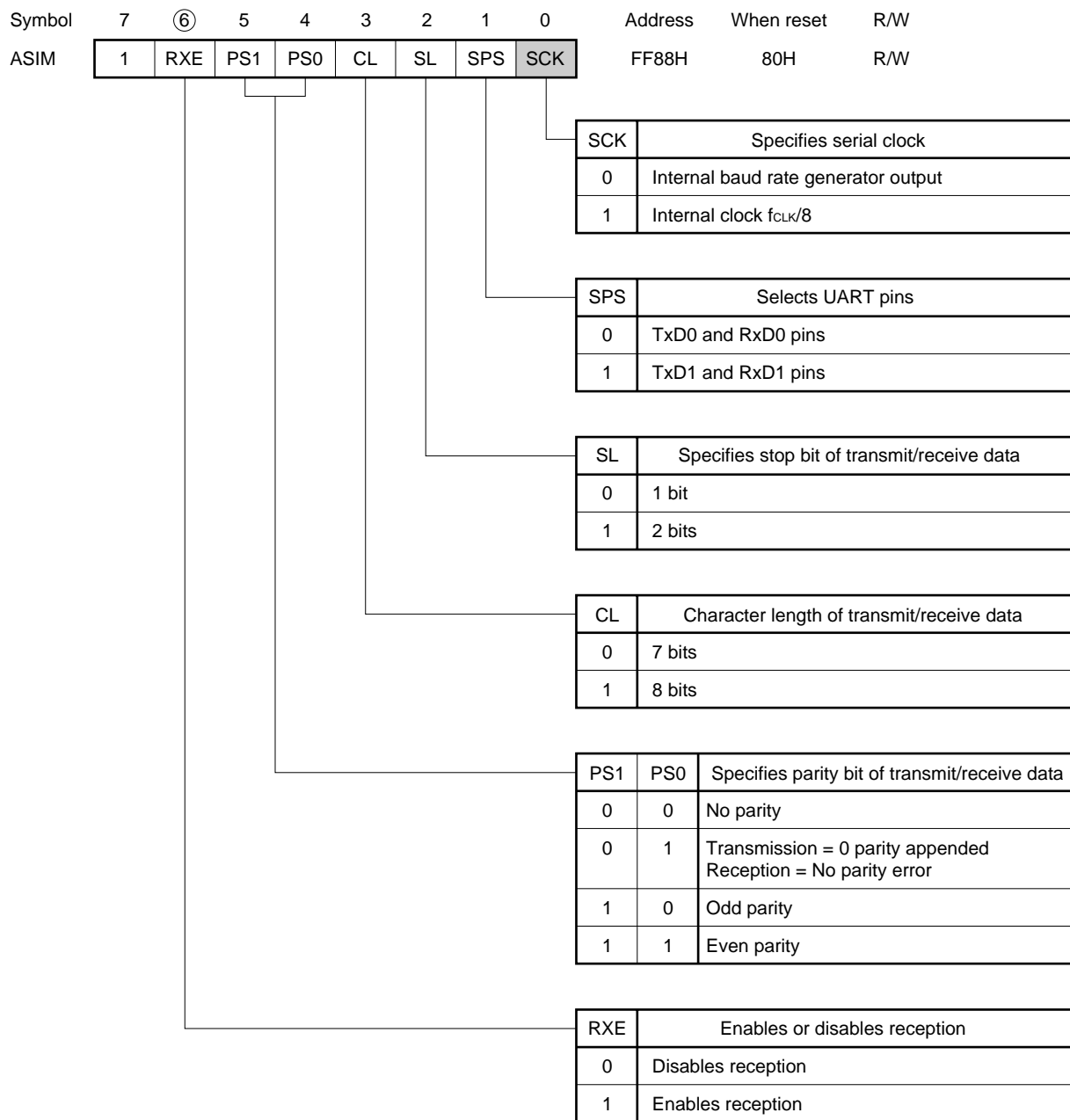
The asynchronous serial interface samples the level of the RxD pin with the serial clock specified by the asynchronous serial interface mode register (ASIM) divided by 16.

(2) Selecting serial clock

The serial clock is selected by using the SCK bit of the asynchronous serial interface mode register (ASIM) (refer to **Figure 9-7**).

When internal system clock $f_{CLK}/8$ is selected, the baud rate is $f_{CLK}/128$. Therefore, when the internal system clock is 16 MHz, the baud rate is 125 Kbps.

Remark The baud rate generator is shared with the clocked serial interface (refer to **10.3 Setting Baud Rate**).

Figure 9-7. Setting of ASIM Register (serial clock)

Caution Bit 7 of the ASIM register is fixed to “1” by hardware. Even if “0” is written, they remain “1”.

Remark f_{CLK} : internal system clock

9.5.1 Baud rate generator configuration

The baud rate generator is controlled by the baud rate generator control register (BRGC) and 8-bit compare register (BRG).

The baud rate generator is configured as shown in Figure 9-8.

(1) Baud rate generator control register (BRGC)

This 8-bit register is used to select the count clock for the 8-bit timer (TMBRG) and control operation of the baud rate generator.

It can be read or written by an 8-bit manipulation instruction or a bit manipulation instruction.

The contents of this register are initialized to 00H when the $\overline{\text{RESET}}$ signal is input.

The format of the BRGC register is shown in Figure 9-9.

(2) Prescaler

The prescaler divides the internal clock ($f_{\text{CLK}}/2$) according to the setting of the BRGC register and generates count clocks.

(3) 8-bit timer (TMBRG)

This 8-bit timer counts the count clock generated by the prescaler.

When a coincidence signal is generated by the 8-bit compare register (BRG), the timer is cleared to 0 by the next count clock.

This timer is started or stopped by the BRGC register.

TMBRG cannot be handled directly from the CPU.

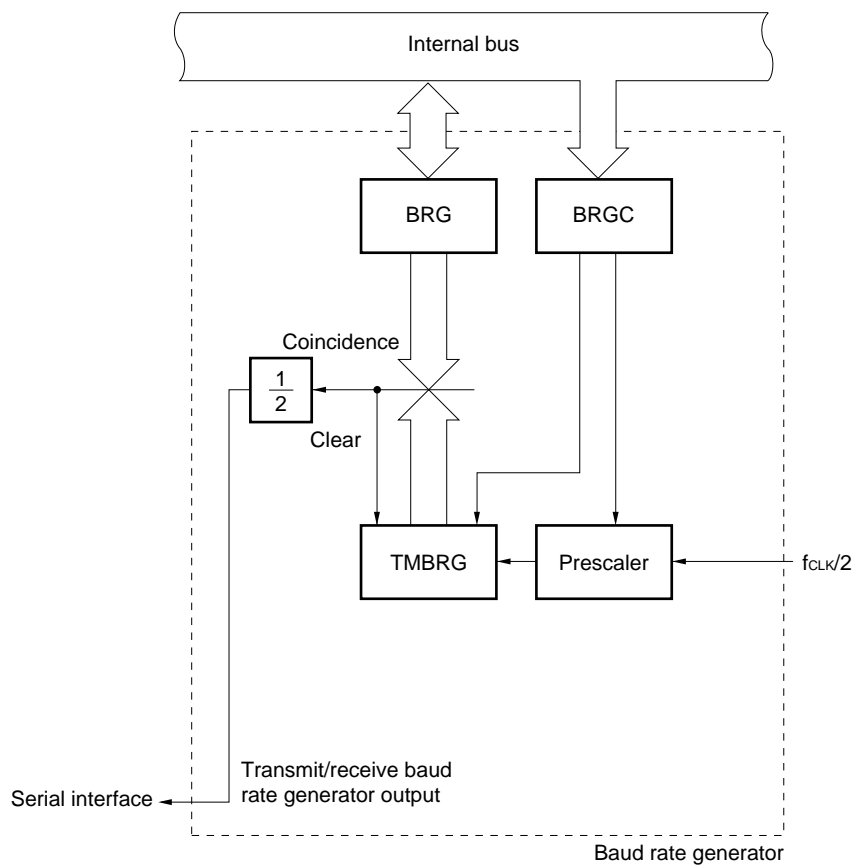
(4) 8-bit baud rate generator compare register (BRG)

This register always compares its value with the contents of the 8-bit timer (TMBRG). When the value written to the register coincides with the value of the timer, the register generates a coincidence signal, which clears the timer to 0.

A signal obtained by dividing this coincidence signal by two is output by the baud rate generator.

This register can be read or written by an 8-bit manipulation instruction.

Its contents are undefined when the $\overline{\text{RESET}}$ signal is input.

Figure 9-8. Block Diagram of Baud Rate Generator**Figure 9-9. Format of Baud Rate Generator Control Register**

Symbol	7	6	5	4	③	2	1	0	Address	When reset	R/W
BRGC	0	0	0	0	CEB0	0	PRMB1	PRMB0	FF84H	00H	R/W

PRMB1	PRMB0	Count clock to TMBRG
0	0	$f_{CLK}/2$ (n = 0)
0	1	$f_{CLK}/4$ (n = 1)
1	0	$f_{CLK}/8$ (n = 2)
1	1	$f_{CLK}/16$ (n = 3)

CEB0	Controls operation of baud rate generator
0	Clears and stops counting
1	Enables counting

Remarks 1. f_{CLK} : internal system clock

2. n in the PRMB1 and PRMB0 descriptions indicates the set values of bits 1 and 0 of the BRGC register, which are used to calculate the baud rate.

9.5.2 Specific baud rate setting

A specified serial clock can be generated by setting the baud rate generator control register (BRGC) and 8-bit compare register (BRG).

(1) Baud rate = serial clock/16

The asynchronous serial interface samples the level of the RxD pin with the serial clock specified by the asynchronous serial interface mode register (ASIM) divided by 16.

(2) Setting a desired baud rate

The baud rate can be calculated by the following expression. Set the value of the BRG register and the count clock of the 8-bit timer (TMBRG) so that the desired baud rate can be obtained, and start operation of the baud rate generator.

Calculating baud rate

$$\text{Baud rate (bps)} = \frac{f_{\text{CLK}}}{2^n} \times \frac{1}{(m+1)} \times \frac{1}{2} \times \frac{1}{2} \times \frac{1}{16}$$

where,

f_{CLK} : internal system clock (External oscillation frequency $f_{\text{xx}} \times 2$)

m : set value of BRG register

n : value corresponding to set value of BRGC register, prescaler value

Table 9-1 shows some typical baud rate settings.

(3) Baud rate error

The asynchronous serial interface samples the level of the RxD pin in synchronization with the start bit. Therefore, if the baud rate error between transmission and reception is greater than the value calculated by the following expression, data including the start and stop bits is shifted more than 0.5 bit, and communication cannot be carried out normally.

$$\text{Maximum permissible error} = \frac{0.5 \text{ bit}}{1 \text{ data frame length}} = \frac{0.5 \text{ bit}}{12 \text{ bits max.}} = 4.1 \%$$

Table 9-1. Typical Baud Rate Settings (asynchronous serial interface)

External Oscillation Frequency f _{xx} (MHz)	8.000			6.250			6.144			5.000			4.9165		
Internal System Clock f _{CLK} (MHz)	16.000			12.500			12.288			10.000			9.833		
Baud Rate (bps)	BRGC (n)	BRG (m)	Error (%)	BRGC (n)	BRG (m)	Error (%)	BRGC (n)	BRG (m)	Error (%)	BRGC (n)	BRG (m)	Error (%)	BRGC (n)	BRG (m)	Error (%)
110	—	—	—	3	221	0.02	3	217	0.08	3	177	0.25	3	174	0.26
150	3	207	0.16	3	162	0.15	3	159	0	3	129	0.16	3	127	0
300	2	207	0.16	2	162	0.15	2	159	0	2	129	0.16	2	127	0
600	1	207	0.16	1	162	0.15	1	159	0	1	129	0.16	1	127	0
1200	0	207	0.16	0	162	0.15	0	159	0	0	129	0.16	0	127	0
2400	0	103	0.16	0	80	0.47	0	79	0	0	64	0.16	0	63	0
4800	0	51	0.16	0	40	0.76	0	39	0	0	32	1.36	0	31	0
9600	0	25	0.16	0	19	1.73	0	19	0	0	15	1.73	0	15	0
19200	0	12	0.16	0	9	1.73	0	9	0	0	7	1.73	0	7	0
38400	0	6	7.0 ^{Note}	0	4	1.73	0	4	0	0	3	1.73	0	3	0

Note Must not be used because error is too much.

Remark BRGC: bits 0, 1 (PRMB0, PRMB1) of baud rate generator control register (BRGC)
BRG : set value of baud rate generator compare register (BRG)

9.6 Transmitting Data

Transmission is started when data is written to the transmit shift register (TXS). The next data is written to the TXS register by the transmission end interrupt (INTST).

(1) Transmitting data

The asynchronous serial interface of the μ PD78366A is always enabled to transmit, and transmission is started when transmit data is written to the transmit shift register (TXS). Start, parity, and stop bits are automatically appended to the transmit data.

When transmission is started, the data in the TXS register is shifted out. When the TXS register becomes empty as a result, a transmission end interrupt (INTST) occurs.

Transmission is aborted unless the data to be transmitted next is written to the TXS register.

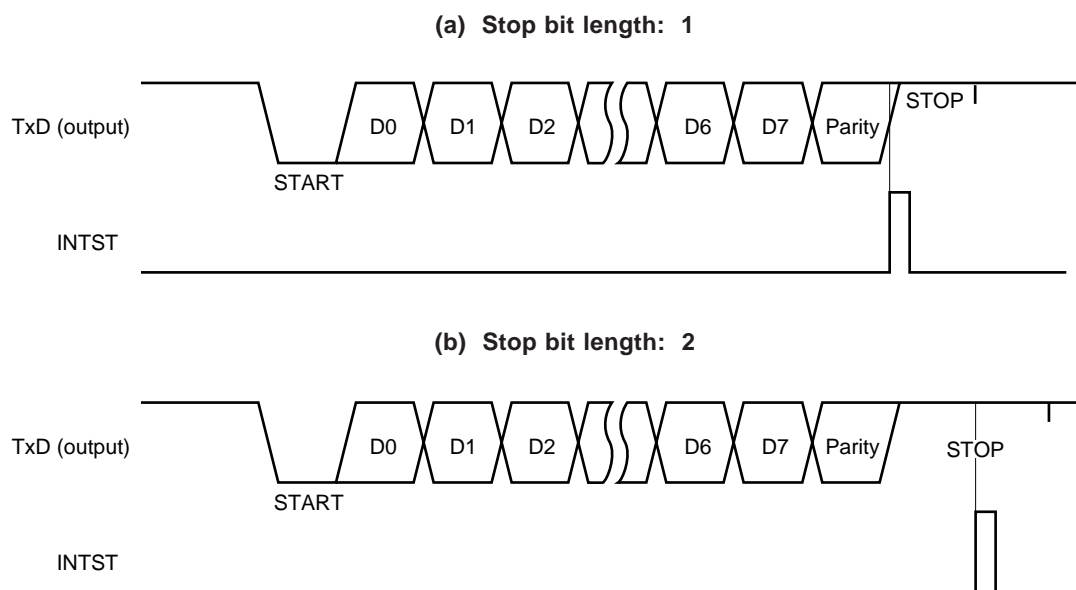
(2) Writing transmit data to TXS register

When transmission has been completed once, the communication rate drops, unless the next transmit data is immediately written to the TXS register.

To write transmit data to the TXS register, use of the block transfer mode (BLKTRS) of the macro service is recommended, because the macro service is started as soon as the transmission end interrupt (INTST) occurs, regardless of the priority, thus improving the communication rate.

Cautions 1. Usually, a transmission end interrupt (INTST) occurs when the transmit shift register (TXS) becomes empty. However, a transmission end interrupt (INTST) does not occur if the transmit shift register (TXS) becomes empty because of RESET input.

2. Data written to the TXS register while transmission is in progress and before INTST occurs is invalid.

Figure 9-10. Asynchronous Serial Interface Transmission End Interrupt Timing

Remark INTST ... vector table address: 0020H (TPF = 0), 8020H (TPF = 1)
 macro service control word address: FE20H

Caution Usually, a transmission end interrupt (INTST) occurs when the transmit shift register (TXS) becomes empty. However, a transmission end interrupt does not occur if the transmit shift register becomes empty because of $\overline{\text{RESET}}$ input.

9.7 Receiving Data

When reception is enabled, sampling of the RxD pin is started. When a start bit is detected, data reception begins. Each time one frame of data has been received, a reception end interrupt (INTSR) occurs. Usually, this interrupt transfers the receive data from the receive buffer (RXB) to memory.

(1) Receiving data

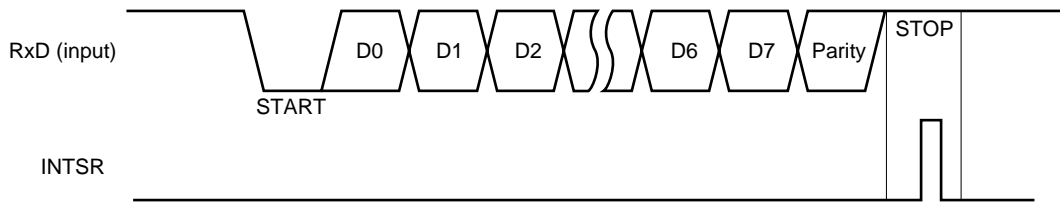
Reception is enabled by setting the RXE bit of the asynchronous serial interface mode register (ASIM) to 1. When reception is enabled, the RxD pin input is sampled with the serial clock specified by the ASIM register. When the RxD pin goes low, the 1/16 counter starts counting. When it counts eight times, a start timing signal for data sampling is output. If the RxD pin is found to be low as a result of sampling the pin again with this start timing signal, the low level is recognized as a start bit. The 1/16 counter is then initialized, counting started and the data sampled. When character data, parity bit, and 1 stop bit are detected after the start bit, reception of one frame of data is completed.

When one frame of data has been received, the data in the receive shift register is transferred to the receive buffer (RXB), and the reception end interrupt (INTSR) occurs.

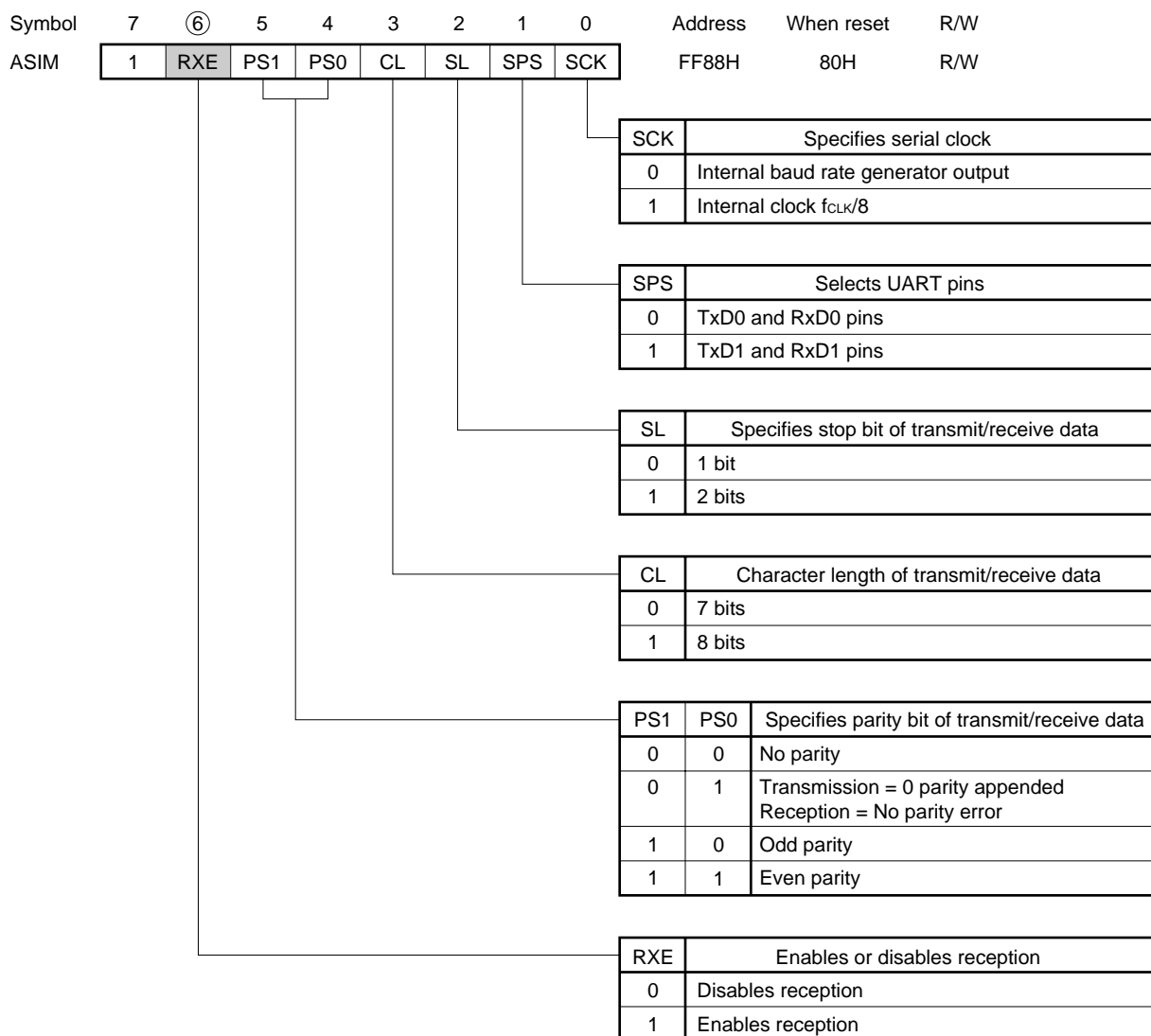
Even if an error occurs, the receive data that has caused the error is transferred to the receive buffer (RXB), and the reception end interrupt (INTSR) and reception error interrupt (INTSER) occur simultaneously.

If the RXE bit is reset to 0 during reception, reception is stopped immediately. At this time, the contents of the receive buffer (RXB) and the asynchronous serial interface status register (ASIS) remain unchanged, and the reception end interrupt (INTSR) and reception error interrupt (INTSER) do not occur.

Figure 9-11. Asynchronous Serial Interface Reception End Interrupt Timing



Remark INTSR ... vector table address: 001EH (TPF = 0), 801EH (TPF = 1)
macro service control word address: FE1EH

Figure 9-12. Setting of ASIM Register (reception enabled)

Caution Bit 7 of the ASIM register is fixed to “1” by hardware. Even if “0” is written, they remain “1”.

Remark f_{CLK} : internal system clock

(2) Transferring receive buffer (RXB) contents to memory

An overrun error occurs unless the contents of the receive buffer (RXB) are read before the next data is received. To transfer the receive data to memory, use of the block transfer mode (BLKTRS) of the macro service is recommended, because the macro service is started as soon as the reception end interrupt (INTSR) occurs, regardless of the priority, so that the contents of the receive buffer (RXB) can always be transferred before the next data is received.

Caution Be sure to read the receive buffer (RXB) even when a reception error occurs. Unless the RXB register is read, an overrun error occurs when the next data is received, and the reception error status persists.

9.8 Transmitting/Receiving Data Using Macro Service

When transmitting data using a macro service, a vectored interrupt request is issued twice. However, during reception, a vectored interrupt request is issued only once.

- **Transmitting/receiving data using a macro service**

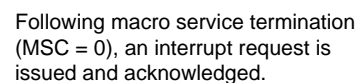
Data transmission is begun by writing data to the transmit shift register (TXS). If this is performed with a macro service, data is written to TXS and transmitted only the specified number of times. After the end of transmission, a transmission end interrupt (INTST) is issued, and the next data writing macro service is executed. When the last data is written to TXS, the macro service terminates (MSC = 0), and a vectored interrupt request is issued (refer to **Figure 9-13**, <1>).

Then, when data transmission is terminated (transmission of 1 frame is completed), INTST is newly issued and a vectored interrupt request is issued again (refer to **Figure 9-13**, <2>).

Therefore, when activating a macro service with INTST as described above, a vectored interrupt may be issued twice by the same vectored interrupt request (in this case, INTST).

On the other hand, during reception, a vectored interrupt request is never issued twice the way it is during transmission. In the case of reception, the reception end interrupt (INTSR) issued when reception is terminated causes execution of the macro service transferring the received data to memory, and therefore, a vectored interrupt request is issued only once upon termination of the macro service.

(a) Transmitting



Following macro service termination (MSC = 0), an interrupt request is issued and acknowledged.

9.9 If Reception Error Occurs

If a reception error occurs, the nature of the error can be identified by reading the asynchronous serial interface status register (ASIS).

- **There are three types of reception errors.**

Three types of errors can occur during reception: parity error, framing error, and overrun error. If an error is detected during reception, the error flag in the ASIS register is set. At the same time, the reception error interrupt (INTSER) occurs. Table 9-2 lists the causes of the errors.

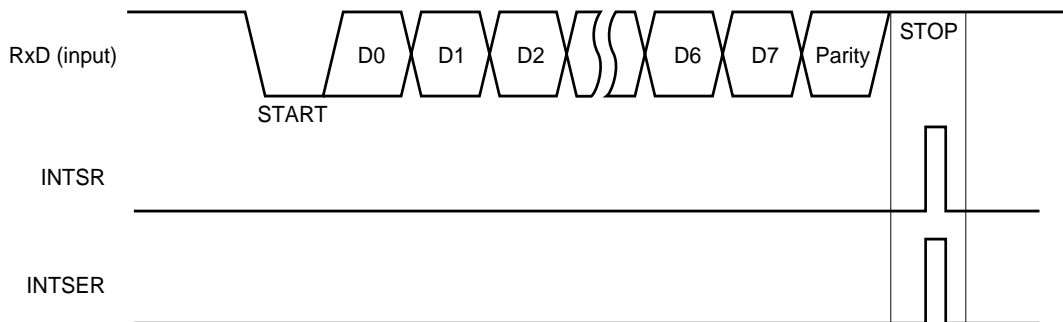
Which error has occurred during reception can be checked by reading the contents of the ASIS register during reception error interrupt service (INTSER) (refer to **Figure 9-15**).

The contents of the ASIS register are reset to 0 when the contents of the receive buffer (RXB) are read or when the next data is received (if an error occurs in the next data, the corresponding error flag is set).

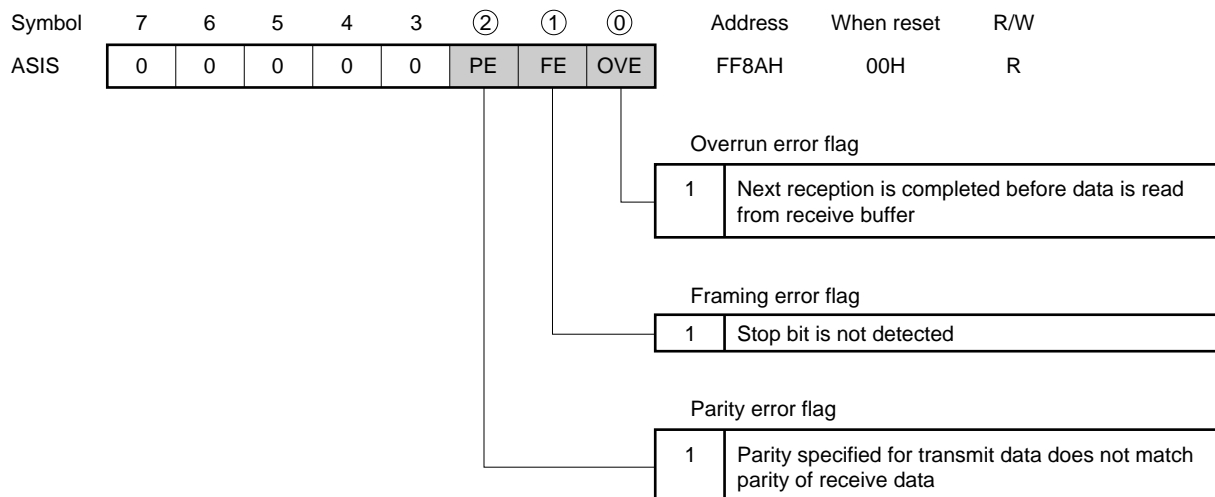
Table 9-2. Causes of Reception Errors

Reception Error	Cause
Parity error	Parity specified during transmission does not match parity of receive data
Framing error	Stop bit is not detected
Overrun error	Next data is received before data is read from receive buffer

Figure 9-14. Reception Error Timing



- Remarks**
1. INTSR vector table address: 001EH (TPF = 0), 801EH (TPF = 1)
macro service control word address: FE1EH
 2. INTSER ... vector table address: 001CH (TPF = 0), 801CH (TPF = 1)
macro service control word address: FE1CH

Figure 9-15. Format of Asynchronous Serial Interface Status Register

- Cautions**
1. The contents of the ASIS register are reset to 0 when the contents of the receive buffer (RXB) are read or when the next data is received. To identify an error, be sure to read the ASIS register before reading the receive buffer (RXB).
If the received data is transferred to memory using a macro service, the receive buffer (RXB) is read during the reception of serial data, and therefore the ASIS register is reset (0). Thus, it is not possible to know more than the fact that an error has occurred. Be sure to check if there is no problem concerning this point before using the register.
An error can be detected when the reception error interrupt request flag (SERIF) is set (1) or by acknowledgment of a reception error interrupt (INTSER).
 2. Be sure to read the receive buffer (RXB) even if a reception error has occurred. Otherwise, an overrun error occurs when the next data is received, and the reception error status persists.
 3. Bits 7-3 of the ASIS register is fixed to “0” by hardware. Even if “1” is written, they remain “0”.

[MEMO]

CHAPTER 10 CLOCKED SERIAL INTERFACE

The μ PD78366A is provided with two clocked serial interface operation mode: three-wire serial I/O mode and serial bus interface (SBI) mode. Therefore, the microcomputer can be flexibly used for various applications.

The clocked serial interface is independent of the asynchronous serial interface.

10.1 Clocked Serial Interface Configuration

The clocked serial interface is controlled by the clocked serial interface mode register (CSIM) and serial bus interface control register (SBIC). The transmit/receive data can be read from or written to the SIO register.

(1) Clocked serial interface mode register (CSIM)

The CSIM register is an 8-bit register that specifies the operation of the clocked serial interface.

This register can be read or written by an 8-bit manipulation instruction or a bit manipulation instruction.

It is initialized to 00H when the $\overline{\text{RESET}}$ signal is input.

(2) Serial bus interface control register (SBIC)

This is an 8-bit register consisting of bits that control the status of the serial bus, and bits that indicate the various statuses of the data input from the serial bus. It can be used only in the SBI mode and cannot be manipulated in the three-wire serial I/O mode.

To manipulate this register, an 8-bit manipulation or a bit manipulation instruction is used. Some bits of this register can be read/written, and some are read-only or write-only bits. If a write-only bit is read, "0" is read.

The contents of the SBIC register are initialized to 00H when the $\overline{\text{RESET}}$ signal is input.

The detection flags ACKD, CMDD, and RELD are cleared when transmission/reception is disabled (by clearing both the CTXE and CRXE bits of the CSIM register to 0).

(3) Shift register (SIO)

The shift register (SIO) is an 8-bit register that converts serial data to parallel data, or vice versa, and is used for both transmission and reception.

Data is shifted in (during reception) or out (during transmission) from the MSB or LSB.

Actual transmission is controlled by reading or writing data from or to the SIO register.

This register can be read or written by an 8-bit manipulation instruction.

Its contents are undefined when the $\overline{\text{RESET}}$ signal is input.

(4) SO latch

The SO latch holds the output level of the SO/SB0 pin. It can be directly controlled by software in the serial bus interface (SBI) mode.

(5) Serial clock selector

This selects the serial clock to be used.

10.2 Setting Pins for Serial Communication

Since the $\overline{\text{SO}}/\text{SB0}$, $\text{SI}/\text{SB1}$, and $\overline{\text{SCK}}$ pins are multiplexed with general-purpose port pins, they must be set in the control mode before communication is executed.

(1) Setting pins for serial communication

The clocked serial interface uses the $\overline{\text{SO}}/\text{SB0}$, $\text{SI}/\text{SB1}$, and $\overline{\text{SCK}}$ pins. These pins, however, are multiplexed with P32, P33, and P34 of a general-purpose port, respectively. Therefore, they must be set in the control mode by using the port 3 mode control register (PMC3) before starting communication.

Figure 10-2. Format of Port 3 Mode Control Register

Symbol	7	6	5	4	3	2	1	0	Address	When reset	R/W
PMC3	×	PMC36	PMC35	PMC34	PMC33	PMC32	PMC31	PMC30	FF43H	×000 0000B	R/W
									PMC30	Specifies control mode of P30 pin	
									0	I/O port mode	
									1	TxD0 output mode	
									PMC31	Specifies control mode of P31 pin	
									0	I/O port mode	
									1	RxD0 input mode	
									PMC32	Specifies control mode of P32 pin	
									0	I/O port mode	
									1	SB0 I/O mode/SO output mode	
									PMC33	Specifies control mode of P33 pin	
									0	I/O port mode	
									1	SB1 I/O mode/SI input mode	
									PMC34	Specifies control mode of P34 pin	
									0	I/O port mode	
									1	$\overline{\text{SCK}}$ I/O mode	
									PMC35	Specifies control mode of P35 pin	
									0	I/O port mode	
									1	TxD1 output mode	
									PMC36	Specifies control mode of P36 pin	
									0	I/O port mode	
									1	RxD1 input mode	

Remark ×: don't care

(2) Reading pin level

When port 3 (P3) is set in the control mode by the port 3 mode control register (PMC3), the following statuses can be read when an instruction that reads port 3 (P3) is executed:

(a) When the corresponding bit of the port 3 mode register (PM3) is set to 1

- Each pin level can be read.

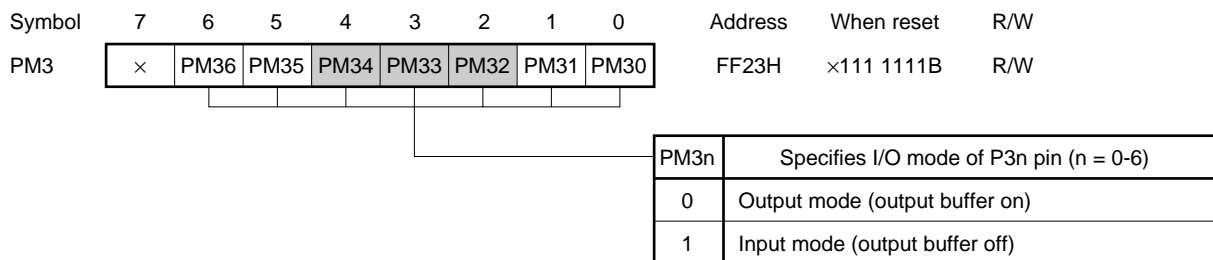
(b) When the corresponding bit of the port 3 mode register (PM3) is reset to 0

- The level of an internal signal is read.

By reading the pin level, contention of the serial bus can be checked.

The level of each pin does not change even if data is written to port 3 (P3) (the data is written to the output buffer of port 3).

Figure 10-3. Format of Port 3 Mode Register



Remark ×: don't care

10.3 Baud Rate Setting

The output of the baud rate generator, internal clocks $f_{CLK}/8$, and $f_{CLK}/32$, or external clock can be selected as the serial clock. When the baud rate generator output is selected, communication can be carried out at a specified baud rate, regardless of the operating frequency.

(1) Baud rate = serial clock

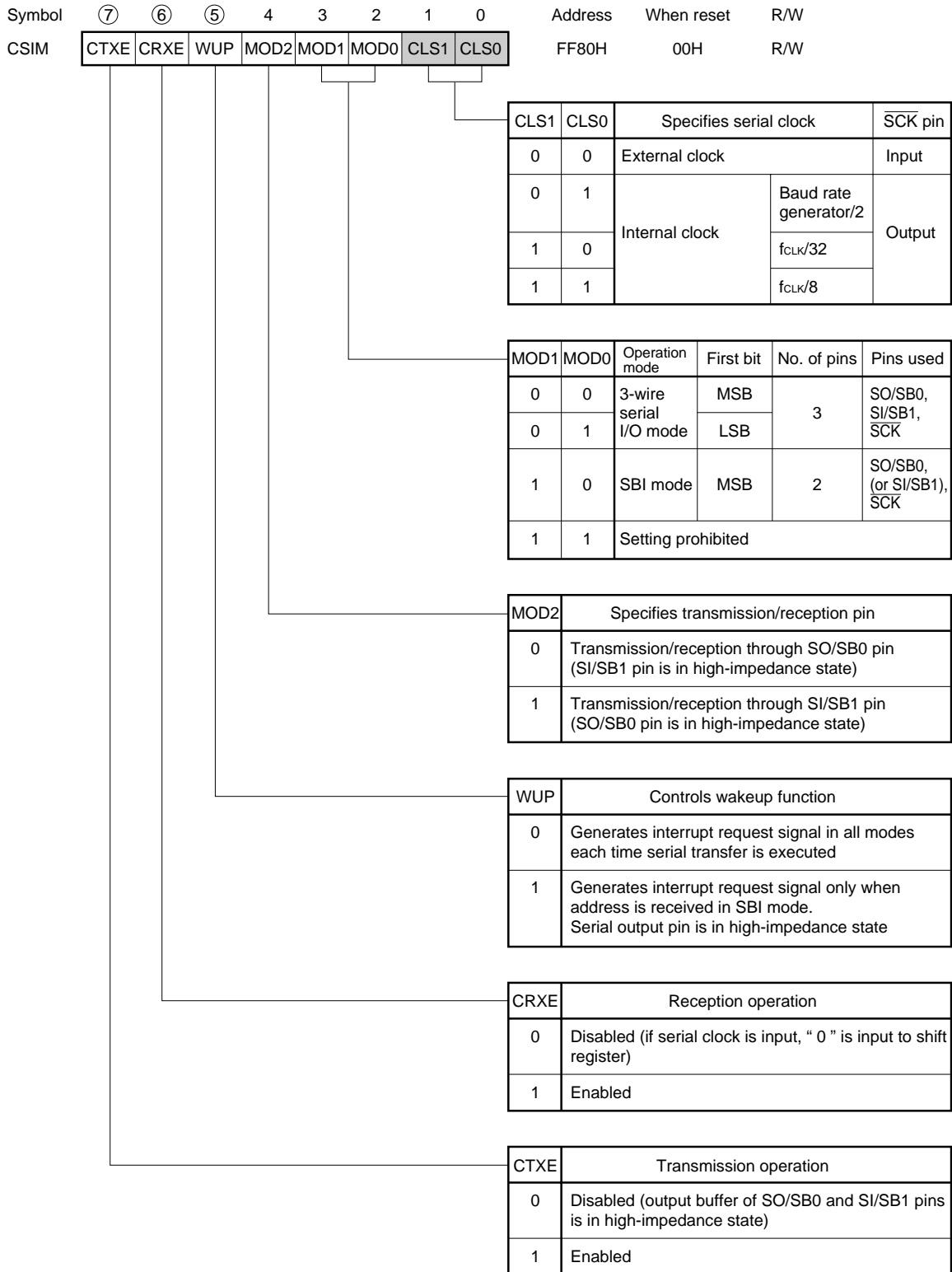
The clocked serial interface samples the receive data at the rising edge of the serial clock. Therefore, the serial clock serves as the baud rate as is.

(2) Selecting serial clock

The serial clock is selected by the CLS1 and CLS0 bits of the clocked serial interface mode register (CSIM). When the external clock is selected, the \overline{SCK} pin is used as an input pin, through which the serial clock is input from the device with which the μ PD78366A communicates.

If $f_{CLK}/8$ is selected when the internal system clock is 16 MHz, the communication rate is 2 Mbps. When $f_{CLK}/32$ is selected, the communication rate is 500 Kbps.

Remark The baud rate generator is shared with the asynchronous serial interface (refer to **9.5 Baud Rate Setting**).

Figure 10-4. Setting of CSIM Register (serial clock)

Remark f_{CLK} : internal system clock

Caution Selection of the serial clock is made asynchronously with the serial clock. Therefore, if the serial clock is changed during communication, a serial clock with an undefined width may be output. Do not change the serial clock during communication.

10.3.1 Baud rate generator configuration

The baud rate generator is controlled by the baud rate generator control register (BRGC) and 8-bit compare register (BRG).

The baud rate generator is configured as shown in Figure 10-5.

(1) Baud rate generator control register (BRGC)

This 8-bit register is used to select the count clock for the 8-bit timer (TMBRG) and control operation of the baud rate generator.

It can be read or written by an 8-bit manipulation instruction or a bit manipulation instruction.

The contents of this register are initialized to 00H when the $\overline{\text{RESET}}$ signal is input.

The format of the BRGC register is shown in Figure 10-6.

(2) Prescaler

The prescaler divides the internal clock ($f_{\text{CLK}}/2$) according to the setting of the BRGC register and generates count clocks.

(3) 8-bit timer (TMBRG)

This 8-bit timer counts the count clocks generated by the prescaler.

When a coincidence signal is generated by the 8-bit compare register (BRG), the timer is cleared to 0 by the next count clock.

This timer is started or stopped by the BRGC register.

TMBRG cannot be handled directly from the CPU.

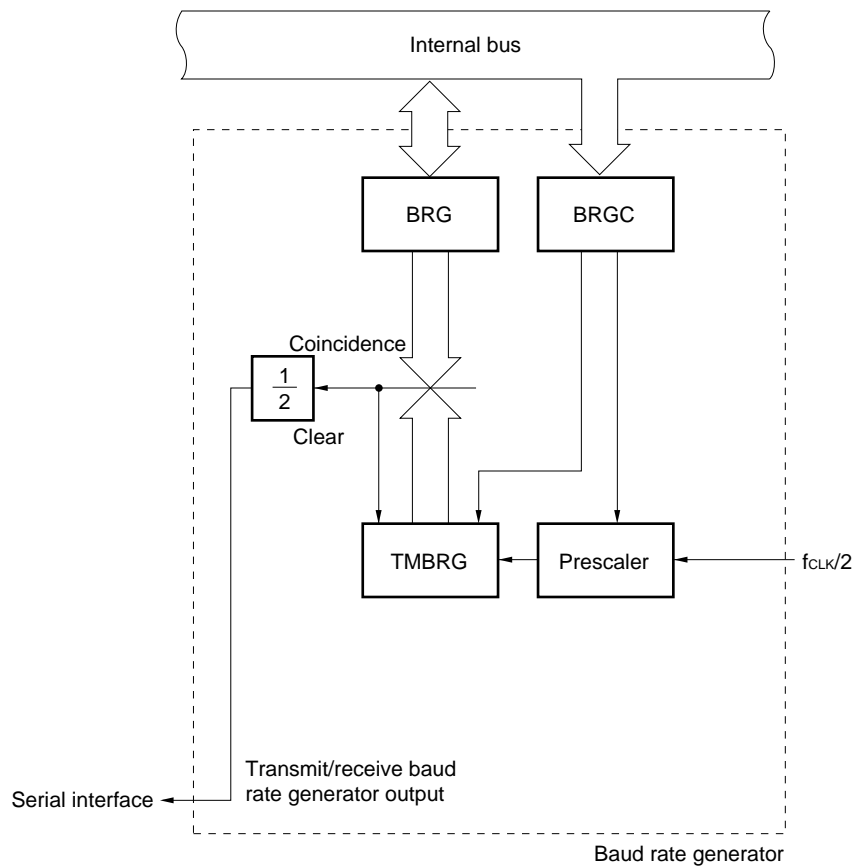
(4) 8-bit baud rate generator compare register (BRG)

This register always compares its value with the contents of the 8-bit timer (TMBRG). When the value written to the register coincides with the value of the timer, the register generates a coincidence signal, which clears the timer to 0.

A signal obtained by dividing this coincidence signal by two is output by the baud rate generator.

This register can be read or written by an 8-bit manipulation instruction.

Its contents are undefined when the $\overline{\text{RESET}}$ signal is input.

Figure 10-5. Block Diagram of Baud Rate Generator

10.3.2 Specific baud rate setting

A specified serial clock can be generated by setting the baud rate generator control register (BRGC) and 8-bit compare register (BRG).

(1) Baud rate = serial clock

The clocked serial interface samples the receive data at the rising edge of the serial clock. Therefore, the serial clock serves as the baud rate as is.

(2) Setting a desired baud rate

The baud rate can be calculated by the following expression. Set the value of the BRG register and the count clock of the 8-bit timer (TMBRG) so that the desired baud rate can be obtained, then start baud rate generator operation.

Calculating baud rate

$$\text{Baud rate (bps)} = \frac{f_{\text{CLK}}}{2^n} \times \frac{1}{(m + 1)} \times \frac{1}{2} \times \frac{1}{2}$$

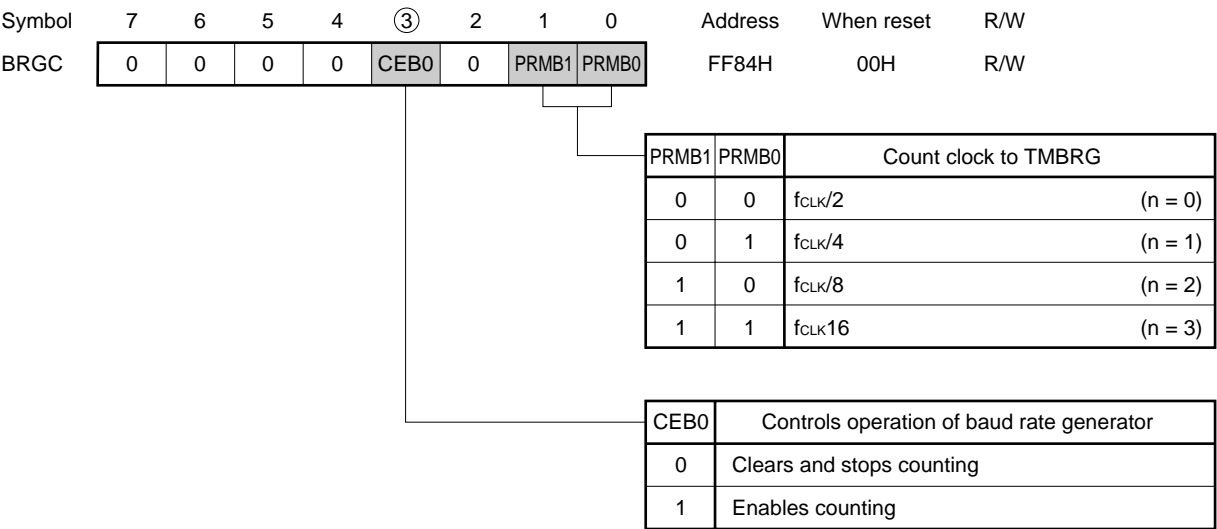
where,

f_{CLK} : internal system clock (External oscillation frequency $f_{\text{xx}} \times 2$)

m : set value of BRG register

n : value corresponding to set value of BRGC register, prescaler value

Figure 10-6. Format of Baud Rate Generator Control Register



- Remarks**
- 1. f_{CLK} : internal system clock
 - 2. n in the PRMB1 and PRMB0 descriptions indicates the set values of bits 1 and 0 of the BRGC register, which are used to calculate the baud rate.

10.4 Two Operation Modes of Clocked Serial Interface

The three-wire serial I/O mode is effective for communicating with a device containing a conventional clocked serial interface.

The SBI mode is NEC's original communication mode which enables the μ PD78366A to communicate with two or more devices by using two signal lines.

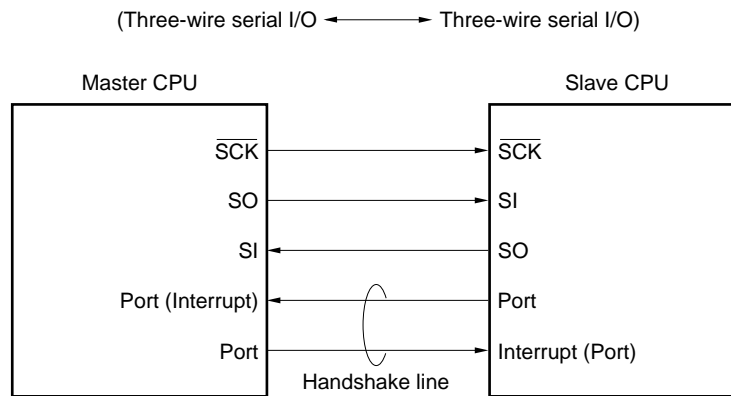
The clocked serial interface of the μ PD78366A can operate in the following two modes:

(1) Three-wire serial I/O mode

In this mode, three signal lines are used to transfer 8-bit data: serial clock ($\overline{\text{SCK}}$), serial input (SI), and serial output (SO) lines. This mode is effective for communicating with devices containing a conventional clocked serial interface, such as peripheral I/Os and display controllers.

To connect two or more devices, handshake lines are necessary.

Figure 10-7. Example of System Configuration in Three-Wire Serial I/O Mode



(2) Serial bus interface mode (SBI mode)

In this mode, two signal lines, serial clock ($\overline{\text{SCK}}$) and serial data bus (SB0 or SB1), are used to communicate with two or more devices.

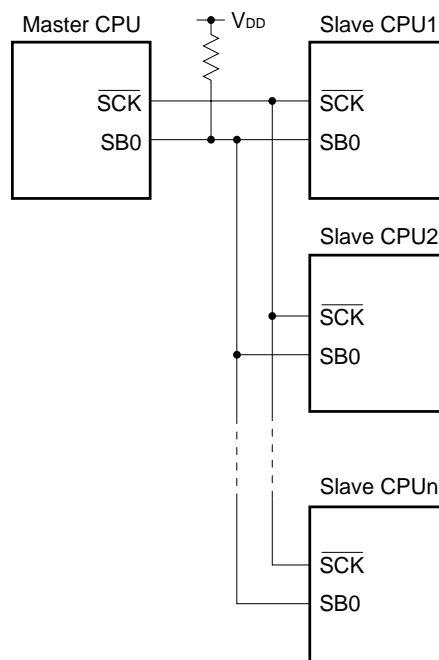
This mode conforms to NEC's serial bus format.

In the SBI mode, an "address" which is used to select the target device to communicate, a "command" which gives a direction to the target device, and actual "data" can be output to the serial data bus.

Therefore, handshake lines, which are necessary for connecting two or more devices with the conventional clocked serial interface, are not necessary. Consequently, the I/O ports can be effectively used and software overheads reduced.

In the SBI mode, the serial data bus pin (SB0 or SB1) serves as an open-drain output pin; therefore, the serial data bus line is wired-ORed. For this reason, a pull-up resistor must be connected to the serial data bus line.

Figure 10-8. Example of System Configuration in Serial Bus Interface (SBI) Mode



10.5 Three-Wire Serial I/O Mode Setting

The three-wire serial I/O mode is set by the clocked serial interface mode register (CSIM). Since the MSB or LSB can be selected as the first bit for communication, communication can be established with various devices.

(1) Setting three-wire serial I/O mode

The three-wire serial I/O mode is set by using the MOD1 and MOD0 bits of the clocked serial interface mode register (CSIM) (refer to **Figure 10-10**). Because the MSB or LSB can be selected as the first bit, the μ PD78366A can communicate with various devices in this mode.

(2) Operation timing in three-wire serial I/O mode

In the three-wire serial I/O mode, data is transmitted or received in 8-bit units. Each bit of data is transmitted or received in synchronization with the serial clock, with the MSB or LSB first (specified by the CSIM register). The transmit data is output in synchronization with the falling edge of $\overline{\text{SCK}}$. The receive data is sampled at the rising edge of $\overline{\text{SCK}}$. In addition, interrupt request INTCSI occurs at the eighth rising edge of $\overline{\text{SCK}}$. When the internal clock is used as $\overline{\text{SCK}}$, output of $\overline{\text{SCK}}$ is stopped at the eighth rising edge and $\overline{\text{SCK}}$ remains high until transmission or reception of the next data is started.

Figure 10-9. Timing in Three-Wire Serial I/O Mode

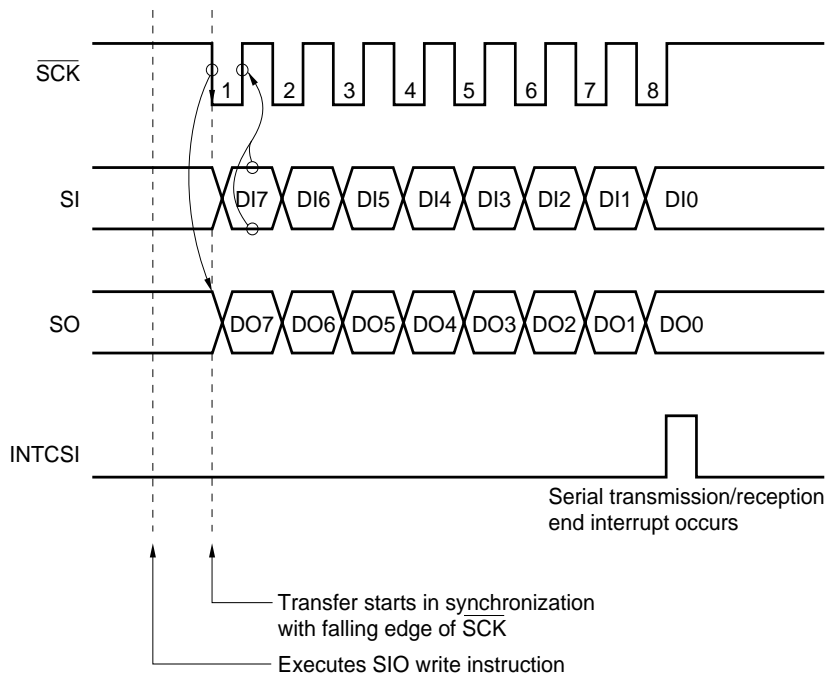
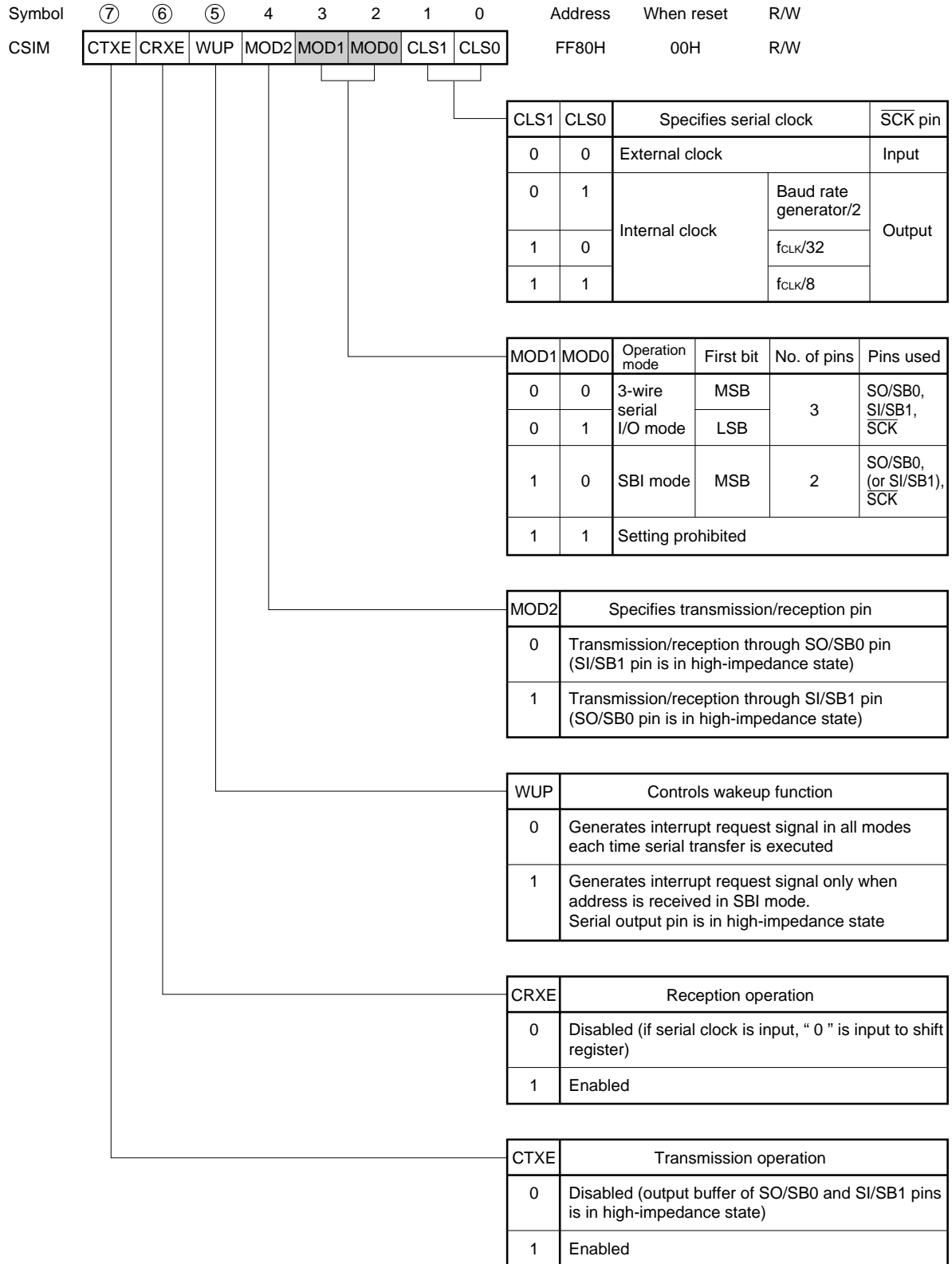


Figure 10-10. Setting of CSIM Register (three-wire serial I/O mode)

Remarks 1. f_{CLK}: internal system clock

2. The setting of the MOD2 bit is valid only in the SBI mode. This setting is invalid in the 3-wire serial I/O mode.

10.5.1 Transmission in three-wire serial I/O mode

When data is written to the SIO register after transmission is enabled by the clocked serial interface mode register (CSIM), transmission is started.

(1) Starting transmission

To start transmission, set the CTXE bit of the clocked serial interface mode register (CSIM) (reset the CRXE bit to "0"), and write the transmit data to the shift register (SIO). The block transfer mode (BLKTRS) of the macro service is useful for writing the transmit data to the SIO register.

When the CTXE bit is reset to 0, the SO pin enters the output high-impedance state.

(2) Transmitting data in synchronization with serial clock**(a) When internal clock is selected as serial clock**

When transmission is started, the serial clock is output from the $\overline{\text{SCK}}$ pin, and data is sequentially output from SIO to the SO pin in synchronization with the falling edge of the serial clock.

(b) When external clock is selected as serial clock

When transmission is started, data is sequentially output from SIO to the SO pin in synchronization with the falling edge of the serial clock input to the $\overline{\text{SCK}}$ immediately after transmission has been started. When transmission is not started, shift operation is not performed even if the serial clock is input to the $\overline{\text{SCK}}$ pin, and the output level of the SO pin remains unchanged.

Figure 10-11. Timing in Three-Wire Serial I/O Mode (transmission)

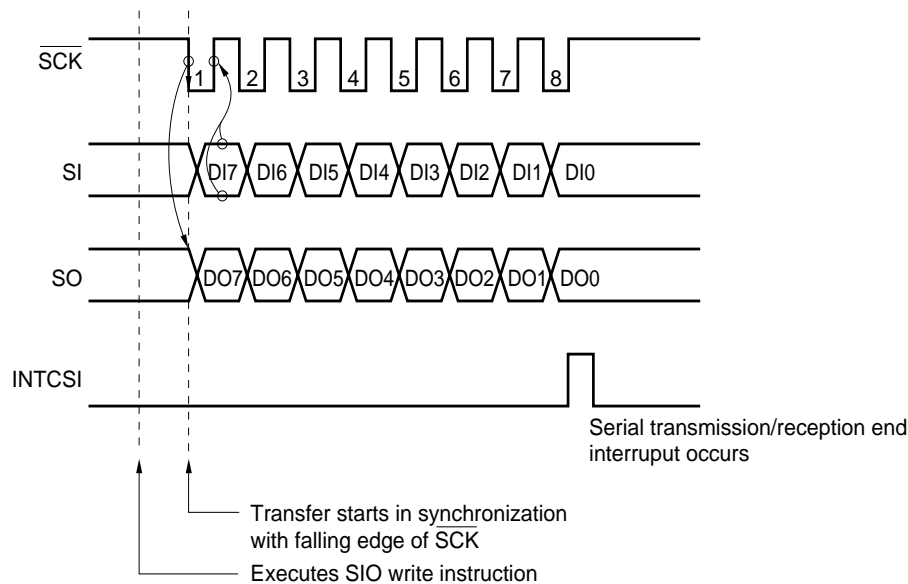
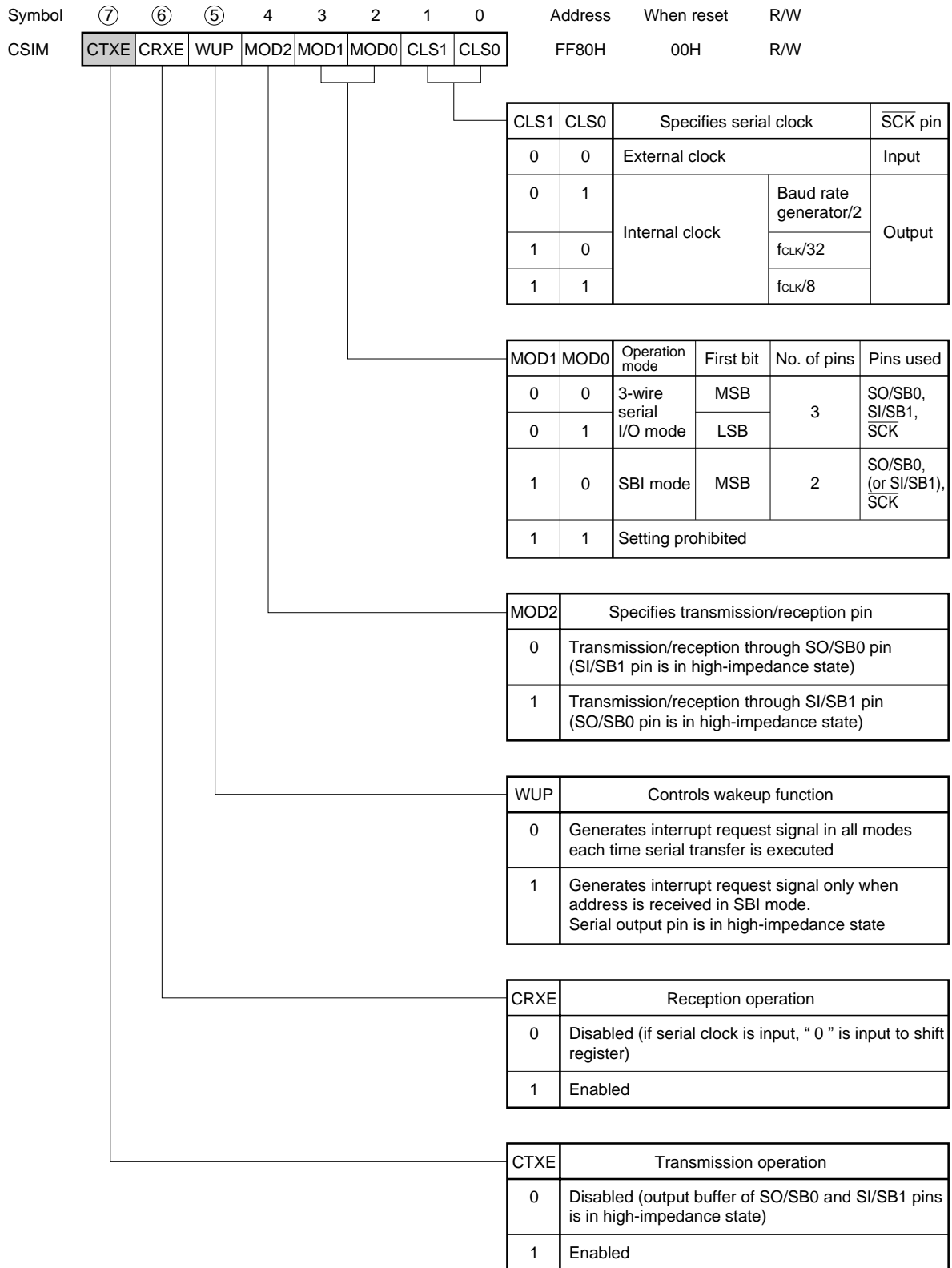


Figure 10-12. Setting of CSIM Register (transmission enabled)**Remark** f_{CLK}: internal system clock

10.5.2 Reception in three-wire serial I/O mode

When reception is enabled by the clocked serial interface mode register (CSIM) (when CTXE = 0) or when the contents of the SIO register are read with reception enabled, reception is started.

(1) Starting reception

Reception can be started in the following two ways:

- <1> By changing the CRXE bit from 0 (reception disable state) to 1 (reception enable state) when the CTXE bit of the CSIM register is 0; or
- <2> By reading receive data from the shift register (SIO) when the CRXE bit of the CSIM register is 1 (reception enable state).

The block transfer mode of the macro service (BLKTRS) is useful for reading receive data from the shift register (SIO). Reception is not started even when "1" is written to the CRXE bit of the CSIM register which has already been set to 1. Reception is not started when CTXE bit is "1", even if the CRXE bit setting is set to 1 from 0.

(2) Receiving data in synchronization with serial clock

(a) When internal clock is selected as serial clock

When reception is started, the serial clock is output from the $\overline{\text{SCK}}$ pin, and the data of the SI pin is sequentially loaded to SIO in synchronization with the rising edge of the serial clock.

(b) When external clock is selected as serial clock

When reception is started, the data of the SI pin is sequentially loaded to SIO in synchronization with the rising edge of the serial clock input to the $\overline{\text{SCK}}$ pin immediately after reception has been started. If reception is not started, shift operation is not performed even if the serial clock is input to the $\overline{\text{SCK}}$ pin.

Figure 10-13. Timing in Three-Wire Serial I/O Mode (reception)

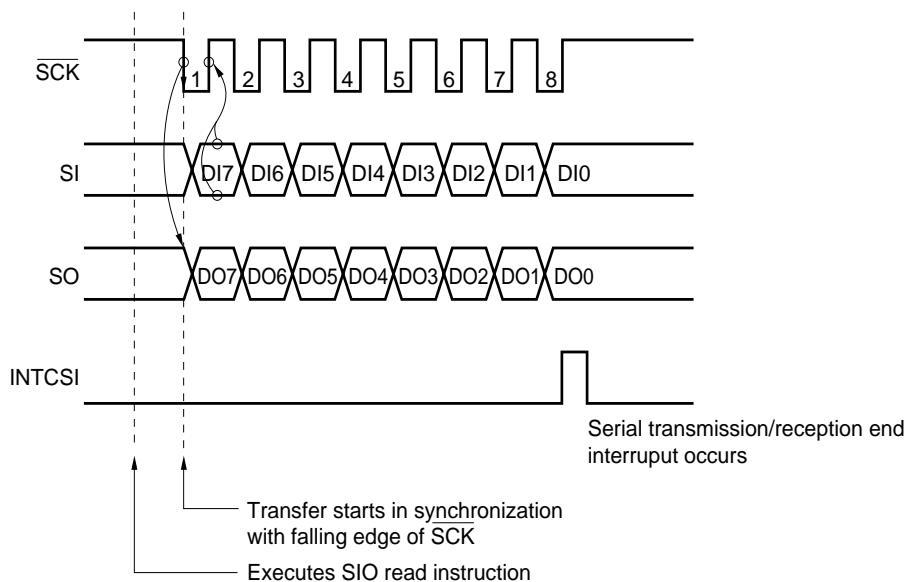
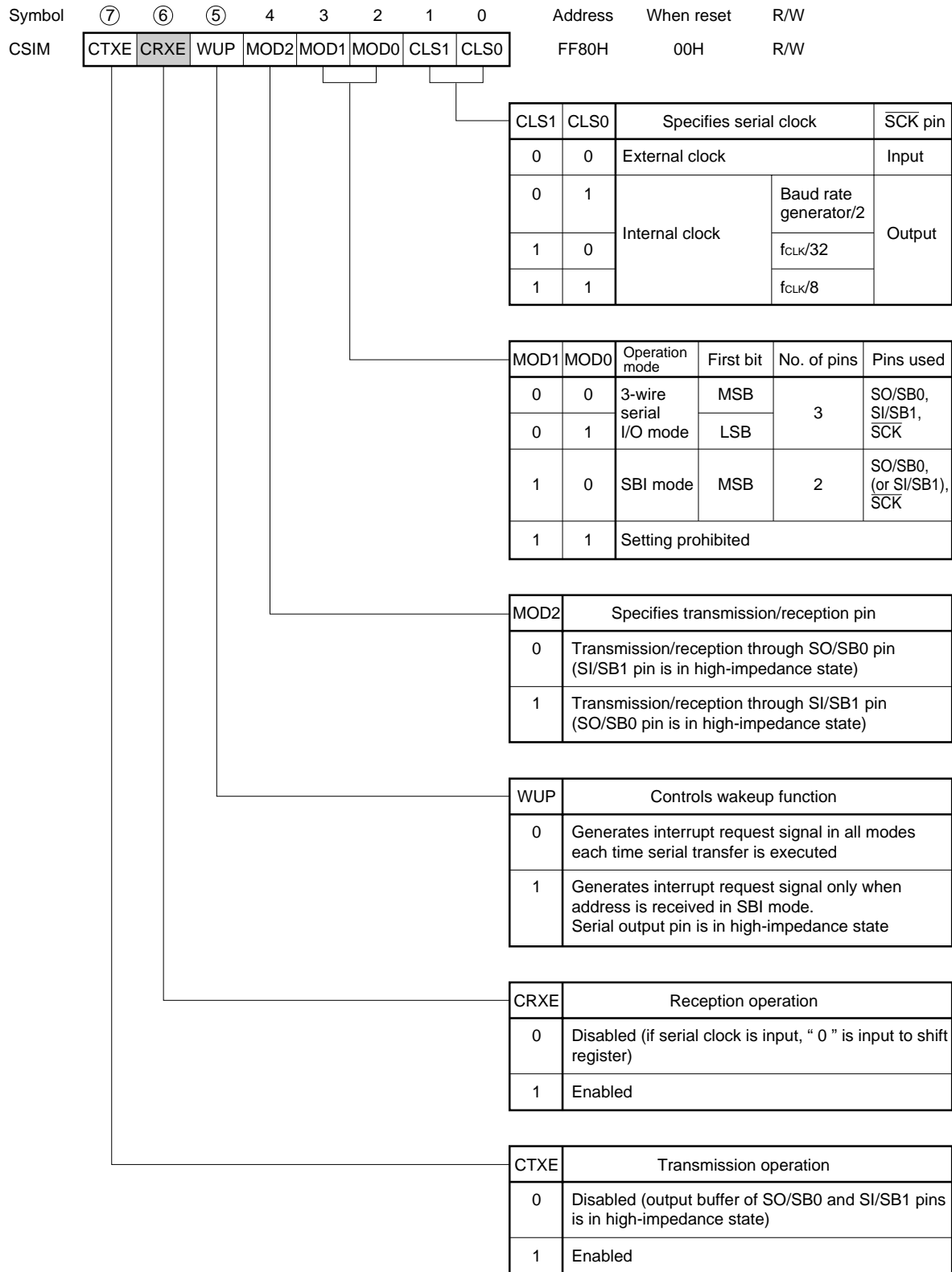


Figure 10-14. Setting of CSIM Register (reception enabled)**Remark** fCLK: internal system clock

10.5.3 Transmission/reception in three-wire serial I/O mode

When both transmission and reception are enabled by the clocked serial interface mode register (CSIM), transmission and reception can be performed simultaneously.

(1) Starting transmission/reception

Transmission and reception can be performed simultaneously when both the CTXE and CRXE bits of the clocked serial interface mode register (CSIM) are set to 1 (transmission/reception operation).

Transmission/reception can be started by writing transmit data to the shift register (SIO) when both the CTXE and CRXE bits of the CSIM register are 1 (transmission/reception enabled)

Transmission/reception operation is not started even if "1" is written to the CRXE bit of the CSIM register which has already been set to 1.

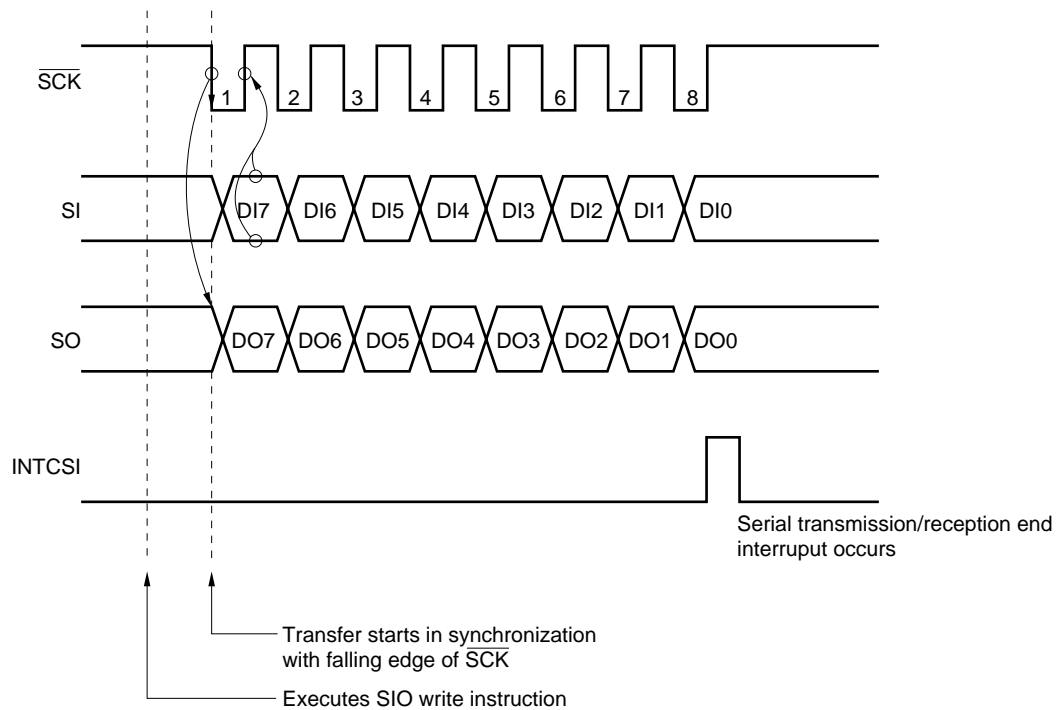
(2) Transmitting/receiving data in synchronization with serial clock

(a) When internal clock is selected as serial clock

When transmission/reception is started, the serial clock is output from the $\overline{\text{SCK}}$ pin, and data is sequentially output from SIO to the SO pin in synchronization with the falling edge of the serial clock. In addition, the data of the SI pin is sequentially loaded to SIO in synchronization with the rising edge of the serial clock.

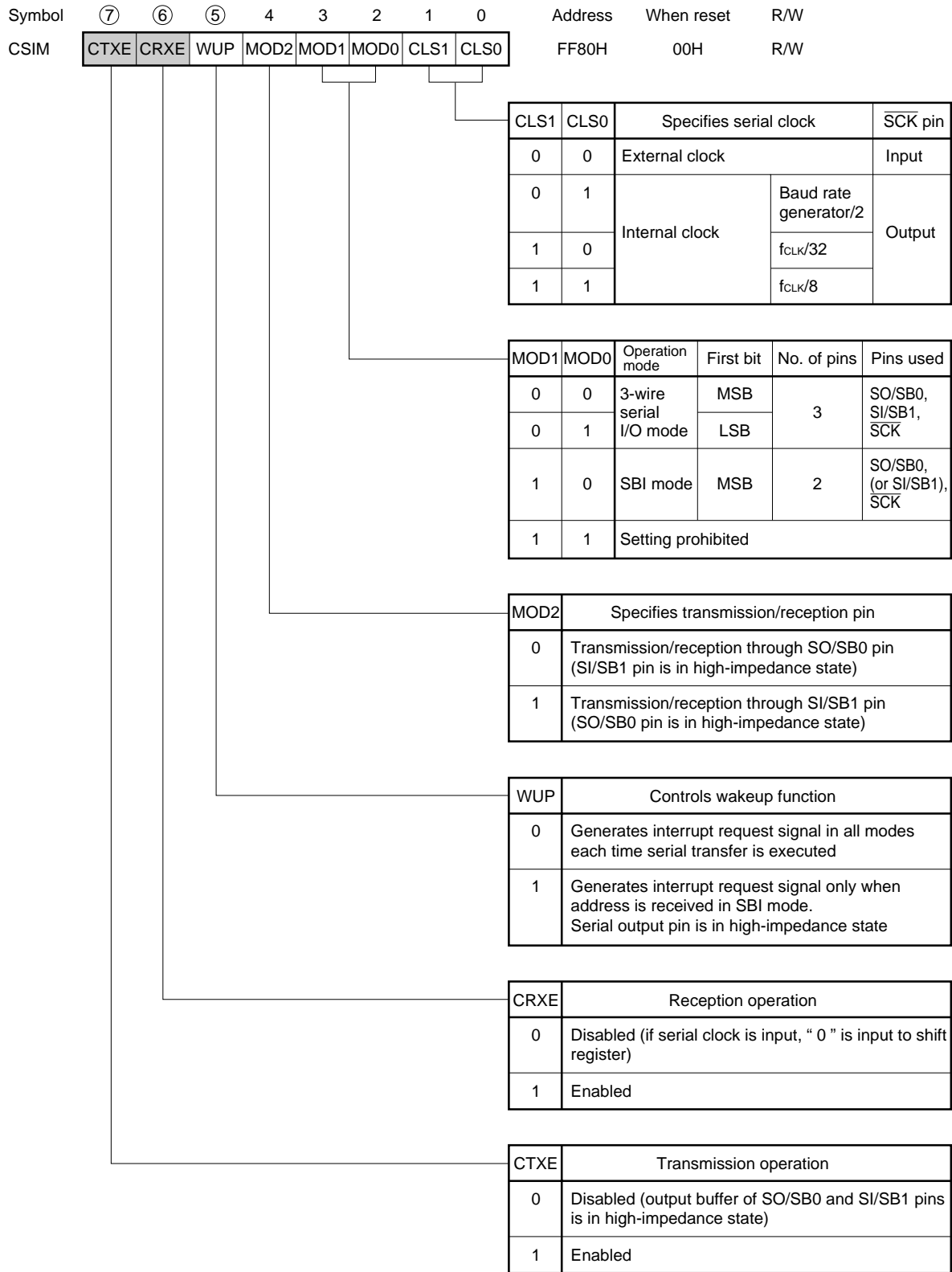
(b) When external clock is selected as serial clock

When transmission/reception is started, data is sequentially output from SIO to the SO pin in synchronization with the falling edge of the serial clock input to the $\overline{\text{SCK}}$ immediately after transmission/reception has been started. The data of the SI pin is sequentially loaded to SIO in synchronization with the rising edge of the serial clock. When transmission/reception is not started, shift operation is not performed even if the serial clock is input to the $\overline{\text{SCK}}$ pin, and the output level of the SO pin remains unchanged.

Figure 10-15. Timing in Three-Wire Serial I/O Mode (transmission/reception)

Remark INTCSI ... vector table address: 0022H (TPF = 0), 8022H (TPF = 1)
 macro service control word address: FE22H

Figure 10-16. Setting of CSIM Register (transmission/reception enabled)



Remark f_{CLK} : internal system clock

10.5.4 Corrective action in case shift operation is not synchronized

If the shift operation is not synchronized with the serial clock, synchronization is restored by disabling both transmission and reception.

- **Corrective action if shift operation is not synchronized with serial clock**

When the external clock is selected as the serial clock, the number of serial clocks may not correctly match the shift operation due to noise. In this case, disable both transmission and reception (by resetting the CTXE and CRXE bits to 0). This initializes the serial clock counter, so that when transmission or reception is enabled the next time, the serial clock input first is treated as the first clock, which is used to restore synchronization between the shift operation and the serial clock.

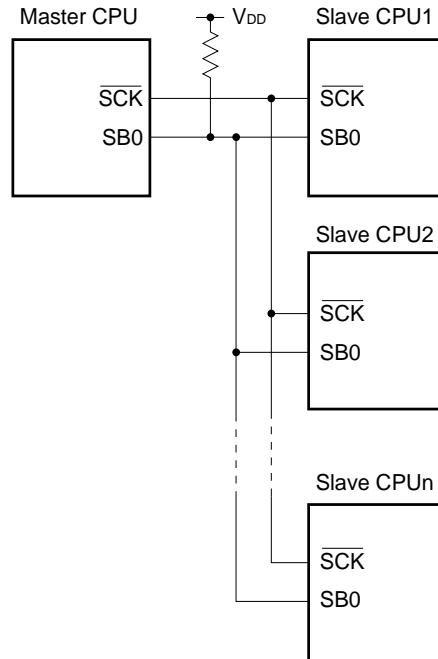
10.6 SBI Mode Setting

The SBI mode is effective for reducing the number of ports used and lowering software overheads because a serial bus can be configured with two signal lines, $\overline{\text{SCK}}$ and SB0 (SB1).

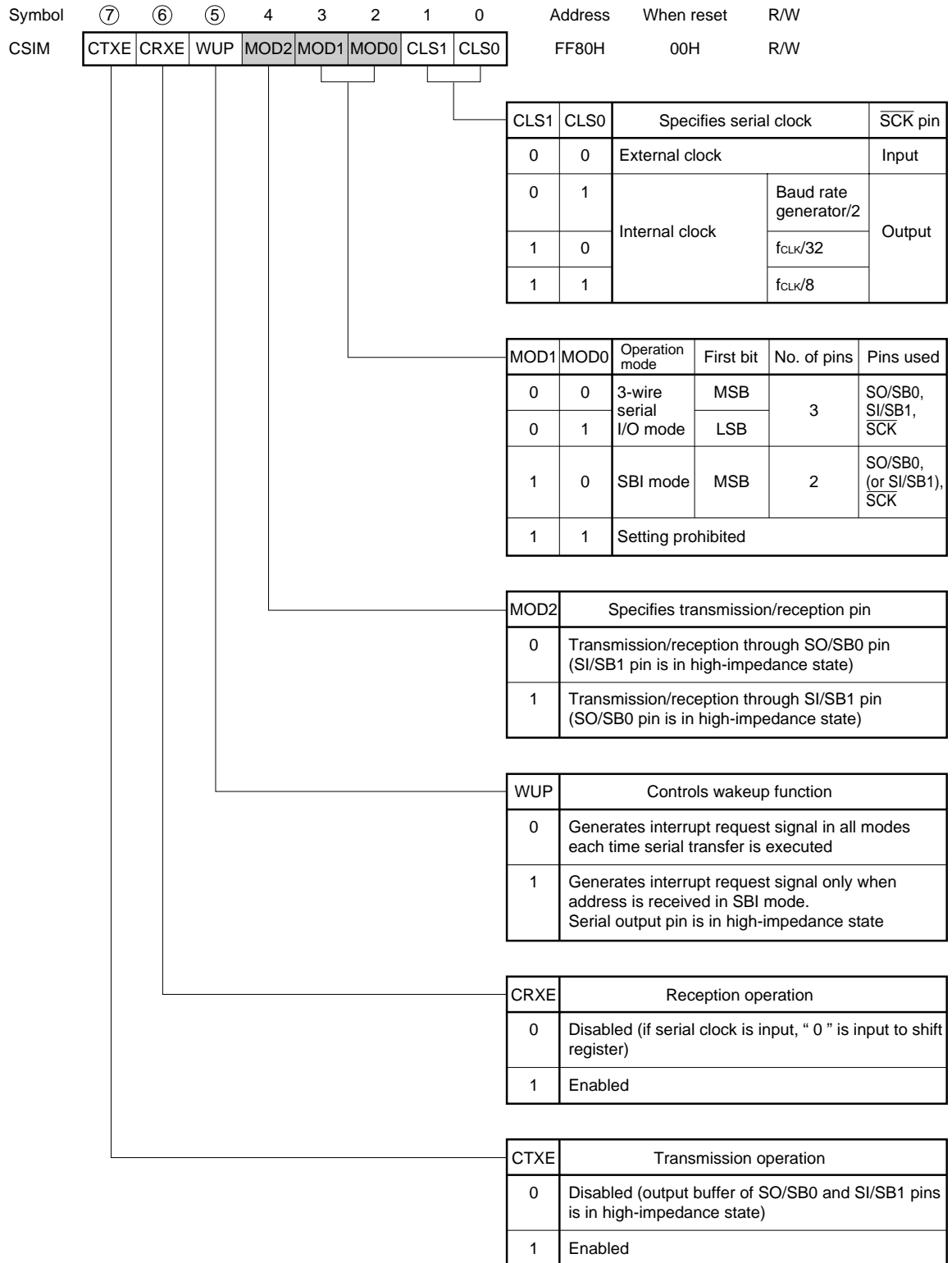
- **Setting SBI mode**

The SBI mode is set by the MOD2, MOD1, and MOD0 bits of the clocked serial interface mode register (CSIM). Since SB0 or SB1 can be selected as a serial data I/O pin, two serial buses can be configured.

Figure 10-17. Example of System Configuration of Serial Bus Interface (SBI)



- Cautions**
1. In the SBI mode, the serial data bus pin (SB0 or SB1) serves as an open-drain output pin. Therefore, the serial data bus line is wired-ORed. For this reason, a pull-up resistor must be connected to the serial data bus line.
 2. To exchange a master with a slave, switching between the input and output modes of $\overline{\text{SCK}}$ is asynchronously performed by the master and slave. Therefore, a pull-up resistor must also be connected to $\overline{\text{SCK}}$.

Figure 10-18. Setting of CSIM Register (SBI mode)

Remarks 1. f_{CLK} : internal system clock

2. The setting of the MOD2 bit is valid only in the SBI mode. This setting is invalid in the 3-wire serial I/O mode.

10.6.1 SBI data format

SBI defines various control signals by using a combination of two signal lines: serial clock $\overline{\text{SCK}}$ and serial data bus SB0 (or SB1).

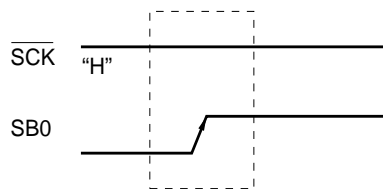
(1) Control signals used with SBI

SBI defines the following eight signals by using a combination of two signal lines, $\overline{\text{SCK}}$ and SB0 (or SB1):

<1> Bus release signal (REL)

This signal is output by the master to inform a slave that the master is to transmit an address to the slave. The slave is provided with hardware that detects the bus release signal.

Figure 10-19. Bus Release Signal

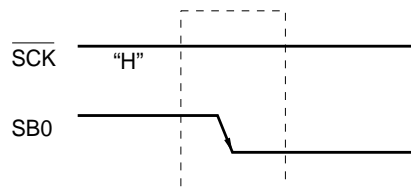


★ **Caution** The transition of the SB0 (SB1) line (transition from low to high) while the $\overline{\text{SCK}}$ line is high is recognized as a bus release signal. If the timing of changes on the bus shifts due to the influence of board capacitance, etc., a bus release signal may be detected even while data is being transmitted. Exercise care when routing the wiring.

<2> Command signal (CMD)

This signal is output by the master to inform a slave that the master is to transmit an address or command. The slave is provided with hardware that detects the command.

Figure 10-20. Command Signal



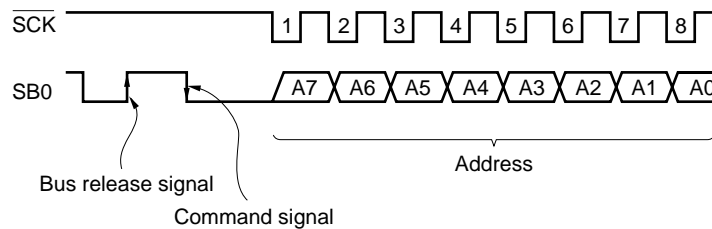
★

Caution The transition of the SB0 (SB1) line (transition from high to low) while the $\overline{\text{SCK}}$ line is high is recognized as a command signal. If the timing of changes on the bus shifts due to the influence of board capacitance, etc., a command signal may be detected even while data is being transmitted. Exercise care when routing the wiring.

<3> Address

This 8-bit data follows the bus release and command signals. It is output by the master to select a slave. The slave compares the address it has received with its own address through software. If the two addresses coincide, the slave performs processing necessary for establishing communication.

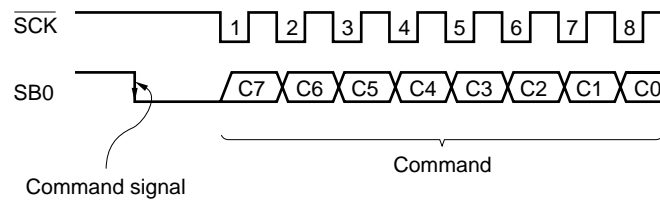
Figure 10-21. Address



<4> Command

This 8-bit data is output by the master after the command signal. In this case, the bus release signal is not output. How a command is used can be determined arbitrarily.

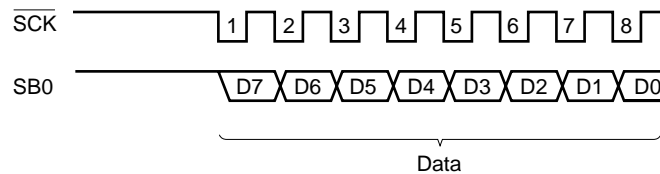
Figure 10-22. Command



<5> Data

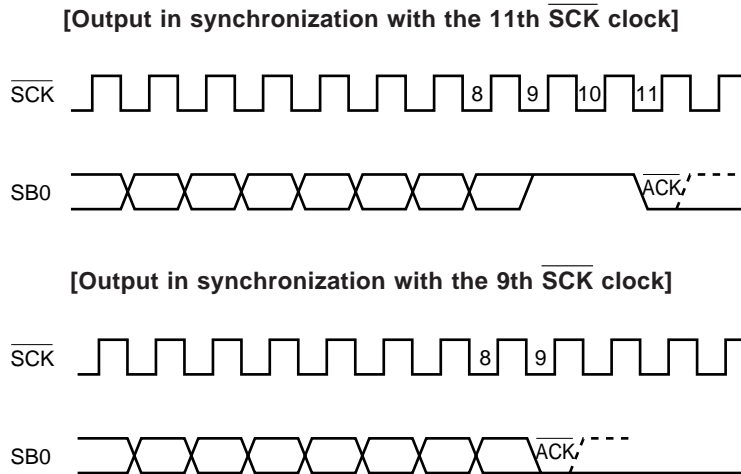
This 8-bit data is output by the master without the bus release and command signals.

Figure 10-23. Data



<6> Acknowledge signal ($\overline{\text{ACK}}$)

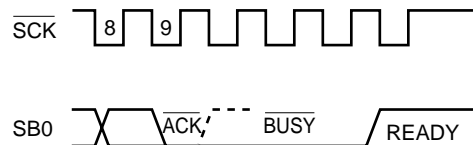
This signal is used to acknowledge reception of data transmitted between the transmission and reception sides. The master is provided with hardware that detects the acknowledge signal. If the acknowledge signal is not returned, reception is checked not to have been correctly carried out.

Figure 10-24. Acknowledge Signal**<7> Busy signal ($\overline{\text{BUSY}}$)**

This signal is output by a slave to indicate that the slave is preparing for data transmission or reception. The master continues outputting the serial clock while the slave is outputting the busy signal, and cannot start the next transmission.

<8> Ready signal ($\overline{\text{READY}}$)

This signal is output by a slave to inform the master that the slave is now ready for transmitting or receiving data.

Figure 10-25. Busy Signal and Ready Signal

(2) Configuration of one data frame of SBI

One data frame of the serial data of SBI is configured as follows:

$(REL) + (CMD) + 8\text{-bit data} + \overline{ACK} + (\overline{BUSY})$

Table 10-1. Signals in SBI Mode (1/2)

Signal Name	Output Device	Definition	Timing Chart	Output Condition	Influence on Flag	Meaning of Signal
Bus release signal (REL)	Master	Rising edge of SB0 when $\overline{SCK} = 1$		<ul style="list-style-type: none"> Setting of RELT 	<ul style="list-style-type: none"> Sets RELD Clears CMDD 	End of series of transmission to selected slave
Command signal (CMD)	Master	Falling edge of SB0 when $\overline{SCK} = 1$		<ul style="list-style-type: none"> Setting of CMDT 	<ul style="list-style-type: none"> Sets CMDD 	i) Address is transmitted after output of REL ii) REL not output. Command is transmitted
Acknowledge signal (ACK)	Slave	Low-level signal output to SB0 during 1-clock period of SCK after serial clock is received	<p>(Sync busy output)</p>	<ul style="list-style-type: none"> ACKE = 1 Setting of ACKT 	<ul style="list-style-type: none"> Sets ACKD 	End of reception
Busy signal (BSY)	Slave	[Sync busy signal] Low-level signal output to SB0 following acknowledge signal		<ul style="list-style-type: none"> BSYE = 1 	—	Serial reception disabled during processing
Ready signal (RDY)	Slave	High-level signal output to SB0 before start or after end after end of serial transfer		<ul style="list-style-type: none"> BSYE = 0 Execution of data write instruction to SIO (transfer start command) 	—	Serial reception enabled

Table 10-1. Signals in SBI Mode (2/2)

Signal Name	Output Device	Definition	Timing Chart	Output Condition	Influence on Flag	Meaning of Signal
Serial clock (SCK)	Master	Sync clock to output address, command, data, ACK, and sync BUSY. Address, command, or data is transferred when first eight clocks are output.		Execution of instruction to write data to SIO when CTXE = 1 (serial transmission start direction) ^{Note 3}	Setting of CSIF ^{Note 1} (rising edge of 8th clock) Note 2	Timing of signal output to serial data bus
Address (A0-A7)	Master	8-bit data transferred in synchronization with SCK after output of REL and CMD signals.			Note 2	Address of slave device on serial bus
Command (D0-D7)	Master	8-bit data transferred in synchronization with SCK after output of only CMD signal. REL signal is not output.			None	Direction message to slave device
Data (D0-D7)	Master/slave	8-bit data transferred in synchronization with SCK. Both REL and CMD signals are not output.			None	Actual data processed by slave or master

Notes 1. An interrupt request flag corresponding to serial transmission/reception interrupt (INTCSI).

2. CSIF is always set at the rising edge of the eighth clock of SCK when WUP = 0.

CSIF is set only when an address is received when WUP = 1.

3. In BUSY status, transmission is not started until READY status is set.

10.6.2 Controlling and detecting status of serial bus

The status of the serial bus can be controlled and detected by using the serial bus interface control register (SBIC).

- **Controlling and detecting status of serial bus**

SBI has a serial bus interface control register (SBIC) that detects and controls the status of the serial bus.

The SBIC register is an 8-bit register consisting of bits that control the status of the serial bus, and flags that indicate various statuses of the data input from the serial bus.

This register can be read or written by an 8-bit manipulation instruction or a bit manipulation instruction. However, some bits are read-only or write-only. If a write-only bit is read, “0” is read.

The contents of the SBIC register are initialized to 00H when the $\overline{\text{RESET}}$ signal is input.

Figure 10-26 shows the format of the SBIC register.

The operations of the respective bits are illustrated in Figures 10-27 through 10-31.

Note that, throughout this section, SB0 is used as the serial data I/O pin. Operation is exactly the same even when SB1 is selected.

Figure 10-26. Format of Serial Bus Interface Control Register

Symbol	⑦	⑥	⑤	④	③	②	①	①	Address	When reset
SBIC	BSYE	ACKD	ACKE	ACKT	CMDD	RELD	CMDT	RELT	FF82H	00H

RELT

W

Controls trigger output of bus release signal (REL)

0

Does not output

1

Outputs

CMDT

W

Controls trigger output of command signal (CMD)

0

Does not output

1

Outputs

RELD

R

Detects bus release signal (REL)

0

Does not detect

1

Detects (serial bus goes high while \overline{SCK} is high)

CMDD

R

Detects command signal (CMD)

0

Does not detect

1

Detects (serial bus goes low while \overline{SCK} is high)

ACKT

W

Controls trigger output of acknowledge signal (\overline{ACK})

0

Does not output

1

Outputs

ACKE

R/W

Controls automatic output of acknowledge signal (\overline{ACK})

0

Disables

1

Enables

ACKD

R

Detects acknowledge signal (\overline{ACK})

0

Does not detect

1

Detects

BSYE

R/W

Controls automatic output of synchronization busy signal (BUSY)

0

Disables

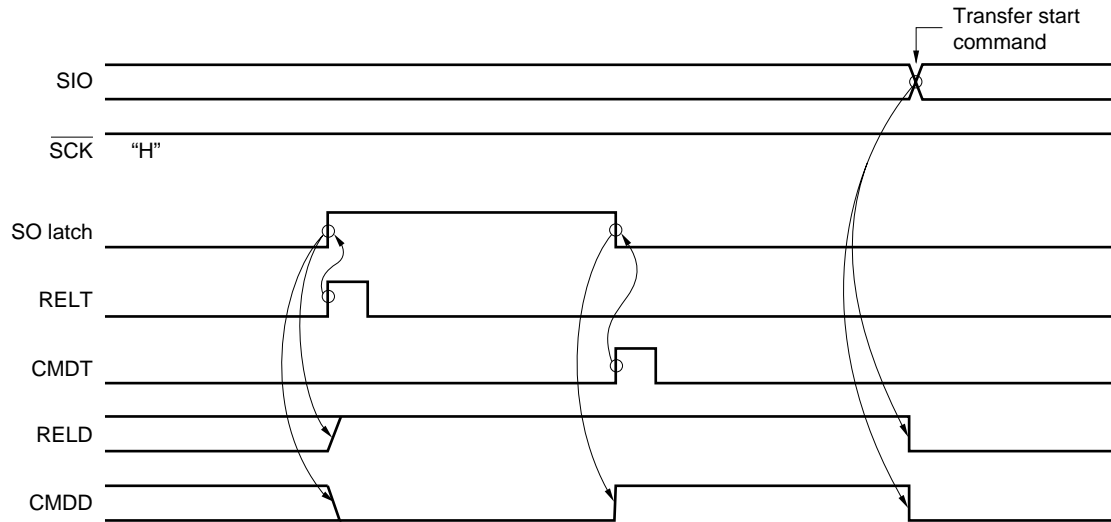
1

Enables

Remark R/W: read/write

R : read only

W : write only

Figure 10-27. Operations of RELT, CMDT, RELD, and CMDD

Caution Do not manipulate the RELT and CMDT bits during transmission or reception.

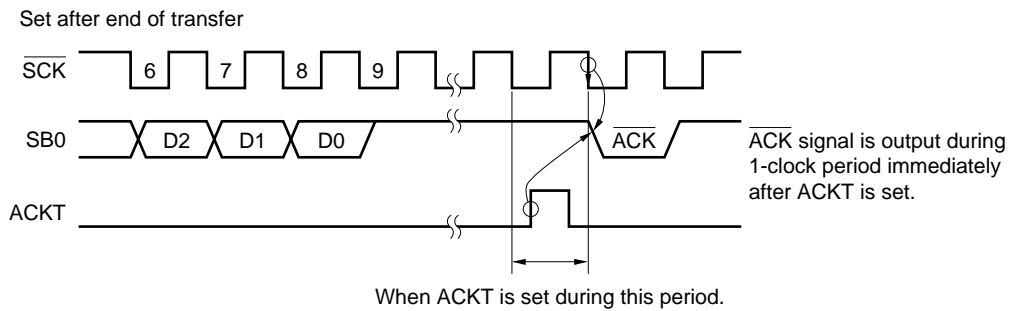
Figure 10-28. Operation of ACKT

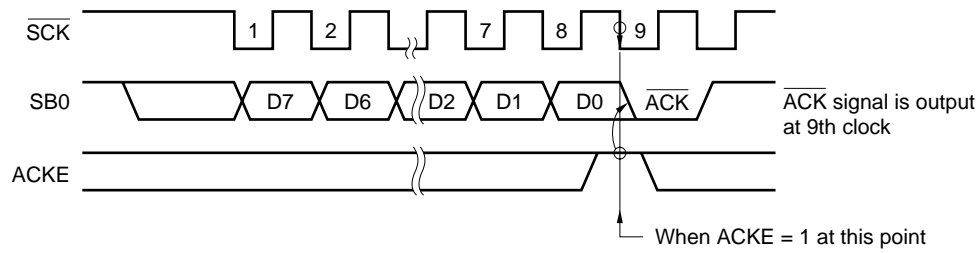
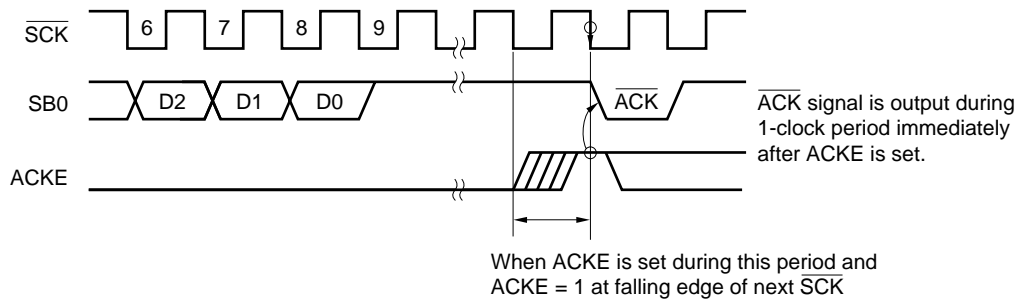
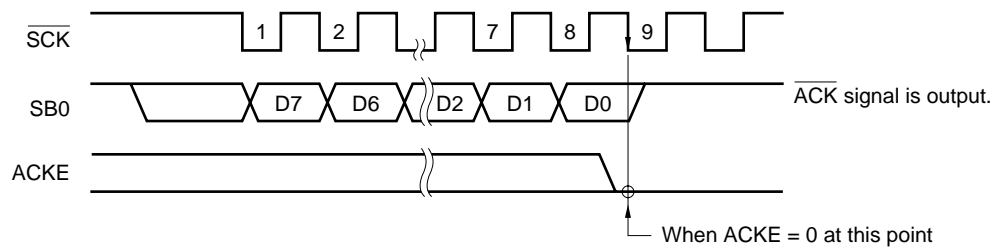
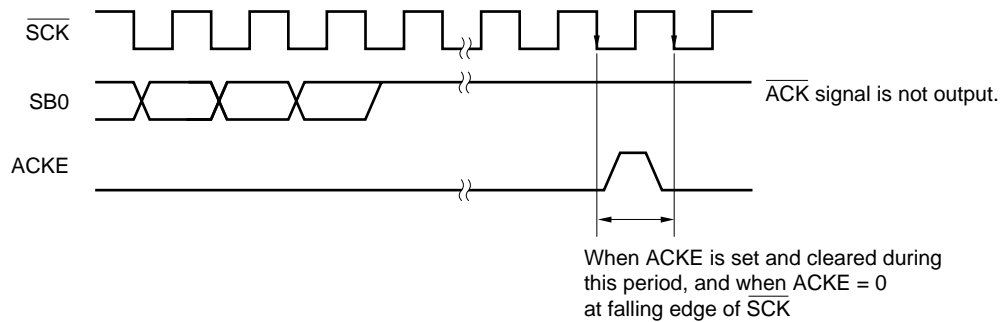
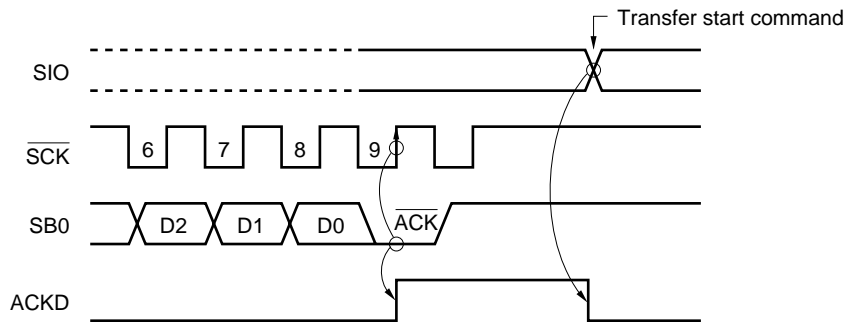
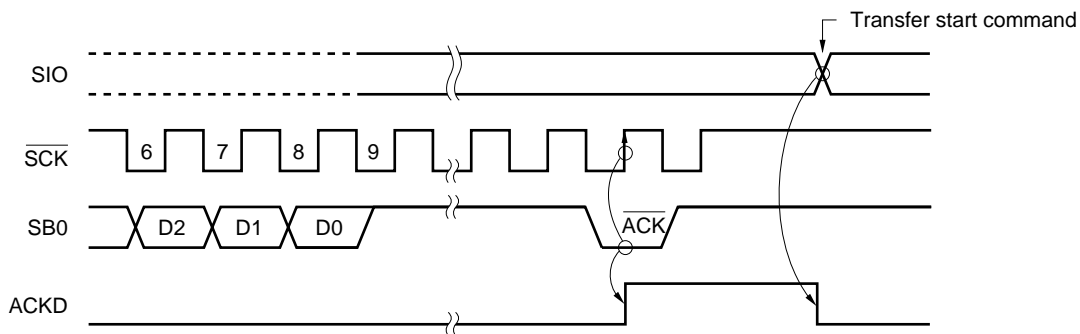
Figure 10-29. Operation of ACKE**a. When ACKE = 1 during transfer****b. When ACKE is set immediately after end of transfer****c. When ACKE = 0 at end of transfer****d. When period of ACKE = 1 is short**

Figure 10-30. Operation of ACKD

a. When $\overline{\text{ACK}}$ signal is output at 9th clock of $\overline{\text{SCK}}$ b. When $\overline{\text{ACK}}$ signal is output after 9th clock of $\overline{\text{SCK}}$ 

c. Clearing timing when transfer start command is issued during BUSY

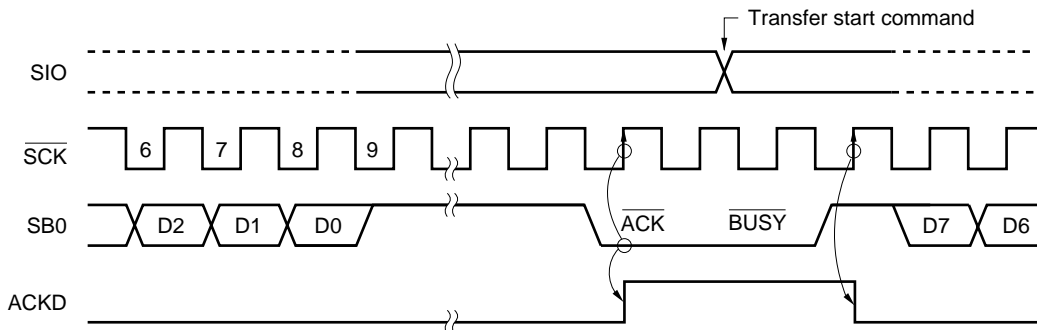
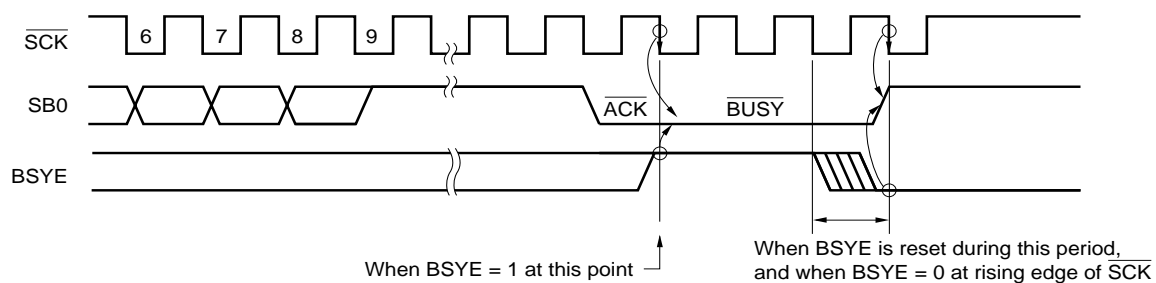


Figure 10-31. Operation of BSYE

10.6.3 Communicating with SBI

Transmit data is written to the shift register (SIO) and transmission/reception is performed after the serial bus has been controlled by the serial bus interface control register (SBIC).

Although the SB0 pin is used in the following description, operation is exactly the same even when the SB1 pin is selected.

(1) Starting transmission

When the transmit data is written to the shift register (SIO) with the CTXE bit of the clocked serial interface mode register (CSIM) set to 1, transmission is started.

The contents of the SIO register are shifted in synchronization with the falling edge of the serial clock (\overline{SCK}), and output from the SB0 pin with the MSB first. After the transmit data has been transmitted, "0" is written to the SIO register, and interrupt request INTCSI occurs.

(2) Starting reception

Reception can be started in the following two ways:

- <1> By setting the CRXE bit of the CSIM register from 0 to 1, thereby enabling reception when CTXE bit is "0"
- <2> By reading the receive data from the SIO register when the CRXE bit of the CSIM register is set to 1 to enable reception

Reception is not started even if "1" is written to the CRXE bit of the CSIM register which has already been set to "1". Reception is not started when CTXE bit is "1", even if the CRXE bit setting is set to 1 from 0. The 8-bit data input to the SB0 pin is latched to the SIO register with the MSB first, at the rising edge of the serial clock (\overline{SCK}), and interrupt request INTCSI occurs. When using the wakeup function, interrupt request INTCSI is generated only when an address is received.

(3) Starting transmission/reception

Transmission/reception is started when the transmit data is written to the SIO register with both the CTXE and CRXE bits of the CSIM register set to "1".

The data on the serial bus is input to the SIO register as is. By comparing the transmit data written to the SIO register with the data input from the serial bus, bus contention can be checked.

Remark INTCSI ... vector table address: 0022H (TPF = 0), 8022H (TPF = 1)
macro service control word address: FE22H

Caution Be sure to specify a pin and serial clock before setting the CTXE and CRXE bits in the SBI mode.

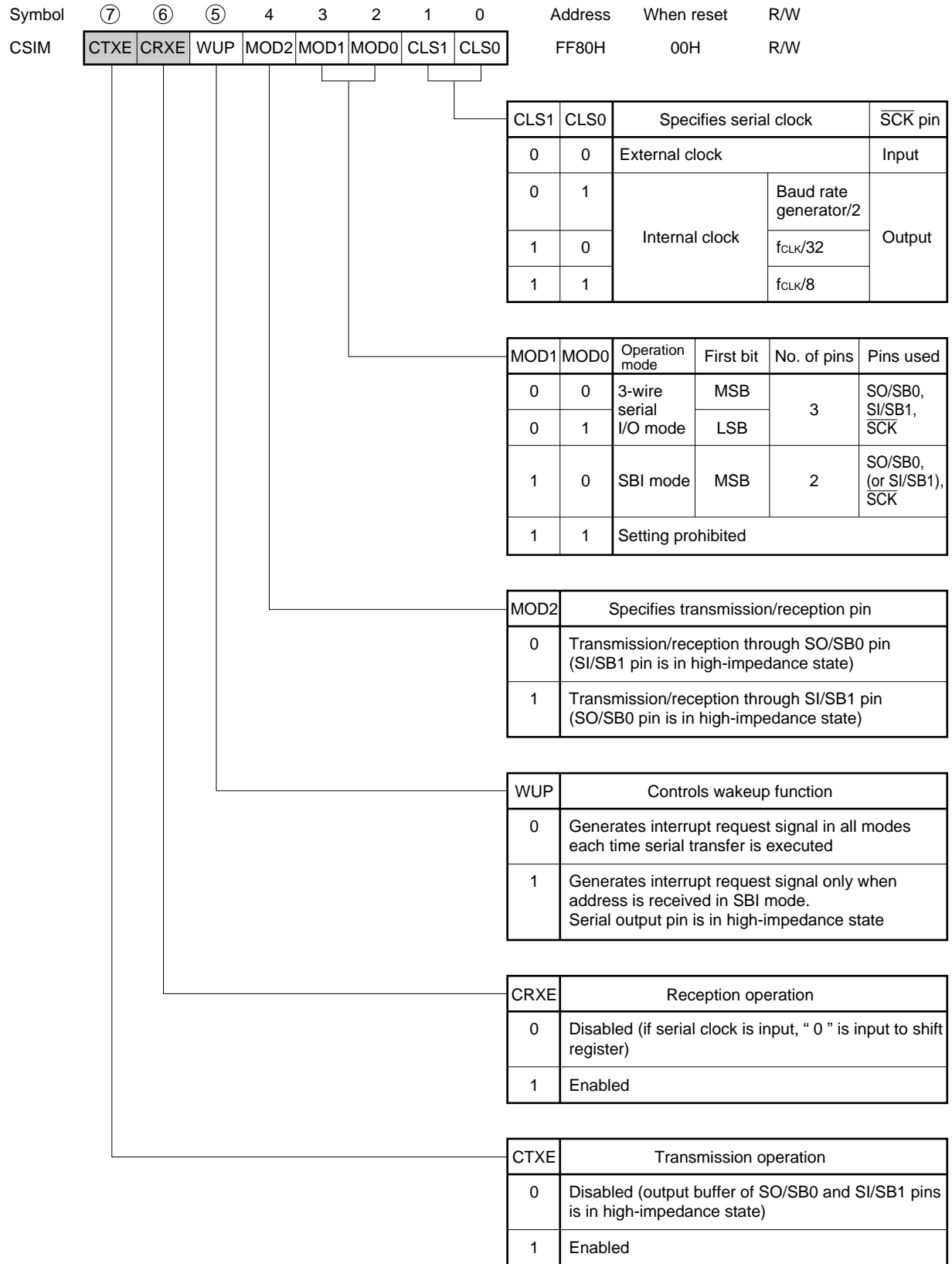
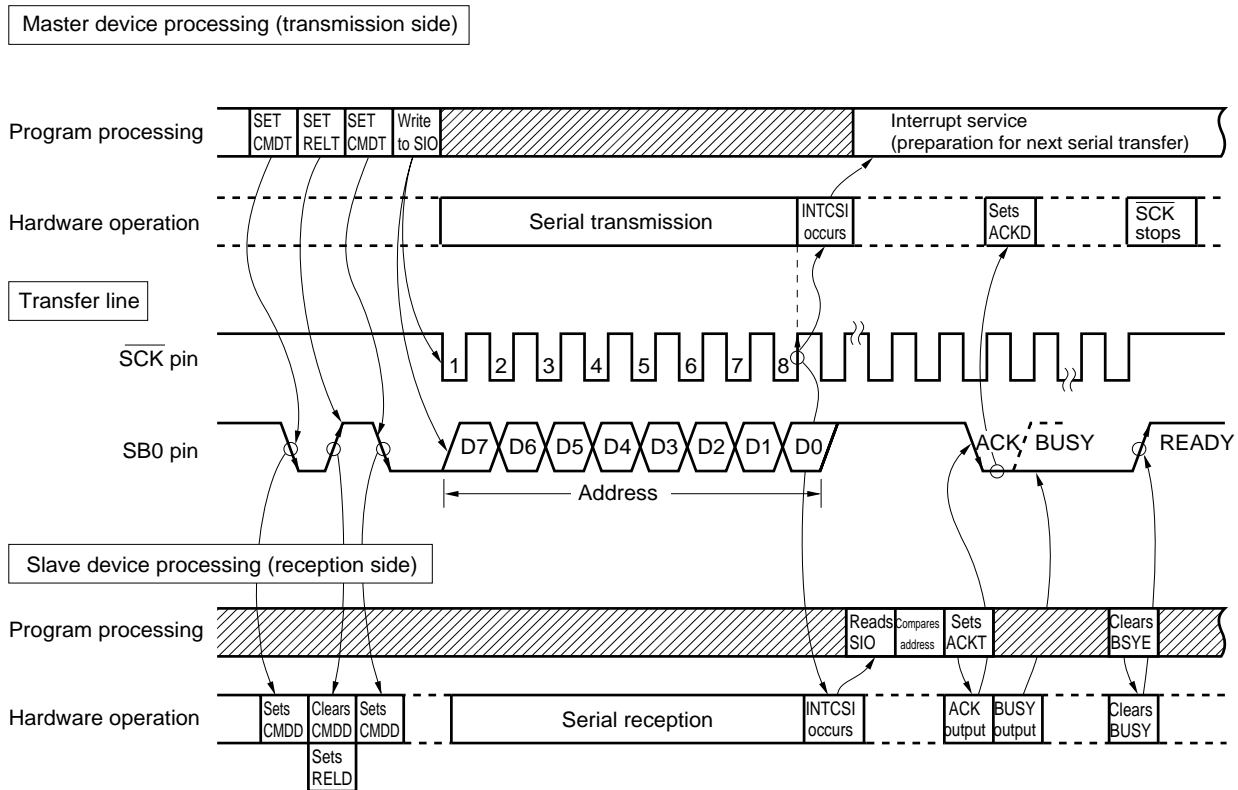
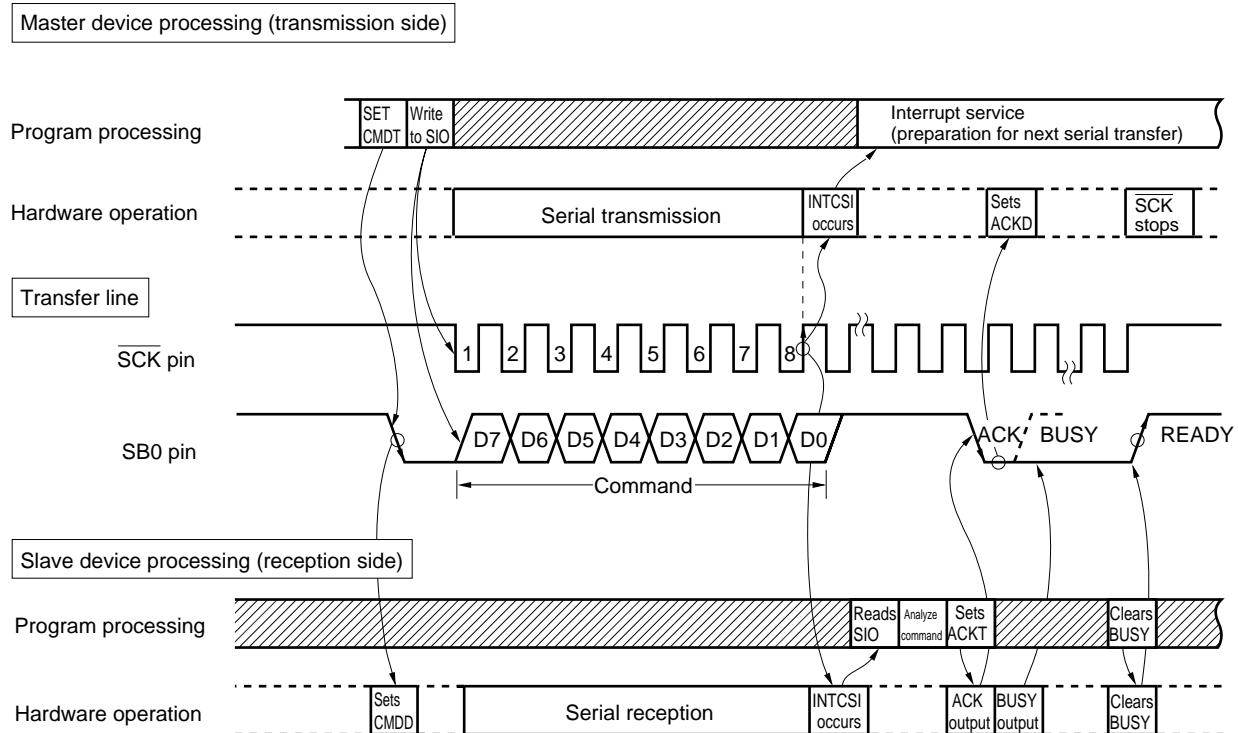
Figure 10-32. Setting of CSIM Register (transmission/reception enabled)**Remark** f_{CLK} : internal system clock

Figure 10-33. Address Transfer from Master Device to Slave Device**Figure 10-34. Command Transfer from Master Device to Slave Device**

The diagram illustrates the timing of an SPI serial transfer between a Master device (transmission side) and a Slave device (reception side). It is divided into two main sections: Master device processing and Slave device processing.

Master device processing (transmission side):

- Program processing:** Includes "Write to SIO" and "Interrupt service (preparation for next serial transfer)".
- Hardware operation:** Shows "Serial transmission" and "SCK stops".
- Transfer line:** Shows the SCK pin (Serial Clock) and SB0 pin (Slave Select).
- SB0 pin:** Shows the data bus (D7-D0) and the ACK signal. The data is transmitted in 8-bit chunks (D7-D0, D7-D0, D7-D0, D7-D0).
- ACK signal:** The ACK signal is active-low and is pulled up to VCC. It is set by the slave device (ACKD) and cleared by the master device (ACK).

Slave device processing (reception side):

- Program processing:** Includes "Sets ACKE", "Sets BSYE", "Reads SIO", and "Receive data processing".
- Hardware operation:** Shows "Serial reception", "INTCSI occurs", "ACK output", "BUSY output", and "Resets BUSY".
- ACK signal:** The ACK signal is active-low and is pulled up to VCC. It is set by the slave device (ACKD) and cleared by the master device (ACK).
- BUSY signal:** The BUSY signal is active-low and is pulled up to VCC. It is set by the slave device (BUSY output) and cleared by the master device (ACK).

The diagram shows the sequence of events: the master device writes to the SIO, the slave device sets the ACK signal, the master device sets the ACK signal, the slave device sets the BUSY signal, the master device reads the SIO, and the slave device resets the BUSY signal. The data is transferred in 8-bit chunks (D7-D0).

The diagram illustrates the timing for SPI reception. It is divided into two main sections: Master device processing (reception side) and Slave device processing (transmission side).

Master device processing (reception side):

- Program processing:** Shows a sequence of operations: Reads SIO, Sets ACK, and Receive data processing. The Receive data processing step is repeated by the READY signal.
- Hardware operation:** Shows the hardware state during reception. SCK stops when the master starts reading SIO. Serial reception occurs while the SCK is stopped. INTCSI occurs when the master sets ACK. ACK output is sent to the slave.

Slave device processing (transmission side):

- Program processing:** Shows a sequence of operations: Writes to SIO and Writes to SIO. The second Writes to SIO step is repeated by the READY signal.
- Hardware operation:** Shows the hardware state during transmission. Resets BUSY when the master starts reading SIO. Serial reception occurs while the SCK is stopped. INTCSI occurs when the master sets ACK. Sets ACKD and Resets BUSY are performed when the master outputs ACK.

Transfer line: Shows the data being transferred. The SCK pin is a square wave. The SB0 pin shows the data being received by the master (D7, D6, D5, D4, D3, D2, D1, D0) and transmitted by the slave (D7, D6, D5). The data is shown as a sequence of bits (1, 2, 3, 4, 5, 6, 7, 8) and a sequence of bytes (D7, D6, D5, D4, D3, D2, D1, D0).

10.6.4 Operation only when address is received

The processing efficiency of the slave CPU can be enhanced by using the wake-up function that is effected only when an address is received.

- **Processing efficiency of slave CPU is enhanced by wake-up function**

In the SBI mode, a wake-up function that generates interrupt request INTCSI only when an address has been received can be used.

When the serial interface is configured as shown in Figure 10-37, and when data is transmitted/received with a certain slave CPU, the other slave CPUs can operate independently of serial communication. If the wake-up function is not used, the interrupt occurs each time data is received, and therefore, all the slave CPUs are influenced each time serial communication is executed. The wake-up function can therefore be used to enhance the efficiency of the slave CPUs.

When a slave CPU receives an address, it compares it with its own address during service of the INTCSI interrupt that has occurred. If the two addresses coincide, the wake-up function is cleared, and the next data is received. If the addresses do not coincide, the wake-up status continues.

Remark INTCSI ... vector table address: 0022H (TPF = 0), 8022H (TPF = 1)
macro service control word address: FE22H

Figure 10-37. Example of System Configuration of Serial Bus Interface (SBI)

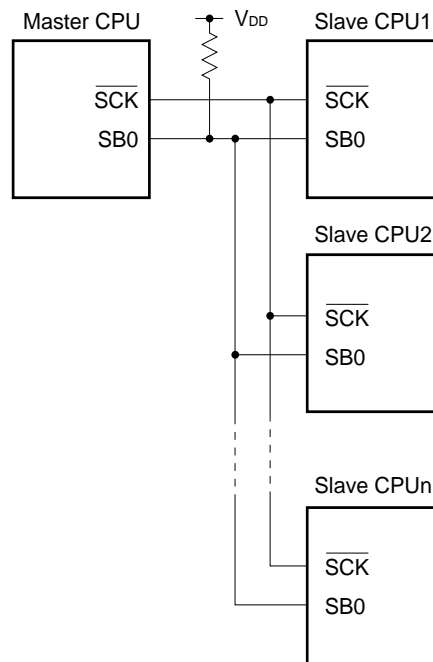
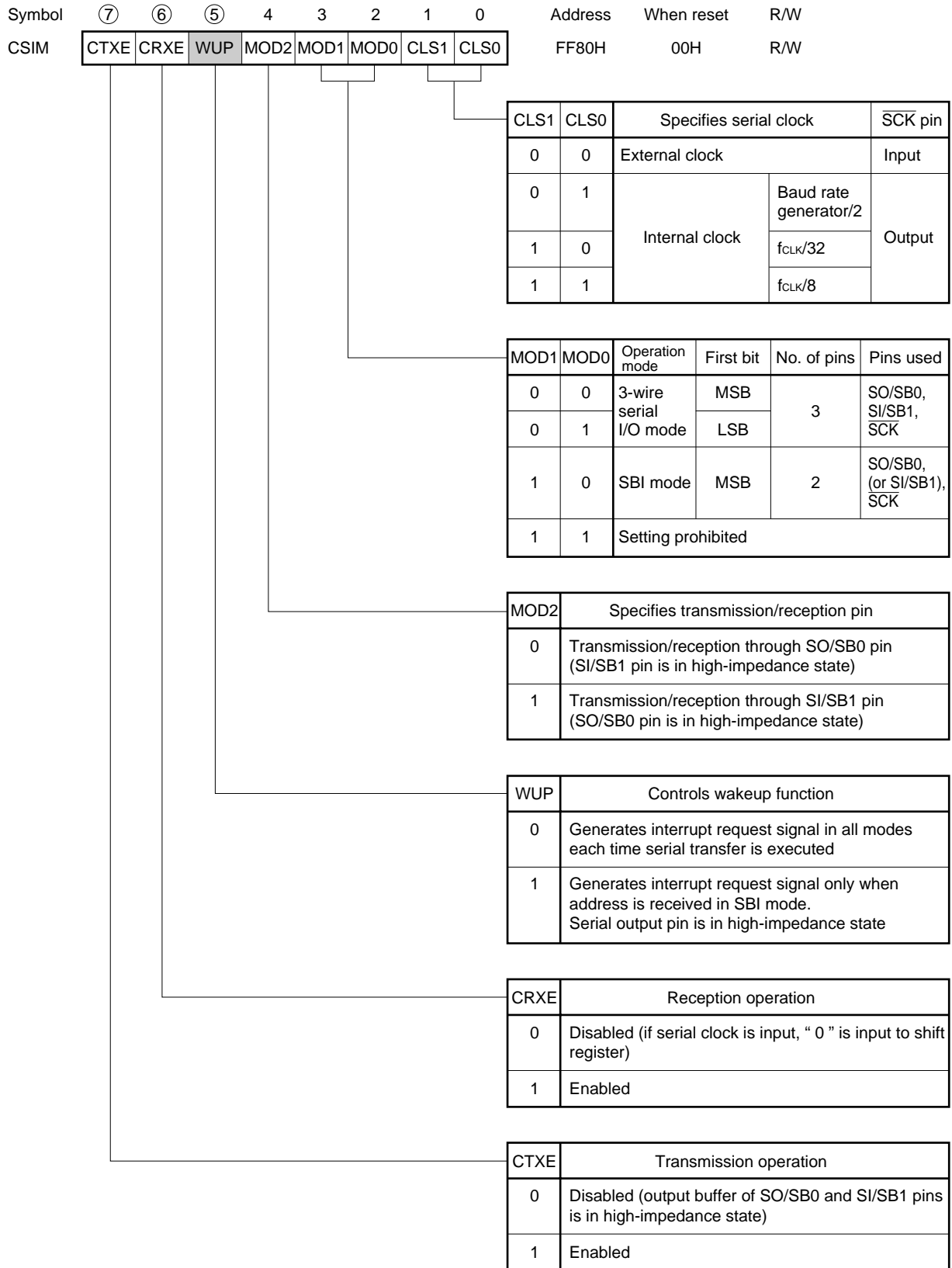


Figure 10-38. Setting of CSIM Register (wake-up function)**Remark** fCLK: internal system clock

[MEMO]

CHAPTER 11 PWM SIGNAL OUTPUT FUNCTION

11.1 Configuration

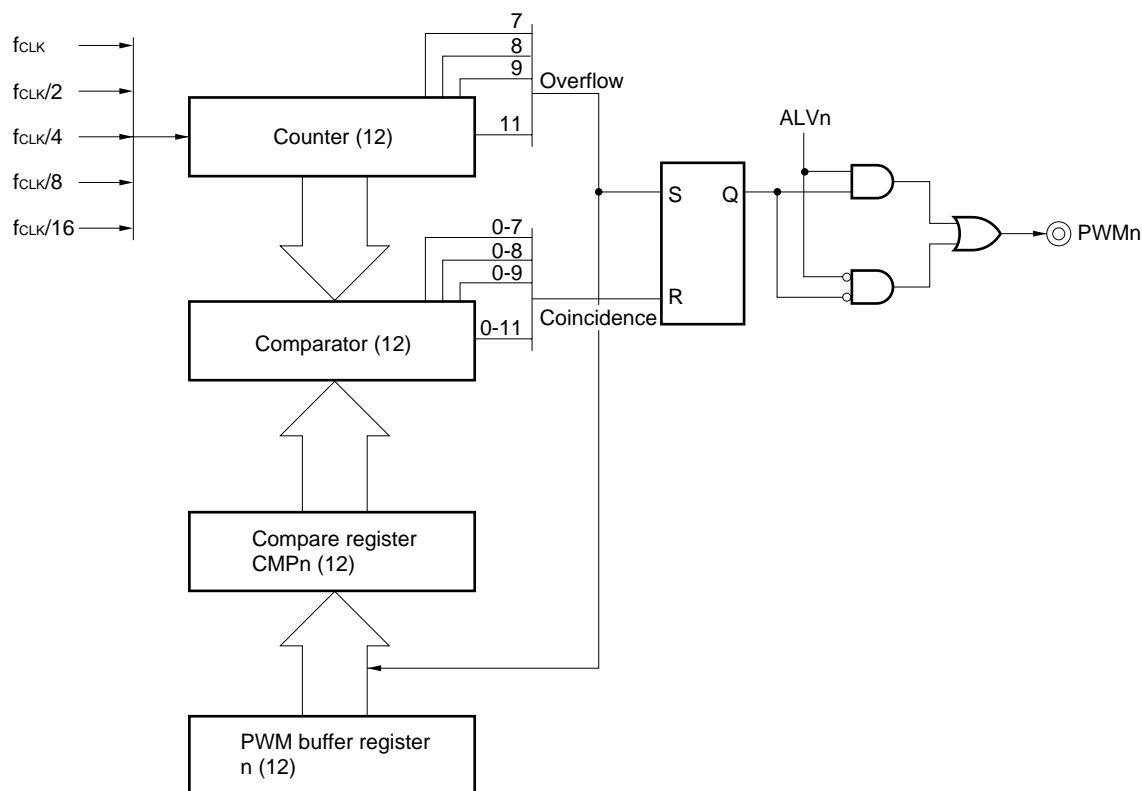
The μ PD78366A has two PWM signal outputs of variable resolution of 8-/9-/10-/12-bit and one 16-bit resolution PWM signal output.

The 16-bit resolution PWM signal output, which is a multifunctional timer output (TO40), is described in **CHAPTER 7 REAL-TIME PULSE UNIT**. Chapter 11 describes PWM dedicated functions PWM0 and PWM1.

By externally connecting a low-pass filter, a PWM output can be used as a digital-to-analog conversion output. The PWM outputs are most suitable, for example, as a control signal for the actuator of a motor.

Figure 11-1 shows the block diagram of the PWM Unit. Table 11-1 lists PWM signal output repetition frequencies.

Figure 11-1. Block Diagram of PWM Unit



Remark $n = 0, 1$

Table 11-1. PWM Signal Repetition Frequencies

Counter Bit Length	Count Clock	Repetition Frequency			
		f _{CLK} = 12.5 MHz		f _{CLK} = 16.0 MHz	
8 bits	f _{CLK}	48.83 kHz	20.48 μ s	62.50 kHz	16.0 μ s
	f _{CLK} /2	24.41 kHz	40.96 μ s	31.25 kHz	32.0 μ s
	f _{CLK} /4	12.21 kHz	81.92 μ s	15.63 kHz	64.0 μ s
	f _{CLK} /8	6.10 kHz	163.84 μ s	7.81 kHz	128.0 μ s
	f _{CLK} /16	3.05 kHz	327.68 μ s	3.91 kHz	256.0 μ s
9 bits	f _{CLK}	24.41 kHz	40.96 μ s	31.25 kHz	32.0 μ s
	f _{CLK} /2	12.21 kHz	81.92 μ s	15.63 kHz	64.0 μ s
	f _{CLK} /4	6.10 kHz	163.84 μ s	7.81 kHz	128.0 μ s
	f _{CLK} /8	3.05 kHz	327.68 μ s	3.91 kHz	256.0 μ s
	f _{CLK} /16	1.53 kHz	655.36 μ s	1.95 kHz	512.0 μ s
10 bits	f _{CLK}	12.21 kHz	81.92 μ s	15.63 kHz	64.0 μ s
	f _{CLK} /2	6.10 kHz	163.84 μ s	7.81 kHz	128.0 μ s
	f _{CLK} /4	3.05 kHz	327.68 μ s	3.91 kHz	256.0 μ s
	f _{CLK} /8	1.53 kHz	655.36 μ s	1.95 kHz	512.0 μ s
	f _{CLK} /16	763 Hz	1.31 ms	977 Hz	1.02 ms
12 bits	f _{CLK}	3.05 kHz	327.68 μ s	3.91 kHz	256.0 μ s
	f _{CLK} /2	1.53 kHz	655.36 μ s	1.95 kHz	512.0 μ s
	f _{CLK} /4	763 Hz	1.31 ms	977 Hz	1.02 ms
	f _{CLK} /8	381 Hz	2.62 ms	488 Hz	2.05 ms
	f _{CLK} /16	191 Hz	5.24 ms	244 Hz	4.1 ms

Remark f_{CLK}: internal system clock

11.2 Control Register

11.2.1 PWM control registers (PWMC0, PWMC1)

The PWM control registers control PWM output. Figures 11-2 and 11-3 show the formats of the PWMC0 and PWMC1 registers.

When a RESET signal is input, the PWMC0 and PWMC1 registers are initialized to 00H.

Figure 11-2. Format of PWM Control Register 0

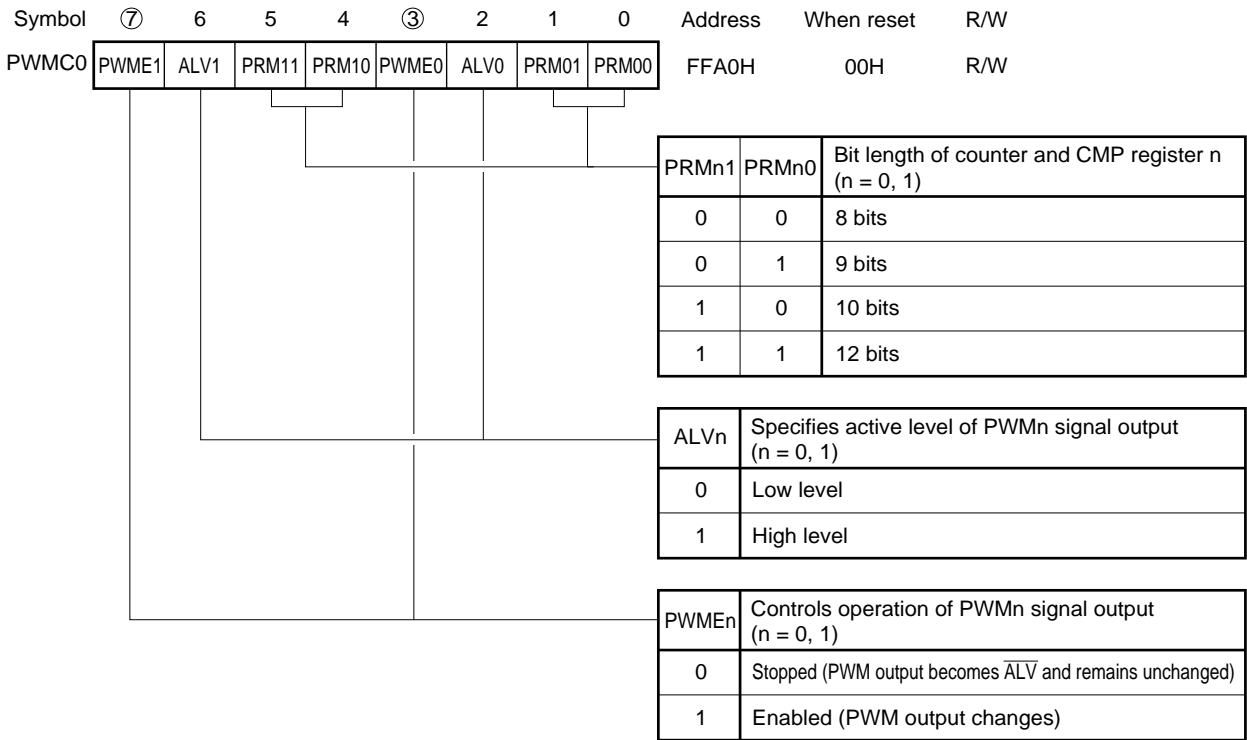
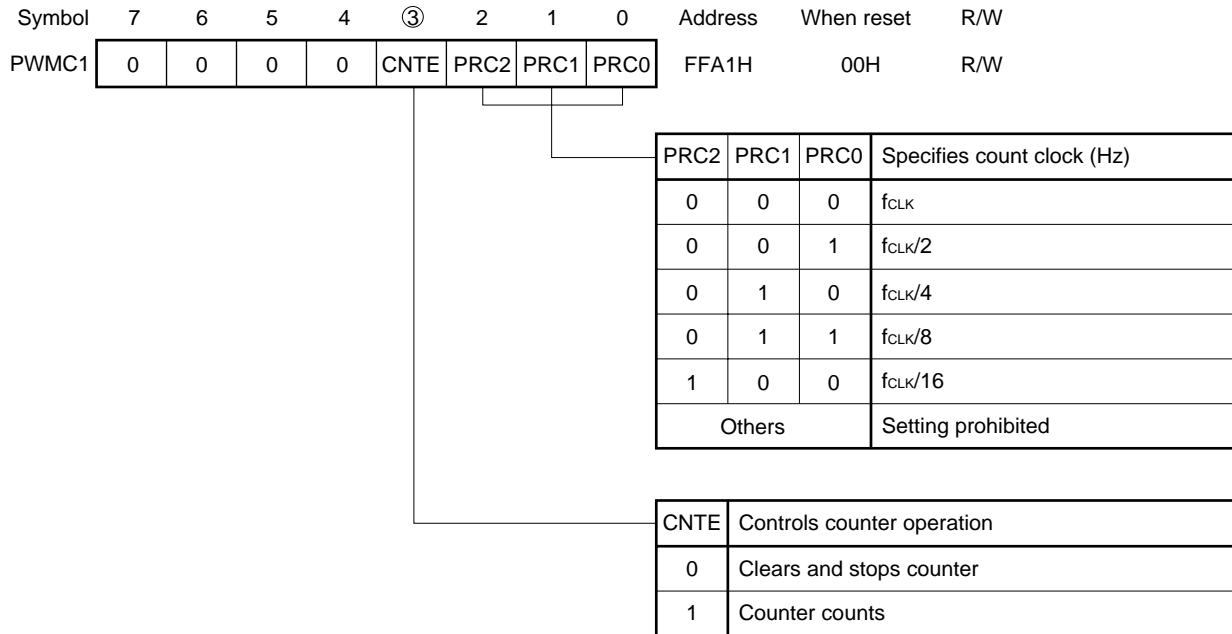


Figure 11-3. Format of PWM Control Register 1



- Cautions**
1. To output PWM signal, be sure to set CNTE bit to 1.
 2. The PWMC1 register is a control register common to PWM0 and PWM1.
 3. Bits 4 to 7 of the PWMC1 register are fixed to “0” by hardware. Even if “1” is written to them, they remain “0”.

Remark f_{CLK} : Internal system clock

11.2.2 PWM buffer registers (PWM0, PWM1)

The PWM0 and PWM1 registers are 12-bit registers that set the data for controlling the active signal width of PWM output. Data can be read from or written into the registers in byte or word units by an instruction.

When an overflow occurs in the counter for PWM output control, the contents of the PWM0 or PWM1 register are sent to compare register CMP0 or CMP1.

Comparison operation between the counter and the PWM0/PWM1 register is performed on bits 0-11 regardless of how the PRM00, 01/PRM10, 11 bits are specified. Therefore, if the bit length of the counter, CMP0/CMP1 register is specified as 10 bits or less, write “0” into the high-order bits.

When RESET is input, the PWM buffer register becomes undefined.

- Cautions**
1. A byte access/bit access can be made to the low-order part of the PWM0/PWM1 register, but cannot be made to the high-order part.
 2. Bits 12 to 15 of the PWM0/PWM1 registers are fixed to “0” by hardware. Even if “1” is written to them, they remain “0”.

11.2.3 Compare registers (CMP0, CMP1)

The CMP0 and CMP1 compare registers are 12-bit registers used to detect coincidence between the values of the register and counter for PWM output control.

The registers cannot be directly manipulated by an instruction.

11.3 Operation

The PWM output function is used in the following procedure:

- The PWM output signal period is specified by bits PRM00, 01/PRM10, 11 of the PRMC0 register and bits PRC0-PRC2 of the PWMC1 register.
- Next, set the control data which has the same active width as that of the PWM output to the PWM0/PWM1 register.
- Next, set the CNTE bit of the PWMC1 register to 1.
- Finally, set the PWME0/PWME1 bit of the PWMC0 register to 1.

After these steps are completed, a PWM signal is output from port P04 or P05. The active signal width of PWM output can be changed by rewriting the contents of the PWM0 or PWM1 register.

The PWM output function sends the contents of the PWM0 or PWM1 register to the CMP0 or CMP1 register when the counter for PWM output control overflows. The PWM output signal goes inactive when the contents of the CMP0 or CMP1 register coincide with the value of the counter for PWM output control. The PWM output signal goes active when the counter for PWM output control overflows.

If PWME0 = 0/PWME1 = 0 is set and the PWM operation is halted, the PWM output reaches the inactive level immediately.

Figure 11-4. Operation of PWM Output Function (high-active setting) (1/2)

(1) Normal operation

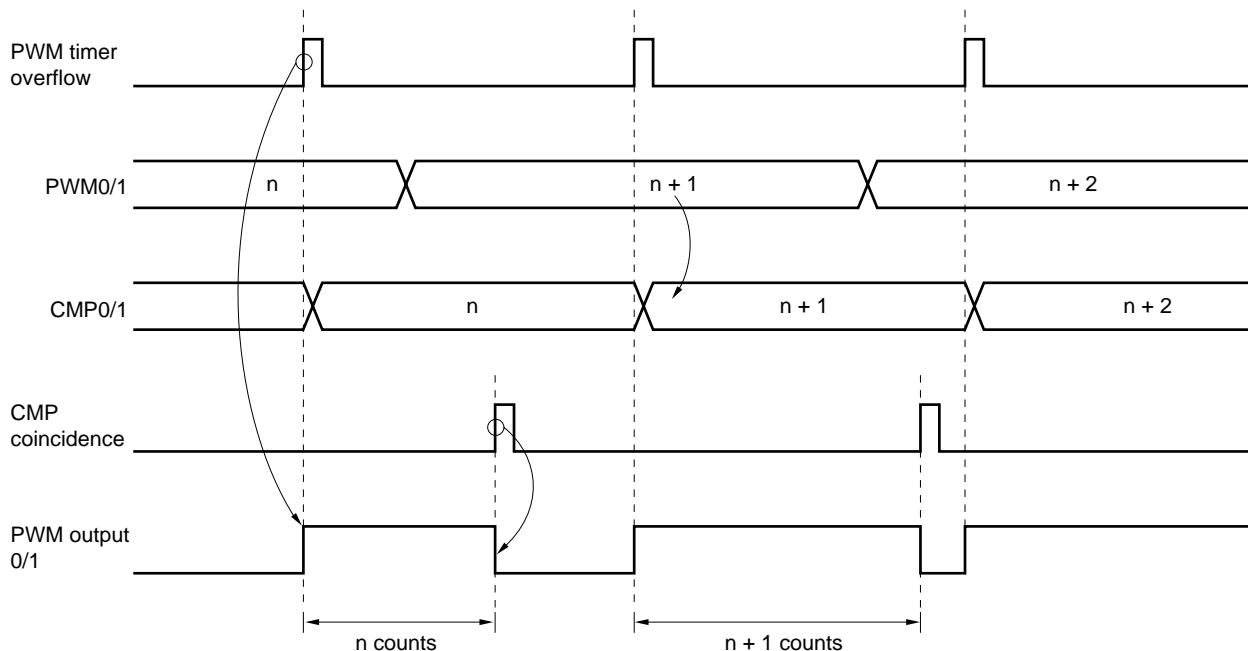


Figure 11-4. Operation of PWM Output Function (high-active setting) (2/2)

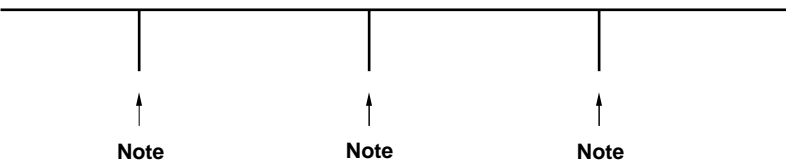
(2) When PWM0/PWM1 = 00H

PWM output 0/1 "L"



(3) When PWM0/PWM1 = 0FH

PWM output 0/1



Note Note Note

Note 1 count clock width of PWM counter

CHAPTER 12 WATCHDOG TIMER

12.1 Configuration

The watchdog timer prevents crashes or deadlocks.

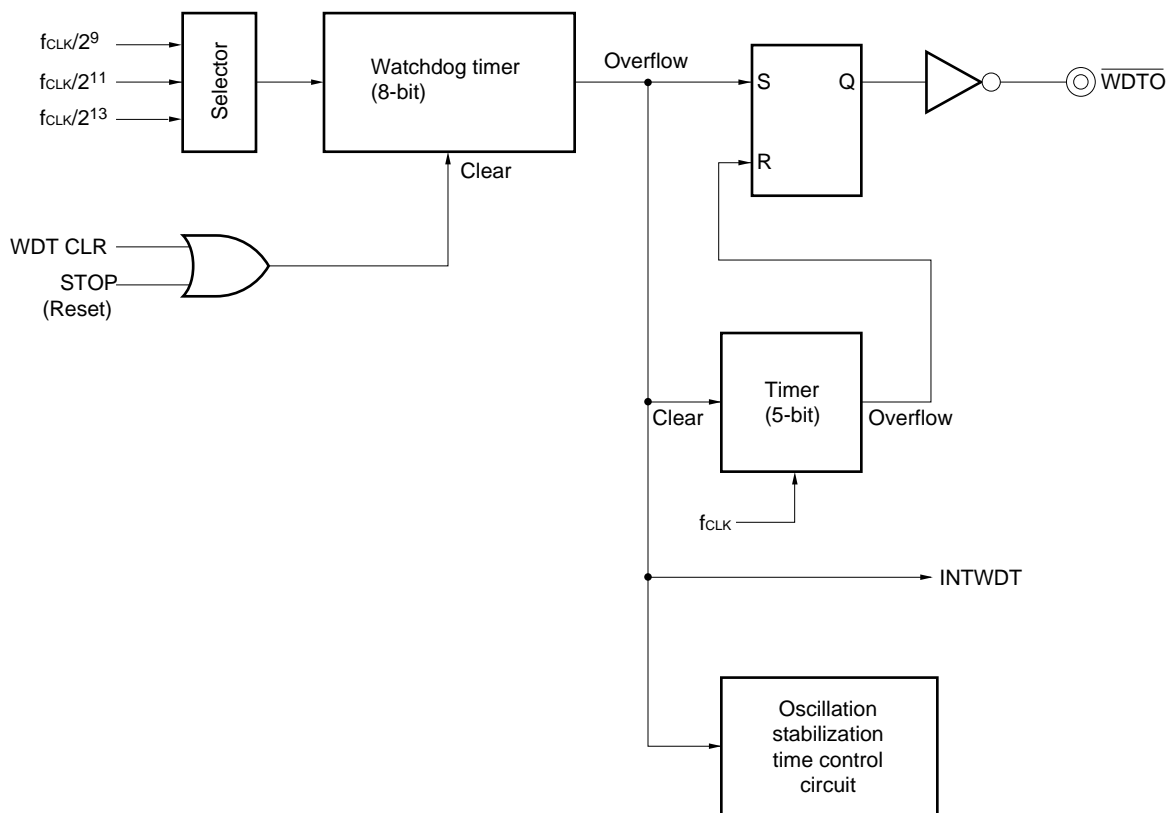
If no watchdog timer interrupt occurs, the program or system is running normally. Each module of a program must have an instruction to clear the watchdog timer and to start counting.

If the instruction to clear the watchdog timer is not executed within a specified time period, an overflow occurs in the watchdog timer and a watchdog timer interrupt (INTWDT) is generated. At the same time, the $\overline{\text{WDTO}}$ pin goes low to indicate that an error has occurred in the program.

The watchdog timer can also be used to guarantee a time required for the oscillator to perform stable operation when the STOP mode is released (refer to **14.3.2 STOP mode**). If the watchdog timer is used as a timer for the oscillation stabilization time, it does not overflow due to a program or system abnormality, and therefore no low-level output of INTWDT and $\overline{\text{WDTO}}$ is generated.

Figure 12-1 shows a block diagram of the watchdog timer.

Figure 12-1. Block Diagram of Watchdog Timer



12.2 Watchdog Timer Mode Register (WDM)

The watchdog timer mode (WDM) register is an 8-bit register which controls the operation of the watchdog timer.

Data can be written into the WDM register only by a special instruction. This prevents the contents of the WDM register from being rewritten accidentally if the program crashes. The specialized instruction is MOV WDM, #byte instruction, consisting of special codes (4 bytes). Data is written only when the op-codes of bytes 3 and 4 complement each other.

Unless the op-codes of bytes 3 and 4 complement each other, data is not written and an op-code trap interrupt occurs. The address of the instruction causing the trap is saved in the stack area. When an RETB instruction is executed, the program can be restarted from the address of the instruction causing the trap.

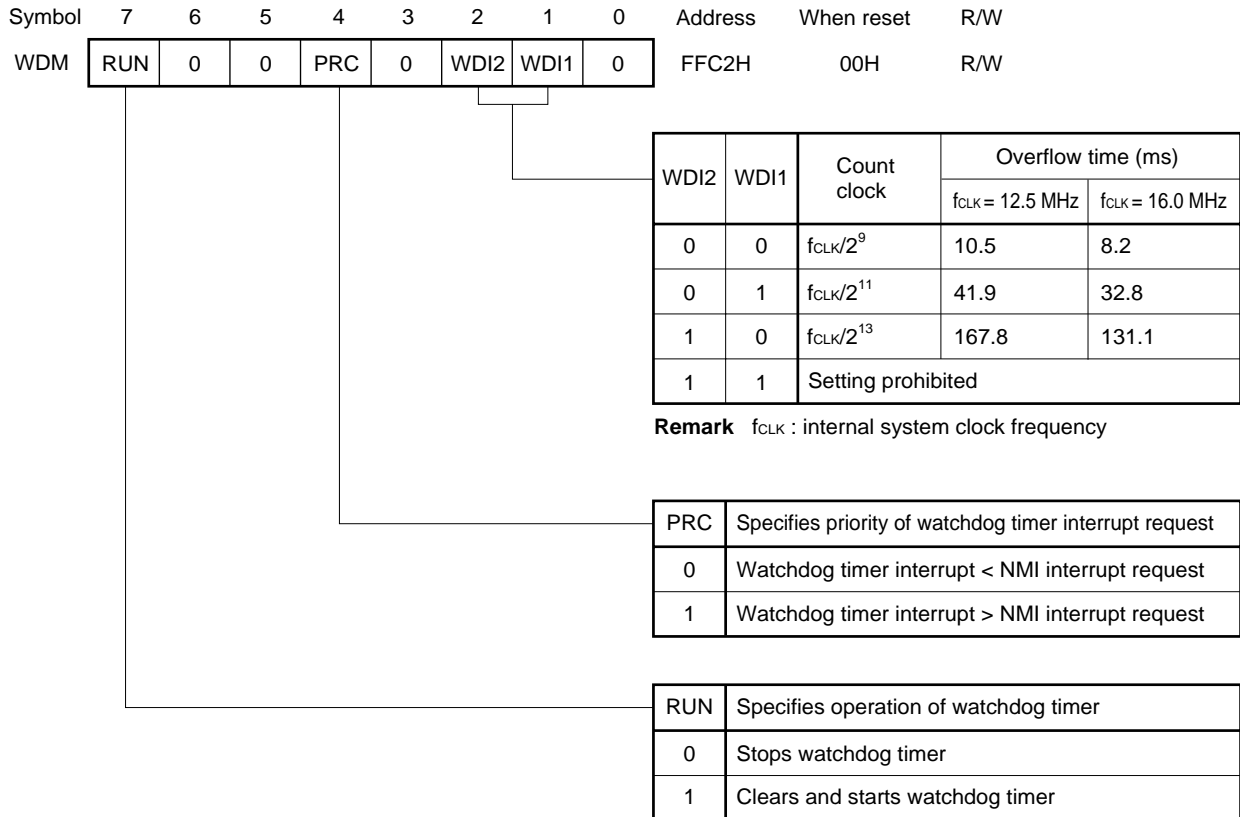
If the RETB instruction is executed before a hardware error or other cause of the op-code trap is eliminated, the program enters an infinite loop.

If the watchdog timer is started after a system reset signal ($\overline{\text{RESET}}$ input) is entered, the contents of the WDM register cannot be rewritten. Only the system reset signal can stop the watchdog timer. The watchdog timer can be cleared at any time by a special instruction.

The contents of the WDM register can be read at any time by a data transfer instruction.

When a $\overline{\text{RESET}}$ signal is input, the WDM register is initialized to 00H.

Figure 12-2 shows the format of the WDM register.

Figure 12-2. Format of Watchdog Timer Mode Register

- Cautions**
1. Data can be written into the WDM register only by a dedicated instruction (MOV WDM, #byte).
 2. Set the priority of interrupt requests at the time of initialization of the application system such as initialization of the stack pointer, and do not dynamically change it in execution of the program.
 3. The RUN bit cannot be reset to 0 by software.
 4. The count clock is not reset even when the watchdog timer is cleared by setting the RUN bit to 1.
 5. If a watchdog timer interrupt and NMI interrupt are generated simultaneously when PRC = 1 (INTWDT > NMI), execute the watchdog timer interrupt service routine after executing the first 1 instruction of the NMI interrupt service routine.
Therefore, when used with PRC = 1 setting, use an NOP instruction as the first instruction of the NMI interrupt service routine.
 6. Bits 6, 5 and 0 of the WDM register are fixed to “0” by hardware. Even if “1” is written to them, they remain “0”.
 7. Be sure to write 0 to bit 3 of the WDM register.

12.3 Watchdog Timer Output Pin

The watchdog timer output pin ($\overline{\text{WDTO}}$) is an output circuit of open-drain type and can report a system error to the outside without software intervention. When the watchdog timer overflows, the watchdog timer output goes low for a period of 32 system clocks.

As the $\overline{\text{RESET}}$ pin can be directly connected, the watchdog timer output pin outputs a low-level signal for a period of 32 clocks even when a system reset occurs.

The initial status of the watchdog timer output pin after a system reset is the inactive level (high-level, open).

Cautions 1. Immediately after the power is turned on, the watchdog timer output pin may go low for a period of up to 32 clocks.

2. If the watchdog timer is operating as a timer for the oscillation stabilization time after release of the STOP mode, the $\overline{\text{WDTO}}$ pin remains at the inactive level even if an overflow occurs.

12.4 Application Example

At an early stage in program development, a program is designed without the watchdog timer. After rough debugging is completed, the program is debugged with the watchdog timer.

The watchdog timer must be set in the mode of short overflow time if the response time after a system error is critical.

When the watchdog timer is not required, it can be used for nonmaskable time base interrupt.

CHAPTER 13 INTERRUPT FUNCTION

The μ PD78366A is provided with the following three modes for interrupt request processing (see **Table 13-1**). These three processing modes can arbitrarily be set by a program. However, regarding selection of interrupt service by a macro service, selection is possible only for the interrupt request sources preparing the macro service processing mode shown in Table 13-2. Furthermore, context switching cannot be selected in the case of non-maskable interrupts and operation code trap interrupts.

Table 13-1. Interrupt Request Processing Modes

Interrupt Request Processing Mode	Processing Source	Contents of PC and PSW	Processing Mode
Vectored interrupt	Software	Performs operation of save/restore to stack.	Branches to address specified by vector table and executes interrupt service routine.
Context switching		Performs operation of save/restore to fixed area in register bank.	Automatically switches to register bank specified by vector table, branches to address specified by fixed area in register bank, and executes interrupt service routine.
Macro service	Hardware (firmware)	Retained	Executes preset processing such as data transfer from/to memory and I/O.

Furthermore, for maskable vectored interrupts, it is possible to easily perform multiplexed processing control with 4 levels of priority.

Table 13-2. Interrupt Sources

Interrupt Request Type	Default Priority	Interrupt Source				Macro Service	Macro Service Control Word Address	Vector Table Address	
		Interrupt Request Signal	Interrupt Request Flag	Source	Unit Requesting Interrupt			TPF = 0	TPF = 1
Software	–	–		Op-code trap	–	Not provided	–	003CH	
				Execution of BRK instruction				003EH	
Non-maskable		NMI	–	Input to NMI pin	External			0002H	8002H
		INTWDT	–	Watchdog timer overflow	WDT			0004H	8004H
Maskable	0	INTOV3	OVIF3	Timer 3 overflow	RPU	Provided	FE06H	0006H	8006H
	1	INTP0/INTCC30	PIF0	INTP0 pin input/CC30 coincidence signal	External/RPU		FE08H	0008H	8008H
	2	INTP1	PIF1	INTP1 pin input	External		FE0AH	000AH	800AH
	3	INTP2	PIF2	INTP2 pin input			FE0CH	000CH	800CH
	4	INTP3/INTCC20	PIF3	INTP3 pin input/CC20 coincidence signal	External/RPU		FE0EH	000EH	800EH
	5	INTP4	PIF4	INTP4 pin input	External		FE10H	0010H	8010H
	6	INTTM0	TMIF0	Timer 0 underflow	RPU		FE12H	0012H	8012H
	7	INTCM03	CMIF03	CM03 coincidence signal			FE14H	0014H	8014H
	8	INTCM10	CMIF10	CM10 coincidence signal			FE16H	0016H	8016H
	9	INTCM40	CMIF40	CM40 coincidence signal			FE18H	0018H	8018H
	10	INTCM41	CMIF41	CM41 coincidence signal			FE1AH	001AH	801AH
	11	INTSER	SERIF	Serial reception error	UART		FE1CH	001CH	801CH
	12	INTSR	SRIF	Serial reception end			FE1EH	001EH	801EH
	13	INTST	STIF	Serial transmission end			FE20H	0020H	8020H
	14	INTCSI	CSIIF	Serial transmission/reception end	CSI		FE22H	0022H	8022H
	15	INTAD	ADIF	A/D conversion end	A/D converter		FE24H	0024H	8024H
Reset	–	RESET	–	RESET pin input	–	Not provided	–	0000H	

Remark Default priority: priority fixed by hardware

13.1 Interrupt Requests

Interrupt requests for the μ PD78366A can be classified into the following four types.

- Non-maskable interrupt
- Maskable interrupt
- Software interrupt
- Op-code trap interrupt

Details of each interrupt are given below.

13.1.1 Non-maskable interrupt

A non-maskable interrupt is generated by the NMI pin input or watchdog timer.

A non-maskable interrupt can be acknowledged unconditionally^{Note} even in the interrupt disabled state. It is not subject to priority control and is given the highest priority over all other interrupts.

Note Except when the non-maskable interrupt is being serviced and when another non-maskable interrupt with higher priority is being executed.

13.1.2 Maskable interrupt

A maskable interrupt is subject to mask control by interrupt mask flag setting. Furthermore, acknowledgment enable/disable can be specified for all maskable interrupts by the IE flag of PSW.

A maskable interrupt can be acknowledged not only by a normal vectored interrupt but also by context switching or a macro service (see **Table 13-2**).

For maskable interrupts, priority is given as shown in Table 13-2 when multiple interrupt requests with the same priority are generated simultaneously (default priority). Furthermore, it is possible to classify interrupt priority into a group of four levels to control multiplexed processing. However, macro services are acknowledged irrespective of priority control or the IE flag.

13.1.3 Software interrupt

A software interrupt includes a BRK instruction that generates a vectored interrupt and a BRKCS instruction that performs context switching.

A software interrupt can be acknowledged even in the interrupt disabled state. This interrupt is not subject to interrupt priority control.

13.1.4 Op-code trap interrupt

An op-code trap interrupt request is generated when the writing to the watchdog timer mode register (WDM) and standby control register (STBC) has not been carried out normally (see **12.2 Watchdog Timer Mode Register (WDM)** and **14.2 Standby Control Register (STBC)**).

An op-code trap interrupt can also be acknowledged in the DI state. This is not subject to interrupt priority control.

13.2 Interrupt Servicing Mode

The μ PD78366A services interrupts in the following three modes:

- Vectored interrupt service
- Macro service
- Context switching

13.2.1 Vectored interrupt service

When an interrupt is accepted, the contents of the program counter (PC) and program status word (PSW) are automatically saved to the stack memory, and execution branches to an address indicated by the data stored in the vector table, and an interrupt service routine is executed.

To return from the interrupt service routine, use the RETI instruction.

13.2.2 Macro service

When an interrupt is accepted, execution of the CPU is temporarily stopped, and data is transferred by hardware. Because macro service is executed without the CPU, the CPU statuses such as PC and PSW need not to be saved or restored. Therefore, the service time of the CPU can be substantially improved (refer to **13.8 Macro Service Function**).

13.2.3 Context switching

When an interrupt is accepted, a specific register bank is selected by hardware, and execution branches to a vector address which has been set in advance in a register bank. At the same time, the current contents of the PC and PSW are saved to the register bank (refer to **13.5.2 Context switching** and **13.6.2 Software interrupt (context switching) acknowledgement operation by BRKCS instruction**).

Remark Context means the registers of the CPU that can be accessed from a program that is executed. These registers include the general registers, PC, PSW, and SP (stack pointer).

13.3 Control Registers

Responses to interrupts in the μ PD78366A are controlled according to the interrupt requests. Table 13-3 lists the control registers.

Table 13-3. Control Registers

Register Name	Symbol	Function
Interrupt control registers	OVIC3 PIC0 PIC1 PIC2 PIC3 PIC4 TMIC0 CMIC03 CMIC10 CMIC40 CMIC41 SERIC SRIC STIC CSIIC ADIC	Registers that record occurrence of each interrupt, control mask, specify vectored interrupt service or macro service processing, enable/disable context switching, and specify priority
Interrupt mask flag registers	MK0	Control masking of maskable interrupt. Associated with mask control flag of interrupt control registers. Can be accessed in byte or word units.
In-service priority register	ISPR	Records priority of interrupt request currently accepted.
Interrupt mode control register	IMC	Controls nesting of maskable interrupt specified to have lowest priority level (level 3).

A control register is assigned to each interrupt source. The flags of each register are used to specify various types of controls which are determined by the bit positions in the register.

Table 13-4 lists the flags in the interrupt control registers for the interrupt request signals.

Table 13-4. Interrupt Control Register Flags for Interrupt Request Signals

Default Priority	Interrupt Request Signal	Interrupt Control Register				
		Interrupt Request Flag	Interrupt Mask Flag	Macro Service Enable Flag	Priority Specification Flag	Context Switching Flag
0	INTOV3	OVIF3	OVMK3	OVISM3	OVPR30 OVPR31	OVCSE3
1	INTP0/INTCC30	PIF0	PMK0	PISM0	PPR00 PPR01	PCSE0
2	INTP1	PIF1	PMK1	PISM1	PPR10 PPR11	PCSE1
3	INTP2	PIF2	PMK2	PISM2	PPR20 PPR21	PCSE2
4	INTP3/INTCC20	PIF3	PMK3	PISM3	PPR30 PPR31	PCSE3
5	INTP4	PIF4	PMK4	PISM4	PPR40 PPR41	PCSE4
6	INTTM0	TMIF0	TMMK0	TMISM0	TMPR00 TMPR01	TMCSE0
7	INTCM03	CMIF03	CMMK03	CMISM03	CMPR030 CMPR031	CMCSE03
8	INTCM10	CMIF10	CMMK10	CMISM10	CMPR100 CMPR101	CMCSE10
9	INTCM40	CMIF40	CMMK40	CMISM40	CMPR400 CMPR401	CMCSE40
10	INTCM41	CMIF41	CMMK41	CMISM41	CMPR410 CMPR411	CMCSE41
11	INTSER	SERIF	SERMK	SERISM	SERPR0 SERPR1	SERCSE
12	INTSR	SRIF	SRMK	SRISM	SRPR0 SRPR1	SRCSE
13	INTST	STIF	STMK	STISM	STPR0 STPR1	STCSE
14	INTCSI	CSIIF	CSIMK	CSIISM	CSIPR0 CSIPR1	CSICSE
15	INTAD	ADIF	ADMK	ADISM	ADPR0 ADPR1	ADCSE

13.3.1 Interrupt control registers

The interrupt control registers are assigned to different interrupt sources. Each register is used to control priority and masking for its associated interrupt source. Figure 13-1 shows the format of the interrupt control registers.

(1) Priority specification flags (xxPR1, xxPR0)

The priority specification flags specify the priority of the associated interrupt source for the 16 maskable interrupts.

Four priority levels can be specified. More than one interrupt source can have the same priority level. Maskable interrupt sources with level 0 have the highest priority.

If more than one interrupt request is generated and the interrupt sources have the same priority level, the requests are accepted according to the default priority.

The flags are set or reset bit by bit by software.

$\overline{\text{RESET}}$ input sets all bits to 1.

(2) Context switching enable flag (xxCSE)

The context switching enable flag specifies whether to respond to a maskable interrupt request with context switching.

The context switching function selects the previously specified register bank on a hardware basis, makes a branch to the vector address stored in the register bank, and also saves the contents of the current program counter (PC) and program status word (PSW) in the register bank.

Context switching can start interrupt service at a higher speed than normal vectored interrupt service. Context switching is therefore appropriate for real-time service.

The flag is set or reset bit by bit by software.

$\overline{\text{RESET}}$ input sets all bits to 0.

(3) Macro service enable flag (xxISM)

The macro service enable flag specifies whether to respond to the associated interrupt request with vectored interrupt service or macro service processing.

If macro service processing is selected, when macro service terminates (when the macro service counter overflows) and a vectored interrupt is generated, the flag is automatically reset to 0 by hardware (vectored interrupt service).

The flag can be set or reset bit by bit by software.

$\overline{\text{RESET}}$ sets all bits to 0.

(4) Interrupt mask flag (××MK)

The interrupt mask flag enables or disables vectored interrupt service or macro service processing for the associated interrupt request.

The interrupt mask flag is not changed by the activation of interrupt service. The contents of the interrupt mask flag and the contents of the interrupt mask flag register are the same (refer to **13.3.2 Interrupt mask flag registers (MK0)**).

As the macro service request is subject to mask control, the macro service request can also be masked by this flag.

This flag can be set or reset by software.

$\overline{\text{RESET}}$ input sets all bits to 1.

Selected interrupt service mode is determined by setting the interrupt mask flag (××MK) and macro service enable flag (××ISM) in combination.

××MK	××ISM	Interrupt Service Mode
0	0	Vectored interrupt
0	1	Vectored interrupt after macro service processing
1	×	Maskable interrupt request is not acknowledged

(5) Interrupt request flag (××IF)

The interrupt request flag is set to 1 when the associated interrupt request is generated. When the interrupt is accepted, the flag is automatically reset to 0 by hardware.

This flag can be set or reset by software.

$\overline{\text{RESET}}$ input sets all bits to 0.

Figure 13-1. Format of Interrupt Control Registers (1/2)

Symbol	⑦	⑥	⑤	④	3	2	①	①	Address	When reset	R/W
OVIC3	OVIF3	OVMK3	OVISM3	OVCSE3	0	0	OVPR31	OVPR30	FFE0H	43H	R/W
PIC0	PIF0	PMK0	PISM0	PCSE0	0	0	PPR01	PPR00	FFE1H	43H	R/W
PIC1	PIF1	PMK1	PISM1	PCSE1	0	0	PPR11	PPR10	FFE2H	43H	R/W
PIC2	PIF2	PMK2	PISM2	PCSE2	0	0	PPR21	PPR20	FFE3H	43H	R/W
PIC3	PIF3	PMK3	PISM3	PCSE3	0	0	PPR31	PPR30	FFE4H	43H	R/W
PIC4	PIF4	PMK4	PISM4	PCSE4	0	0	PPR41	PPR40	FFE5H	43H	R/W
TMIC0	TMIF0	TMMK0	TMISM0	TMCSE0	0	0	TMPR01	TMPR00	FFE6H	43H	R/W
CMIC03	CMIF03	CMMK03	CMISM03	CMCSE03	0	0	CMPR031	CMPR030	FFE7H	43H	R/W

xxPR1	xxPR0	Specifies priority
0	0	Priority 0 (highest)
0	1	Priority 1
1	0	Priority 2
1	1	Priority 3

xxCSE	Context switching enable flag
0	Served by vectored interrupt
1	Served by context switching

xxISM	Macro service enable flag
0	Served by vectored interrupt
1	Served by macro service

xxMK	Enables/disables interrupt request
0	Enables interrupt service
1	Disables interrupt service

xxIF	Interrupt request flag
0	No interrupt request. Interrupt request signal is not generated.
1	Interrupt request signal is generated and interrupt is requested.

Caution Bits 3 and 2 of the interrupt control register are fixed to “0” by hardware. Even if “1” is written to them, they remain “0”.

Figure 13-1. Format of Interrupt Control Registers (2/2)

Symbol	⑦	⑥	⑤	④	3	2	①	①	Address	When reset	R/W
CMIC10	CMIF10	CMMK ₁₀	CMISM ₁₀	CMCSE ₁₀	0	0	CMPR ₁₀₁	CMPR ₁₀₀	FFE8H	43H	R/W
CMIC40	CMIF40	CMMK ₄₀	CMISM ₄₀	CMCSE ₄₀	0	0	CMPR ₄₀₁	CMPR ₄₀₀	FFE9H	43H	R/W
CMIC41	CMIF41	CMMK ₄₁	CMISM ₄₁	CMCSE ₄₁	0	0	CMPR ₄₁₁	CMPR ₄₁₀	FFEAH	43H	R/W
SERIC	SERIF	SERMK	SERISM	SERCSE	0	0	SERPR1	SERPR0	FFEBH	43H	R/W
SRIC	SRIF	SRMK	SRISM	SRCSE	0	0	SRPR1	SRPR0	FFECH	43H	R/W
STIC	STIF	STMK	STISM	STCSE	0	0	STPR1	STPR0	FFEDH	43H	R/W
CSIIC	CSIIF	CSIMK	CSIISM	CSICSE	0	0	CSIPR1	CSIPR0	FFEEH	43H	R/W
ADIC	ADIF	ADMK	ADISM	ADCSE	0	0	ADPR1	ADPR0	FFEFH	43H	R/W

xxPR1	xxPR0	Specifies priority
0	0	Priority 0 (highest)
0	1	Priority 1
1	0	Priority 2
1	1	Priority 3

xxCSE	Context switching enable flag
0	Serviced by vectored interrupt
1	Serviced by context switching

xxISM	Macro service enable flag
0	Serviced by vectored interrupt
1	Serviced by macro service

xxMK	Enables/disables interrupt request
0	Enables interrupt service
1	Disables interrupt service

xxIF	Interrupt request flag
0	No interrupt request. Interrupt request signal is not generated.
1	Interrupt request signal is generated and interrupt is requested.

Caution Bits 3 and 2 of the interrupt control register are fixed to “0” by hardware. Even if “1” is written to them, they remain “0”.

13.3.2 Interrupt mask flag registers (MK0)

The interrupt mask flag register (MK0) is comprised of interrupt mask flags corresponding to 16 types of maskable interrupt requests.

The MK0 register is a 16-bit register and can be manipulated not only in 16-bit units but also in 8-bit units as MK0L and MK0H. In addition, each bit of the MK0 register can be manipulated in 1-bit units by a bit manipulation instruction. Each interrupt mask flag controls enable/disable 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 reset (0), acknowledgment of the corresponding interrupt request is enabled as a vectored interrupt or macro service.

Each interrupt mask flag in the MK0 register is the same as the interrupt mask flag in the interrupt control register. The MK0 register is provided to perform mask-related control for all interrupts at a time.

$\overline{\text{RESET}}$ input sets each interrupt mask flag to "1" disabling all maskable interrupts.

The interrupt mask flag register format is shown in Figure 13-2.

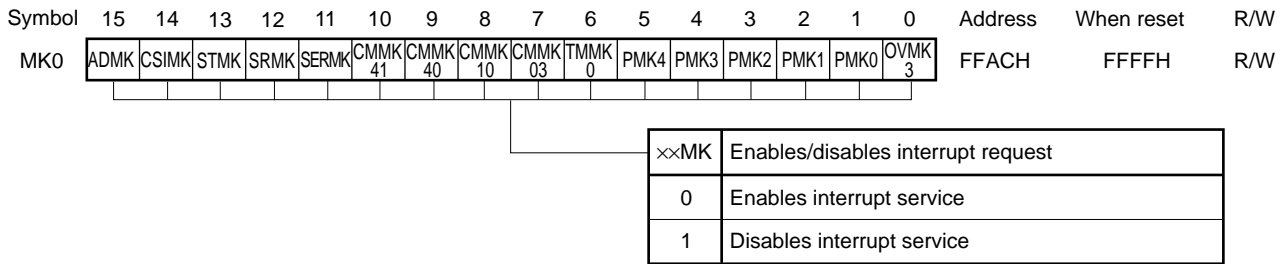
Figure 13-2. Format of Interrupt Mask Flag Register (1/2)

(Byte access)

Symbol	7	6	5	4	3	2	1	0	Address	When reset	R/W
MK0L	CMMK03	TMMK0	PMK4	PMK3	PMK2	PMK1	PMK0	OVMK3	FFACH	FFH	R/W
MK0H	ADMK	CSIMK	STMK	SRMK	SERMK	CMMK41	CMMK40	CMMK10	FFADH	FFH	R/W
									xxMK	Enables/disables interrupt request	
									0	Enables interrupt service	
									1	Disables interrupt service	

Figure 13-2. Format of Interrupt Mask Flag Register (2/2)

(Word access)



Relations between each bit and interrupt source

Bit Position	Bit Name	Interrupt Request Source
Bit 0	OVMK3	Overflow of timer 3 (INTOV3)
Bit 1	PMK0	INTP0 pin input/CC30 capture signal (INTP0) Coincidence of CC30 (INTCC30)
Bit 2	PMK1	INTP1 pin input/CT30 capture signal (INTP1)
Bit 3	PMK2	INTP2 pin input (INTP2)
Bit 4	PMK3	INTP3 pin input/CC20 capture signal (INTP3) Coincidence of CC20 (INTCC20)
Bit 5	PMK4	INTP4 pin input/CT31 capture signal (INTP4)
Bit 6	TMMK0	Underflow of timer 0 (INTTM0)
Bit 7	CMMK03	Coincidence of CM03 (INTCM03)
Bit 8	CMMK10	Coincidence of CM10 (INTCM10)
Bit 9	CMMK40	Coincidence of CM40 (INTCM40)
Bit 10	CMMK41	Coincidence of CM41 (INTCM41)
Bit 11	SERMK	Serial error interrupt (INTSER)
Bit 12	SRMK	Serial reception end interrupt (INTSR)
Bit 13	STMK	Serial transmission end interrupt (INTST)
Bit 14	CSIMK	Serial transmission/reception end interrupt (INTCSI)
Bit 15	ADMK	A/D conversion end interrupt (INTAD)

13.3.3 Interrupt mode control register (IMC)

The interrupt mode control register (IMC) has a PRSL flag. The PRSL flag enables or disables nesting of maskable interrupts for which the lowest priority (level 3) is specified.

Interrupts having the same level if the level is 0, 1, or 2 cannot be nested regardless of the setting of the IMC register.

To manipulate the IMC register, enter the interrupt disable status (DI status) to prevent an error.

The register can be set or reset by software.

$\overline{\text{RESET}}$ input sets the PRSL flag to 1.

Figure 13-3 shows the format of the IMC register.

Figure 13-3. Format of Interrupt Mode Control Register

Symbol	⑦	6	5	4	3	2	1	0	Address	When reset	R/W
IMC	PRSL	0	0	0	0	0	0	0	FFAAH	80H	R/W
									PRSL	Nesting control	
									0	Enables nesting between interrupts for which level 3 (lowest level) is set.	
									1	Disables nesting between interrupts for which level 3 (lowest level) is set.	

Caution Bits 6 to 0 of the IMC register are fixed to 0 by hardware. Even if “1” is written to them, they remain “0”.

13.3.4 In-service priority register (ISPR)

The in-service priority register (ISPR) holds the priority level of the interrupt request being serviced. When the interrupt request is accepted, the bit corresponding to the priority level for the request is set to 1, and the level is held during service.

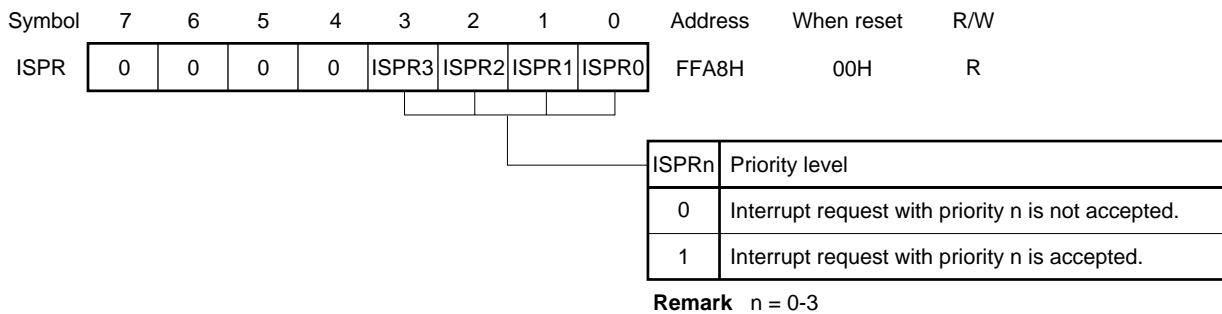
When the RETI or RETCS instruction is executed, the bit for an interrupt request with the highest priority among the bits that are set to 1 in the ISPR register is reset to 0 automatically by hardware.

The ISPR register contents are not changed by the execution of the RETB or RETCSB instruction.

RESET input sets the register to 00H.

Figure 13-4 shows the format of the ISPR register.

Figure 13-4. Format of In-service Priority Register



Cautions 1. The ISPR register can be accessed for read only. Write to the register may cause an error.

2. Bits 7 to 4 of the ISPR register are fixed to “0” by hardware. Even if “1” is written to them, they remain “0”.

13.3.5 Program status word (PSW)

PSW is a register to retain the execution result of an instruction and the current state of interrupt requests. The IE flag that sets enable/disable of a maskable interrupt is mapped onto the lower 8 bits of PSW (PSWL).

PSWL can not only be read/written in 8-bit units but also can be manipulated by a bit manipulation instruction or a dedicated instruction (EI, DI).

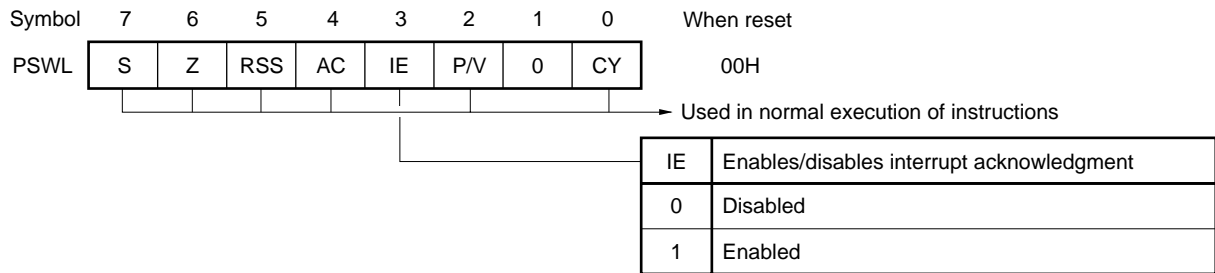
When a vectored interrupt is acknowledged or the BRK instruction is executed, it is saved to the stack and the IE flag is reset (0). It is also saved to the stack by the PUSH PSW instruction. It is restored from the stack by the RETI, RETB or POP PSW instruction.

When the context switching or BRKCS instruction is executed, it is saved to a fixed area of the register bank and the IE flag is reset (0). It is restored from the fixed area in the register bank by the RETCS or RETCSB instructions.

A request for a non-maskable interrupt or macro service is acknowledged irrespective of the IE flag. In the case of a macro service, the IE flag contents are unchanged.

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

Figure 13-5. Program Status Word (PSWL) Format



13.4 Non-Maskable Interrupt Acknowledgment Operation

A non-maskable interrupt is acknowledged even in the interrupt disable state. A non-maskable interrupt is always acknowledged except when the non-maskable interrupt or a non-maskable interrupt with higher priority is being serviced.

The priority among non-maskable interrupts is set by the PRC bit of the watchdog timer mode register (WDM) (see **12.2 Watchdog Timer Mode Register (WDM)**).

A non-maskable interrupt request is acknowledged immediately except for the states described in **13.9 Cases where Interrupt Request and Macro Service Are Temporarily Held Pending**. After a non-maskable interrupt request is acknowledged, PSW and PC are saved to the stack in that order, the IE flag of PSW is reset (0), the vector table contents are loaded to PC, and a branch is made.

When a non-maskable interrupt is being serviced, a request for the same non-maskable interrupt as that being serviced or a request for a non-maskable interrupt with priority lower than that being serviced is held pending. The pending non-maskable interrupt is acknowledged after the service of the non-maskable interrupt is completed (after execution of the RETI instruction). However, if the same non-maskable interrupt request is generated twice or more while a non-maskable interrupt is being serviced, only one non-maskable interrupt is acknowledged after completion of the non-maskable interrupt service.

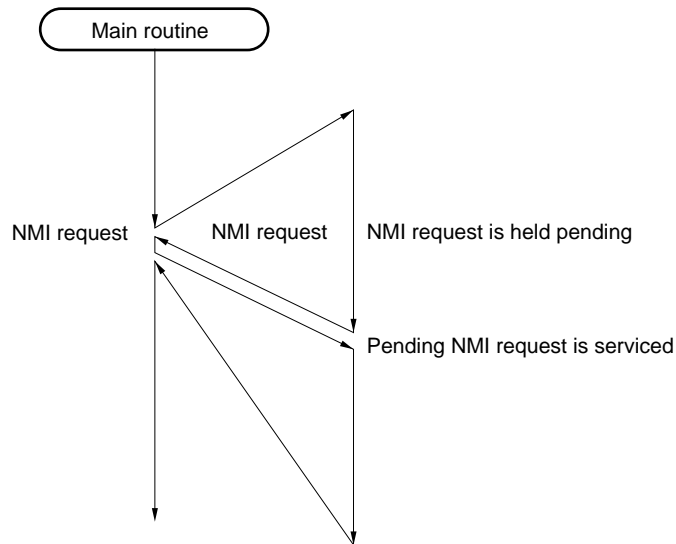
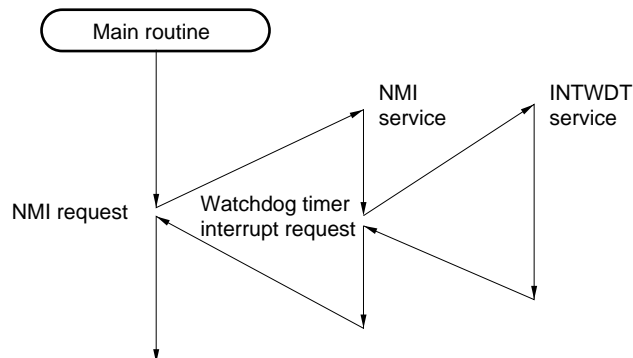
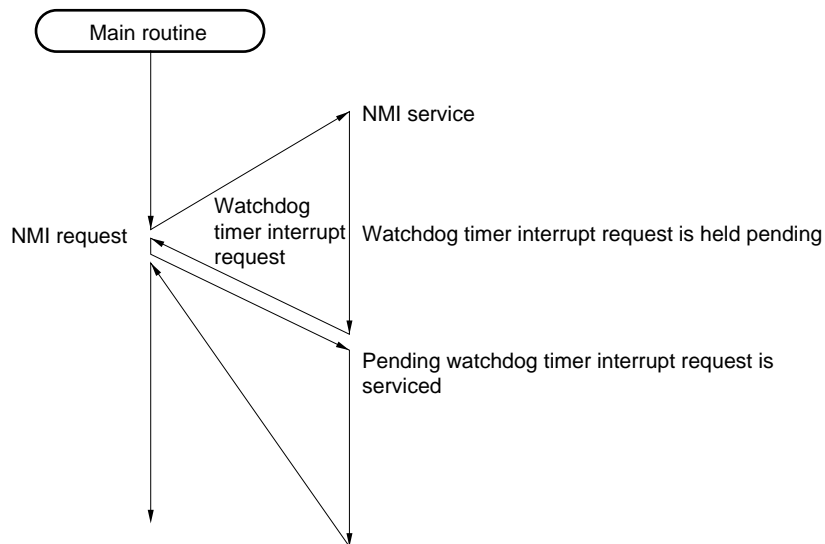
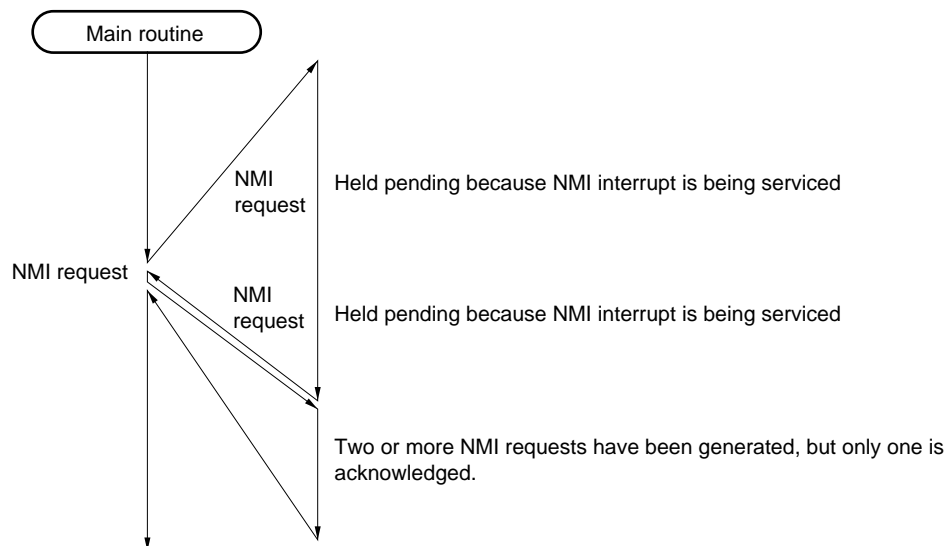
Figure 13-6. Non-Maskable Interrupt Request Acknowledgment Operation (1/2)**(a) When a new NMI request is generated while NMI interrupt is being serviced****(b) When watchdog timer interrupt request is generated while NMI interrupt is being serviced (when watchdog timer interrupt has higher priority (PRC bit of WDM register = 1))**

Figure 13-6. Non-Maskable Interrupt Request Acknowledgment Operation (2/2)

- (c) When watchdog timer interrupt request is generated while NMI interrupt is being serviced (when NMI interrupt has higher priority (PRC bit of WDM register = 0))



- (d) When two additional NMI requests are generated while NMI interrupt is being serviced



- Cautions**
1. A macro service request is acknowledged and processed even in a non-maskable interrupt service routine. If it is desirable not to execute a macro service in the non-maskable interrupt service routine, manipulate the interrupt mask flag register during the non-maskable interrupt service routine to prevent any macro service from being generated.
 2. Be sure to use the RETI instruction for restoring from the non-maskable interrupt. Acknowledgment of subsequent interrupts otherwise would not be carried out normally by other instructions.
 3. A non-maskable interrupt always is acknowledged except when another non-maskable interrupt is being serviced (unless a non-maskable interrupt request with higher priority is generated while a non-maskable interrupt with lower priority is being serviced) and except for a certain period after execution of a specific instruction shown in 13.9. Therefore, a non-maskable interrupt is also acknowledged when the stack pointer value is undefined, especially after release of a reset. In this case, depending on the stack pointer value, the program counter (PC) or program status word (PSW) may be written to the address where writing to the special function register is disabled (see Table 3-4 in 3.2.3 Special function register (SFR)). This will cause the CPU to deadlock, or an unexpected signal to be output from the pin or PC or PSW to be written to the address where no RAM is mounted, in which case the program cannot return normally from the non-maskable interrupt service routine to the main routine and it goes into an inadvertent loop.

Therefore, be sure to program as follows after release of $\overline{\text{RESET}}$.

```
CSEG AT 0
DW   STRT
STRT:
MOVW SP, #imm16
```

13.5 Maskable Interrupt Acknowledgment Operation

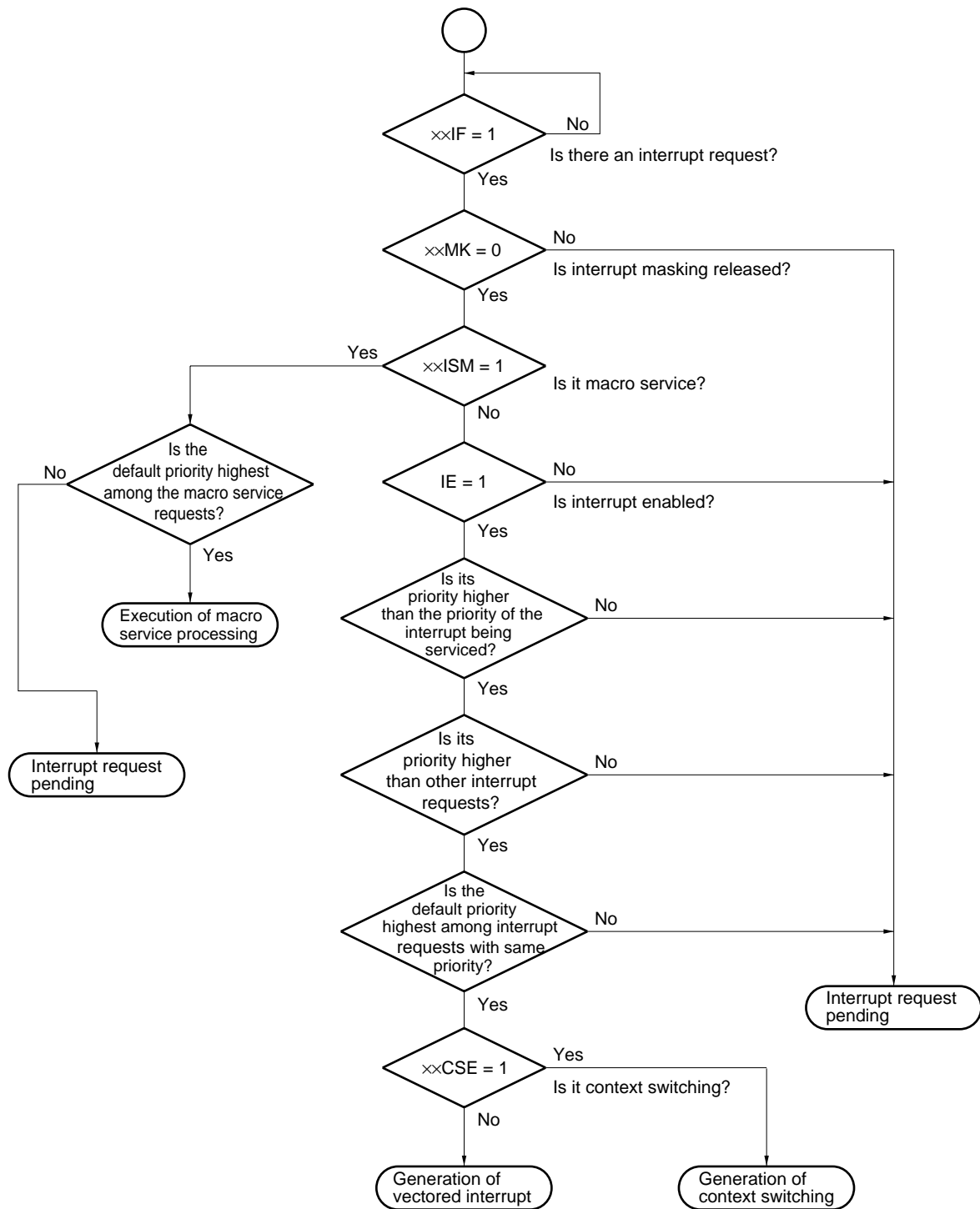
Acknowledgment of a maskable interrupt is enabled when the interrupt request flag is set (1) and the mask flag for the interrupt is reset (0). When servicing is carried out by a macro service, a maskable interrupt is acknowledged immediately and serviced by the macro service. In the case of a vectored interrupt or context switching, a maskable interrupt is acknowledged if interrupts are enabled (when the IE flag is set (1)) and if the interrupt priority is acknowledgeable-priority.

If multiple maskable interrupt requests are generated simultaneously, the one with high priority specified by the priority specification flag is acknowledged first, followed by others in the order of priority. If the same priority is specified to multiple maskable interrupts, acknowledgment of interrupt requests accords to the default priority.

The pending interrupts are acknowledged when acknowledgment capability is enabled.

The interrupt acknowledgment algorithm is shown in Figure 13-7.

Figure 13-7. Interrupt Acknowledgment Service Algorithm



13.5.1 Vectored interrupt

If a maskable interrupt request by a vectored interrupt is acknowledged, PSW and PC are saved to the stack in that order, the IE flag is reset (0) (interrupt disabled state), and the bit of the ISPR register corresponding to the priority of the acknowledged interrupt is set (1). Furthermore, data in the vector table specified for each interrupt request is loaded to PC and a branch is made. The restoration from the vectored interrupt is carried out by the RETI instruction.

Caution If a maskable interrupt is acknowledged by a vectored interrupt, be sure to restore the program by the RETI instruction. Operation of subsequent interrupts otherwise would not be carried out normally by other instructions.

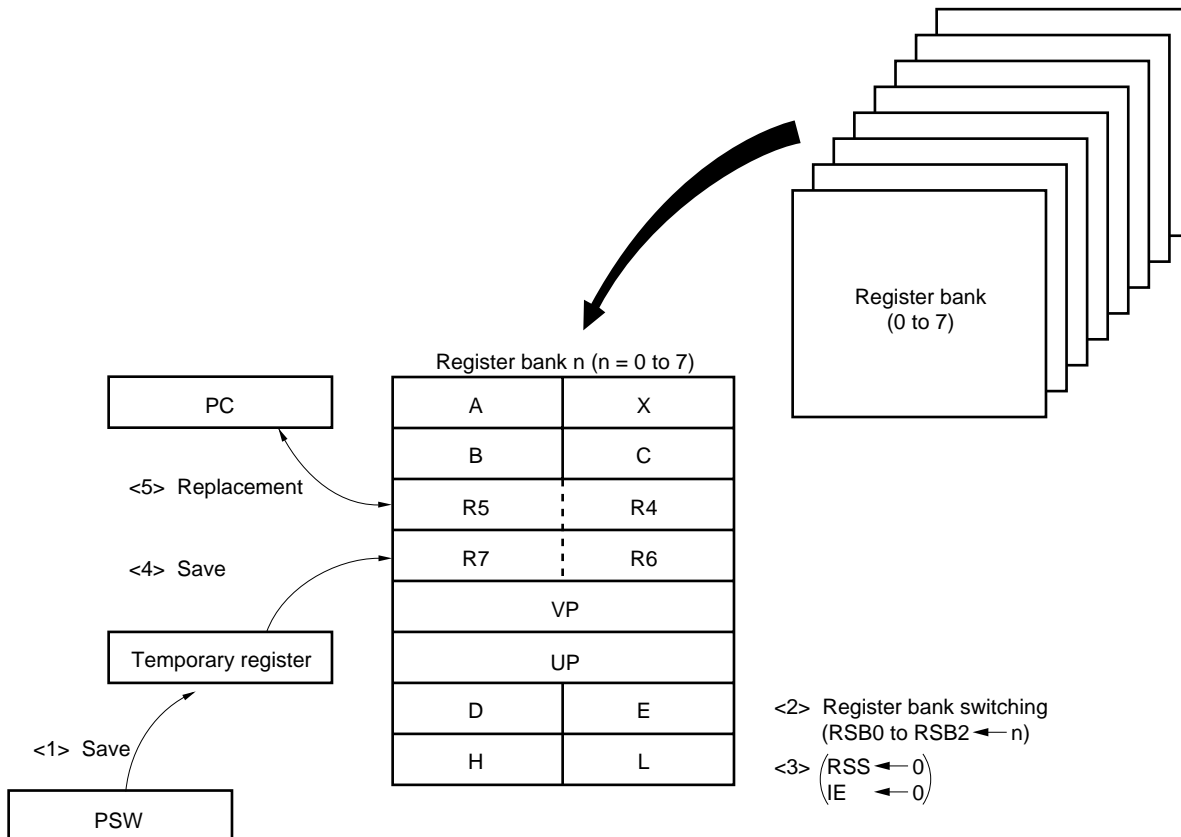
13.5.2 Context switching

Setting (1) the context switching enable flag of the interrupt control register (see **Table 13-4** and **Figure 13-1**) enables the context switching function to start.

If an interrupt request that is not masked in the EI state and whose context switching function is enabled is generated, a register bank specified by the lower 3 bits of the lower address (even address) among the corresponding vector table addresses is selected.

The vector address that has been stored beforehand in the selected register bank is transferred to PC and simultaneously the contents of PC and PSW at that time are saved to the register bank and a branch is made to the interrupt service routine.

Figure 13-8. Context Switching Operation by Generation of Interrupt Request

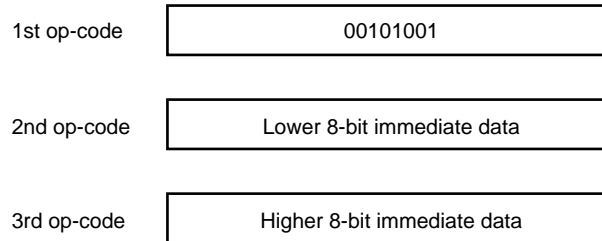


Restoration from an interrupt using the context switching function is performed by executing the RETCS instruction or RETCSB instruction.

By executing the RETCS instruction the contents of the R4 and R5 registers and the contents of the R6 and R7 registers in the selected register bank are transferred to PC and PSW, respectively at this time. And at the same time, the 16-bit immediate data specified by the 2nd and 3rd op-codes of the RETCS instruction is stored in the R4 and R5 registers in the register bank.

Therefore, if the same register bank is selected again by the context switching function, the 16-bit immediate data specified by the 2nd and 3rd op-codes of the RETCS instruction becomes the branch address.

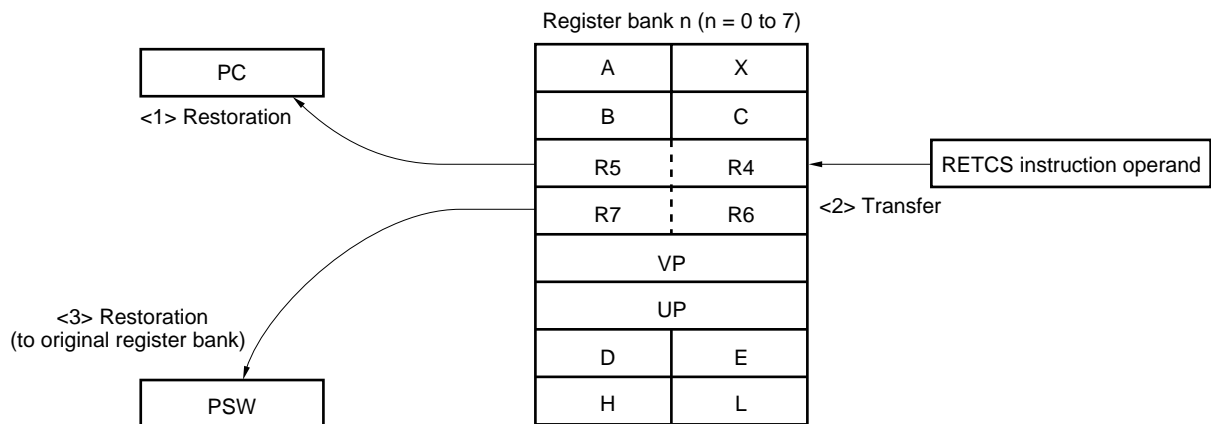
Figure 13-9. RETCS Instruction Format



When returning from the branch processing by executing the RETCS instruction, the bit set (1) in the in-service priority register (ISPR) which corresponds to the highest priority is reset (0).

Caution Be sure to use the RETCS instruction for returning from the interrupt by context switching. Operation of subsequent interrupts otherwise would not be carried out normally by other instructions.

Figure 13-10. Restoration Operation from Interrupt Using Context Switching Function by RETCS Instruction



13.5.3 Maskable interrupt priority

The μ PD78366A handles multiplexed interrupt service by acknowledging an interrupt while servicing another interrupt. Multiplexed interrupt can be controlled by priority.

Priority control includes control by the default priority as well as programmable priority control by setting the priority specification flag. In priority control by the default priority, if multiple interrupts are generated simultaneously, interrupts are serviced according to the priority assigned to each interrupt request beforehand (default priority) (see **Table 13-2**). In programmable priority control, interrupt requests are classified into four levels by setting the priority specification flag. The interrupt requests to which multiplexed interrupt service is applicable are shown in Table 13-5.

When an interrupt is acknowledged, the IE flag is automatically reset (0). Therefore, when using multiplexed interrupt service, execute the EI instruction in the interrupt service routine, etc., to set (1) the IE flag so as to set the interrupt enabled state.

Table 13-5. Multiplexed Interrupt Service

Priority of Interrupt being Acknowledged	ISPR Value	IE Flag of PSW	PRSL Flag of IMC Register	Acknowledgeable Maskable Interrupt
No interrupt being acknowledged	0 0 0 0 0 0 0 0	0	×	• All macro services only
		1	×	• All maskable interrupts only
3	0 0 0 0 1 0 0 0	0	×	• All macro services only
		1	0	• All maskable interrupts only
		1	1	• All macro services • Maskable interrupts with priority specified to 0, 1 and 2
2	0 0 0 0 × 1 0 0	0	×	• All macro services only
		1	×	• All macro services • Maskable interrupts with priority specified to 0 and 1
1	0 0 0 0 × × 1 0	0	×	• All macro services only
		1	×	• All macro services • Maskable interrupts with priority specified to 0
0	0 0 0 0 × × × 1	×	×	• All macro services only

Figure 13-11. Examples of Servicing an Interrupt Request Generated while Another Interrupt is being Serviced (1/3)

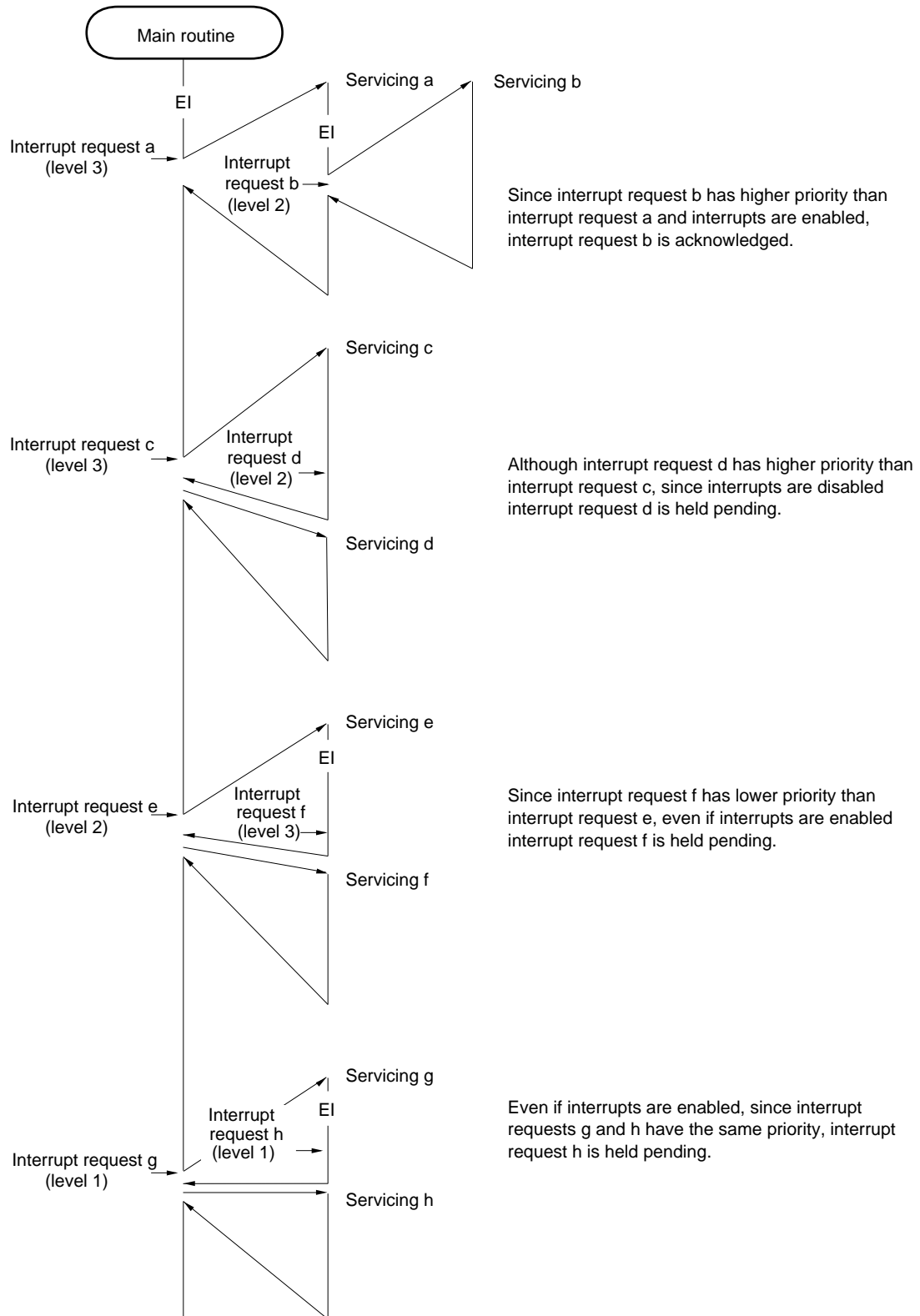
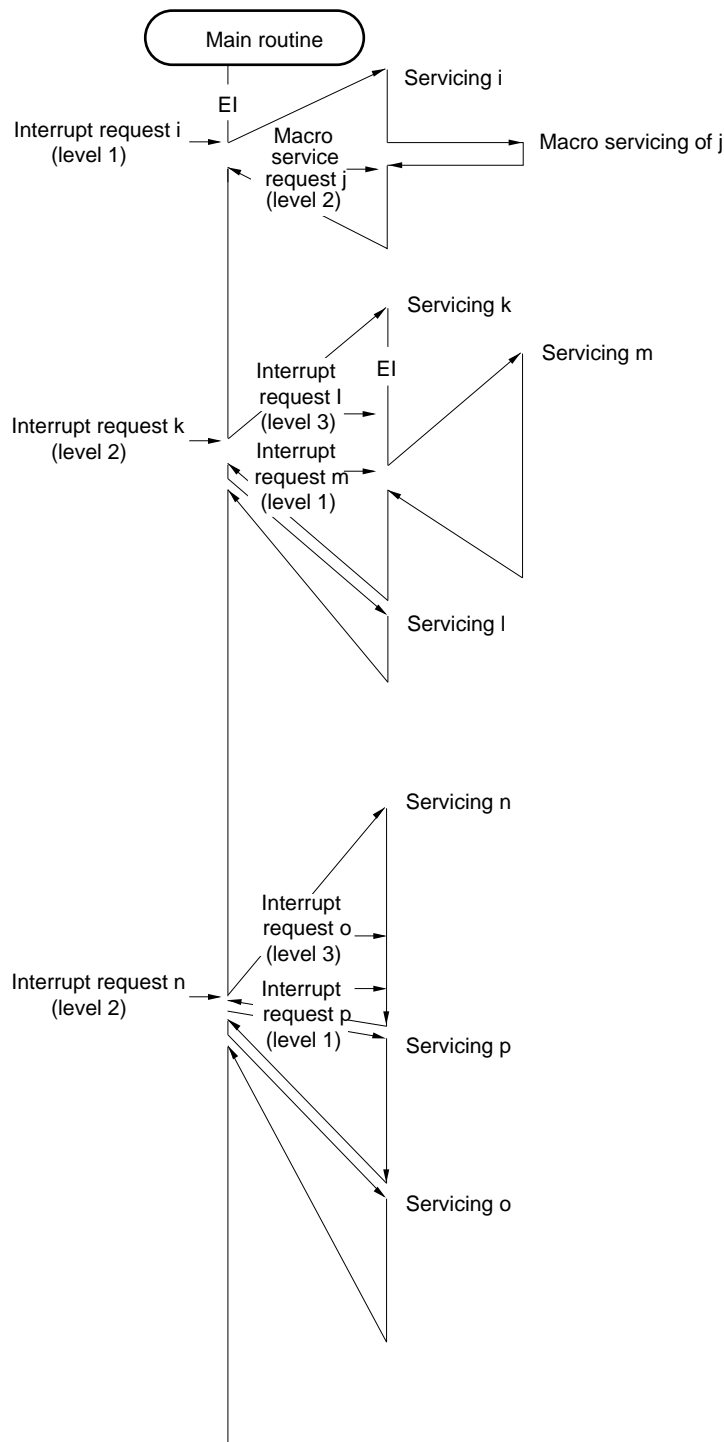


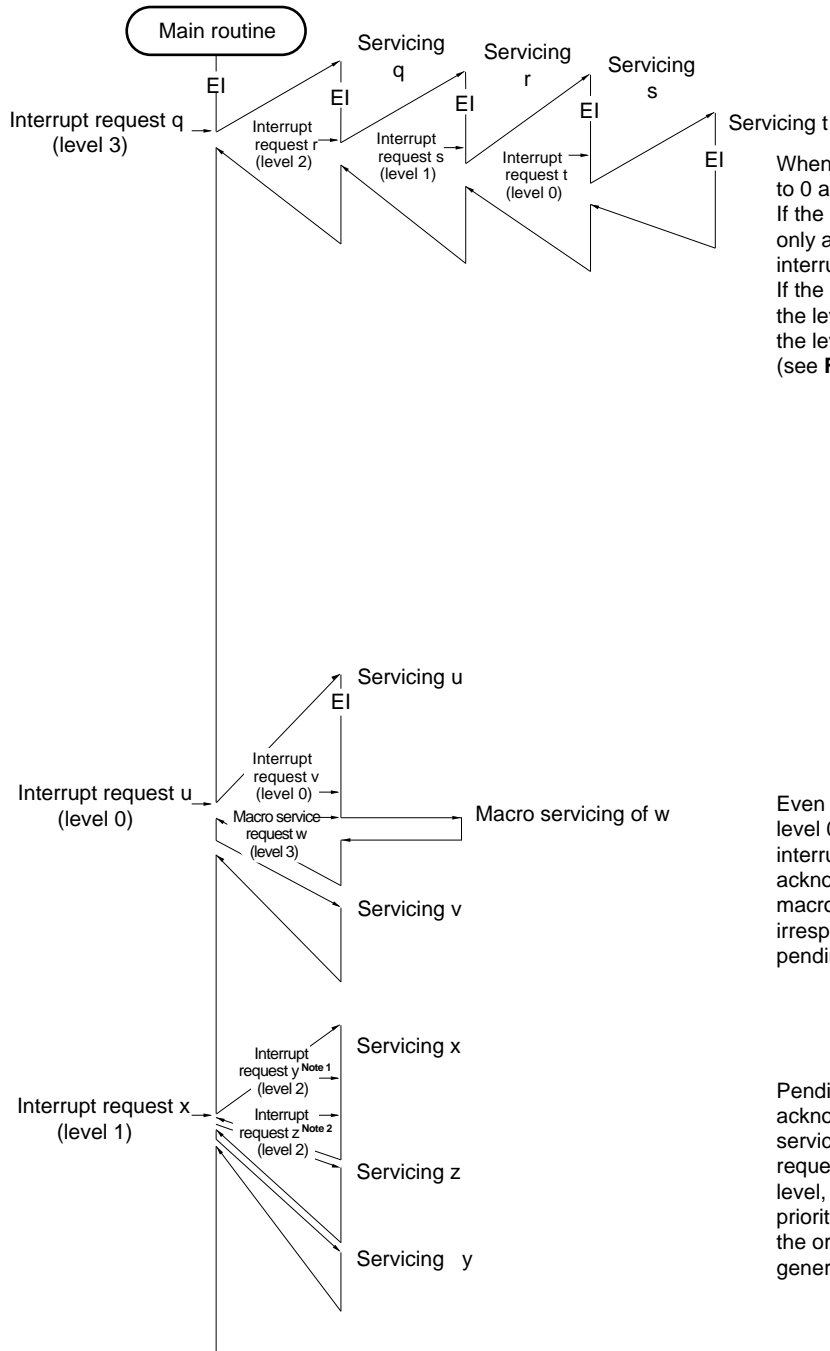
Figure 13-11. Examples of Servicing an Interrupt Request Generated while Another Interrupt is being Serviced (2/3)



Macro service request is serviced irrespective of interrupt enable/disable or priority of interrupts.

Interrupt request l with lower priority than interrupt request k is held pending. Interrupt request m, which is generated after l, is acknowledged first because it has higher priority.

Since interrupt request n is serviced in the interrupt disabled state, interrupt requests o and p are held pending. After servicing of interrupt request n, pending interrupt requests are acknowledged. In this case, although interrupt request o has been generated first, interrupt request p with higher priority is acknowledged first.

Figure 13-11. Examples of Servicing an Interrupt Request Generated while Another Interrupt is being Serviced (3/3)

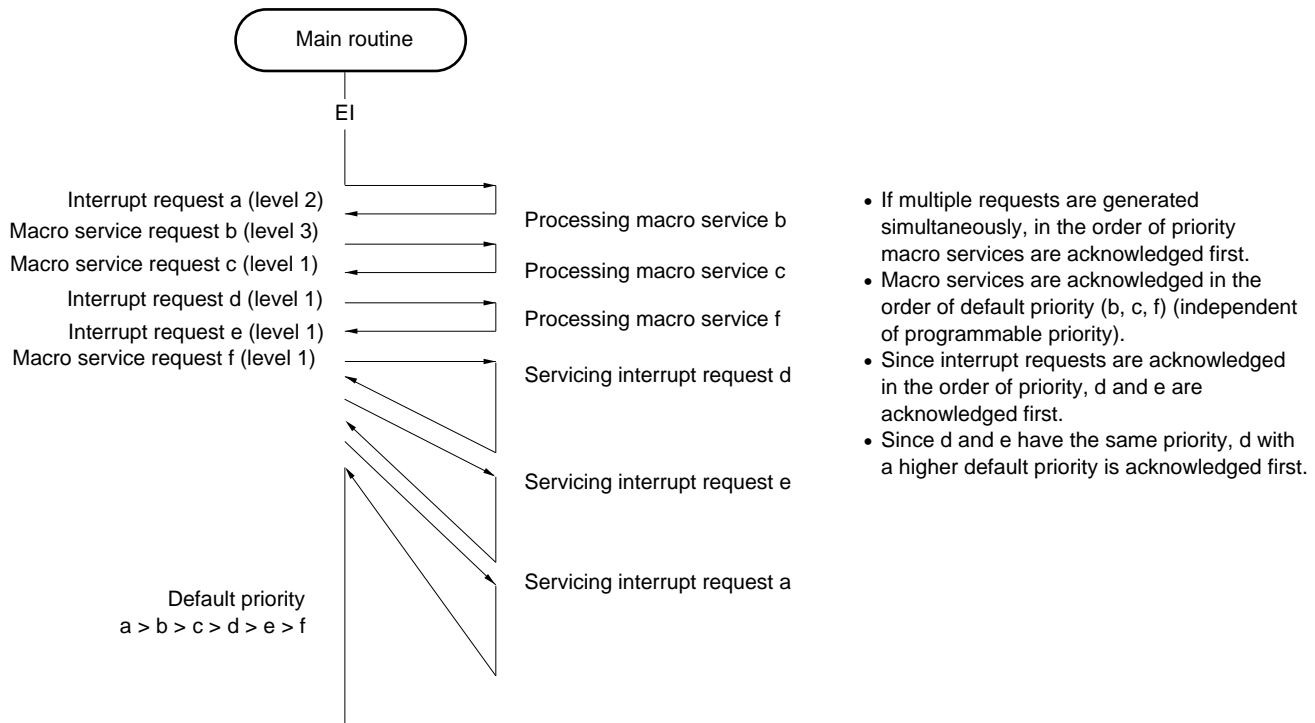
When multiple interrupts with priority levels 3 to 0 are acknowledged. If the PRSL bit of the IMC register is set (1), only a macro service request and non-maskable interrupt will generate further nesting. If the PRSL bit of the IMC register is cleared (0), the level 3 interrupt can also be nested while the level 3 interrupt is being serviced (see **Figure 13-13**).

Even if the interrupt enabled state is set while level 0 interrupt request u is being serviced an interrupt request whose priority level is 0 is not acknowledged but held pending. However, a macro service is acknowledged and serviced irrespective of the level or even if there is a pending interrupt request with higher priority.

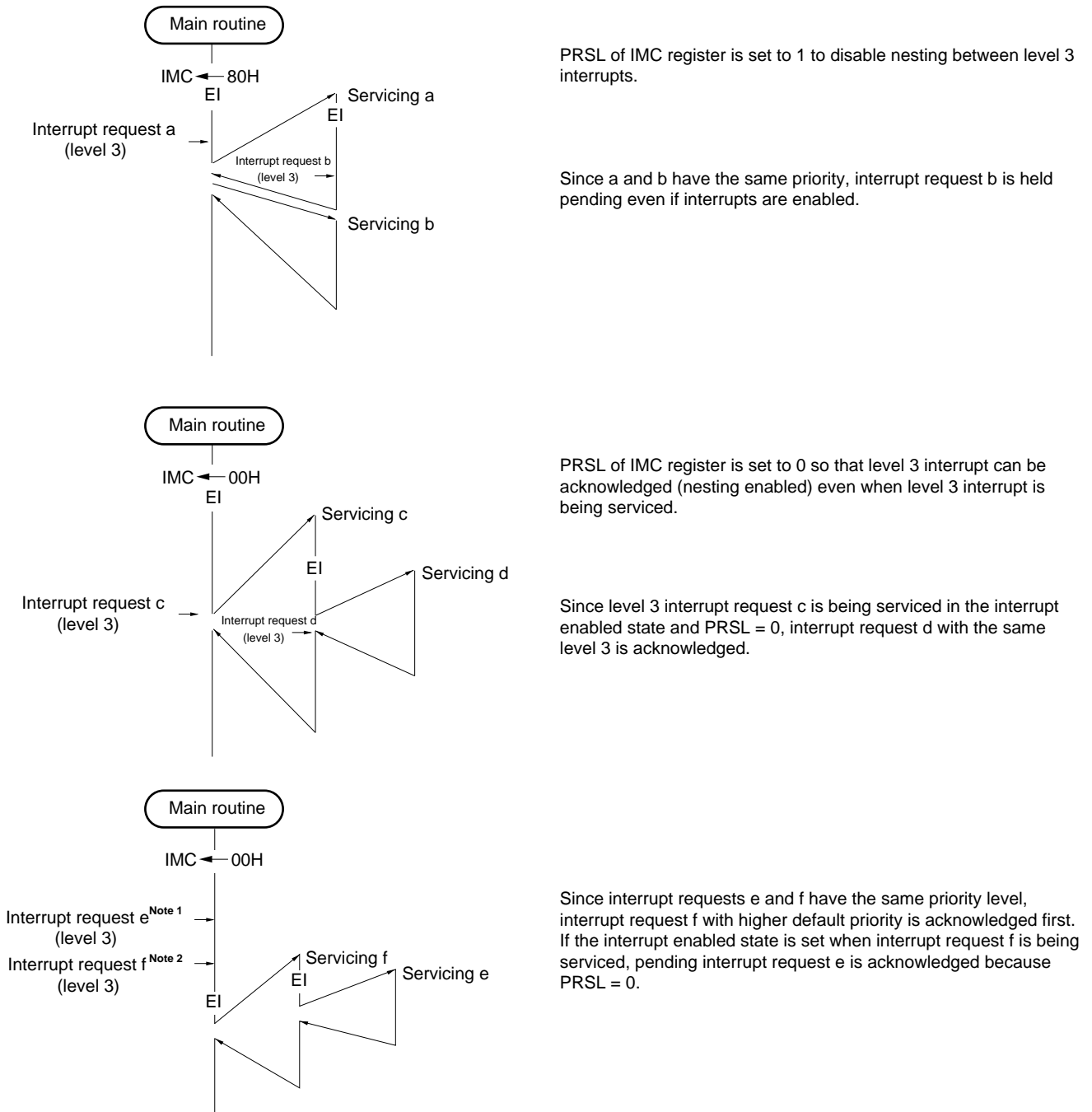
Pending interrupt requests y and z are acknowledged after interrupt request x is serviced. In this case, since interrupt requests y and z have the same priority level, interrupt request z with a higher default priority is acknowledged first irrespective of the order in which the interrupt requests are generated.

Notes 1. Lower default priority
2. Higher default priority

Remarks 1. a to z in the figures are fictitious names to distinguish interrupt requests and macro service requests.
2. Higher/lower default priority attributes in the figures represent relative priority levels between two interrupt requests.

Figure 13-12. Example of Servicing Interrupt Requests Generated Simultaneously

Remark a to f in the figure are fictitious names to distinguish interrupt requests and macro services.

Figure 13-13. Differences in Level 3 Interrupt Acknowledgment Operation by Setting IMC Register

Notes 1. Lower default priority
2. Higher default priority

Remarks 1. a to z in the figure are fictitious names to distinguish interrupt requests.
2. Higher/lower default priority levels in the figure represent relative priority levels between two interrupt requests.

13.6 Software Interrupt Acknowledgment Operation

Software interrupts are acknowledged by executing the BRK and BRKCS instructions. Software interrupts cannot be disabled.

13.6.1 Software interrupt acknowledgment operation by BRK instruction

When the BRK instruction is executed, PSW and PC are saved to the stack in that order, the IE flag is reset (0), the contents of the vector table (003EH, 003FH) are loaded to PC, and a branch is made.

To return from a software interrupt by the BRK instruction, the RETB instruction is used.

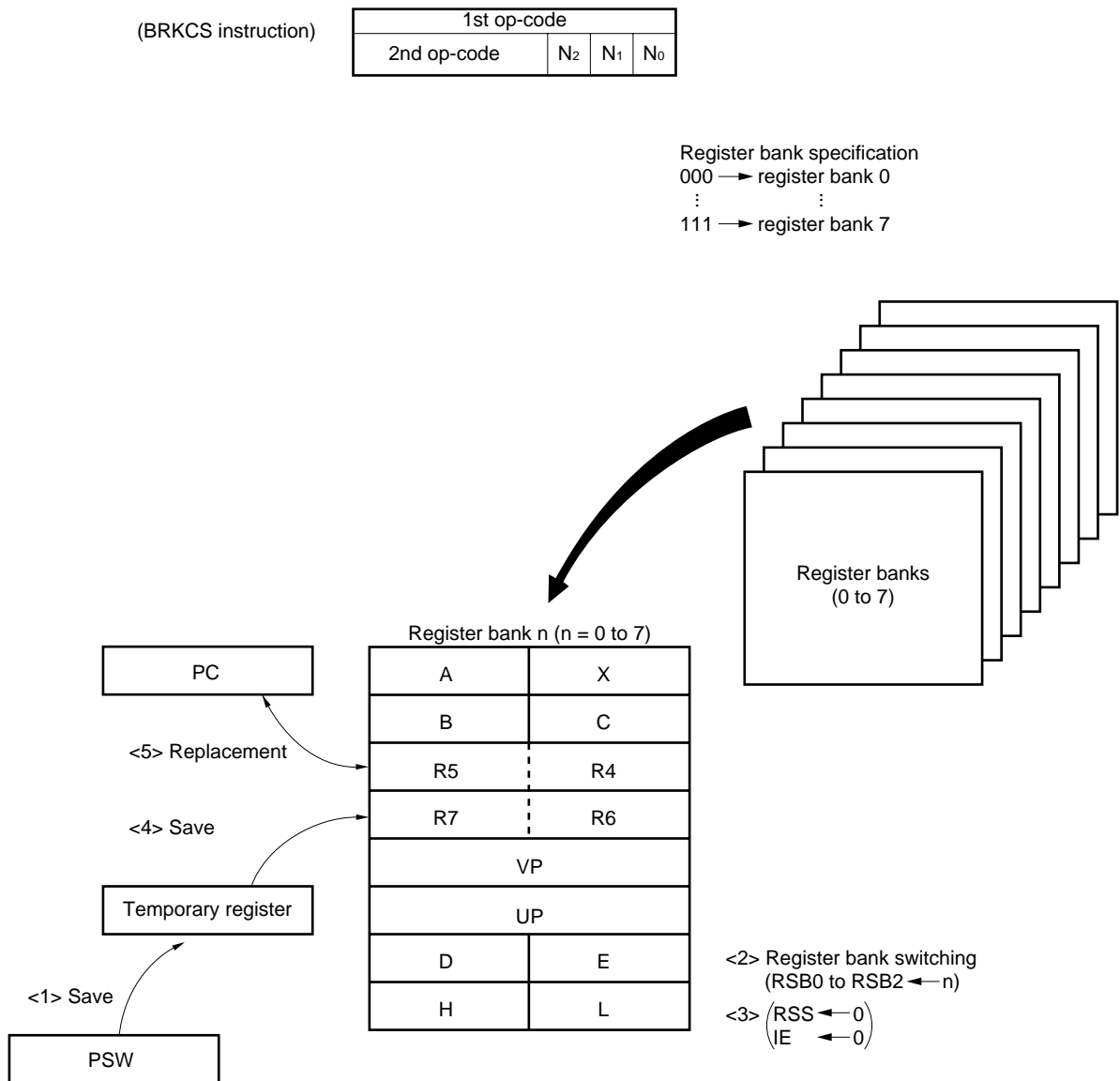
Caution Do not use the RETI instruction to return from a software interrupt executed by the BRK instruction.

13.6.2 Software interrupt (context switching) acknowledgment operation by BRKCS instruction

Executing the BRKCS instruction can activate the context switching function.

The register bank following context switching is specified by the lower 3 bits of immediate data (N₂ to N₀) of the 2nd op-code of the BRKCS instruction.

When the BRKCS instruction is executed, the program branches to the vector address stored beforehand in this register bank and saves the contents of PC and PSW at that time to the register bank.

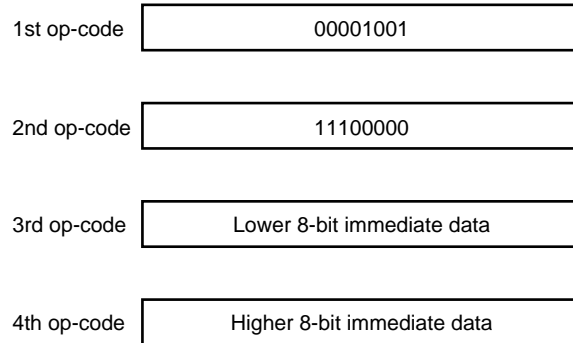
Figure 13-14. Context Switching Operation by Executing BRKCS Instruction

Restoration from the software interrupt executed by the BRKCS instruction is performed by executing the RETCSB instruction.

By executing the RETCSB instruction, the contents of the R4 and R5 registers and those of the R6 and R7 registers in the register bank selected at this time are transferred to PC and PSW, respectively. At the same time, the 16-bit immediate data specified by the 3rd and 4th op-codes of the RETCSB instruction are stored in the R4 and R5 registers in the register bank.

Therefore, if the same register bank is selected again by the context switching function, the 16-bit immediate data specified by the 3rd and 4th op-codes of the RETCSB instruction become the branch address.

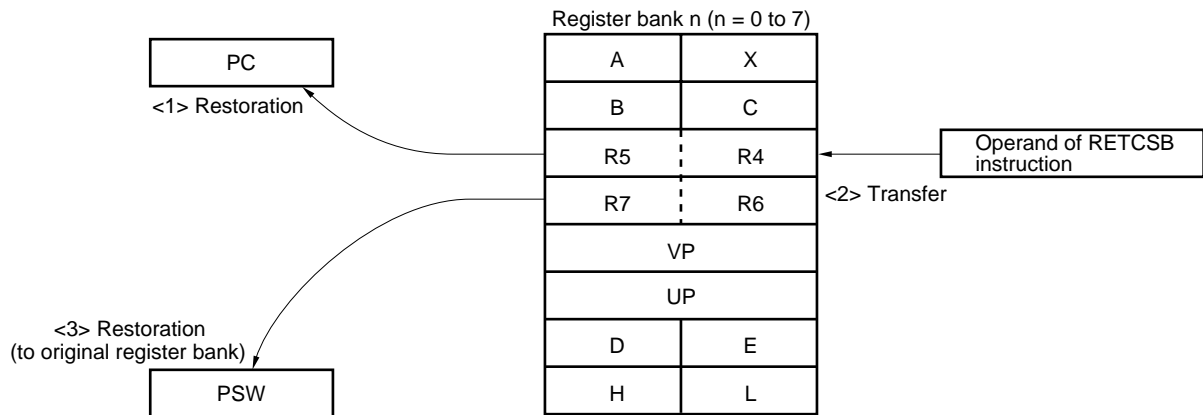
Figure 13-15. RETCSB Instruction Format



Caution If the context switching function is activated by executing the BRKCS instruction, resetting (0) of the ISPR register bits by execution of the RETCS instruction may destroy the interrupt nesting control.

Be sure to use the RETCSB instruction to restore from the service activated by the BRKCS instruction.

**Figure 13-16. Restoration Operation from Software Interrupt Executed by BRKCS Instruction
(Operation of RETCSB Instruction)**



13.7 Op-Code Trap Interrupt Acknowledgment Operation

An op-code trap interrupt is generated when the data obtained by inverting all bits of the 3rd byte of the operands of MOV STBC, #byte instruction, and MOV WDM, #byte instruction, does not match with the 4th byte of the operands. An op-code trap interrupt cannot be disabled.

When an op-code trap interrupt is generated, PSW and the lead address of the instruction in which the error occurred are saved to the stack, the IE flag is reset (0), the value of the vector table is loaded to PC, and a branch is made.

Since the address saved to the stack is the lead address of the instruction in which the error occurred, simply executing the RETB instruction at the end of the op-code trap interrupt service routine may not prevent the generation of another op-code trap interrupt.

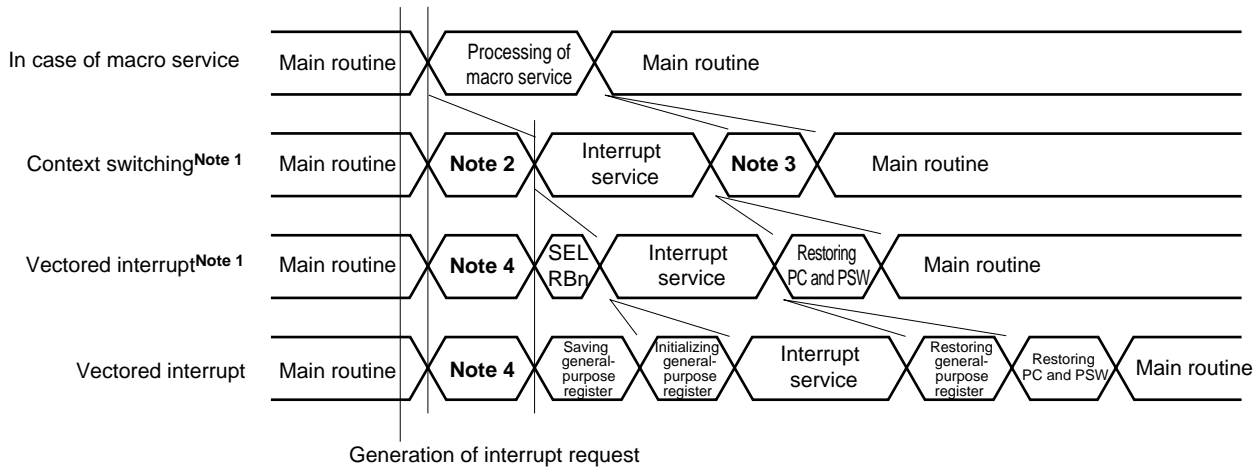
13.8 Macro Service Function

13.8.1 Outline of macro service

Macro service is an interrupt service method. In a normal interrupt, the program counter (PC) and program status word (PSW) are saved and the lead address of the interrupt service program is loaded to PC. A macro service performs processing (mainly data transfer) differing from this processing. Therefore, it can respond to interrupt requests faster. Furthermore, it can perform transfer processing faster than a program, thus shortening the servicing time.

Also, since it generates a vectored interrupt after servicing a specified number of times, vectored interrupt also can be simplified.

Figure 13-17. Differences between Vectored Interrupt and Macro Service Processing



Notes 1. When register bank switching is used and the initial value has been preset in the register

2. Switching register bank and saving PC and PSW by context switching
3. Restoring register bank, PC and PSW by context switching
4. Saving PC and PSW to stack, loading vector address to PC

Table 13-6. Interrupts for which Macro Service Is Available

Default Priority	Interrupt Request	Interrupt Generation Source	Generation Unit	Macro Service Control Word Address
0	INTOV3	Timer 3 overflow	RPU	FE06H
1	INTP0/INTCC30	INTP0 pin input/CC30 match signal	External/RPU	FE08H
2	INTP1	INTP1 pin input	External	FE0AH
3	INTP2	INTP2 pin input		FE0CH
4	INTP3/INTCC20	INTP3 pin input/CC20 match signal	External/RPU	FE0EH
5	INTP4	INTP4 pin input	External	FE10H
6	INTTM0	Timer 0 underflow	RPU	FE12H
7	INTCM03	CM03 match signal		FE14H
8	INTCM10	CM10 match signal		FE16H
9	INTCM40	CM40 match signal		FE18H
10	INTCM41	CM41 match signal		FE1AH
11	INTSER	Generation of serial reception error	Asynchronous serial interface	FE1CH
12	INTSR	Serial reception end		FE1EH
13	INTST	Serial transmission end		FE20H
14	INTCSI	Serial transmission/reception end	Clocked serial interface	FE22H
15	INTAD	A/D conversion end	A/D converter	FE24H

Remark The default priority is a fixed value. It indicates priority when multiple macro service requests are generated simultaneously.

The macro service operation includes the following five modes.

(1) Counter mode: EVTCNT

Every time an interrupt request is generated, the macro service counter (MSC) is incremented (+1) or decremented (−1) and a vectored interrupt request is generated as MSC becomes 00H.

This mode can be used to scale the number of interrupt request occurrences, etc.

(2) Block transfer mode: BLKTRS

Every time an interrupt request is generated, 1-byte or 1-word data is transferred between the special function register (SFR) specified by the SFR pointer (SFRP) and the buffer, and when the specified number of data transfers is performed, a vectored interrupt request is generated.

The buffer for the transfer is limited to the main RAM of FE00H to FEFFH.

As this mode has a simple specification method, it is used for small-capacity, high-speed data transfer.

(3) Block transfer mode (with memory pointer): BLKTRS-P

As is the case with the block transfer mode, every time an interrupt request is generated, 1-byte or 1-word data is transferred between the SFR specified by SFRP and the buffer, and when the specified number of data transfers is performed, a vectored interrupt request is generated.

The buffer for the transfer is specified by the memory pointer (MEM.PTR) (memory is an entire space of 64K bytes).

This is a general-purpose type of block transfer mode, and is used for large-volume transfers.

(4) Data difference mode: DTADIF

Every time an interrupt request is generated, the difference between the current value of SFR specified by SFRP and the “immediately preceding value” stored in memory is written to the buffer, and this current value is set as the “immediately preceding value”.

When the specified number of data transfers is performed, a vectored interrupt request is generated.

The buffer for the transfer is limited to the main RAM of FE00H to FEFFH.

This mode is used to measure the cycle and the pulse width of input pulses by the capture register.

(5) Data difference mode (with memory pointer): DTADIF-P

As is the case with the data difference mode, every time an interrupt request is generated, the difference between the current value of SFR specified by SFRP and the “immediately preceding value” stored in memory is written to the buffer and this current value is set as the “immediately preceding value”.

When the specified number of data transfers is performed, a vectored interrupt request is generated.

The buffer for the transfer is specified by the memory pointer (MEM.PTR) (memory is an entire space of 64K bytes).

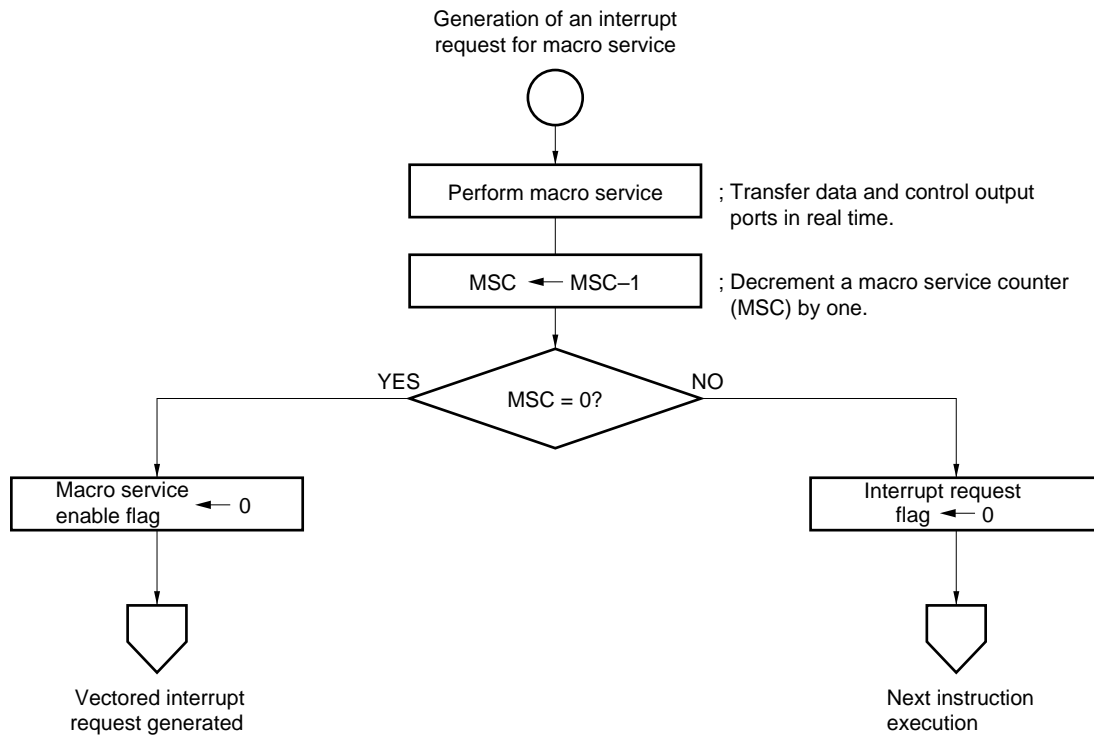
This is a general-purpose type of data difference mode and is used for large-volume data.

13.8.2 Basic function of macro service

The macro service function is used to transfer data on a hardware basis between the special function register area and memory space in response to an interrupt request.

When a macro service request is generated, the CPU temporarily stops program execution, and one- or two-byte data is transferred automatically between special function registers (SFRs) and memory. When data transfer ends, the interrupt request flag is reset to 0, then the CPU restarts program execution. After data transfer is performed as many times as the value set in the macro service counter (MSC), a vectored interrupt request is generated.

Figure 13-18. Sequence Example of Macro Service Processing



Unlike other interrupt service that is accompanied by the activation of an interrupt service program, macro service is performed automatically. This means that macro service does not require a series of the operations of making a branch to an interrupt service routine, saving and restoring register contents, and returning from the interrupt service routine. It is therefore possible to improve the CPU service time and to reduce the number of program steps.

During macro service, the general-purpose registers and instruction queue in the CPU hold the data present immediately before macro service is activated.

Interrupt requests for which macro service is specified are not influenced by the status of the IE flag in the program status word (PSW). Macro service can be performed even in the interrupt disable status or while the interrupt service program is being executed. Macro service is disabled only when the corresponding bit of the interrupt mask flag register (MK0) is set to 1.

If more than one macro service request is present, the service order is determined by the default priority. Instruction execution is suspended until all of these macro service requests are serviced.

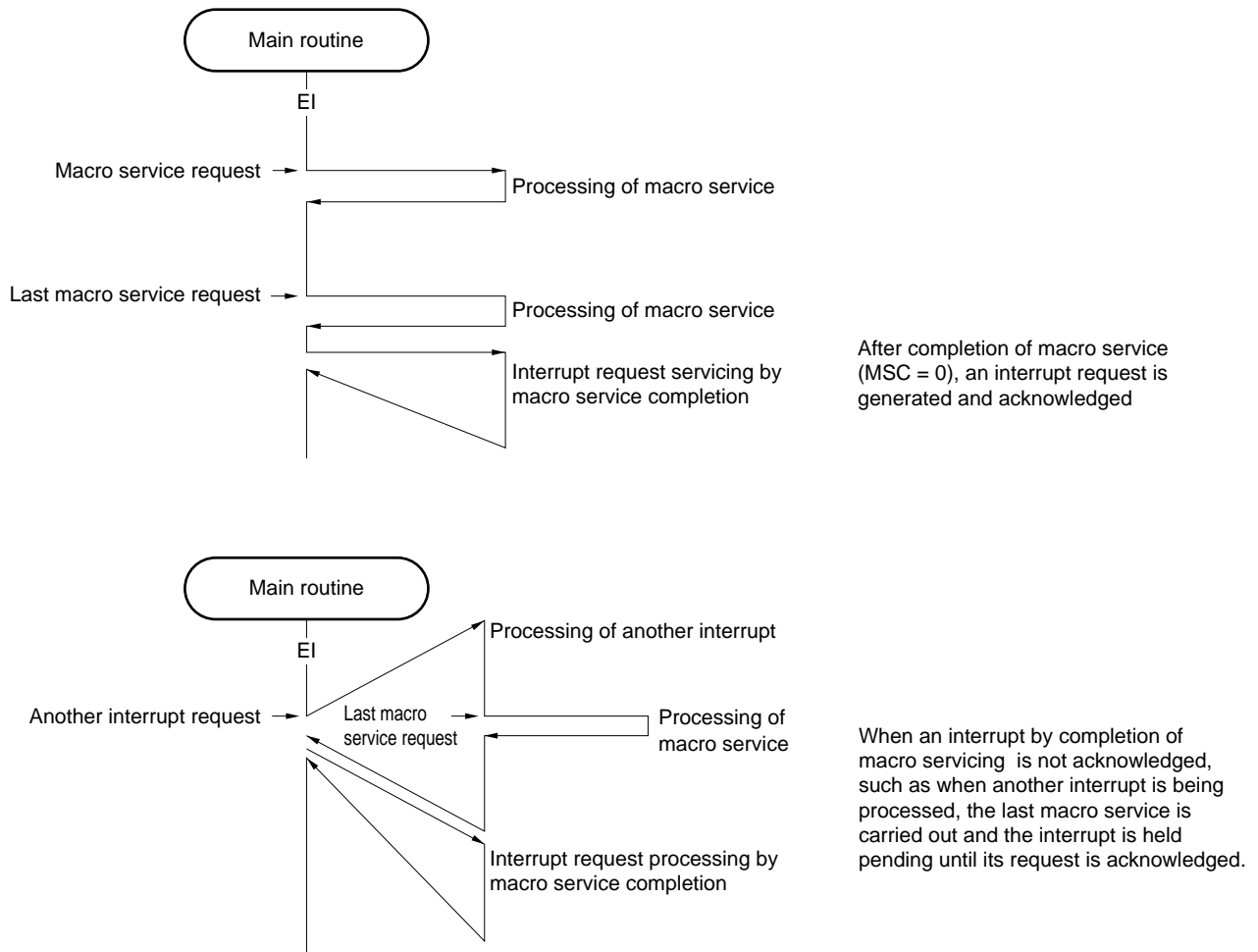
The μ PD78366A supports macro service for all included interrupt requests.

Macro service performs the following two basic operations:

- Data transfer from memory to a special function register (SFR)
- Data transfer from a special function register (SFR) to memory

13.8.3 Operation at completion of macro service

A macro service performs processing the number of times specified during execution of other programs. When processing is performed the specified number of times (when the macro service counter (MSC) becomes 0), the macro service is completed.

Figure 13-19. Operation at Completion of Macro Service

Caution If data is transmitted by UART, using the macro service, a vectored interrupt request is generated two times (refer to 9.8 Transmitting/Receiving Data Using Macro Service).

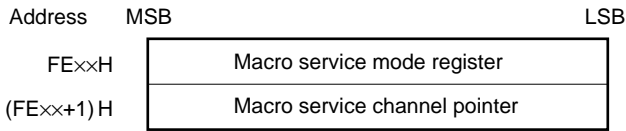
13.8.4 Macro service control register

(1) Macro service control word

A macro service control word is made up of a macro service mode register which controls the macro service function and a macro service channel pointer. These control words are located in addresses FE06H to FE25H in the main RAM area (refer to **Figure 13-21**).

Figure 13-20 shows the basic structure of a macro service control word.

Figure 13-20. Basic Structure of Macro Service Control Word



The macro service mode register sets the macro service processing mode, and the macro service channel pointer specifies the address of a macro service channel.

Before macro service can be performed, values must be loaded into the macro service mode register and channel pointer corresponding to the interrupt request for which macro service processing can be specified.

Figure 13-21. Format of Macro Service Control Word

Address		Source
FE06H	Mode register	} INTOV3
FE07H	Channel pointer	
FE08H	Mode register	} INTP0/INTCC30
FE09H	Channel pointer	
FE0AH	Mode register	} INTP1
FE0BH	Channel pointer	
FE0CH	Mode register	} INTP2
FE0DH	Channel pointer	
FE0EH	Mode register	} INTP3/INTCC20
FE0FH	Channel pointer	
FE10H	Mode register	} INTP4
FE11H	Channel pointer	
FE12H	Mode register	} INTTM0
FE13H	Channel pointer	
FE14H	Mode register	} INTCM03
FE15H	Channel pointer	
FE16H	Mode register	} INTCM10
FE17H	Channel pointer	
FE18H	Mode register	} INTCM40
FE19H	Channel pointer	
FE1AH	Mode register	} INTCM41
FE1BH	Channel pointer	
FE1CH	Mode register	} INTSER
FE1DH	Channel pointer	
FE1EH	Mode register	} INTSR
FE1FH	Channel pointer	
FE20H	Mode register	} INTST
FE21H	Channel pointer	
FE22H	Mode register	} INTCSI
FE23H	Channel pointer	
FE24H	Mode register	} INTAD
FE25H	Channel pointer	

(2) Macro service mode register

The macro service mode registers are 8-bit registers that specify macro service operation. They are mapped as part of the macro service control words in the main RAM area (refer to **Figure 13-20**).

13.8.5 Macro service mode

Operation of the macro service is specified by setting a macro service mode register. The macro service mode is determined by the low-order six bits of the macro service mode register. The modes are classified into two groups: Group 0 and group 1.

- Group 0: Control word only; no channel provided
- Group 1: Control word and channel

The high-order two bits of the macro service mode register function as a subcommand (refer to **Table 13-7**).

13.8.6 Macro service operation

There are five modes of macro service operation.

Table 13-7. Classification of Macro Service Modes

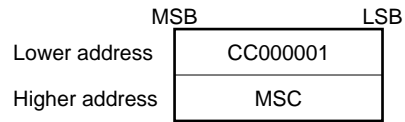
Group	Macro Service Mode Register	Function	
Group 0	CC000001	Counter mode	EVTCNT
Group 1	CC010011	Block transfer mode	BLKTRS
	CC010100	Block transfer mode (with memory pointer)	BLKTRS-P
	10011001	Data difference mode	DTADIF
	10011010	Data difference mode (with memory pointer)	DTADIF-P

C in the most significant bit (MSB) of a macro service mode register indicates the length of the data to be operated on except for EVTCNT.

- C = 0: Byte data
- C = 1: Word data

For BLKTRS and BLKTRS-P, byte buffer representation is always used. If word specification is made, byte buffers must be replaced by word buffers.

Caution Word buffers must be placed at even addresses.

(1) Counter mode: EVTCNT**[Macro service control word]****[Operation]**

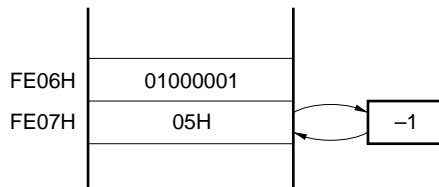
Every time the macro service is generated, the macro service counter (MSC) is incremented or decremented by one. When the MSC reaches 00H (overflow), a vectored interrupt request is generated.

Table 13-8. Counter Mode Operation Specification

CC	Operation
00	Increment
01	Decrement
10	Setting prohibited
11	

In the counter mode, the macro service function operates as a counter to divide the number of interrupt requests.

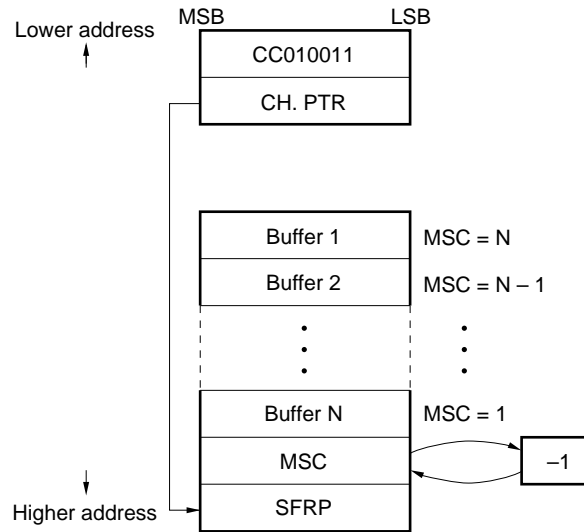
Example Dividing the number of INTOV3 interrupt requests by five using macro service

**[Application]**

Event counter, or measurement of the number of capture times

(2) Block transfer mode: BLKTRS

[Macro service control word]



[Operation]

The channel pointer (CH.PTR) specifies the SFR pointer (SFRP). A buffer is addressed by the channel pointer (CH.PTR) and macro service counter (MSC).

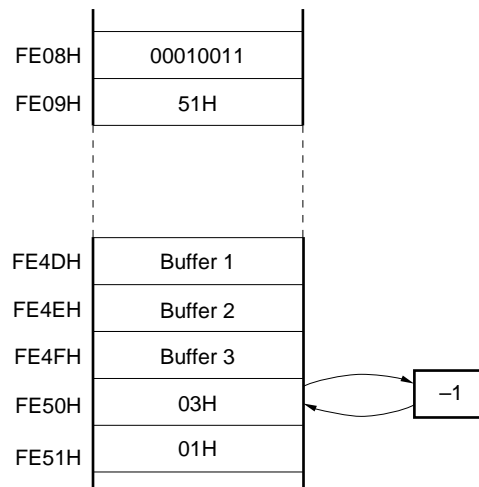
Data is transferred between the SFR pointed to by the SFRP and buffers. Data transfer starts from buffer 1.

Every time transfer terminates, the MSC is decremented by one. When the MSC reaches 0, a vectored interrupt request is generated.

Table 13-9. Specification of Block Transfer Mode Operation

CC	Operation	Transferred Data	Buffer Address
00	Buffer \leftarrow SFR	Byte	$(\text{CH.PTR content}) - (\text{MSC content}) - 1$
01	SFR \leftarrow buffer		
10	Buffer \leftarrow SFR	Word	$(\text{CH.PTR content}) - (\text{MSC content} \times 2) - 1$
11	SFR \leftarrow buffer		

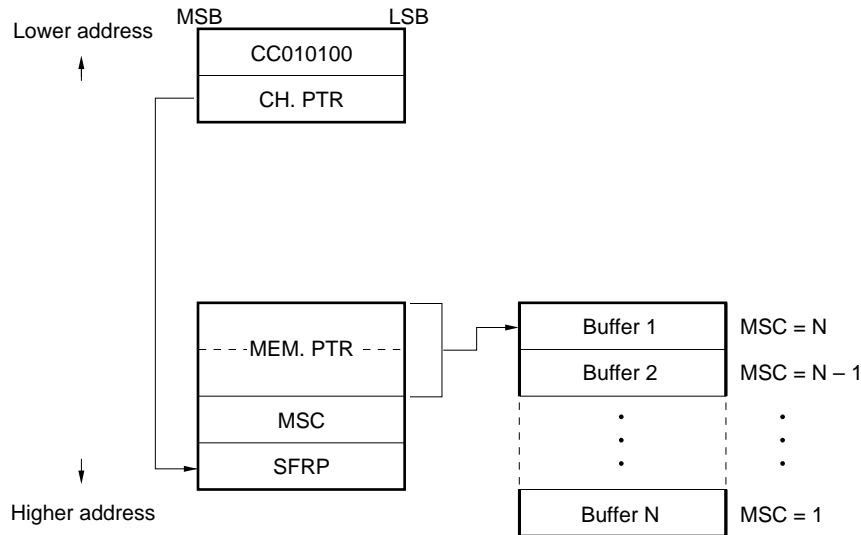
Example Transferring the contents of port 1 (FF01H) to buffers in response to the INTP0 interrupt request



[Application]

This mode can be used to transmit/receive data with serial interface.

Caution Word buffers must be placed at even addresses.

(3) Block transfer mode (with a memory pointer): BLKTRS-P**[Macro service control word]****[Operation]**

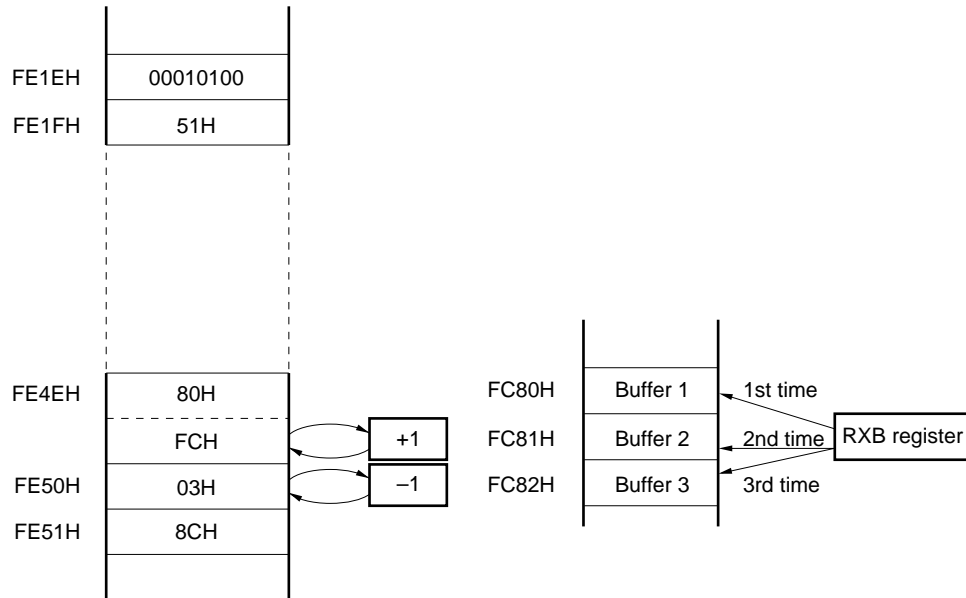
The channel pointer (CH.PTR) specifies the SFR pointer (SFRP). Data is transferred between the SFR pointed to by the SFRP and the buffer addressed by MEM.PTR. Data transfer starts from buffer 1. When byte data transfer terminates, the MEM.PTR is incremented by one. When word data transfer terminates, the MEM.PTR is incremented by two.

Every time transfer terminates, the macro service counter (MSC) is decremented by one. When the MSC reaches 0, a vectored interrupt request is generated.

Table 13-10. Specification of Block Transfer Mode (with a memory pointer) Operation

CC	Operation	Transferred Data
00	Buffer \leftarrow SFR	Byte
01	SFR \leftarrow buffer	Byte
10	Buffer \leftarrow SFR	Word
11	SFR \leftarrow buffer	Word

Example Transferring the contents of serial receive buffer RXB (FF8CH) to buffer in response to INTSR interrupt request



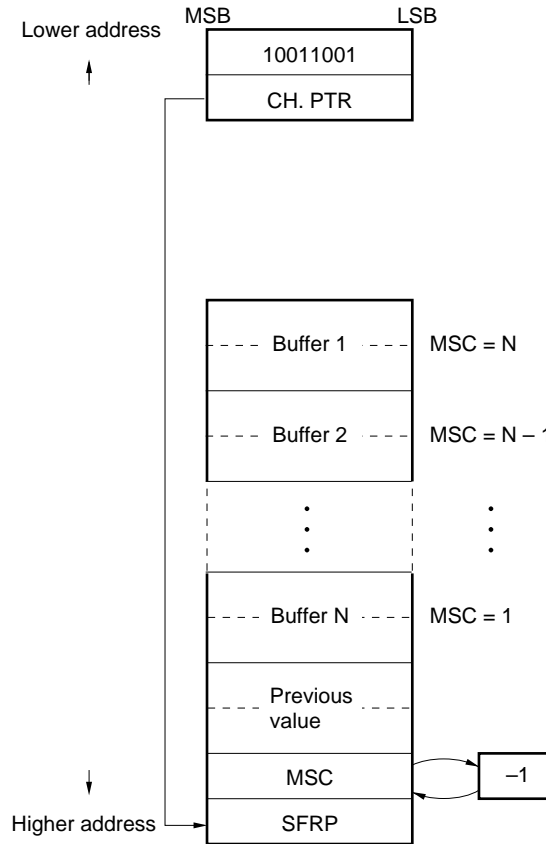
[Application]

This mode can be used to transmit/receive data with serial interface.

- Cautions**
1. Word buffers must be placed at even addresses.
 2. MEM.PTR must be placed at an even address.

(4) Data difference mode: DTADIF

[Macro service control word]



[Operation]

The channel pointer (CH.PTR) specifies the SFR pointer (SFRP). A buffer is addressed by the channel pointer (CH.PTR) and macro service counter (MSC).

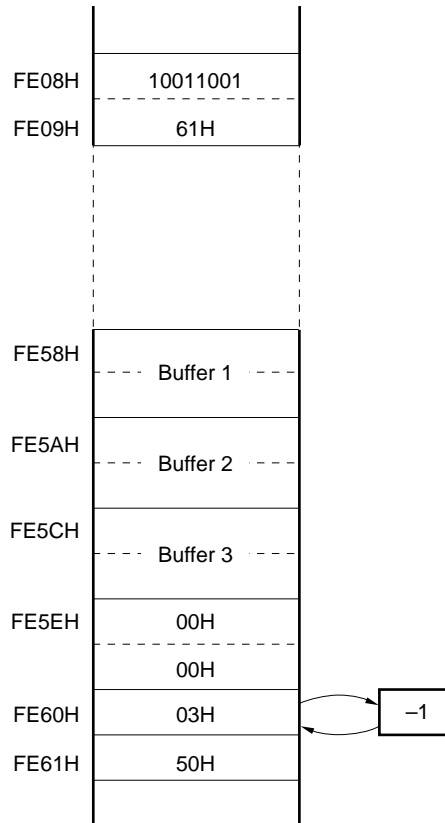
The difference between the current value of the SFR (especially capture register) pointed to by the SFRP and the previous value is written to the buffer. The current value of the SFR is then regarded as the new "previous value." Data write starts from buffer 1.

Every time data is written, the MSC is decremented by one. When the MSC reaches 0, a vectored interrupt request is generated.

The buffer address is obtained as follows:

$$(\text{Buffer address}) = (\text{value of CH.PTR}) - (\text{value of MSC} \times 2) - 3$$

Example The difference between the current value and previous value of capture/compare register CC30 (FF50H) is written into a buffer with the INTP0 input signal used as the trigger signal. The period of the INTP0 input signal is then measured using the difference by the vectored interrupt service routine.



[Application]

This mode can be used to measure periods or pulse widths using a capture register.

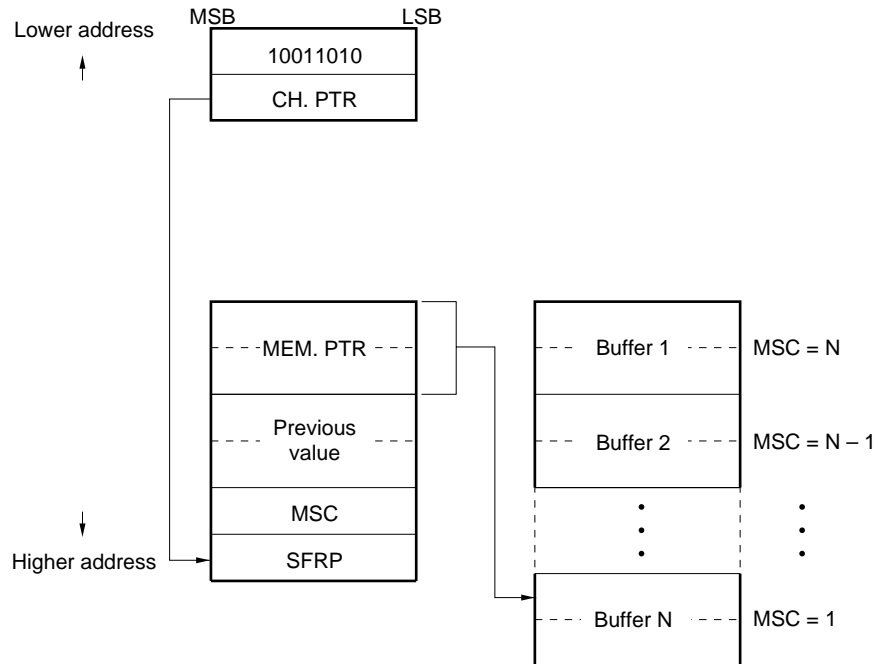
Cautions 1. Don't set 00H in the MSC.

2. Buffers must be placed at even addresses.

3. The "previous value" must be initialized to dummy data in advance.

4. The SFRP can specify 16-bit SFRs only.

(5) Data difference mode (with a memory pointer): DTADIF-P

[Macro service control word]**[Operation]**

The channel pointer (CH.PTR) specifies the SFR pointer (SFRP). A buffer is addressed by the MEM.PTR and macro service counter (MSC).

The difference between the current value of the SFR (especially capture register) specified by the SFRP and the previous value is written to the buffer. The current value of the SFR is then regarded as the new "previous value." Data write starts from buffer 1.

Every time data is written, the MSC is decremented by one. When the MSC reaches 0, a vectored interrupt request is generated.

The MEM.PTR remains unchanged.

The buffer address is obtained as follows:

$$(\text{Buffer address}) = (\text{value of MEM.PTR}) - (\text{value of MSC} \times 2) + 2$$

[Application]

This mode is used to measure periods and pulse widths using a capture register.

Cautions 1. Don't set 00H in the MSC.

- 2. Buffers must be placed at even addresses.**
- 3. The MEM.PTR must be placed at an even address.**
- 4. The "previous value" must be initialized to dummy data in advance.**
- 5. The SFRP can specify 16-bit SFRs only.**

13.9 Cases where Interrupt Request and Macro Service are Temporarily Held Pending**(1) Instructions that do not acknowledge interrupt requests**

When the following instructions are executed, acknowledgment of interrupts and processing of macro services are temporarily held pending. However, software interrupts are not held pending.

MOV1	PSWL.bit, CY	RETB	
MOV1	PSWH.bit, CY	RETI	
SET1	PSWL.bit	RETCS	!addr16
CLR1	PSWL.bit	RETCSB	!addr16
NOT1	PSWL.bit	POP	PSW
BFSET	PSWL.bit, \$addr16	POPU	post
BFSET	PSWH.bit, \$addr16	EI	
BTCLR	PSWL.bit, \$addr16	DI	
BTCLR	PSWH.bit, \$addr16		
BRK			
BRKCS	RBn		

(2) Instructions that sometimes do not acknowledge interrupt requests

The instructions shown in Table 13-11 are mainly instructions that perform writing to various control registers (INTCreg) for interrupt service. When these instructions are executed, acknowledgment of interrupts and processing of macro services may be temporarily held pending depending on the CPU condition.

Remark INTCreg: Registers MK0H, MK0L, ISPR, IMC and interrupt control registers (OVIC3, PIC0 to PIC4, TMIC0, CMIC03, CMIC10, CMIC40, CMIC41, SERIC, SRIC, STIC, CSIIC, ADIC)

Table 13-11. List of Instructions that Sometimes do not Acknowledge Interrupt Requests

Instruction Group	Mnemonic	Operand	Condition
8-bit data transfer	MOV	INTCreg, #byte	
		INTCreg, A	
		mem, A	When mem is set to INTCreg address
		[saddrp], A	When saddrp is set to INTCreg address
		!addr16, A	When addr16 is set to INTCreg address
		PSWL, #byte	
		PSWH, #byte	
		PSWL, A	
		PSWH, A	
	XCH	A, mem	When mem is set to INTCreg address
		A, [saddrp]	When saddrp is set to INTCreg address
16-bit data transfer	MOVW	INTCreg, #word	
		INTCreg, AX	
		!INTCreg, rp1	
		mem, AX	When mem is set to INTCreg address
	XCHW	AX, INTCreg	
		AX, mem	When mem is set to INTCreg address
8-bit operation	ALU	INTCreg, #byte	ALU: ADD, ADDC, SUB, SUBC, AND, OR, XOR
		mem, A	When mem is set to INTCreg address ALU: ADD, ADDC, SUB, SUBC, AND, OR, XOR
16-bit operation	ALUW	INTCreg, #word	ALUW: ADDW, SUBW
Bit manipulation	MOV1	INTCreg. bit, CY	
	BIT	INTCreg. bit	BIT: SET1, CLR1, NOT1
Stack manipulation	POP	INTCreg	
Conditional branch	BTCLR	INTCreg. bit, \$addr16	
	BFSET	INTCreg. bit, \$addr16	
String	STRING	[DE+], A [DE-], A [DE+], [HL+] [DE-], [HL-]	When INTCreg address is set to destination (DE register) STRING: MOVW, MOVW, XCHW, XCHW, CMPME, CMPBKE, CMPMNE, CMPBKNE, CMPMC, CMPBKC, CMPMNC, CMPBKNC

Caution 1. When executing polling to the interrupt-related registers by using the BTCLR instruction, etc., ensure that the branch destination of the BTCLR instruction, etc. is not the instruction itself. If a program that branches to the instruction itself is written, all interrupts and macro services will be held pending until a condition is established under which the program does not branch by that instruction.

```

Bad example
:
LOOP:  BTCLR PIC0.7,$LOOP    All interrupts and macro services are held pending until PIC0.7
                               becomes 1.
      xxx                    ← Interrupts and macro services are processed first after execution
                               of instruction following BTCLR instruction.
      :

Good example (1)
:
LOOP:  NOP
      BTCLR PIC0.7,$LOOP    ← Since interrupts and macro services are processed after
                               execution of NOP instruction, interrupts are not held pending
                               for a long time.
      :

Good example (2)
:
LOOP:  BTCLR PIC0.7,$NEXT
      BR $LOOP              ← Interrupts and macro services are processed after execution
                               of BR instruction, and thus interrupts are not held pending for
                               a long time.

NEXT:  :

```

Caution 2. When a group of the above-mentioned instructions is consecutively used for the same reason and interrupts and macro services you should not hold pending for a long time, insert an NOP instruction, etc. during the interim to create timing for interrupts or macro services to be acknowledged.

13.10 Instructions whose Execution Is Temporarily Suspended by Interrupts and Macro Services

Execution of the following instructions is temporarily suspended by an acknowledgeable interrupt request or macro service request in order to acknowledge it. The suspended instructions are restarted after completion of the interrupt service or the macro service.

<Instructions suspended temporarily>

```

MOVM, XCHM, MOVBK, XCHBK
CMPME, CMPMNE, CMPMC, CMPMNC
CMPBKE, CMPBKNE, CMPBKC, CMPBKNC
SACW

```


CHAPTER 14 STANDBY FUNCTION

14.1 Function Overview

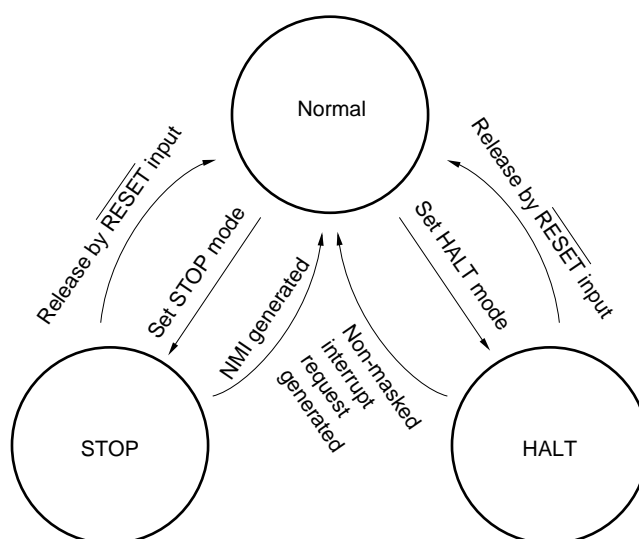
The μ PD78366A has a standby function to reduce power consumption of the system. With the standby function, two modes are available:

- **HALT mode:** In this mode, the CPU operation clock is stopped. Intermittent operation, when combined with the normal operation mode, can reduce overall system power consumption.
- **STOP mode:** In this mode, the oscillator is stopped to stop the entire system. Since only leakage currents may flow in this mode, system power consumption can be minimized.

Each mode is set by software.

Figure 14-1 is the transition diagram of the standby modes (STOP and HALT modes).

Figure 14-1. Transition Diagram of Standby Modes



14.2 Standby Control Register (STBC)

The standby control register (STBC) is an 8-bit register which controls the standby mode.

As the contents of the STBC register can only be changed by a dedicated instruction, it will not be changed by a program crash. The dedicated instruction MOV STBC, #byte is shown below.

Figure 14-2. STBC Register Write Instruction

	B1	B2	B3	B4
MOV STBC, #byte	09H	C0H	data0	data1

This instruction has a data check function to prevent the application system stopping unintentionally by a program crash. Data can be written in the STBC register only when the last two bytes of the instruction are the complement of each other, i.e., when $\overline{\text{data0}} = \text{data1}$.

If they are not complementary, the STBC register is not written to and an op-code trap interrupt occurs. In this case, the address of the instruction which causes the trap is saved in the stack area as return address. Therefore, the program can be restarted from the return address by checking the trap address or executing the RETB instruction. (refer to **13.1.4 Op-code trap interrupt**, **13.7 Op-Code Trap Interrupt Acknowledgment Operation**)

However, if the cause of an op-code trap cannot be removed due to a hardware error, the RETB instruction may cause an infinite loop.

The SBF bit of the STBC register is a standby flag. It is used to determine whether to release the STOP mode by the $\overline{\text{RESET}}$ input. This bit is reset to 0 only when the supply voltage (V_{DD}) is supplied (power-on reset). It cannot be reset by software. It is not affected simply by supplying the $\overline{\text{RESET}}$ signal.

The contents of the STBC register can be read by the data transfer instruction at any time.

$\overline{\text{RESET}}$ input sets the STBC register to 0000x000B.

Figure 14-3 shows the format of the STBC register.

Caution After the SBF flag is read, set it to 1. Software can then discriminate a power-on reset from release of the STOP or HALT mode.

Figure 14-3. Format of Standby Control Register (STBC)

Symbol	7	6	5	4	3	2	1	0	Address	When reset	R/W
STBC	0	0	0	0	SBF	0	STP	HLT	FFC0H	0000×000B	R/W

HALT mode specification bit^{Note}

When this bit is set to 1, the system enters the HALT mode.
When the HALT mode is released, this bit is automatically reset to 0.

STOP mode specification bit

When this bit is set to 1, the system enters the STOP mode.
When the STOP mode is released, this bit is automatically reset to 0.

This flag is reset to 0 only when power supply voltage (V_{DD}) is supplied being increased from 0 V.

Note If the HALT mode is released by macro service activation, this bit is reset to 0 when a vectored interrupt request is issued at the end of the macro service.

Caution Do not set the STP bit of the standby control register (STBC) to 1 when using the external clock.

14.3 Operation

14.3.1 HALT mode

(1) Setting the HALT mode and operation states in the HALT mode

In the HALT mode, the CPU clock is stopped.

The total power consumption of the system can be reduced by setting the system to the HALT mode during CPU dead time. The system is put in the HALT state by setting the HLT bit in the STBC register to 1.

In the HALT mode, the CPU clock and program execution are stopped. Nevertheless, the contents of all registers and internal RAM immediately before the HALT mode is set are retained. On-chip peripheral hardware can operate. Hardware functions enter the states shown in Table 14-1.

Caution If the interrupt request flag ($\times\times IF$) is set to 1 and the interrupt is not masked ($\times\times MK = 0$), the system does not enter the HALT mode. When macro service processing ($\times\times ISM = 1$) is performed, the system enters the HALT mode after the macro service terminates.

Table 14-1. Operation States in HALT Mode

Function		Operation State
Clock generator		Operating
Internal system clock		
CPU		Stopped
I/O line		Retained
On-chip peripheral hardware		Operating
Internal data		Internal data such as the CPU status, data, and contents of the internal RAM is retained as it was before setting the HALT mode.
When external devices are expanded	AD0-AD7	High-impedance
	A8-A15	
	ASTB	0
	$\overline{\text{RD}}$	1
	$\overline{\text{WR}}$	

(2) Releasing the HALT mode

The HALT mode can be released by a non-maskable interrupt request, non-masked maskable interrupt request, non-masked macro service request, or $\overline{\text{RESET}}$ input.

(a) Releasing the HALT mode by an interrupt request

The HALT mode is released by a non-maskable interrupt request, a maskable interrupt request which is not masked, or macro service request, regardless of the priority of the request. If the HALT mode is set by the interrupt service routine, however, the following operation takes place (refer to **Table 14-2**).

- (i) When an interrupt request, having higher priority than the interrupt request being serviced, is issued, the HALT mode is released, and this new interrupt request is accepted. Such interrupt requests include a non-maskable interrupt request. When the PRSL bit of an interrupt mode control register (IMC) is set to 0, however, the system only accepts multiple interrupt requests having the lowest priority.
- (ii) When an interrupt request, having the same priority as or lower priority than the interrupt request being serviced, is issued, the HALT mode is only released. In this case, the new interrupt request is not accepted, and the next instruction is executed. The interrupt request is retained.

Table 14-2. Acceptance of Interrupts Generated during Interrupt Servicing

Priority of Interrupt Service Routines in HALT Mode	Priority and State of Generated Interrupts	
	Accepted	Retained
0	—	0-3
1	0	1-3
2	0, 1	2, 3
3	0-2, 3 (PRSL = 0)	3 (PRSL = 1)

Table 14-3. Operations after HALT Mode Is Released by an Interrupt Request

Releasing Request	EI State	DI State
Non-maskable interrupt	Branches to the vector address.	
Maskable interrupt	Branches to the vector address or executes the next instruction.	Executes the next instruction.

Table 14-4. Releasing HALT Mode by a Macro Service Request

Conditions for Vectored Interrupt at End of Macro Service	HALT Mode	Operations after Releasing HALT Mode	
		EI State	DI State
Satisfied	Released	Branches to vector address or executes next instruction. Note	Executes next instruction
Not satisfied	System returns to HALT mode.	—	

Note If the HALT mode is set by an interrupt service routine then an interrupt request which is not masked and has higher priority is generated, processing branches to the vector address.

(b) Releasing the HALT mode by RESET input

Same as the normal reset operation except that data in the internal RAM before setting the HALT mode is retained.

14.3.2 STOP mode

(1) Setting the STOP mode and operation states in the STOP mode

In the STOP mode, the oscillator is stopped.

Power consumption can be substantially reduced when the entire application system stops. The system is placed in the STOP state by setting the STP bit in the STBC register to 1.

In the STOP mode, generation of the internal clock is stopped when the oscillator stops operating. At the same time, the watchdog timer for reserving the oscillation stabilization time is automatically cleared by the hardware. Although program execution is stopped, the contents of all registers and internal RAM immediately before the STOP mode is set are retained. Hardware functions enter the states shown in Table 14-5.

Table 14-5. Operation States in STOP Mode

Function		Operation State
Clock generator		Stopped
Internal system clock		
CPU		
I/O line		Retained when V_{DD} is within the operable range
On-chip peripheral hardware		Stopped
Internal data		Internal data such as the CPU status, data, and contents of the internal RAM is retained as it was before setting the STOP mode when V_{DD} is within the operating range ^{Note} .
When external devices are expanded	AD0-AD7	High-impedance
	A8-A15	
	ASTB	0
	\overline{RD}	1
	\overline{WR}	

Note Only the contents of internal RAM are retained by keeping the data retention voltage when V_{DD} is lower than the minimum operable voltage.

(2) Releasing the STOP mode

The STOP mode can be released by NMI or $\overline{\text{RESET}}$ input.

The operation after release of the STOP mode depends on the source of the STOP mode release and the condition under which the STOP mode was set.

(a) Releasing the STOP mode by NMI input

The oscillator restarts when a valid edge is applied to the NMI input. While the NMI input is active, the watchdog timer remains cleared and does not start counting. When the NMI input level returns to the inactive level, the watchdog timer starts counting. When the watchdog timer overflows, generation of the internal system clock is started.

Therefore, the $\mu\text{PD78366A}$ is in the wait state during the following period:

$$\begin{aligned} & \text{(Active level duration after detecting a valid edge in NMI input)} \\ & \quad + \\ & \text{(Watchdog timer overflow time)} \end{aligned}$$

The overflow time for the watchdog timer is specified using the WDM register.

The operation after release of the STOP mode depends on the condition under which the STOP mode was set and the priority of watchdog timer interrupt requests and NMI interrupt requests.

Priority is specified for a watchdog timer input interrupt or NMI interrupt using the WDM register (refer to **Figure 14-4**).

Figure 14-4. Format of Watchdog Timer Mode Register

Symbol	7	6	5	4	3	2	1	0	Address	When reset	R/W
WDM	RUN	0	0	PRC	0	WDI2	WDI1	0	FFC2H	00H	R/W

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 ¹³	167.8	131.1
1	1	Setting prohibited		

Remark f_{CLK} : internal system clock frequency

PRC	Specifies priority of watchdog timer interrupt request
0	Watchdog timer interrupt < NMI interrupt request
1	Watchdog timer interrupt > NMI interrupt request

RUN	Specifies operation of watchdog timer
0	Stops watchdog timer
1	Clears and starts watchdog timer

- Cautions**
1. Data can be written into the WDM register only by a dedicated instruction (MOV WDM, #byte).
 2. The priority of interrupt requests should be set at initialization of the application system, such as initialization of the stack pointer, and should not be dynamically changed during execution of the program.
 3. Once it is set to 1, the RUN bit cannot be reset to 0 by software.
 4. The count clock is not reset even when the watchdog timer is cleared by setting the RUN bit to 1.
 5. If a watchdog timer interrupt and NMI interrupt are generated simultaneously with PRC = 1 (INTWDT > NMI), after the first 1 instruction of the NMI interrupt service routine is executed, the watchdog timer interrupt service routine is executed. Therefore, when used with PRC = 1 setting, the first instruction of the NMI interrupt service routine should be an NOP instruction.
 6. Bits 6, 5 and 0 of the WDM register are fixed to “0” by hardware. Even if “1” is written to them, they remain “0”.
 7. Be sure to write 0 to bit 3 of the WDM register.

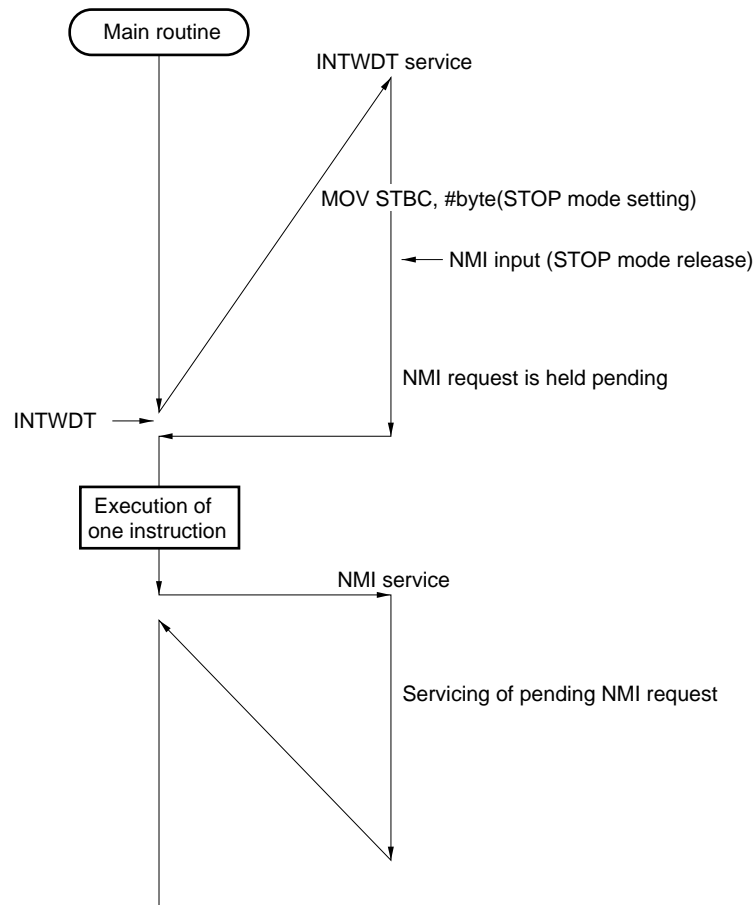
(i) When set to STOP mode in watchdog timer interrupt service routine

<1> When priority is watchdog timer interrupt request > NMI interrupt request (PRC bit of WDM register = 1)

After the STOP mode is released by NMI input, the program starts execution of the instruction following the instruction by which the STOP mode was set (NMI interrupt request is held pending).

When the RETI instruction is executed during the watchdog timer interrupt service routine, the program restores from the watchdog timer interrupt service routine. Then, if one instruction is executed, the program branches to the NMI interrupt service routine.

Figure 14-5. Operation after Release of STOP Mode (1)



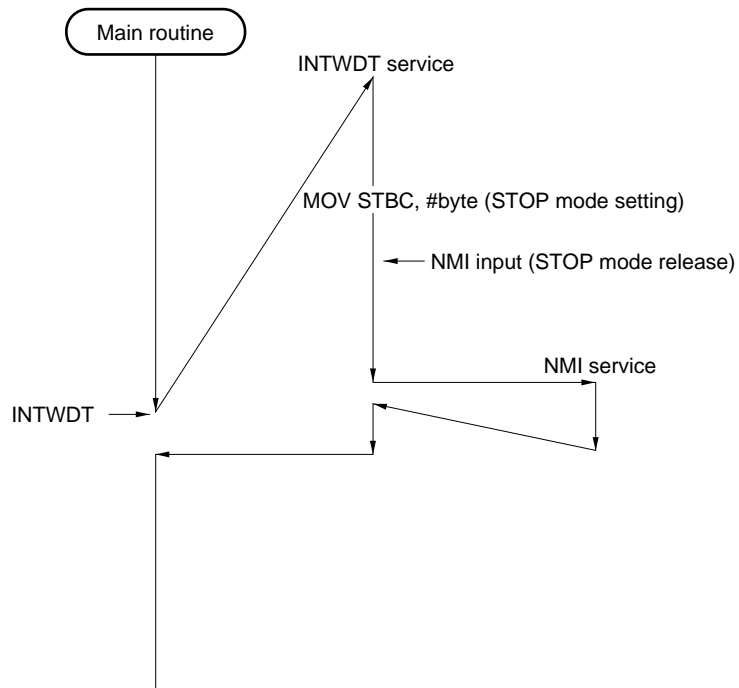
<2> When priority is watchdog timer interrupt request < NMI interrupt request (PRC bit of WDM register = 0)

After the STOP mode is released by NMI input, the program branches to the NMI interrupt service routine immediately.

In the NMI interrupt service routine, if the RETI instruction is executed, the program returns to the instruction immediately following that by which the STOP mode was set in the watchdog timer interrupt service routine.

Then, if the RETI instruction is executed, the program restores from the watchdog timer interrupt service routine.

Figure 14-6. Operation after Release of STOP Mode (2)



Then, if one instruction is executed, the program branches to the NMI interrupt service routine again.

```
graph TD
    MR(Main routine) --> J1(( ))
    J1 --> EOI[Execution of one instruction]
    EOI --> J2(( ))
    J2 --> NMI[NMI]
    NMI --> J1
    J2 --> NMS1[NMI service]
    NMS1 --> MOV[MOV STBC, #byte (STOP mode setting)]
    MOV --> NMIIN[NMI input (STOP mode release)]
    NMIIN --> J2
    NMS1 --> J2
    J2 --> NMS2[NMI service]
    NMS2 --> J2
```

The flowchart illustrates the NMI interrupt processing in STOP mode. It starts with the 'Main routine' (rounded rectangle) which leads to a junction point. From this junction, the flow goes to 'Execution of one instruction' (rectangle). After instruction execution, it reaches another junction point. This junction can be triggered by an 'NMI' (indicated by an arrow from the left), which loops back to the first junction. From the second junction, the flow proceeds to 'NMI service'. This service routine includes the instruction 'MOV STBC, #byte (STOP mode setting)', followed by 'NMI input (STOP mode release)' (indicated by an arrow from the right), which loops back to the second junction. The 'NMI service' routine then loops back to the second junction. Finally, the flow returns to the first junction, completing the cycle.

(iii) When set to STOP mode other than during non-maskable interrupt service routine

After the STOP mode is released by NMI input, the program immediately branches to the NMI interrupt service routine.

When the RETI instruction is executed during the NMI interrupt service routine, the program restores to the instruction following that by which the STOP mode was set.

Figure 14-8. Operation after Release of STOP Mode (4)

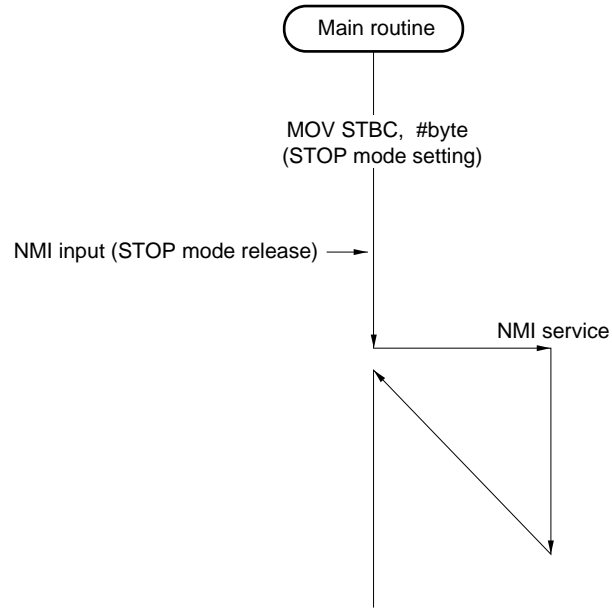
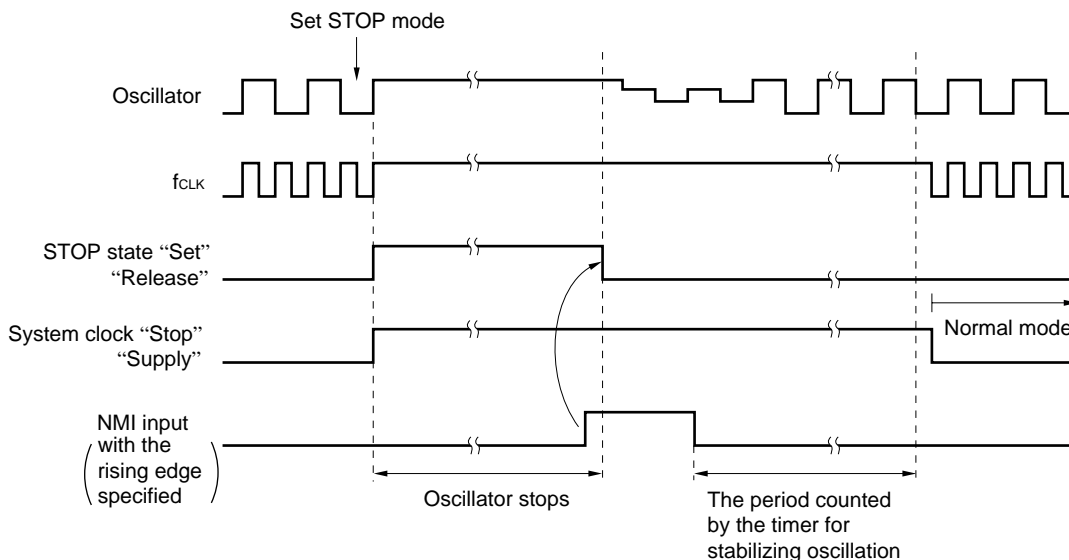


Table 14-6. Release of STOP Mode and Operation after Release

Release Source	Condition when STOP Mode was Set	PRC ^{Note}	Operation after Release
RESET input	–	×	Starts operation from reset address.
NMI input	INTWDT routine	1	Starts execution of instruction following MOV STBC, #byte instruction (NMI interrupt request is held pending). After completion of watchdog timer interrupt service that set STOP mode, NMI interrupt request is acknowledged (see Figure 14-5).
		0	Acknowledges NMI interrupt request (see Figure 14-6).
	NMI routine	×	Starts execution of instruction following MOV STBC, #byte instruction (NMI interrupt request is held pending). After completion of NMI interrupt service that set STOP mode, NMI interrupt request is acknowledged again (see Figure 14-7).
	Other than non-maskable interrupt routine	×	Acknowledges NMI interrupt request (see Figure 14-8).

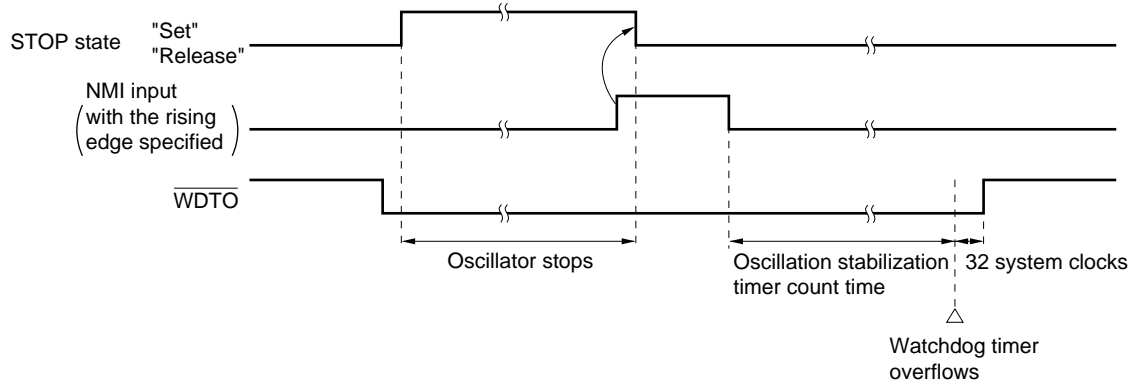
Note Priority specification flag in watchdog timer mode register (WDM)
 PRC = 1 ... Watchdog timer interrupt request > NMI interrupt request
 PRC = 0 ... Watchdog timer interrupt request < NMI interrupt request
 PRC = x ... don't care

Figure 14-9. Releasing STOP Mode by NMI Input (1)



If INTWDT occurs immediately before the STOP mode is set, and the STOP mode is set with the $\overline{\text{WDTO}}$ pin low, the STOP mode is released by NMI input as shown in Figure 14-10.

Figure 14-10. Releasing STOP Mode by NMI Input (2)



(b) Releasing by $\overline{\text{RESET}}$ input

The oscillator restarts at the same time $\overline{\text{RESET}}$ input level is changed from high to low to put the system in the reset state.

Apply the $\overline{\text{RESET}}$ active period for the oscillation stabilization time. When $\overline{\text{RESET}}$ goes high, operation starts from the address stored in the reset vector.

[MEMO]

CHAPTER 15 RESET FUNCTION

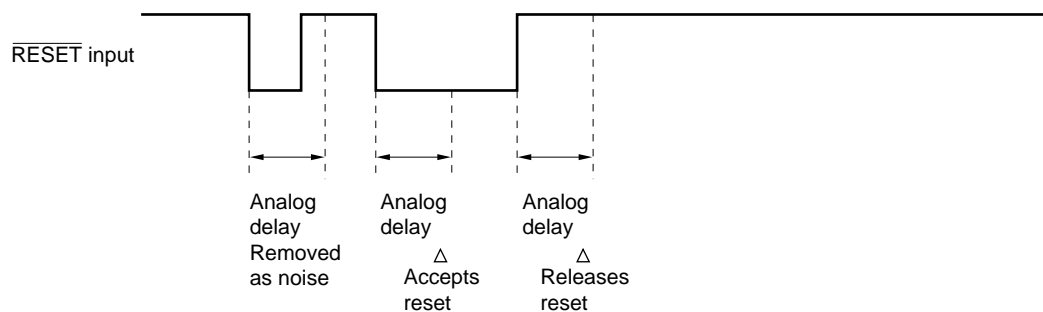
When the signal applied to the $\overline{\text{RESET}}$ pin is low, the system is reset, and each hardware component is placed in the status indicated in Table 15-1.

When the signal applied to the $\overline{\text{RESET}}$ input port goes high, the reset status is released, and program execution starts. The contents of registers must be initialized in the program as required. In particular, the number of cycles specified in the programmable wait control register (PWC) must be changed as required. After resetting, the initial setting in the PWC register is effective: Three wait cycles are added to the bus cycle, and the fetch cycle mode is normal (refer to **Figure 16-7. Format of Programmable Wait Control Register**).

The $\overline{\text{RESET}}$ pin contains a noise eliminator based on analog delays to prevent abnormal operation due to noise.

- Cautions 1.** When $\overline{\text{RESET}}$ is active, all pins except $\overline{\text{WDTO}}$, AV_{REF} , AV_{DD} , AV_{SS} , V_{DD} , V_{SS} , X1 , and X2 go into the high-impedance state.
- 2.** When RAM is expanded externally, attach a pull-up resistor to the $\text{P90}/\overline{\text{RD}}$ pin and $\text{P91}/\overline{\text{WR}}$ pin. Otherwise, these pins may go into the high-impedance state, and the contents of the external RAM may be lost. Or signal collision on the address/data bus may damage the input/output circuit.

Figure 15-1. Acceptance of RESET Signal



In reset operation at power-on, a time for stabilized operation between power-on to reset acceptance is required as shown in Figure 15-2.

Figure 15-2. Reset Operation at Power-On

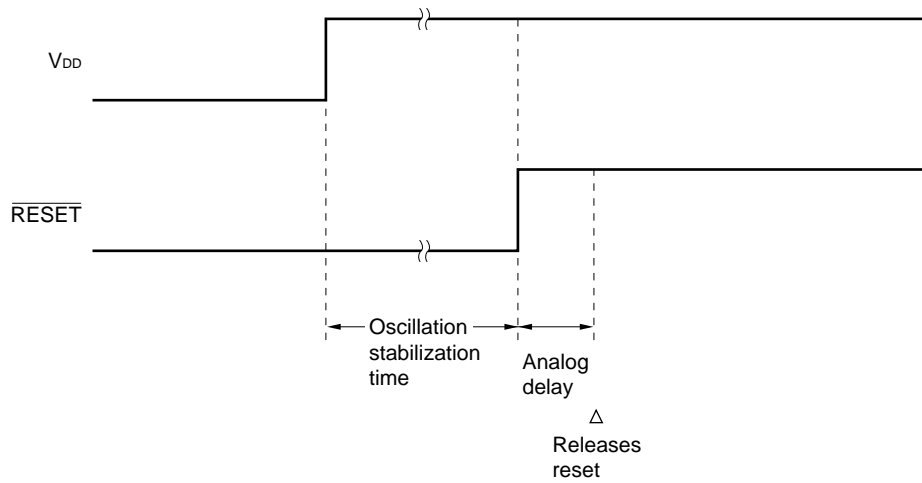


Table 15-1. Hardware Statuses after Reset (1/2)

Hardware			Status after Reset
Control register	Program counter (PC)		Contents of reset vector table (0000H, 0001H) are set
	Stack pointer (SP)		Undefined ^{Note}
	Program status word (PSW)		0000H
	CPU control word (CCW)		00H
Internal RAM	Data memory		Undefined ^{Note}
	General-purpose register (R0-R15)		
Port	Output latch (P0, P1, P3-P5, P8, P9)		Undefined
	Mode register	(PM0, PM1, PM5)	FFH
		(PM3)	×111 1111B
		(PM8)	××11 1111B
		(PM9)	×××× 1111B
	Mode control register	(PMC0)	00H
		(PMC3)	×000 0000B
		(PMC8)	××00 0000B
	Pull-up resistor option register (PUOL, PUOH)		00H
	Port read control register (PRDC)		
Real-time output port (RTP)	Real-time output port register (RTP)		Undefined
	Real-time output port mode register (RTPM)		00H
Real-time pulse unit (RPU)	Timer register (TM0-TM4)		00H
	Timer unit mode register (TUM0, TUM1)		
	Timer control register (TMC0-TMC4)		
	Compare register (CM00-CM03, CM10, CM40, CM41)		Undefined
	Capture register (CT20, CT30, CT31)		
	Capture/compare register (CC20, CC30)		
	Reload register (DTIME)		
	Buffer register (BFCM00-BFCM03, SBUF0-SBUF5, MBUF0-MBUF5)		
	Timer out register (TOUT)		
	Sampling control register (SMPC0, SMPC1)		00H
A/D converter	A/D converter mode register (ADM)		00H
	A/D conversion result register (ADCR0-ADCR7, ADCR0H-ADCR7H)		Undefined

Note When STOP mode is released by $\overline{\text{RESET}}$ input, the value immediately before the STOP mode is set is retained.

Table 15-1. Hardware Statuses after Reset (2/2)

Hardware			Status after Reset
Serial interface	Asynchronous serial interface mode register (ASIM)		80H
	Asynchronous serial interface status register (ASIS)		00H
	Clocked serial interface mode register (CSIM)		
	Serial bus interface control register (SBIC)		
	Serial I/O shift register (SIO)		Undefined
	Serial receive buffer (RXB)		
	Serial transmit shift register (TXS)		
	Baud rate generator compare register (BRG)		
	Baud rate generator control register (BRGC)		00H
PWM output function	PWM control register (PWMC0, PWMC1)		00H
	PWM buffer register (PWM0, PWM0L, PWM1, PWM1L)		Undefined
Watchdog timer	Watchdog timer mode register (WDM)		00H
Interrupt function	External interrupt mode register (INTM0, INTM1)		00H
	Interrupt mode control register (IMC)		80H
	Interrupt mask flag register	(MK0L, MK0H)	FFH
		(MK0)	FFFFH
	Interrupt control register (OVIC3, PIC0-PIC4, TMIC0, CMIC03, CMIC10, CMIC40, CMIC41, SERIC, SRIC, STIC, CSIIC, ADIC)		43H
	In-service priority register (ISPR)		00H
External expansion function	Memory expansion mode register (MM)		Note
	Programmable wait control register (PWC)		C0AAH
CPU control	Standby control register (STBC)		0000 × 000B

Note The condition of the MM register differs after a reset depending on the product.

μPD78363A ... 60H

μPD78365A, 78366A ... 20H

μPD78368A, 78P368A ... 00H

CHAPTER 16 BUS INTERFACE FUNCTION

The bus interface function is to control external memory (ROM, RAM) and I/O devices.

16.1 External Device Expansion Function for μ PD78366A

For the μ PD78366A, external devices (data memory, program memory, and peripheral devices) can be expanded in the external memory area (8000H-F6FFH)^{Note}.

When an external device is connected, port 4 (P40 to P47) is used as the multiplexed address/data bus (AD0 to AD7) and port 5 (P50 to P57) is used as the address bus (A8 to A15) by setting the memory expansion mode register (MM). The \overline{RD} , \overline{WR} , and ASTB signals are used to access the external device.

Table 16-1 lists the pins used for access to an external device and indicates how to assign functions to the pins.

Note μ PD78363A : Address 6000H to FBFFH
 μ PD78368A, 78P368A : Address C000H to F6FFH

Table 16-1. Assigning Functions to Pins (μ PD78366A)

Memory Expansion Mode Register	Pin Function				
MM0-MM2	P40-P47	P50-P57	P90	P91	
Port mode	General-purpose port				
Expansion mode	AD0-AD7	Set to A8-A15 according to the expanded memory size	\overline{RD}	\overline{WR}	

The number of port 5 pins which operate as the address bus can be changed according to the size of memory expanded externally. Pins not used as the address bus can be used as general-purpose I/O ports (refer to **Table 16-2**).

Table 16-2. Operation of Port 5 (expansion mode)

P50	P51	P52	P53	P54	P55	P56	P57	External address space
General-purpose I/O ports								Within 256 bytes
A8	A9	A10	A11					Within 4K bytes
				A12	A13			Within 16K bytes
						A14	A15	Full expansion mode ^{Note}

Note 32K bytes or fewer in case of the μ PD78363A

When an external device reference instruction is executed in the 256-, 4K-, or 16K-byte expansion mode, the system operates as follows. Figures 16-1 through 16-4 show the memory maps for expansion mode.

(1) 256-byte expansion mode

Masks the high-order 8 bits of the 16 external reference address bits and outputs 00H-FFH as address information from pins AD0-AD7.

(2) 4K-byte expansion mode

Masks the high-order 4 bits among 16 external reference address bits and outputs 000H-FFFH as address information from pins A8-A11 and AD0-AD7.

(3) 16K-byte expansion mode

Masks the high-order 2 bits of the 16 external reference address bits and output 0000H-3FFFH as address information from pins A8-A13 and AD0-AD7.

As described above, the 8, 4, and 2 high-order bits of the 16-bit address are masked in the 256-, 4K-, and 16K-byte expansion modes, respectively.

External extension is basically carried out for the area starting from the address following internal ROM to the address preceding internal RAM. However, if the μ PD78363A (internal ROM: 24K bytes) is set to the 16K-byte expansion mode, note that the program counter (PC) differs from the address actually output from pins A13 to A8 and AD7 to AD0 as follows.

PC	A13 to A8, AD7 to AD0
6000H	2000H
⋮	⋮
7FFFH	3FFFH
8000H	0000H
⋮	⋮
9FFFH	1FFFH

If the external ROM addresses are used as consecutive addresses, set the external ROM area to addresses 4000H to 7FFFH. In this case, since the internal ROM and external ROM addresses are not consecutive addresses, it is necessary to use a jump instruction to move the program between each area. The same applies if the external ROM area is set to addresses 8000H to BFFFH.

CHAPTER 16 BUS INTERFACE FUNCTION

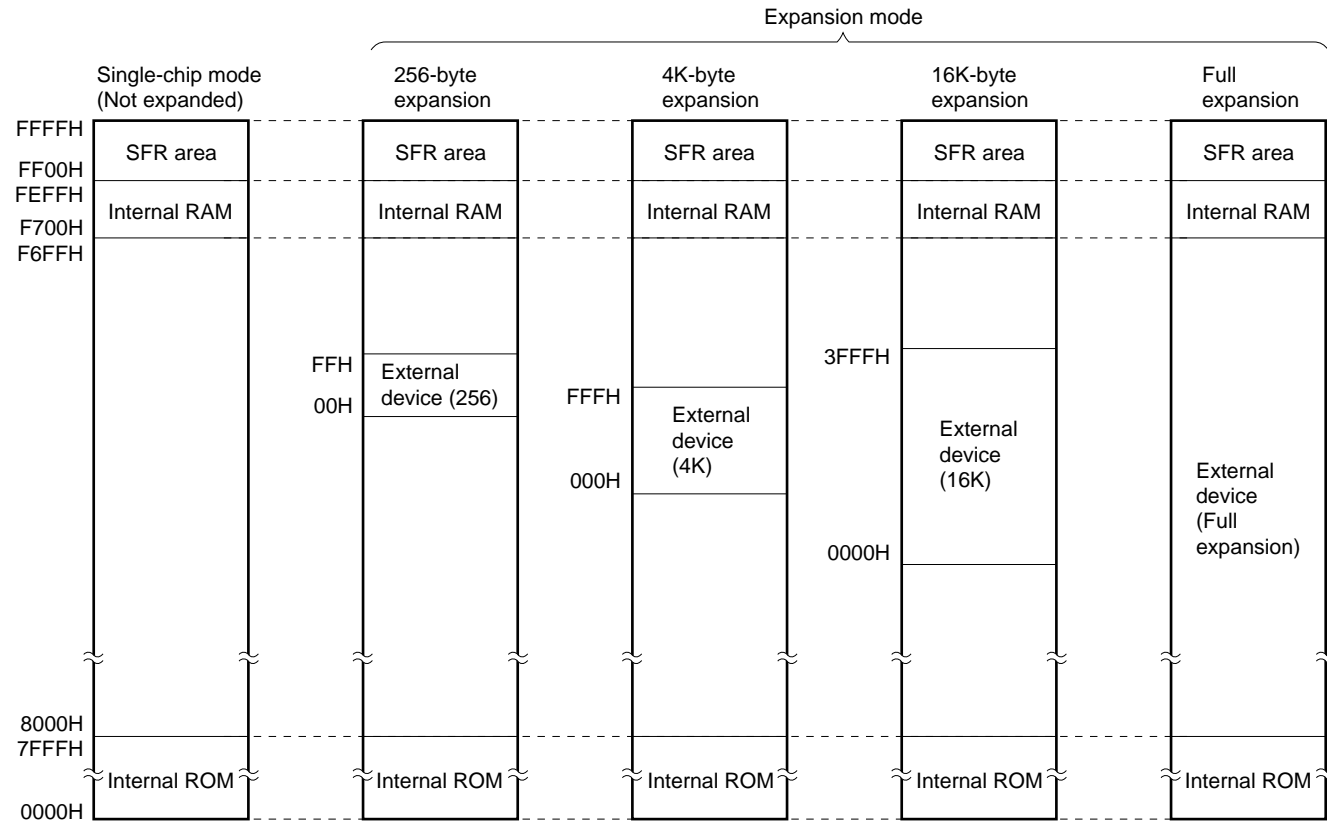


Figure 16-3. Memory Map in Expansion Mode (μ PD78363A)

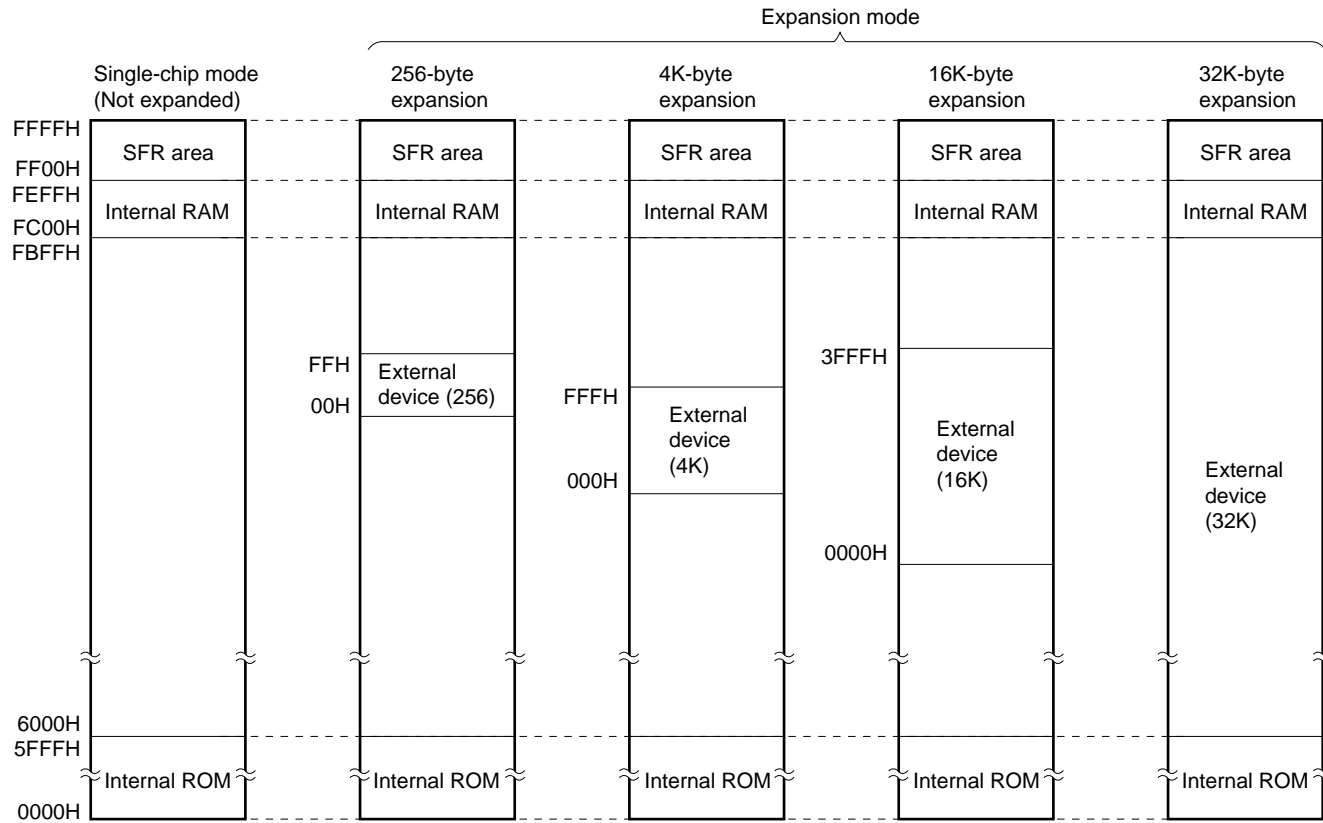
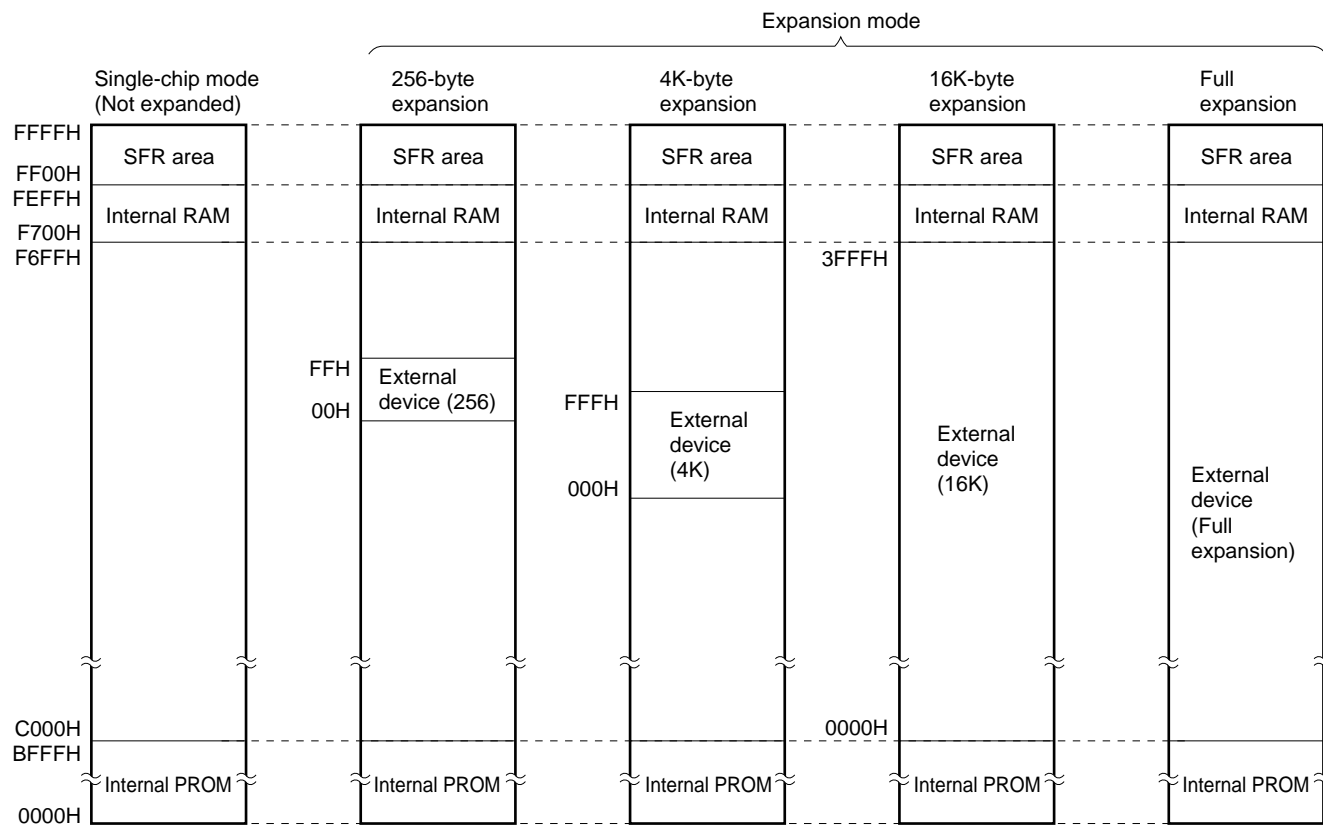


Figure 16-4. Memory Map in Expansion Mode (μ PD78P368A)

Caution The internal memory (ROM, RAM) capacity of the μ PD78P368A can be changed by setting the memory expansion mode register (MM) (see Figure 16-5. Format of Memory Expansion Mode Register).

16.2 Access to External Devices by μ PD78365A

The μ PD78365A does not contain ROM. External devices (data memory, program memory, and peripheral devices) can be expanded in the area (0000H-F6FFH) other than internal RAM.

When an external device is connected, port 4 (P40-P47) is used as the multiplexed address/data bus (AD0-AD7), and port 5 (P50-P57) is used as the address bus (A8-A15). To access the external device, \overline{RD} , \overline{WR} , and ASTB signals are used.

Be sure to fix both the MODE0 and MODE1 pins to the high level.

In the μ PD78365A, specification in the MM0 to MM2 bits of the memory expansion mode register (MM) is invalidated, and ports 4 and 5 always function as AD0-AD7 and A8-A15 pins.

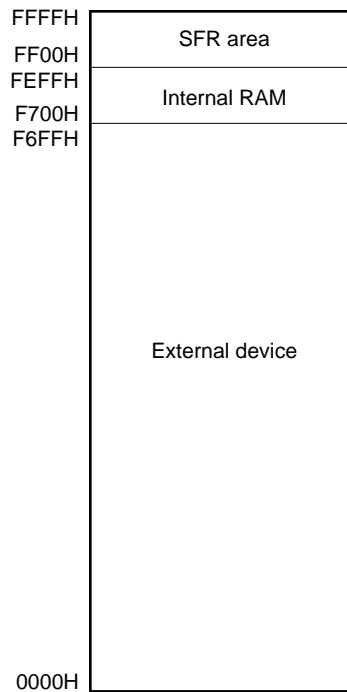
Unlike the μ PD78366A, the memory of the μ PD78365 cannot be expanded in steps.

Table 16-3 shows the pins used to access an external device, and how to set the function of each pin.

Table 16-3. Setting Pin Function (μ PD78365A)

MODE0	MODE1	Pin Function			
		P40-P47	P50-P57	P90	P91
H	H	AD0-AD7	A8-A15	\overline{RD}	\overline{WR}

Figure 16-5. Memory Map of μ PD78365A



16.3 Control Registers

16.3.1 Memory expansion mode register

The memory expansion mode register (MM) is an 8-bit register which controls the address bus, address/data bus, and signals such as \overline{RD} and \overline{WR} when memory or I/O is externally expanded.

The bits MM0 to MM2 are valid only when MODE0, 1 = LL and used to specify the functions of pins of Port 4 and Port 5.

When the expansion mode is specified using bits MM0 to MM2, pins P90 and P91 function as the \overline{RD} and \overline{WR} pins automatically.

The MM register value after \overline{RESET} input depends on the product as follows.

- μ PD78363A ... 60H
- μ PD78365A, 78366A ... 20H
- μ PD78368A, 78P368A ... 00H

Figure16-6 shows the format of the MM register.

Figure 16-6. Format of Memory Expansion Mode Register

Symbol	7	6	5	4	3	2	1	0	Address	When reset	R/W
MM	0	MM6	MM5	0	0	MM2	MM1	MM0	FFC4H	Note 1	R/W

MM2	MM1	MM0	Operation modes of the P40-P47 and P50-P57 pins		
0	0	0	Port mode	Single chip mode	P40-47 : Input port P50-57 : Port mode
0	0	1			P40-47 : Output port P50-57 : Port mode
0	1	0	Setting prohibited		
0	1	1	Expansion mode	256 bytes	P40-47 : Expansion mode P50-57 : Port mode
1	0	0		4K bytes	P40-47 : P50-53 : Expansion mode P54-57 : Port mode
1	0	1		16K bytes	P40-47 : P50-55 : Expansion mode P56-57 : Port mode
1	1	0	Setting prohibited		
1	1	1	Expansion mode	Full Expansion	P40-47, P50-57 : Expansion mode

μPD78P368A only

MM6	MM5	Specifies internal memory capacity of the μPD78P368A ^{Note 2}	
		Internal ROM size	Internal RAM size
0	0	48K bytes	2K bytes
0	1	32K bytes	
1	0		
1	1	24K bytes	768 bytes

Notes 1. The MM register value after reset depends on the product.

μPD78363A ... 60H

μPD78365A, 78366A ... 20H

μPD78368A, 78P368A ... 00H

2. Function to change the internal memory capacity of the μPD78P368A. With the μPD78363A, 78365A, 78366A, 78368A, it is fixed at the state on reset.

Cautions 1. Bits 3, 4, and 7 of the MM register are fixed to “0” by hardware. Even if “1” is written to them, they remain “0”.

2. Invalid combinations are specified as “Setting prohibited” in Figure 16-5. Never write these combinations.

16.3.2 Programmable wait control register

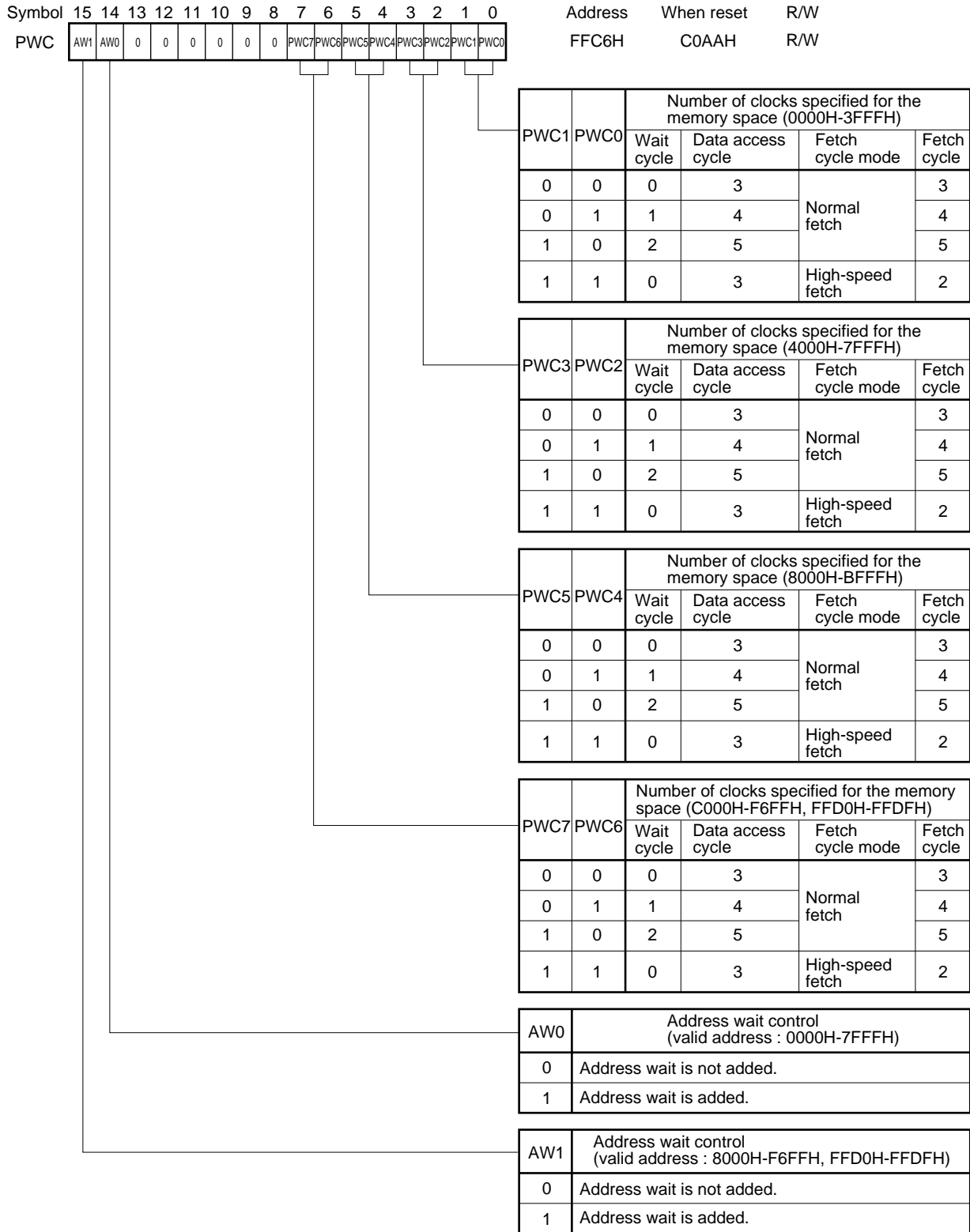
The programmable wait control register (PWC) is a 16-bit register for programmable wait control to the bus cycle (internal^{Note} and external memory access) generated by the μ PD78366A. This register enables low-speed memory and peripheral devices to be connected externally.

The PWC register cannot be accessed in 8-bit mode. Use a 16-bit data transfer instruction to access the PWC register.

Figure 16-7 shows the format of the PWC register.

Note The internal memory refers to the internal ROM area (only for the μ PD78363A, 78366A, 78368A, and 78P368A)

- Cautions**
1. The number of cycles shown in Figure 16-7 is when no address wait cycle is appended. If an address wait cycle is appended, one cycle must be added.
 2. Instruction fetch from and data access to the peripheral RAM area (F700H-FDFFH, FC00H-FDFFH for μ PD78363A) are enabled, but wait specification by setting the PWC register is invalid. The peripheral RAM area operates with a 16-bit bus. Instruction fetch becomes high-speed fetch.
 3. Instructions cannot be fetched from the main RAM area (FE00H-FEFFFH). Wait specification by setting the PWC register when data is accessed is invalid. The main RAM area is accessed in a 16-bit units. (For bus cycles, special two bus cycles are started.)
 4. To make a word access to the main RAM area (containing stack handling), only even addresses can be specified in operands.
 5. The internal ROM area operates with 16-bit bus regardless how the PWC register is set. Wait specification can be set in the PWC register.
 6. To access external memory, do not specify high-speed fetch.
 7. External SFR area (FFD0H-FFDFH) wait specification is set in the PWC7 and PWC6 bits.

Figure 16-7. Format of Programmable Wait Control Register

- Cautions**
1. To access the internal memory in the high-speed fetch mode, set 0 in the AW0 and AW1 bits of the PWC register.
 2. Bits 8-13 of the PWC register are fixed to "0" by hardware. Even if "1" is written to them, they remain "0".

[MEMO]

CHAPTER 17 PROGRAMMING FOR μ PD78P368A

The μ PD78P368A contains an electrically writable PROM of 49152×8 bits.

Use the MODE0/ V_{PP} , and MODE1 pins to set the μ PD78P368A to the PROM programming mode when programming on the PROM.

The μ PD78P368A provides programming characteristics which are compatible with the μ PD27C1001A.

Table 17-1. Pin Functions in Programming Mode

Function	Normal Operating Mode	Programming Mode
Address input	P00-P07, P21, P20, P80-P85, P30	A0-A16
Data input	P40-P47	D0-D7
Program pulse	ASTB	$\overline{\text{PGM}}$
Chip enable	P91	$\overline{\text{CE}}$
Output enable	P90	$\overline{\text{OE}}$
Program voltage	MODE0/ V_{PP}	
Mode control	MODE1	

17.1 Operating Mode

When the MODE0/V_{PP} pin is set to H, and the MODE1 pin is set to L, the μ PD78P368A enters the programming write/verify mode. This mode varies to each operating mode shown in Table 17-2 according to how to set the $\overline{\text{CE}}$, $\overline{\text{OE}}$, and $\overline{\text{PGM}}$ pins.

Setting the μ PD78P368A to the read mode enables it to read the contents of PROM.

Process the unused pins by referring to **1.4 Pin Configuration (Top View)**.

Table 17-2. Operating Modes for PROM Programming

Mode	MODE1	$\overline{\text{CE}}$	$\overline{\text{OE}}$	$\overline{\text{PGM}}$	V _{PP} /MODE0	V _{DD}	D0-D7
Page data latch	L	H	L	H	+12.5 V	+6.5 V	Data input
Page program		H	H	L			High impedance
Byte program		L	H	L			Data input
Program verify		L	L	H			Data output
Program inhibit		×	L	L			High impedance
		×	H	H			
Read		L	L	H	+5 V	+5 V	Data output
Output disable		L	H	×			High impedance
Standby		H	×	×			High impedance

Remark × indicates L or H.

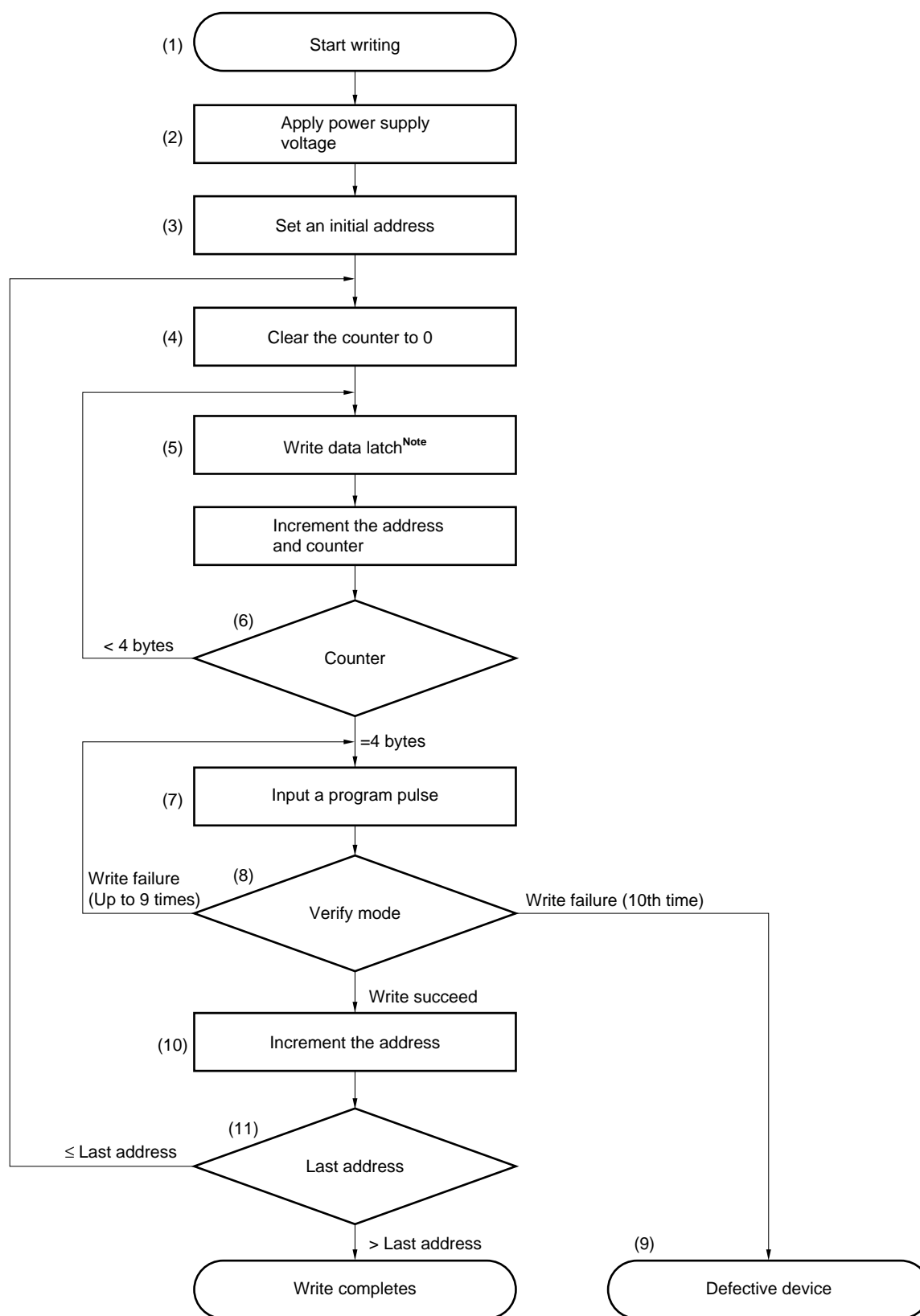
17.2 Procedure for Writing on PROM (Page Program Mode)

The following is a procedure for writing on PROM (refer to **Figure 17-1**).

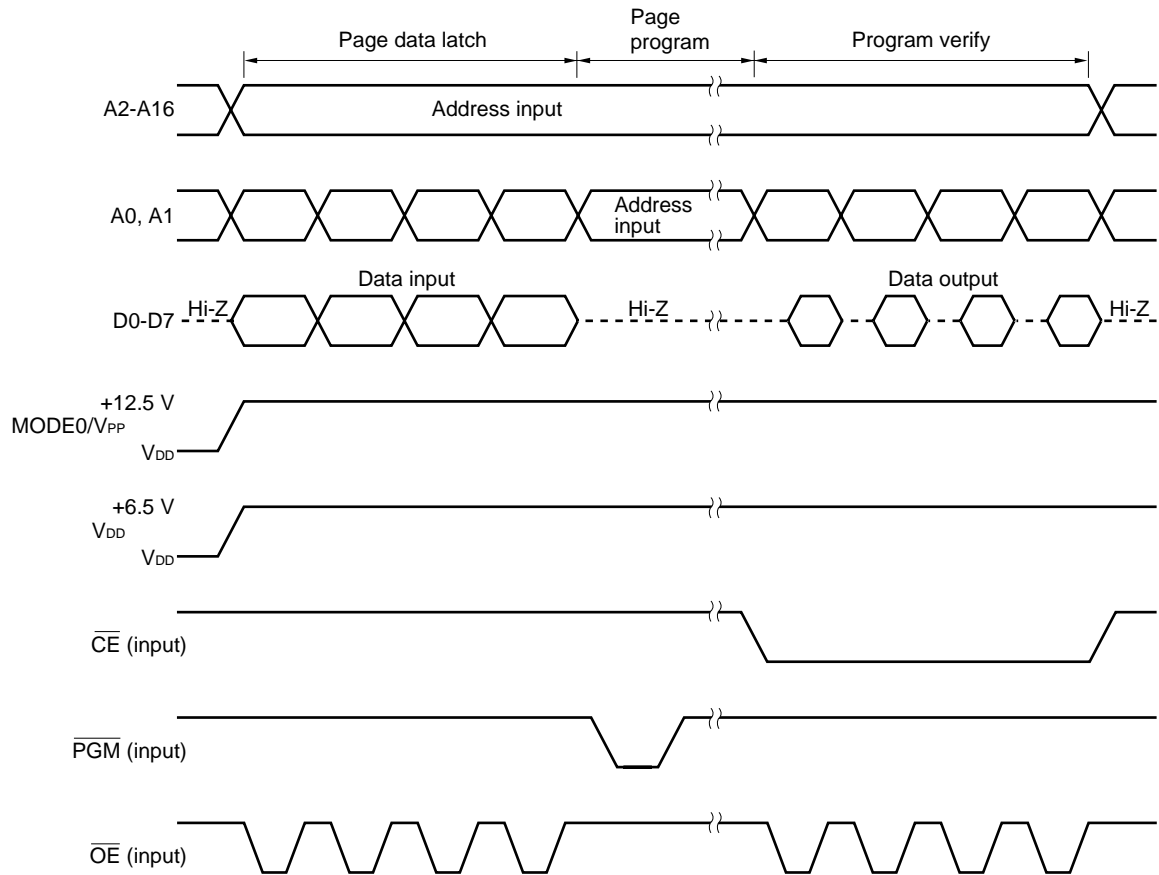
In the page program mode, data is written in units of pages (four bytes). When write data completes midway of a page, latch FFH after the data so that the data fits into pages.

- (1) Always set each pin as follows: MODE0/V_{PP} = H, MODE1 = L. Process unused pins by referring to **1.4 Pin Configuration (Top View)**.
- (2) Apply +6.5 V to the V_{DD} pin and +12.5 V to the MODE0/V_{PP} pin.
- (3) Input an initial address to the A0 to A16 pins.
- (4) Clear the page counter.
- (5) Data latch mode. Input write data to the D0 to D7 pins and input an active-low pulse to the $\overline{\text{OE}}$ pin. Increment the address and the page counter.
- (6) Repeat step (5) for a page (four bytes).
- (7) Input a 0.1 ms program pulse (active low) to the $\overline{\text{PGM}}$ pin.
- (8) Verify mode. Checks if data has been written in PROM.
Apply a low level to the $\overline{\text{CE}}$ pin, input an active-low pulse to the $\overline{\text{OE}}$ pin, and then read the write data from the D0 to D7 pins. Repeat this for a page (four bytes). When verification completes, apply a high level to the $\overline{\text{CE}}$ pin.
 - If data has been written, go to step (10).
 - If not, repeat steps (7) and (8). If no data is written yet after the steps have been repeated 10 times, go to step (9).
- (9) Assume the device to be defective and stop write operation.
- (10) Increment the address.
- (11) Repeat steps (4) to (10) until the address exceeds the last address.

Figure 17-2 is a timing chart of these steps (2) to (9).

Figure 17-1. Flowchart of Procedure for Writing (page program mode)

Note When write data completes midway of a page, latch FFH after the data so that the data fits into pages.

Figure 17-2. Timing Chart of PROM Write and Verify Operation (page program mode)

17.3 Procedure for Writing on PROM (Byte Program Mode)

The procedure for writing data into PROM is as follows (refer to **Figure 17-3**).

- (1) Always set each pin as follows: $\text{MODE0}/V_{PP} = \text{H}$, and $\text{MODE1} = \text{L}$. Connect unused pins by referring to **1.4 Pin Configuration (Top View)**.
- (2) Apply +6.5 V to the V_{DD} pin and +12.5 V to the $\text{MODE0}/V_{PP}$ pin. Input a low level to the $\overline{\text{CE}}$ pin.
- (3) Input an initial address to the A0 to A16 pins.
- (4) Input write data to the D0 to D7 pins.
- (5) Input a program pulse (active low) which has a period of 0.1 ms to the $\overline{\text{PGM}}$ pin.
- (6) Verify mode. Check that data has been written in PROM.
Input an active-low pulse to the $\overline{\text{OE}}$ pin and read the written data from the D0 to D7 pins.
 - When data has been written, go to step (8).
 - If not, repeat steps (4) to (6). If no data is written after the steps are repeated ten times, go to step (7).
- (7) Assume the device to be defective and stop write operation.
- (8) Increment the address.
- (9) Repeat steps (4) to (8) until the address exceeds the last address.

Figure 17-4 is a timing chart of steps (2) to (7) above.

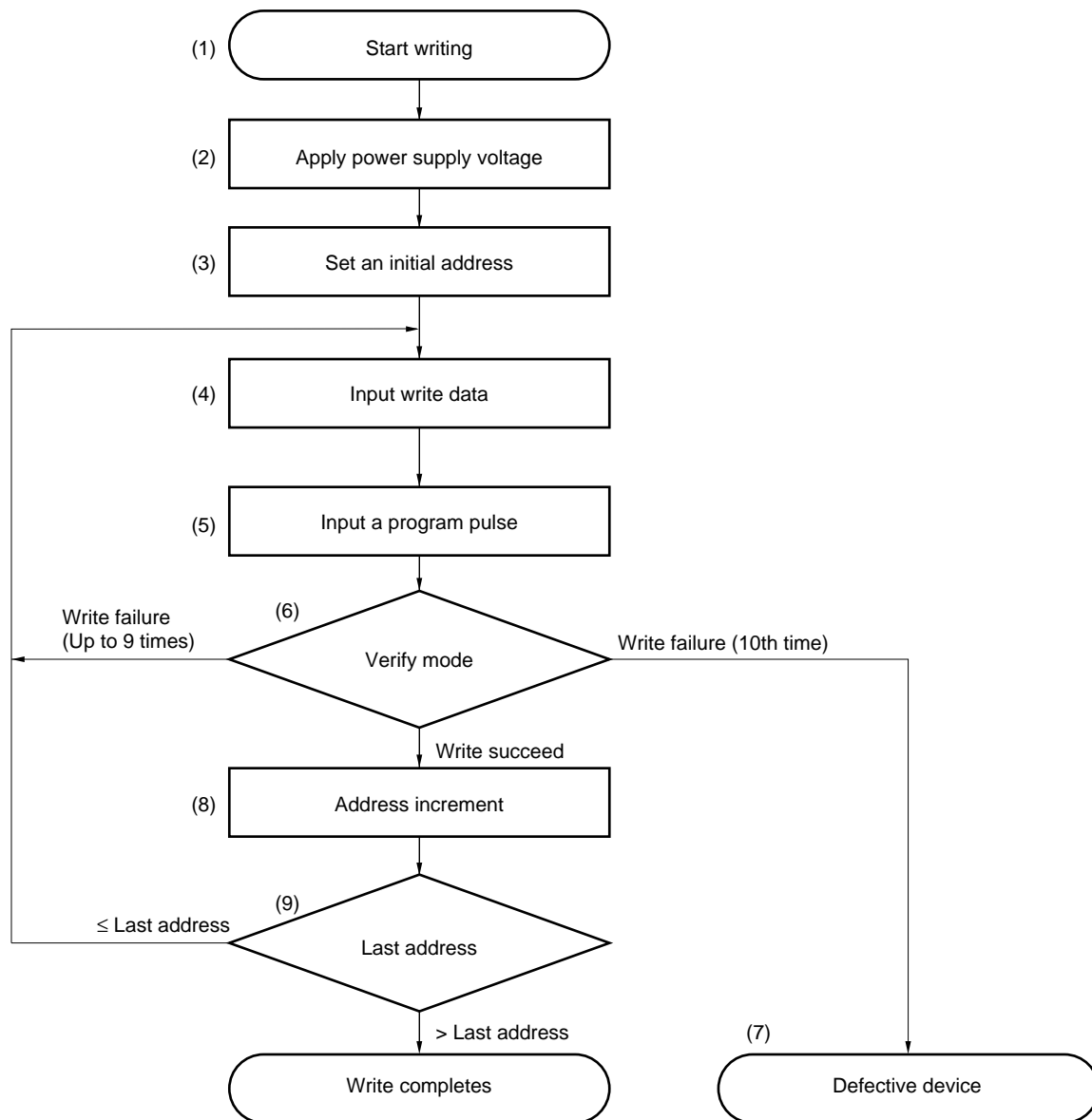
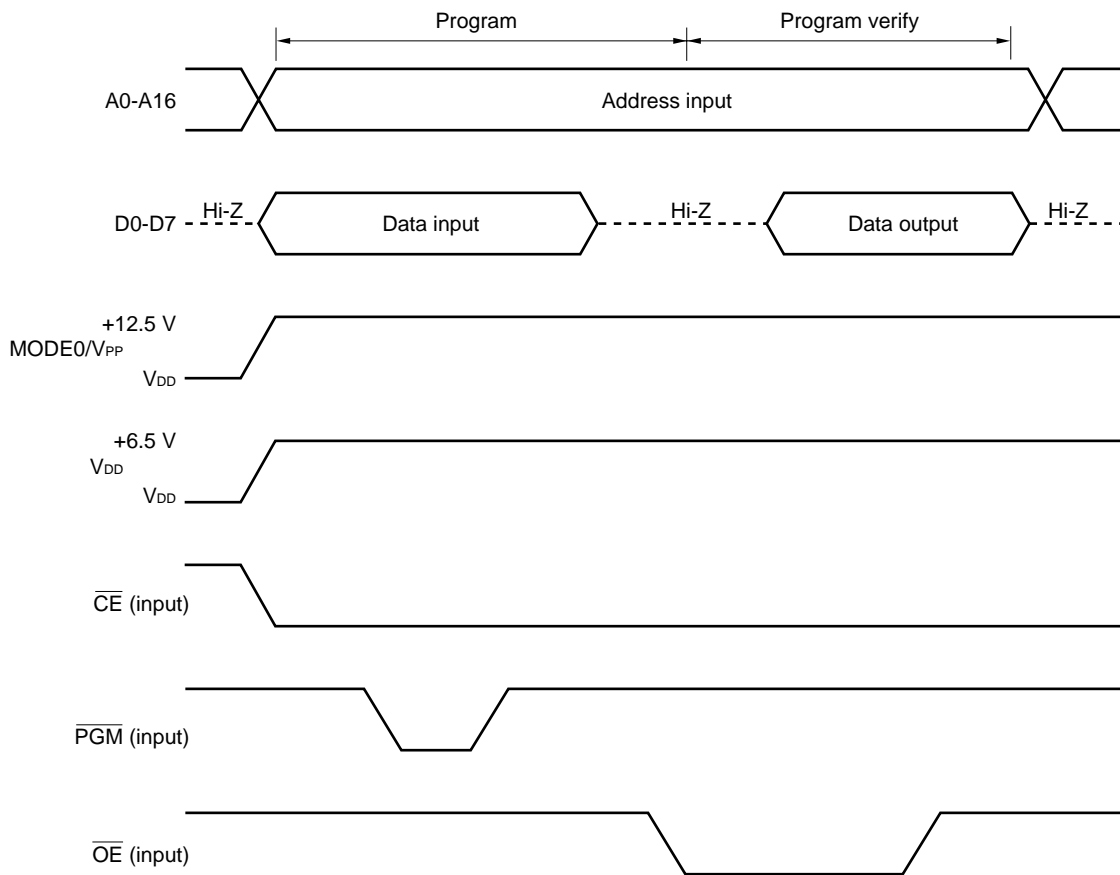
Figure 17-3. Flowchart of Write Operation (byte program mode)

Figure 17-4. Timing Chart of PROM Write and Verify Operation (byte program mode)

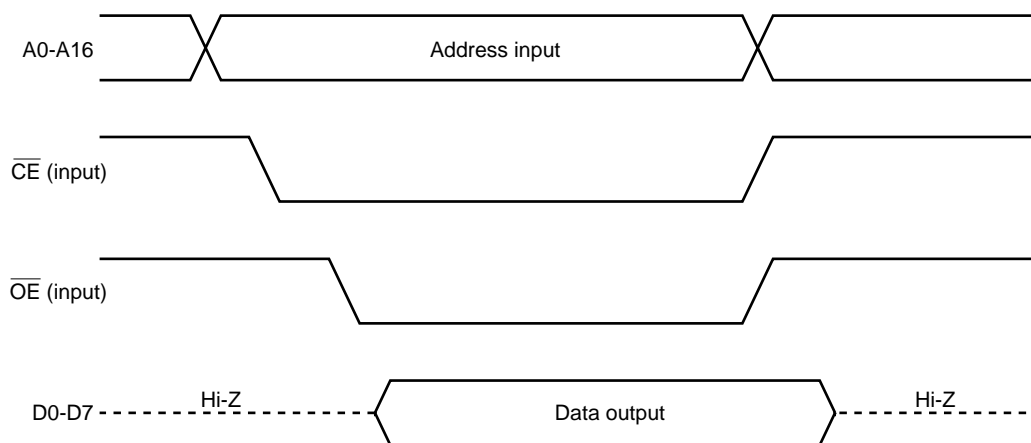
17.4 Procedure for Reading from PROM

The contents of PROM can be read out to the external data bus (D0 to D7) in the following steps:

- (1) Always set the MODE0/V_{PP} pin to H, and the MODE1 pin to L. Handle other unused pins by referring to **1.4 Pin Configuration (Top View)**.
- (2) Apply +5 V to the V_{DD} and MODE0/V_{PP} pins.
- (3) Input the address of data to be read into the A0 to A16 pins.
- (4) Read mode ($\overline{CE} = L$, $\overline{OE} = L$)
- (5) Output the data on the D0 to D7 pins.

Figure 17-5 is a timing chart of these steps (2) to (5).

Figure 17-5. PROM Read Timing Chart



17.5 Erasure Characteristics (μ PD78P368AKL only)

The μ PD78P368AKL allows the data written to the program memory to be erased (FFH) and new data to be rewritten.

To erase the data, expose the erasure window of the program memory to a light having a wavelength of shorter than 400 nm. Usually, ultraviolet ray with a wavelength of 254 nm is used. The quantity of light necessary for erasing the data completely is as follows:

- Ultraviolet ray intensity \times erasure time: 15 W \cdot s/cm² min.
- Erasure time: 15 to 20 minutes (with an ultraviolet lamp of 12,000 μ W/cm². Note, however, the erasure time is extended depending on the performances of the ultraviolet lamp used, and dirt collecting on the erasure window.)

To erase the data, locate the ultraviolet lamp within 2.5 cm from the erasure window. If a filter is attached to the ultraviolet lamp, remove the filter.

17.6 Sticker on Erasure Window (μ PD78P368AKL only)

If the erasure window of the μ PD78P368AKL is exposed to sun light or the light of a fluorescent lamp for a long time, the data of the EPROM may be erased, or the internal circuit may malfunction. To prevent this, usually attach a protection sticker to the erasure window.

The package product with EPROM erasure window is supplied with a protection sticker whose quality is guaranteed by NEC.

17.7 Screening of One-Time PROM Products

The one-time PROM product (μ PD78P368AGF-3B9) cannot be completely tested by NEC for shipment because of its structure. It is therefore recommended that screening is conducted by verifying the PROM after the PROM has been stored under the following conditions:

Storage Temperature	Storage Time
125 °C	24 hours

NEC provides a one-time PROM writing, marking, screening, and verification service under the name of QTOP™ Microcontrollers. For details, consult our salesperson.

CHAPTER 18 INSTRUCTION SET

This chapter describes each instruction operation of the μ PD78366A.

For details of the operation, the operation code and clock cycles, refer to the μ PD78356 **User's Manual-Instruction (U12117E)**.

18.1 Operand Identifier and Format

Operands are written in the operand field of each instruction as listed in the column of Table 18-1. For details of the operand format, refer to the relevant assembler specifications. When several formats are presented, any one of them can be selected. Uppercase letters and the symbols, +, −, #, \$, !, and [], are keywords and must be written as they are.

For immediate data, an appropriate numeric or label must be written. The symbols #, \$, !, and [] must not be omitted when writing labels.

Table 18-1. Operand Identifier and Format

Identifier	Format
r	R0, R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13, R14, R15
r1	R0, R1, R2, R3, R4, R5, R6, R7
r2	C, B
rp	RP0, RP1, RP2, RP3, RP4, RP5, RP6, RP7
rp1	RP0, RP1, RP2, RP3, RP4, RP5, RP6, RP7
rp2	DE, HL, VP, UP
sfr	Special function register name (refer to Table 3-4).
sfrp	Special function register name (16-bit manipulation register. Refer to Table 3-4).
post	RP0, RP1, RP2, RP3, RP4, RP5/PSW, RP6, RP7 (Can be used more than once. However, RP5 can only be used in a PUSH or POP instruction and PSW can only be used in a PUSHU or POPU instruction.)
mem	[DE], [HL], [DE+], [HL+], [DE-], [HL-], [VP], [UP]: Register indirect mode [DE + A], [HL + A], [DE + B], [HL + B], [VP + DE], [VP + HL]: Based indexed mode [DE + byte], [HL + byte], [VP + byte], [UP + byte], [SP + byte]: Based mode word [A], word [B], word [DE], word [HL]: Indexed mode
saddr	FE20H-FF1FH Immediate data or label
saddrp	FE20H-FF1EH Immediate data (bit 0 = 0) or label (for 16-bit manipulation)
\$addr16	0000H-FDFFH Immediate data or label: Relative addressing
!addr16	0000H-FDFFH Immediate data or label: Immediate addressing (Data at an address up to FFFFH can be used in an MOV instruction. Data at an address from FE00H to FEFFH can be used in an MOVTBLW instruction.)
addr11	800H-FFFH Immediate data or label
addr5	40H-7EH Immediate data (bit 0 = 0) ^{Note} 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 (0 to 7)

Note Do not attempt to access word data at an odd-numbered address (bit 0 = 1).

- Remarks**
1. The same register name can be specified in rp and rp1, but different codes are generated.
 2. Immediate addressing is effective for entire address spaces. Relative addressing is effective for the locations within a displacement range of -128 to +127 from the starting address of the next instruction.

The 8-bit registers (r, r1) and 16-bit register pairs (rp, rp1, post) can be represented by either absolute names (R0-R15, RP0-RP7) or function names. Table 18-2 and Table 18-3 list the absolute names and corresponding function names.

Table 18-2. Absolute Names and Their Corresponding Function Names of 8-bit Register

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

Table 18-3. Absolute Names and Their Corresponding Function Names of 16-bit Register

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

RSS stands for the register set selection flag (bit 5 of PSW). Setting or resetting RSS switches function names for correspondence with an absolute name.

18.2 Legend

A	: A register; 8-bit accumulator
X	: X register
B	: B register
C	: C register
D	: D register
E	: E register
H	: H register
L	: L register
R0-R15	: Register 0 to register 15 (absolute name)
AX	: Register pair (AX); 16 bit accumulator
BC	: Register pair (BC)
DE	: Register pair (DE)
HL	: Register pair (HL)
RP0-RP7	: Register pair 0 to register pair 7 (absolute name)
PC	: Program counter
SP	: Stack pointer
UP	: User stack pointer
PSW	: Program status word
CY	: Carry flag
AC	: Auxiliary carry flag
Z	: Zero flag
P/V	: Parity/overflow flag
S	: Sign flag
TPF	: Table position flag
RBS	: Register bank selecting flag
RSS	: Register set selecting flag
IE	: Interrupt request enable flag
STBC	: Standby control register
WDM	: Watchdog timer mode register
jdisp8	: Signed 8-bit data (displacement value: -128 to +127)
()	: Contents at an address enclosed in parentheses or at an address indicated in a register indicated in parentheses. (+) and (-) indicate that an address or the contents of a register indicated in parentheses are incremented and decremented by one after execution of the instruction, respectively.
(())	: Contents at an address indicated by the contents at an address indicated in parentheses (()).
xxH	: Hexadecimal number
xH, xL	: High-order 8 bits and low-order 8 bits of 16-bit register

18.3 Notational Symbols in Flag Operation Field

Table 18-4. Notational Symbols in Flag Operation Field

Symbol	Explanation
(Blank)	No change
0	Cleared to zero.
1	Set to 1.
×	Set or clear according to the result.
P	P/V flag operates as a parity flag.
V	P/V flag operates as an overflow flag.
R	Saved values are restored.

18.4 Differences between μ PD78366A and μ PD78328 in Instruction Set

The instruction set of the μ PD78366A has the following four additional instructions which are not provided to the μ PD78328. The other instructions are the same as those of the μ PD78328.

- Sum-of-products instruction
- Sum-of-products instruction with saturation function
- Relative operation instruction
- Table shift instruction

18.5 Operations of Basic Instructions

(1) 8-bit data transfer instructions: MOV, XCH

Mnemonic	Operand	Byte	Operation	Flag					
				S	Z	AC	P/V	CY	
MOV	r1, #byte	2	$r1 \leftarrow \text{byte}$						
	saddr, #byte	3	$(saddr) \leftarrow \text{byte}$						
	sfr ^{Note} , #byte	3	$\text{sfr} \leftarrow \text{byte}$						
	r, r1	2	$r \leftarrow r1$						
	A, r1	1	$A \leftarrow r1$						
	A, saddr	2	$A \leftarrow (saddr)$						
	saddr, A	2	$(saddr) \leftarrow A$						
	saddr, saddr	3	$(saddr) \leftarrow (saddr)$						
	A, sfr	2	$A \leftarrow \text{sfr}$						
	sfr, A	2	$\text{sfr} \leftarrow A$						
	A, mem	1-4	$A \leftarrow (\text{mem})$						
	mem, A	1-4	$(\text{mem}) \leftarrow A$						
	A, [saddrp]	2	$A \leftarrow ((saddrp))$						
	[saddrp], A	2	$((saddrp)) \leftarrow A$						
	A, !addr16	4	$A \leftarrow (\text{addr16})$						
	!addr16, A	4	$(\text{addr16}) \leftarrow A$						
	PSWL, #byte	3	$\text{PSWL} \leftarrow \text{byte}$	x	x	x	x	x	
	PSWH, #byte	3	$\text{PSWH} \leftarrow \text{byte}$						
	PSWL, A	2	$\text{PSWL} \leftarrow A$	x	x	x	x	x	
	PSWH, A	2	$\text{PSWH} \leftarrow A$						
	A, PSWL	2	$A \leftarrow \text{PSWL}$						
	A, PSWH	2	$A \leftarrow \text{PSWH}$						
XCH	A, r1	1	$A \leftrightarrow r1$						
	r, r1	2	$r \leftrightarrow r1$						
	A, mem	2-4	$A \leftrightarrow (\text{mem})$						
	A, saddr	2	$A \leftrightarrow (saddr)$						
	A, sfr	3	$A \leftrightarrow \text{sfr}$						
	A, [saddrp]	2	$A \leftrightarrow ((saddrp))$						
	saddr, saddr	3	$(saddr) \leftrightarrow (saddr)$						

Note If STBC or WDM is coded in sfr, a different instruction having the different byte count is generated.

(2) 16-bit data transfer instructions: MOVW, XCHW

Mnemonic	Operand	Byte	Operation	Flag				
				S	Z	AC	P/V	CY
MOVW	rp1, #word	3	rp1 ← word					
	saddrp, #word	4	(saddrp) ← word					
	sfrp, #word	4	sfrp ← word					
	rp, rp1	2	rp ← rp1					
	AX, saddrp	2	AX ← (saddrp)					
	saddrp, AX	2	(saddrp) ← AX					
	saddrp, saddrp	3	(saddrp) ← (saddrp)					
	AX, sfrp	2	AX ← sfrp					
	sfrp, AX	2	sfrp ← AX					
	rp1, !addr16	4	rp1 ← (addr16)					
	!addr16, rp1	4	(addr16) ← rp1					
	AX, mem	2-4	AX ← (mem)					
	mem, AX	2-4	(mem) ← AX					
XCHW	AX, saddrp	2	AX ↔ (saddrp)					
	AX, sfrp	3	AX ↔ sfrp					
	saddrp, saddrp	3	(saddrp) ↔ (saddrp)					
	rp, rp1	2	rp ↔ rp1					
	AX, mem	2-4	AX ↔ (mem)					

(3) 8-bit arithmetic/logical instructions: ADD, ADDC, SUB, SUBC, AND, OR, XOR, CMP

(1/2)

Mnemonic	Operand	Byte	Operation	Flag					
				S	Z	AC	P/V	CY	
ADD	A, #byte	2	$A, CY \leftarrow A + \text{byte}$	x	x	x	V	x	
	saddr, #byte	3	$(saddr), CY \leftarrow (saddr) + \text{byte}$	x	x	x	V	x	
	sfr, #byte	4	$sfr, CY \leftarrow sfr + \text{byte}$	x	x	x	V	x	
	r, r1	2	$r, CY \leftarrow r + r1$	x	x	x	V	x	
	A, saddr	2	$A, CY \leftarrow A + (saddr)$	x	x	x	V	x	
	A, sfr	3	$A, CY \leftarrow A + sfr$	x	x	x	V	x	
	saddr, saddr	3	$(saddr), CY \leftarrow (saddr) + (saddr)$	x	x	x	V	x	
	A, mem	2-4	$A, CY \leftarrow A + (\text{mem})$	x	x	x	V	x	
	mem, A	2-4	$(\text{mem}), CY \leftarrow (\text{mem}) + A$	x	x	x	V	x	
ADDC	A, #byte	2	$A, CY \leftarrow A + \text{byte} + CY$	x	x	x	V	x	
	saddr, #byte	3	$(saddr), CY \leftarrow (saddr) + \text{byte} + CY$	x	x	x	V	x	
	sfr, #byte	4	$sfr, CY \leftarrow sfr + \text{byte} + CY$	x	x	x	V	x	
	r, r1	2	$r, CY \leftarrow r + r1 + CY$	x	x	x	V	x	
	A, saddr	2	$A, CY \leftarrow A + (saddr) + CY$	x	x	x	V	x	
	A, sfr	3	$A, CY \leftarrow A + sfr + CY$	x	x	x	V	x	
	saddr, saddr	3	$(saddr), CY \leftarrow (saddr) + (saddr) + CY$	x	x	x	V	x	
	A, mem	2-4	$A, CY \leftarrow A + (\text{mem}) + CY$	x	x	x	V	x	
	mem, A	2-4	$(\text{mem}), CY \leftarrow (\text{mem}) + A + CY$	x	x	x	V	x	
SUB	A, #byte	2	$A, CY \leftarrow A - \text{byte}$	x	x	x	V	x	
	saddr, #byte	3	$(saddr), CY \leftarrow (saddr) - \text{byte}$	x	x	x	V	x	
	sfr, #byte	4	$sfr, CY \leftarrow sfr - \text{byte}$	x	x	x	V	x	
	r, r1	2	$r, CY \leftarrow r - r1$	x	x	x	V	x	
	A, saddr	2	$A, CY \leftarrow A - (saddr)$	x	x	x	V	x	
	A, sfr	3	$A, CY \leftarrow A - sfr$	x	x	x	V	x	
	saddr, saddr	3	$(saddr), CY \leftarrow (saddr) - (saddr)$	x	x	x	V	x	
	A, mem	2-4	$A, CY \leftarrow A - (\text{mem})$	x	x	x	V	x	
	mem, A	2-4	$(\text{mem}), CY \leftarrow (\text{mem}) - A$	x	x	x	V	x	
SUBC	A, #byte	2	$A, CY \leftarrow A - \text{byte} - CY$	x	x	x	V	x	
	saddr, #byte	3	$(saddr), CY \leftarrow (saddr) - \text{byte} - CY$	x	x	x	V	x	
	sfr, #byte	4	$sfr, CY \leftarrow sfr - \text{byte} - CY$	x	x	x	V	x	
	r, r1	2	$r, CY \leftarrow r - r1 - CY$	x	x	x	V	x	
	A, saddr	2	$A, CY \leftarrow A - (saddr) - CY$	x	x	x	V	x	
	A, sfr	3	$A, CY \leftarrow A - sfr - CY$	x	x	x	V	x	
	saddr, saddr	3	$(saddr), CY \leftarrow (saddr) - (saddr) - CY$	x	x	x	V	x	
	A, mem	2-4	$A, CY \leftarrow A - (\text{mem}) - CY$	x	x	x	V	x	
	mem, A	2-4	$(\text{mem}), CY \leftarrow (\text{mem}) - A - CY$	x	x	x	V	x	

(2/2)

Mnemonic	Operand	Byte	Operation	Flag				
				S	Z	AC	P/V	CY
AND	A, #byte	2	$A \leftarrow A \wedge \text{byte}$	x	x		P	
	saddr, #byte	3	$(\text{saddr}) \leftarrow (\text{saddr}) \wedge \text{byte}$	x	x		P	
	sfr, #byte	4	$\text{sfr} \leftarrow \text{sfr} \wedge \text{byte}$	x	x		P	
	r, r1	2	$r \leftarrow r \wedge r1$	x	x		P	
	A, saddr	2	$A \leftarrow A \wedge (\text{saddr})$	x	x		P	
	A, sfr	3	$A \leftarrow A \wedge \text{sfr}$	x	x		P	
	saddr, saddr	3	$(\text{saddr}) \leftarrow (\text{saddr}) \wedge (\text{saddr})$	x	x		P	
	A, mem	2-4	$A \leftarrow A \wedge (\text{mem})$	x	x		P	
	mem, A	2-4	$(\text{mem}) \leftarrow (\text{mem}) \wedge A$	x	x		P	
OR	A, #byte	2	$A \leftarrow A \vee \text{byte}$	x	x		P	
	saddr, #byte	3	$(\text{saddr}) \leftarrow (\text{saddr}) \vee \text{byte}$	x	x		P	
	sfr, #byte	4	$\text{sfr} \leftarrow \text{sfr} \vee \text{byte}$	x	x		P	
	r, r1	2	$r \leftarrow r \vee r1$	x	x		P	
	A, saddr	2	$A \leftarrow A \vee (\text{saddr})$	x	x		P	
	A, sfr	3	$A \leftarrow A \vee \text{sfr}$	x	x		P	
	saddr, saddr	3	$(\text{saddr}) \leftarrow (\text{saddr}) \vee (\text{saddr})$	x	x		P	
	A, mem	2-4	$A \leftarrow A \vee (\text{mem})$	x	x		P	
	mem, A	2-4	$(\text{mem}) \leftarrow (\text{mem}) \vee A$	x	x		P	
XOR	A, #byte	2	$A \leftarrow A \nabla \text{byte}$	x	x		P	
	saddr, #byte	3	$(\text{saddr}) \leftarrow (\text{saddr}) \nabla \text{byte}$	x	x		P	
	sfr, #byte	4	$\text{sfr} \leftarrow \text{sfr} \nabla \text{byte}$	x	x		P	
	r, r1	2	$r \leftarrow r \nabla r1$	x	x		P	
	A, saddr	2	$A \leftarrow A \nabla (\text{saddr})$	x	x		P	
	A, sfr	3	$A \leftarrow A \nabla \text{sfr}$	x	x		P	
	saddr, saddr	3	$(\text{saddr}) \leftarrow (\text{saddr}) \nabla (\text{saddr})$	x	x		P	
	A, mem	2-4	$A \leftarrow A \nabla (\text{mem})$	x	x		P	
	mem, A	2-4	$(\text{mem}) \leftarrow (\text{mem}) \nabla A$	x	x		P	
CMP	A, #byte	2	$A - \text{byte}$	x	x	x	V	x
	saddr, #byte	3	$(\text{saddr}) - \text{byte}$	x	x	x	V	x
	sfr, #byte	4	$\text{sfr} - \text{byte}$	x	x	x	V	x
	r, r1	2	$r - r1$	x	x	x	V	x
	A, saddr	2	$A - (\text{saddr})$	x	x	x	V	x
	A, sfr	3	$A - \text{sfr}$	x	x	x	V	x
	saddr, saddr	3	$(\text{saddr}) - (\text{saddr})$	x	x	x	V	x
	A, mem	2-4	$A - (\text{mem})$	x	x	x	V	x
	mem, A	2-4	$(\text{mem}) - A$	x	x	x	V	x

(4) 16-bit arithmetic/logical instructions: ADDW, SUBW, CMPW

Mnemonic	Operand	Byte	Operation	Flag					
				S	Z	AC	P/V	CY	
ADDW	AX, #word	3	$AX, CY \leftarrow AX + \text{word}$	x	x	x	V	x	
	saddrp, #word	4	$(\text{saddrp}), CY \leftarrow (\text{saddrp}) + \text{word}$	x	x	x	V	x	
	sfrp, #word	5	$\text{sfrp}, CY \leftarrow \text{sfrp} + \text{word}$	x	x	x	V	x	
	rp, rp1	2	$\text{rp}, CY \leftarrow \text{rp} + \text{rp1}$	x	x	x	V	x	
	AX, saddrp	2	$AX, CY \leftarrow AX + (\text{saddrp})$	x	x	x	V	x	
	AX, sfrp	3	$AX, CY \leftarrow AX + \text{sfrp}$	x	x	x	V	x	
	saddrp, saddrp	3	$(\text{saddrp}), CY \leftarrow (\text{saddrp}) + (\text{saddrp})$	x	x	x	V	x	
SUBW	AX, #word	3	$AX, CY \leftarrow AX - \text{word}$	x	x	x	V	x	
	saddrp, #word	4	$(\text{saddrp}), CY \leftarrow (\text{saddrp}) - \text{word}$	x	x	x	V	x	
	sfrp, #word	5	$\text{sfrp}, CY \leftarrow \text{sfrp} - \text{word}$	x	x	x	V	x	
	rp, rp1	2	$\text{rp}, CY \leftarrow \text{rp} - \text{rp1}$	x	x	x	V	x	
	AX, saddrp	2	$AX, CY \leftarrow AX - (\text{saddrp})$	x	x	x	V	x	
	AX, sfrp	3	$AX, CY \leftarrow AX - \text{sfrp}$	x	x	x	V	x	
	saddrp, saddrp	3	$(\text{saddrp}), CY \leftarrow (\text{saddrp}) - (\text{saddrp})$	x	x	x	V	x	
CMPW	AX, #word	3	$AX - \text{word}$	x	x	x	V	x	
	saddrp, #word	4	$(\text{saddrp}) - \text{word}$	x	x	x	V	x	
	sfrp, #word	5	$\text{sfrp} - \text{word}$	x	x	x	V	x	
	rp, rp1	2	$\text{rp} - \text{rp1}$	x	x	x	V	x	
	AX, saddrp	2	$AX - (\text{saddrp})$	x	x	x	V	x	
	AX, sfrp	3	$AX - \text{sfrp}$	x	x	x	V	x	
	saddrp, saddrp	3	$(\text{saddrp}) - (\text{saddrp})$	x	x	x	V	x	

(5) Multiply/divide instructions: MULU, DIVUW, MULUW, DIVUX

Mnemonic	Operand	Byte	Operation	Flag					
				S	Z	AC	P/V	CY	
MULU	r1	2	$AX \leftarrow A \times r1$						
DIVUW	r1	2	$AX \text{ (quotient)}, r1 \text{ (remainder)} \leftarrow AX \div r1$						
MULUW	rp1	2	$AX \text{ (high-order 16 bits)}, rp1 \text{ (low-order 16 bits)} \leftarrow AX \times rp1$						
DIVUX	rp1	2	$AXDE \text{ (quotient)}, rp1 \text{ (remainder)} \leftarrow AXDE \div rp1$						

(6) Signed multiply instruction: **MULW**

Mnemonic	Operand	Byte	Operation	Flag					
				S	Z	AC	P/V	CY	
MULW	rp1	2	AX (high-order 16 bits), rp1 (low-order 16 bits) \leftarrow AX \times rp1						

(7) Sum-of-products instruction: **MACW**

Mnemonic	Operand	Byte	Operation	Flag					
				S	Z	AC	P/V	CY	
MACW	n	3	AXDE \leftarrow (B) \times (C) + AXDE B \leftarrow B + 2, C \leftarrow C + 2, n \leftarrow n - 1 End if n = 0 or P/V = 1	\times	\times	\times	V	\times	

(8) Sum-of-products instruction with saturation function: **MACSW**

Mnemonic	Operand	Byte	Operation	Flag					
				S	Z	AC	P/V	CY	
MACSW	n	3	AXDE \leftarrow (B) \times (C) + AXDE B \leftarrow B + 2, C \leftarrow C + 2, n \leftarrow n - 1 if overflow (P/V = 1) then AXDE \leftarrow 7FFFFFFFH if underflow (P/V = 1) then AXDE \leftarrow 80000000H end if n = 0 or P/V = 1	\times	\times	\times	V	\times	

(9) Relative operation instruction: **SACW**

Mnemonic	Operand	Byte	Operation	Flag					
				S	Z	AC	P/V	CY	
SACW	[DE+], [HL+]	4	AX \leftarrow AX + (DE) - (HL) DE \leftarrow DE + 2, HL \leftarrow HL + 2, C \leftarrow C - 1 end if C = 0 or CY = 1	\times	\times	\times	V	\times	

(10) Table shift instruction: **MOVTBLW**

Mnemonic	Operand	Byte	Operation	Flag					
				S	Z	AC	P/V	CY	
MOVTBLW	!addr16, n	4	(addr16 + 2) \leftarrow (addr16), n \leftarrow n - 1 addr16 \leftarrow addr16 - 2, End if n = 0						

Remark The addressing range of the table shift instruction is FE00H to FEFH.

(11) Increment/decrement instructions: INC, DEC, INCW, DECW

Mnemonic	Operand	Byte	Operation	Flag					
				S	Z	AC	P/V	CY	
INC	r1	1	$r1 \leftarrow r1 + 1$	x	x	x	V		
	saddr	2	$(saddr) \leftarrow (saddr) + 1$	x	x	x	V		
DEC	r1	1	$r1 \leftarrow r1 - 1$	x	x	x	V		
	saddr	2	$(saddr) \leftarrow (saddr) - 1$	x	x	x	V		
INCW	rp2	1	$rp2 \leftarrow rp2 + 1$						
	saddrp	3	$(saddrp) \leftarrow (saddrp) + 1$						
DECW	rp2	1	$rp2 \leftarrow rp2 - 1$						
	saddrp	3	$(saddrp) \leftarrow (saddrp) - 1$						

(12) Shift/rotate instructions: ROR, ROL, RORC, ROLC, SHR, SHL, SHRW, SHLW, ROR4, ROL4

Mnemonic	Operand	Byte	Operation	Flag					
				S	Z	AC	P/V	CY	
ROR	r1, n	2	$(CY, r1_7 \leftarrow r1_0, r1_{m-1} \leftarrow r1_m) \times n \text{ times}$ $n = 0-7$				P	x	
ROL	r1, n	2	$(CY, r1_0 \leftarrow r1_7, r1_{m+1} \leftarrow r1_m) \times n \text{ times}$ $n = 0-7$				P	x	
RORC	r1, n	2	$(CY \leftarrow r1_0, r1_7 \leftarrow CY, r1_{m-1} \leftarrow r1_m) \times n \text{ times}$ $n = 0-7$				P	x	
ROLC	r1, n	2	$(CY \leftarrow r1_7, r1_0 \leftarrow CY, r1_{m+1} \leftarrow r1_m) \times n \text{ times}$ $n = 0-7$				P	x	
SHR	r1, n	2	$(CY \leftarrow r1_0, r1_7 \leftarrow 0, r1_{m-1} \leftarrow r1_m) \times n \text{ times}$ $n = 0-7$	x	x	0	P	x	
SHL	r1, n	2	$(CY \leftarrow r1_7, r1_0 \leftarrow 0, r1_{m+1} \leftarrow r1_m) \times n \text{ times}$ $n = 0-7$	x	x	0	P	x	
SHRW	rp1, n	2	$(CY \leftarrow rp1_0, rp1_{15} \leftarrow 0, rp1_{m-1} \leftarrow rp1_m) \times n \text{ times}$ $n = 0-7$	x	x	0	P	x	
SHLW	rp1, n	2	$(CY \leftarrow rp1_{15}, rp1_0 \leftarrow 0, rp1_{m+1} \leftarrow rp1_m) \times n \text{ times}$ $n = 0-7$	x	x	0	P	x	
ROR4	[rp1]	2	$A_{3-0} \leftarrow (rp1)_{3-0}, (rp1)_{7-4} \leftarrow A_{3-0},$ $(rp1)_{3-0} \leftarrow (rp1)_{7-4}$						
ROL4	[rp1]	2	$A_{3-0} \leftarrow (rp1)_{7-4}, (rp1)_{3-0} \leftarrow A_{3-0},$ $(rp1)_{7-4} \leftarrow (rp1)_{3-0}$						

Remark n indicates the number of shifts or rotations.

(13) BCD adjustment instructions: ADJBA, ADJBS

Mnemonic	Operand	Byte	Operation	Flag					
				S	Z	AC	P/V	CY	
ADJBA		2	Decimal Adjust Accumulator	×	×	×	P	×	
ADJBS									

(14) Data conversion instruction: CVTBW

Mnemonic	Operand	Byte	Operation	Flag					
				S	Z	AC	P/V	CY	
CVTBW		1	When $A_7 = 0$, $X \leftarrow A$, $A \leftarrow 00H$ When $A_7 = 1$, $X \leftarrow A$, $A \leftarrow FFH$						

(15) Bit manipulation instructions: MOV1, AND1, OR1, XOR1, SET1, CLR1, NOT1

(1/2)

Mnemonic	Operand	Byte	Operation	Flag					
				S	Z	AC	P/V	CY	
MOV1	CY, saddr.bit	3	$CY \leftarrow (\text{saddr.bit})$						×
	CY, sfr.bit	3	$CY \leftarrow \text{sfr.bit}$						×
	CY, A.bit	2	$CY \leftarrow A.\text{bit}$						×
	CY, X.bit	2	$CY \leftarrow X.\text{bit}$						×
	CY, PSWH.bit	2	$CY \leftarrow \text{PSWH}.\text{bit}$						×
	CY, PSWL.bit	2	$CY \leftarrow \text{PSWL}.\text{bit}$						×
	saddr.bit, CY	3	$(\text{saddr.bit}) \leftarrow CY$						
	sfr.bit, CY	3	$\text{sfr.bit} \leftarrow CY$						
	A.bit, CY	2	$A.\text{bit} \leftarrow CY$						
	X.bit, CY	2	$X.\text{bit} \leftarrow CY$						
	PSWH.bit, CY	2	$\text{PSWH}.\text{bit} \leftarrow CY$						
	PSWL.bit, CY	2	$\text{PSWL}.\text{bit} \leftarrow CY$						
AND1	CY, saddr.bit	3	$CY \leftarrow CY \wedge (\text{saddr.bit})$						×
	CY, /saddr.bit	3	$CY \leftarrow CY \wedge \overline{(\text{saddr.bit})}$						×
	CY, sfr.bit	3	$CY \leftarrow CY \wedge \text{sfr.bit}$						×
	CY, /sfr.bit	3	$CY \leftarrow CY \wedge \overline{\text{sfr.bit}}$						×
	CY, A.bit	2	$CY \leftarrow CY \wedge A.\text{bit}$						×
	CY, /A.bit	2	$CY \leftarrow CY \wedge \overline{A.\text{bit}}$						×
	CY, X.bit	2	$CY \leftarrow CY \wedge X.\text{bit}$						×
	CY, /X.bit	2	$CY \leftarrow CY \wedge \overline{X.\text{bit}}$						×
	CY, PSWH.bit	2	$CY \leftarrow CY \wedge \text{PSWH}.\text{bit}$						×
	CY, /PSWH.bit	2	$CY \leftarrow CY \wedge \overline{\text{PSWH}.\text{bit}}$						×
	CY, PSWL.bit	2	$CY \leftarrow CY \wedge \text{PSWL}.\text{bit}$						×
	CY, /PSWL.bit	2	$CY \leftarrow CY \wedge \overline{\text{PSWL}.\text{bit}}$						×
OR1	CY, saddr.bit	3	$CY \leftarrow CY \vee (\text{saddr.bit})$						×
	CY, /saddr.bit	3	$CY \leftarrow CY \vee \overline{(\text{saddr.bit})}$						×
	CY, sfr.bit	3	$CY \leftarrow CY \vee \text{sfr.bit}$						×
	CY, /sfr.bit	3	$CY \leftarrow CY \vee \overline{\text{sfr.bit}}$						×
	CY, A.bit	2	$CY \leftarrow CY \vee A.\text{bit}$						×
	CY, /A.bit	2	$CY \leftarrow CY \vee \overline{A.\text{bit}}$						×
	CY, X.bit	2	$CY \leftarrow CY \vee X.\text{bit}$						×
	CY, /X.bit	2	$CY \leftarrow CY \vee \overline{X.\text{bit}}$						×
	CY, PSWH.bit	2	$CY \leftarrow CY \vee \text{PSWH}.\text{bit}$						×
	CY, /PSWH.bit	2	$CY \leftarrow CY \vee \overline{\text{PSWH}.\text{bit}}$						×
	CY, PSWL.bit	2	$CY \leftarrow CY \vee \text{PSWL}.\text{bit}$						×
	CY, /PSWL.bit	2	$CY \leftarrow CY \vee \overline{\text{PSWL}.\text{bit}}$						×

(2/2)

Mnemonic	Operand	Byte	Operation	Flag				
				S	Z	AC	P/V	CY
XOR1	CY, saddr.bit	3	$CY \leftarrow CY \vee (\text{saddr.bit})$					×
	CY, sfr.bit	3	$CY \leftarrow CY \vee \text{sfr.bit}$					×
	CY, A.bit	2	$CY \leftarrow CY \vee A.\text{bit}$					×
	CY, X.bit	2	$CY \leftarrow CY \vee X.\text{bit}$					×
	CY, PSWH.bit	2	$CY \leftarrow CY \vee \text{PSWH.bit}$					×
	CY, PSWL.bit	2	$CY \leftarrow CY \vee \text{PSWL.bit}$					×
SET1	saddr.bit	2	$(\text{saddr.bit}) \leftarrow 1$					
	sfr.bit	3	$\text{sfr.bit} \leftarrow 1$					
	A.bit	2	$A.\text{bit} \leftarrow 1$					
	X.bit	2	$X.\text{bit} \leftarrow 1$					
	PSWH.bit	2	$\text{PSWH.bit} \leftarrow 1$					
	PSWL.bit	2	$\text{PSWL.bit} \leftarrow 1$	×	×	×	×	×
	CY	1	$CY \leftarrow 1$					1
CLR1	saddr.bit	2	$(\text{saddr.bit}) \leftarrow 0$					
	sfr.bit	3	$\text{sfr.bit} \leftarrow 0$					
	A.bit	2	$A.\text{bit} \leftarrow 0$					
	X.bit	2	$X.\text{bit} \leftarrow 0$					
	PSWH.bit	2	$\text{PSWH.bit} \leftarrow 0$					
	PSWL.bit	2	$\text{PSWL.bit} \leftarrow 0$	×	×	×	×	×
	CY	1	$CY \leftarrow 0$					0
NOT1	saddr.bit	3	$(\text{saddr.bit}) \leftarrow \overline{(\text{saddr.bit})}$					
	sfr.bit	3	$\text{sfr.bit} \leftarrow \overline{\text{sfr.bit}}$					
	A.bit	2	$A.\text{bit} \leftarrow \overline{A.\text{bit}}$					
	X.bit	2	$X.\text{bit} \leftarrow \overline{X.\text{bit}}$					
	PSWH.bit	2	$\text{PSWH.bit} \leftarrow \overline{\text{PSWH.bit}}$					
	PSWL.bit	2	$\text{PSWL.bit} \leftarrow \overline{\text{PSWL.bit}}$	×	×	×	×	×
	CY	1	$CY \leftarrow \overline{CY}$					×

(16) Call/return instructions: CALL, CALLF, CALLT, BRK, RET, RETB, RETI

Mnemonic	Operand	Byte	Operation	Flag					
				S	Z	AC	P/V	CY	
CALL	!addr16	3	$(SP - 1) \leftarrow (PC + 3)_H$, $(SP - 2) \leftarrow (PC + 3)_L$, $PC \leftarrow \text{addr16}$, $SP \leftarrow SP - 2$						
	rp1	2	$(SP - 1) \leftarrow (PC + 2)_H$, $(SP - 2) \leftarrow (PC + 2)_L$, $PC_H \leftarrow rp1_H$, $PC_L \leftarrow rp1_L$, $SP \leftarrow SP - 2$						
	[rp1]	2	$(SP - 1) \leftarrow (PC + 2)_H$, $(SP - 2) \leftarrow (PC + 2)_L$, $PC_H \leftarrow (rp1 + 1)$, $PC_L \leftarrow (rp1)$, $SP \leftarrow SP - 2$						
CALLF	!addr11	2	$(SP - 1) \leftarrow (PC + 2)_H$, $(SP - 2) \leftarrow (PC + 2)_L$, $PC_{15:11} \leftarrow 00001$, $PC_{10:0} \leftarrow \text{addr11}$, $SP \leftarrow SP - 2$						
CALLT	[addr5]	1	$(SP - 1) \leftarrow (PC + 1)_H$, $(SP - 2) \leftarrow (PC + 1)_L$, $PC_H \leftarrow (TPF, 00000000, \text{addr5} + 1)$, $PC_L \leftarrow (TPF, 00000000, \text{addr5})$, $SP \leftarrow SP - 2$						
BRK		1	$(SP - 1) \leftarrow PSW_H$, $(SP - 2) \leftarrow PSW_L$, $(SP - 3) \leftarrow (PC + 1)_H$, $(SP - 4) \leftarrow (PC + 1)_L$, $PC_L \leftarrow (003EH)$, $PC_H \leftarrow (003FH)$, $SP \leftarrow SP - 4$, $IE \leftarrow 0$						
RET		1	$PC_L \leftarrow (SP)$, $PC_H \leftarrow (SP + 1)$, $SP \leftarrow SP + 2$						
RETB		1	$PC_L \leftarrow (SP)$, $PC_H \leftarrow (SP + 1)$, $PSW_L \leftarrow (SP + 2)$, $PSW_H \leftarrow (SP - 3)$, $SP \leftarrow SP + 4$	R	R	R	R	R	
RETI		1	$PC_L \leftarrow (SP)$, $PC_H \leftarrow (SP + 1)$, $PSW_L \leftarrow (SP + 2)$, $PSW_H \leftarrow (SP + 3)$, $SP \leftarrow SP + 4$, $ISPR_n \leftarrow 0$ ^{Note}	R	R	R	R	R	

Note When using the RETI instruction, reset (0) the interrupt request corresponding bit that has the highest priority among the bits (n = 0 to 3) set (1) in the ISPR register.

(17) Stack manipulation instructions: PUSH, PUSHU, POP, POPU, MOVW, INCW, DECW

Mnemonic	Operand	Byte	Operation	Flag					
				S	Z	AC	P/V	CY	
PUSH	sfrp	3	$(SP - 1) \leftarrow sfr_H, (SP - 2) \leftarrow sfr_L, SP \leftarrow SP - 2$						
	post	2	$\{(SP - 1) \leftarrow post_H, (SP - 2) \leftarrow post_L, SP \leftarrow SP - 2\} \times n \text{ times}$						
	PSW	1	$(SP - 1) \leftarrow PSW_H, (SP - 2) \leftarrow PSW_L, SP \leftarrow SP - 2$						
PUSHU	post	2	$\{(UP - 1) \leftarrow post_H, (UP - 2) \leftarrow post_L, UP \leftarrow UP - 2\} \times n \text{ times}$						
POP	sfrp	3	$sfr_L \leftarrow (SP), sfr_H \leftarrow (SP + 1), SP \leftarrow SP + 2$						
	post	2	$\{post_L \leftarrow (SP), post_H \leftarrow (SP + 1), SP \leftarrow SP + 2\} \times n \text{ times}$						
	PSW	1	$PSW_L \leftarrow (SP), PSW_H \leftarrow (SP + 1), SP \leftarrow SP + 2$	R	R	R	R	R	
POPU	post	2	$\{post_L \leftarrow (UP), post_H \leftarrow (UP + 1), UP \leftarrow UP + 2\} \times n \text{ times}$						
MOVW	SP, #word	4	$SP \leftarrow \text{word}$						
	SP, AX	2	$SP \leftarrow AX$						
	AX, SP	2	$AX \leftarrow SP$						
INCW	SP	2	$SP \leftarrow SP + 1$						
DECW	SP	2	$SP \leftarrow SP - 1$						

Remark n indicates the number of registers specified in post.

(18) Special instructions: CHKL, CHKLA

Mnemonic	Operand	Byte	Operation	Flag					
				S	Z	AC	P/V	CY	
CHKL	sfr	3	$(\text{Pin level}) \nabla (\text{Signal level before output buffer})$	×	×		P		
CHKLA	sfr	3	$A \leftarrow \{(\text{Pin level}) \nabla (\text{Signal level before output buffer})\}$	×	×		P		

(19) Unconditional branch instruction: BR

Mnemonic	Operand	Byte	Operation	Flag					
				S	Z	AC	P/V	CY	
BR	!addr16	3	$PC \leftarrow \text{addr16}$						
	rp1	2	$PC_H \leftarrow rp1_H, PC_L \leftarrow rp1_L$						
	[rp1]	2	$PC_H \leftarrow (rp1 + 1), PC_L \leftarrow (rp1)$						
	\$addr16	2	$PC \leftarrow PC + 2 + jdisp8$						

(20) Conditional branch instructions: BC, BL, BNC, BNL, BZ, BE, BNZ, BNE, BV, BPE, BN, BP, BGT, BGE, BLT, BLE, BH, BNH, BT, BF, BTCLR, BFSET, DBNZ (1/2)

Mnemonic	Operand	Byte	Operation	Flag					
				S	Z	AC	P/V	CY	
BC	\$addr16	2	$PC \leftarrow PC + 2 + \text{jdisp8}$ if $CY = 1$						
BL									
BNC	\$addr16	2	$PC \leftarrow PC + 2 + \text{jdisp8}$ if $CY = 0$						
BNL									
BZ	\$addr16	2	$PC \leftarrow PC + 2 + \text{jdisp8}$ if $Z = 1$						
BE									
BNZ	\$addr16	2	$PC \leftarrow PC + 2 + \text{jdisp8}$ if $Z = 0$						
BNE									
BV	\$addr16	2	$PC \leftarrow PC + 2 + \text{jdisp8}$ if $P/V = 1$						
BPE									
BNV	\$addr16	2	$PC \leftarrow PC + 2 + \text{jdisp8}$ if $P/V = 0$						
BPO									
BN	\$addr16	2	$PC \leftarrow PC + 2 + \text{jdisp8}$ if $S = 1$						
BP	\$addr16	2	$PC \leftarrow PC + 2 + \text{jdisp8}$ if $S = 0$						
BGT	\$addr16	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if $(P/V \nrightarrow S) \vee Z = 0$						
BGE	\$addr16	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if $P/V \nrightarrow S = 0$						
BLT	\$addr16	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if $P/V \nrightarrow S = 1$						
BLE	\$addr16	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if $(P/V \nrightarrow S) \vee Z = 1$						
BH	\$addr16	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if $Z \vee CY = 0$						
BNH	\$addr16	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if $Z \vee CY = 1$						
BT	saddr.bit, \$addr16	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if (saddr.bit) = 1						
	sfr.bit, \$addr16	4	$PC \leftarrow PC + 4 + \text{jdisp8}$ if sfr.bit = 1						
	A.bit, \$addr16	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if A.bit = 1						
	X.bit, \$addr16	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if X.bit = 1						
	PSWH.bit, \$addr16	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if PSWH.bit = 1						
	PSWL.bit, \$addr16	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if PSLW.bit = 1						
BF	saddr.bit, \$addr16	4	$PC \leftarrow PC + 4 + \text{jdisp8}$ if (saddr.bit) = 0						
	sfr.bit, \$addr16	4	$PC \leftarrow PC + 4 + \text{jdisp8}$ if sfr.bit = 0						
	A.bit, \$addr16	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if A.bit = 0						
	X.bit, \$addr16	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if X.bit = 0						
	PSWH.bit, \$addr16	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if PSWH.bit = 0						
	PSWL.bit, \$addr16	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if PSLW.bit = 0						

(2/2)

Mnemonic	Operand	Byte	Operation	Flag				
				S	Z	AC	P/V	CY
BTCLR	saddr.bit, \$addr16	4	$PC \leftarrow PC + 4 + \text{jdisp8}$ if (saddr.bit) = 1 then reset (saddr.bit)					
	sfr.bit, \$addr16	4	$PC \leftarrow PC + 4 + \text{jdisp8}$ if sfr.bit = 1 then reset sfr.bit					
	A.bit, \$addr16	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if A.bit = 1 then reset A.bit					
	X.bit, \$addr16	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if X.bit = 1 then reset X.bit					
	PSWH.bit, \$addr16	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if PSWH.bit = 1 then reset PSWH.bit					
	PSWL.bit, \$addr16	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if PSWL.bit = 1 then reset PSWL.bit	x	x	x	x	x
BFSET	saddr.bit, \$addr16	4	$PC \leftarrow PC + 4 + \text{jdisp8}$ if (saddr.bit) = 0 then set (saddr.bit)					
	sfr.bit, \$addr16	4	$PC \leftarrow PC + 4 + \text{jdisp8}$ if sfr.bit = 0 then set sfr.bit					
	A.bit, \$addr16	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if A.bit = 0 then set A.bit					
	X.bit, \$addr16	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if X.bit = 0 then set X.bit					
	PSWH.bit, \$addr16	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if PSWH.bit = 0 then set PSWH.bit					
	PSWL.bit, \$addr16	3	$PC \leftarrow PC + 3 + \text{jdisp8}$ if PSWL.bit = 0 then set PSWL.bit	x	x	x	x	x
DBNZ	r2, \$addr16	2	$r2 \leftarrow r2 - 1$, then $PC \leftarrow PC + 2 + \text{jdisp8}$ if $r2 \neq 0$					
	saddr, \$addr16	3	(saddr) \leftarrow (saddr) - 1 then $PC \leftarrow PC + 3 + \text{jdisp8}$ if (saddr) $\neq 0$					

(21) Context switching instructions: BRKCS, RETCS, RETCSB

Mnemonic	Operand	Byte	Operation	Flag					
				S	Z	AC	P/V	CY	
BRKCS	R _{Bn}	2	$RBS2 - 0 \leftarrow n$, $PC_H \leftrightarrow R5$, $PC_L \leftrightarrow R4$, $R7 \leftarrow PSW_H$, $R6 \leftarrow PSW_L$, $RSS \leftarrow 0$, $IE \leftarrow 0$						
RETCS	!addr16	3	$PC_H \leftarrow R5$, $PC_L \leftarrow R4$, $R5 \leftarrow addr16_H$, $R4 \leftarrow addr16_L$, $PSW_H \leftarrow R7$, $PSW_L \leftarrow R6$	R	R	R	R	R	
RETCSB	!addr16	4	$PC_H \leftarrow R5$, $PC_L \leftarrow R4$, $R5 \leftarrow addr16_H$, $R4 \leftarrow addr16_L$, $PSW_H \leftarrow R7$, $PSW_L \leftarrow R6$	R	R	R	R	R	

(22) String instructions: MOV_M, MOV_{BK}, XCH_M, XCH_{BK}, CMP_{ME}, CMP_{BKE}, CMP_{MNE}, CMP_{BKNE}, CMP_{MC}, CMP_{BKC}, CMP_{MNC}, CMP_{BKNC}

Mnemonic	Operand	Byte	Operation	Flag					
				S	Z	AC	P/V	CY	
MOV_M	[DE+], A	2	$(DE+) \leftarrow A$, $C \leftarrow C - 1$, End if $C = 0$						
	[DE-], A	2	$(DE-) \leftarrow A$, $C \leftarrow C - 1$, End if $C = 0$						
MOV_{BK}	[DE+], [HL+]	2	$(DE+) \leftarrow (HL+)$, $C \leftarrow C - 1$, End if $C = 0$						
	[DE-], [HL-]	2	$(DE-) \leftarrow (HL-)$, $C \leftarrow C - 1$, End if $C = 0$						
XCH_M	[DE+], A	2	$(DE+) \leftrightarrow A$, $C \leftarrow C - 1$, End if $C = 0$						
	[DE-], A	2	$(DE-) \leftrightarrow A$, $C \leftarrow C - 1$, End if $C = 0$						
XCH_{BK}	[DE+], [HL+]	2	$(DE+) \leftrightarrow (HL+)$, $C \leftarrow C - 1$, End if $C = 0$						
	[DE-], [HL-]	2	$(DE-) \leftrightarrow (HL-)$, $C \leftarrow C - 1$, End if $C = 0$						
CMP_{ME}	[DE+], A	2	$(DE+) - A$, $C \leftarrow C - 1$, End if $C = 0$ or $Z = 0$	x	x	x	V	x	
	[DE-], A	2	$(DE-) - A$, $C \leftarrow C - 1$, End if $C = 0$ or $Z = 0$	x	x	x	V	x	
CMP_{BKE}	[DE+], [HL+]	2	$(DE+) - (HL+)$, $C \leftarrow C - 1$, End if $C = 0$ or $Z = 0$	x	x	x	V	x	
	[DE-], [HL-]	2	$(DE-) - (HL-)$, $C \leftarrow C - 1$, End if $C = 0$ or $Z = 0$	x	x	x	V	x	
CMP_{MNE}	[DE+], A	2	$(DE+) - A$, $C \leftarrow C - 1$, End if $C = 0$ or $Z = 1$	x	x	x	V	x	
	[DE-], A	2	$(DE-) - A$, $C \leftarrow C - 1$, End if $C = 0$ or $Z = 1$	x	x	x	V	x	
CMP_{BKNE}	[DE+], [HL+]	2	$(DE+) - (HL+)$, $C \leftarrow C - 1$, End if $C = 0$ or $Z = 1$	x	x	x	V	x	
	[DE-], [HL-]	2	$(DE-) - (HL-)$, $C \leftarrow C - 1$, End if $C = 0$ or $Z = 1$	x	x	x	V	x	
CMP_{MC}	[DE+], A	2	$(DE+) - A$, $C \leftarrow C - 1$, End if $C = 0$ or $CY = 0$	x	x	x	V	x	
	[DE-], A	2	$(DE-) - A$, $C \leftarrow C - 1$, End if $C = 0$ or $CY = 0$	x	x	x	V	x	
CMP_{BKC}	[DE+], [HL+]	2	$(DE+) - (HL+)$, $C \leftarrow C - 1$, End if $C = 0$ or $CY = 0$	x	x	x	V	x	
	[DE-], [HL-]	2	$(DE-) - (HL-)$, $C \leftarrow C - 1$, End if $C = 0$ or $CY = 0$	x	x	x	V	x	
CMP_{MNC}	[DE+], A	2	$(DE+) - A$, $C \leftarrow C - 1$, End if $C = 0$ or $CY = 1$	x	x	x	V	x	
	[DE-], A	2	$(DE-) - A$, $C \leftarrow C - 1$, End if $C = 0$ or $CY = 1$	x	x	x	V	x	
CMP_{BKNC}	[DE+], [HL+]	2	$(DE+) - (HL+)$, $C \leftarrow C - 1$, End if $C = 0$ or $CY = 1$	x	x	x	V	x	
	[DE-], [HL-]	2	$(DE-) - (HL-)$, $C \leftarrow C - 1$, End if $C = 0$ or $CY = 1$	x	x	x	V	x	

(23) CPU control instructions: MOV, SWRS, SEL, NOP, EI, DI

Mnemonic	Operand	Byte	Operation	Flag					
				S	Z	AC	P/V	CY	
MOV	STBC, #byte	4	$STBC \leftarrow \text{byte}$ ^{Note}						
	WDM, #byte	4	$WDM \leftarrow \text{byte}$ ^{Note}						
SWRS		1	$RSS \leftarrow \overline{RSS}$						
SEL	RBn	2	$RBS2 - 0 \leftarrow n, RSS \leftarrow 0$						
	RBn, ALT	2	$RBS2 - 0 \leftarrow n, RSS \leftarrow 1$						
NOP		1	No operation						
EI		1	$IE \leftarrow 1$ (Enable interrupt)						
DI		1	$IE \leftarrow 0$ (Disable interrupt)						

Note An op-code trap interrupt occurs if an invalid op-code is specified in an STBC or WDM register manipulation instruction.

Trap operation: $(SP - 1) \leftarrow PSW_H, (SP - 2) \leftarrow PSW_L,$
 $(SP - 3) \leftarrow (PC - 4)_H, (SP - 4) \leftarrow (PC - 4)_L,$
 $PC_L \leftarrow (003CH), PC_H \leftarrow (003DH),$
 $SP \leftarrow SP - 4, IE \leftarrow 0$

[MEMO]

CHAPTER 19 INSTRUCTION EXECUTION RATE

19.1 Memory Space and Access Speed

19.1.1 Main RAM and peripheral RAM

The μ PD78366A has 2048-byte internal RAM^{Note 1} in the area between F700H and FEFH. The internal RAM is divided into two large groups, main RAM and peripheral RAM, according to the access speed.

Main RAM is in the execution unit (EXU) and allows access at the highest speed.

- Main RAM (FE00H to FEFH) : 1 clock per word
- Peripheral RAM (F700H to FDFFH)^{Note 2}: 3 + n clocks per word (When a word access to even addresses is executed.)

Notes 1. FC00H to FEFH (768 bytes) in case of μ PD78363A

2. FC00H to FDFFH in case of μ PD78363A

Caution When a word access to the main RAM area (FE00H to FEFH) (containing stack handling) is executed, addresses specified in operands are limited to even addresses.

Remark n is the number of wait cycles set by the programmable wait control register (PWC).

19.1.2 Memory access

(1) Op-code fetch

(a) Access range

The μ PD78366A allows op-code fetch in the area between 0000H and FDFFH (refer to **Figure 19-1**). Op-code fetch is not allowed in the area between FE00H and FFFFH.

(b) Number of clocks required for access

The number of clocks required for op-code fetch can be specified by the PWC register in units of 16K bytes. The number of clocks depends on the area to be accessed.

Only an area in internal memory can be specified in the high-speed fetch cycle mode.

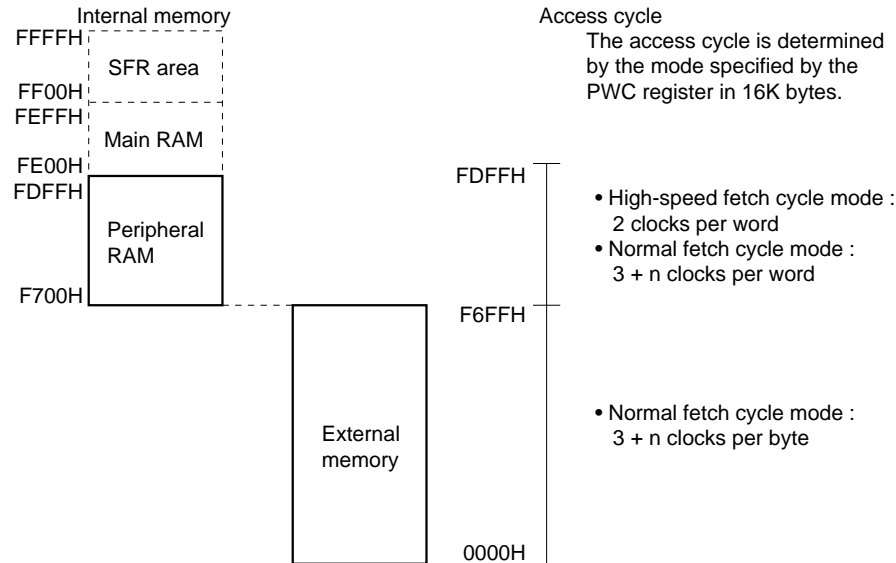
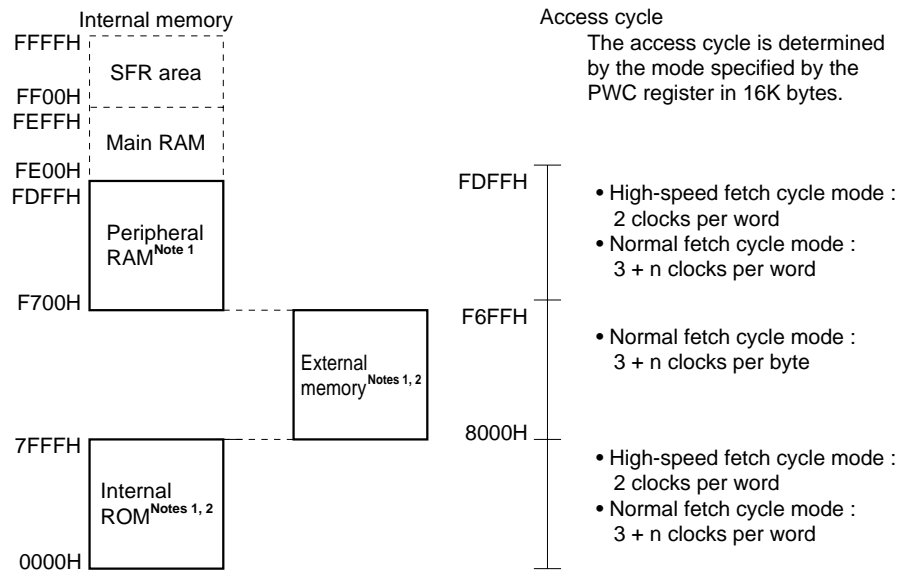
Table 19-1. Number of Clocks Required for Op-Code Fetch

Area to Be Accessed		Access Cycle
Peripheral RAM	Normal fetch	3 + n clocks per word
	High-speed fetch	2 clocks per word
External memory		3 + n clocks per byte
Internal ROM ^{Note}	Normal fetch	3 + n clocks per word
	High-speed fetch	2 clocks per word

Note μ PD78363A, 78366A and 78P368A only

Remark n is the wait count specified by the PWC register.

Caution The PWC register specifies the mode of an area, irrespective of whether the area is in internal memory or external memory. After reset, the whole space is set in the normal fetch cycle mode.

Figure 19-1. Concept of Memory Access in Op-Code Fetch**(1) μ PD78365A****(2) μ PD78363A, 78366A, 78368A, 78P368A**

Notes 1. The following addresses are used in the case of the μ PD78363A.

Peripheral RAM (FC00H to FDFFH), external memory (6000H to FBFFH), internal ROM (0000H to 5FFFH)

2. The following addresses are used in the case of the μ PD78368 and 78P368A.

External memory (C000H to F6FFH), internal ROM (0000H to BFFFH)

Remark n is the wait count specified by the PWC register.

(c) Bus control signals (ASTB, \overline{RD} , and \overline{WR}) to be output during memory access

The μ PD78366A outputs bus control signals ASTB, \overline{RD} , and \overline{WR} during op-code fetch. Bus control signals to be output depend on areas to be accessed (refer to **Table 19-2**).

Table 19-2. Bus Control Signals to Be Output during Op-Code Fetch**(1) μ PD78365A**

Area to Be Accessed		Address	ASTB	\overline{RD}	\overline{WR}
Peripheral RAM	Normal fetch	F700H-FDFFH	○	—	—
	High-speed fetch		_Note	—	—
External memory		0000H-F6FFH	○	○	—

Note The μ PD78365A outputs the ASTB signal only when the BR instruction is fetched.

Remark ○: indicates that the signal is output.

—: indicates that the signal is not output (inactive-level signal output).

(2) μ PD78363A, 78366A, 78368A, 78P368A

Area to Be Accessed		Address	ASTB	\overline{RD}	\overline{WR}
Peripheral RAM	Normal fetch	F700H-FDFFH ^{Note 1}	○	—	—
	High-speed fetch		_Note 3	—	—
External memory		8000H-F6FFH ^{Notes 1, 2}	○	○	—
Internal ROM	Normal fetch	0000H-7FFFH ^{Notes 1, 2}	○	—	—
	High-speed fetch		_Note 3	—	—

Notes 1. The following addresses are used in the case of the μ PD78363A.

Peripheral RAM (FC00H to FDFFH), external memory (6000H to FBFFH), internal ROM (0000H to 5FFFH)

2. The following addresses are used in the case of the μ PD78368A and 78P368A.

External memory (C000H to F6FFH), internal ROM (0000H to BFFFH)

3. The ASTB signal is output only in the case of the BR instruction.

Remark ○: indicates that the signal is output.

—: indicates that the signal is not output (inactive-level signal output).

(2) Data access

While instructions such as MOV A,[HL] and SUB [DE+],A are being executed, data is read from or written into memory.

(a) Access range

The μ PD78366A allows data access to the whole 64K-byte range. In the address range between FE00H and FFFFH, access to internal memory has precedence. In the 16-byte external SFR area between FFD0H and FFDFH, however, external memory is accessed (refer to **Figure 19-2**).

(b) Number of clocks required for access

The number of clocks required for data access depends on the area to be accessed.

Table 19-3. Number of Clocks Required for Data Access

Area to Be Accessed	Address	Read Access	Write Access
Main RAM	FE00H-FEFFFH	1 clock per word	1 clock per word
SFR	FF00H-FFFFH	4 clocks per word	4 clocks per word
Internal ROM ^{Note 1}	0000H-7FFFFH ^{Notes 2, 3}	3 + n clocks per word	—
Peripheral RAM	F700H-FDFFFH ^{Note 2}		1 clock per word ^{Note 4}
External SFR	FFD0H-FFDFH	3 + n clocks per byte	1 clock per byte ^{Note 4}
External memory	—		

Remark n is the wait count specified by the PWC register.

Notes 1. μ PD78363A, 78366A, 78368A and 78P368A only

2. In case of the μ PD78363A, internal ROM (0000H to 5FFFFH), peripheral RAM (FC00H to FDFFFH).

3. In case of the μ PD78368A and 78P368A, internal ROM (0000H to BFFFFH).

4. One clock is required to execute data write access to peripheral RAM, external SFR or external memory, in which only the address and data are passed to the bus control unit (BCU). However, the bus is occupied for the same time period as in data read access (3 + n clocks). When data access to this area is executed continuously after data write access, more clocks than indicated in the table may be required.

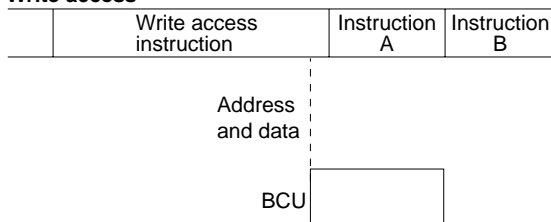
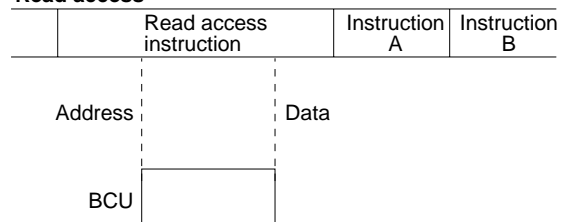
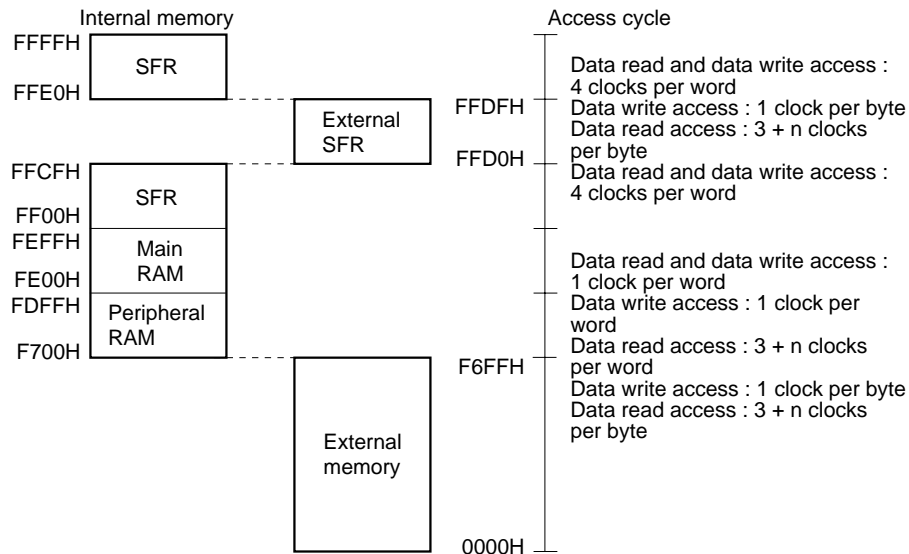
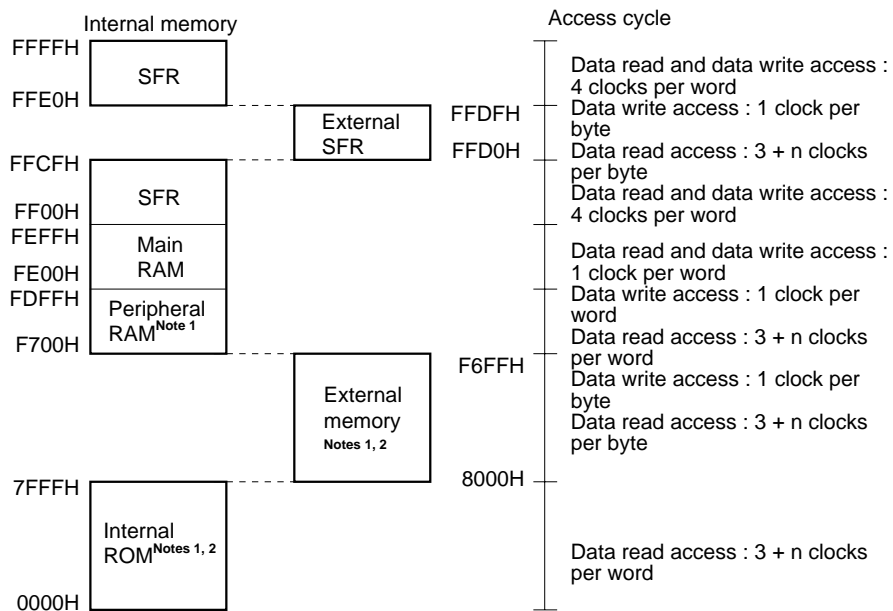
Write access**Read access**

Figure 19-2. Concept of Memory Access in Data Access**(1) μ PD78365A****(2) μ PD78363A, 78366A, 78368A, 78P368A**

Notes 1. The following addresses are used in the case of the μ PD78363A.

Peripheral RAM (FC00H to FDFFH), external memory (6000H to FBFFH), internal ROM (0000H to 5FFFH)

2. The following addresses are used in the case of the μ PD78368A and 78P368A.

External memory (C000H to F6FFH), internal ROM (0000H to BFFFH)

Remark n is the wait count specified by the PWC register.

(c) Bus control signals (ASTB, \overline{RD} , and \overline{WR}) to be output during memory access

The μ PD78366A outputs bus control signals ASTB, \overline{RD} , and \overline{WR} during data access. Bus control signals to be output depend on areas to be accessed (refer to **Table 19-4**).

Table 19-4. Bus Control Signals to Be Output during Data Access**(1) μ PD78365A**

Area to Be Accessed	Address	ASTB	\overline{RD}	\overline{WR}
SFR	FFE0H-FFFFH FF00H-FFCFH	—	—	—
External SFR	FFD0H-FFDFH	○	○	○
Main RAM	FE00H-FEFFH	—	—	—
Peripheral RAM	F700H-FDFFH	○	—	—
External memory	0000H-F6FFH	○	○	○

Remark ○: indicates that the signal is output.

—: indicates that the signal is not output (inactive-level signal output).

(2) μ PD78363A, 78366A, 78368A, 78P368A

Area to Be Accessed	Address	ASTB	\overline{RD}	\overline{WR}
SFR	FFE0H-FFFFH FF00H-FFCFH	—	—	—
External SFR	FFD0H-FFDFH	○	○	○
Main RAM	FE00H-FEFFH	—	—	—
Peripheral RAM	F700H-FDFFH ^{Note 1}	○	—	—
External memory	8000H-F6FFH ^{Notes 1, 2}	○	○	○
Internal ROM	0000H-7FFFH ^{Notes 1, 2}	○	—	—

Notes 1. The following addresses are used in the case of the μ PD78363A.

Peripheral RAM (FC00H to FDFFH), external memory (6000H to FBFFH), internal ROM (0000H to 5FFFH)

2. The following addresses are used in the case of the μ PD78368A and 78P368A.

External memory (C000H to F6FFH), internal ROM (0000H to BFFFH)

Remark ○: indicates that the signal is output.

—: indicates that the signal is not output (inactive-level signal output).

19.2 Interrupt Execution Rate

The following tables list interrupt execution rates. The time required to determine the priority is ignored in the tables. The priority is determined every two clocks. The time required to determine the priority ranges from zero to two clocks, depending on when the interrupt occurs.

n is the wait count specified by the PWC register.

(1) Vectored interrupt service

Stack	Number of Clocks
Main RAM (FE00H-FEFFFH)	$21 + 2n$
Peripheral RAM (F700H-FDFFFH) ^{Note 1}	$25 + 2n/33 + 2n$ ^{Note 2}
External memory	$33 + 6n$

Notes 1. In case of μ PD78363A, FC00H to FDFFFH

2. Even address/odd address

(2) Context switching processing

Number of clocks: $17 + 2n$ clocks

(3) Macro service processing

Macro Service	Number of Clocks	
	Byte Operation	Word Operation
EVTCNT	12	
BLKTRS mem \rightarrow SFR	18	19
BLKTRS SFR \rightarrow mem	17	18
BLKTRS-P mem \rightarrow SFR	(IRAM)	21
	(PRAM)	23/27 ^{Note}
	(EMEM)	$27 + 2n$
BLKTRS-P SFR \rightarrow mem	(IRAM)	21
	(PRAM)	23/27 ^{Note}
	(EMEM)	$27 + 2n$
DTADIF	–	22
DTADIF-P	(IRAM)	26
	(PRAM)	28/32 ^{Note}
	(EMEM)	$32 + 2n$

Note Even address/odd address

19.3 Calculating Number of Execution Clocks

This section describes the procedure for calculating the number of instruction execution clocks.

(1) Calculating number of basic clocks

- (a) When a program is held in internal ROM or peripheral RAM (F700H to FDFFH) in the high-speed fetch mode
 → The total number of clocks is the number of basic clocks.
- (b) When a program is usually held in external memory in the normal fetch mode
 → The larger value of the following two is taken as the number of basic clocks:
- Total number of clocks
 - Total number of bytes of executed instruction $\times (3 + n)$
 n is the wait count specified by the PWC register.

When the high-speed fetch mode is not specified, the number of basic clocks is calculated according to procedure (b) above even if the program is in internal ROM or peripheral RAM.

Example of calculation μ PD78366A (Internal clock of 16 MHz, wait 0, main RAM specified by [HL])

		Number of bytes	Number of clocks
MOV	A, [HL]	1	6
ADD	A, B	2	3
MOV	[HL], A	1	5
		4	14

The time to execute this program is calculated as follows:

(a) When the program is held in internal ROM

Number of execution clocks: 14

$$14 \times 0.0625 \mu\text{s} = 0.875 \mu\text{s}$$

(b) When the program is held in external memory

Number of execution clocks: $4 \times 3 = 12 < 14$

Taking the number of execution clocks as 14, the calculation shown in (a) above is performed. The calculated execution time is $0.875 \mu\text{s}$.

(2) Adding a correction factor to the number of basic clocks**(a) Accessing the SFR area (FF00H to FF1FH) by an instruction having saddr or saddrp as an operand**

→ Each time an access is made four clocks are added.

Table 19-5. Number of saddr Accesses by Instruction (1/2)

Instruction		Number of Accesses
MOV	saddr, #byte	1
	A, saddr	
	saddr, A	
	saddr, saddr	1/2 Note 2
	A, [saddrp]	1
	[saddrp], A	
XCH	A, saddr	2
	A, [saddrp]	1
	saddr, saddr	2/4 Note 2
MOVW	saddrp, #word	1
	AX, saddrp	
	saddrp, AX	
	saddrp, saddrp	1/2 Note 2
XCHW	AX, saddrp	2
	saddrp, saddrp	2/4 Note 2
ALU Note 1	saddr, #byte	2
	A, saddr	1
	saddr-D, saddr-S	1/2/3 Note 3
CMP	saddr, #byte	1
	A, saddr	
	saddr, saddr	1/2 Note 2

Notes 1. ALU: ADD, ADDC, SUB, SUBC, AND, OR, XOR**2.** When either or both are in the SFR area**3.** When only the source (saddr-S), only the destination (saddr-D), or both are in the SFR area

Table 19-5. Number of saddr Accesses by Instruction (2/2)

Instruction		Number of Accesses
ADDW, SUBW	saddrp, #word	2
	AX, saddrp	1
	saddrp-D, saddrp-S	1/2/3 ^{Note 1}
CMPW	saddrp, #word	1
	AX, saddrp	
	saddrp-D, saddrp-S	1/2 ^{Note 2}
INC, DEC	saddr	2
INCW, DECW	saddrp	2
MOV1	CY, saddr.bit	1
	saddr.bit, CY	2
AND1, OR1	CY, saddr.bit	1
	CY, /saddr.bit	
XOR1	CY, saddr.bit	1
SET1, CLR1, NOT1	saddr.bit	2
BT, BF	saddr.bit, \$saddr16	1
BTCLR, BFSET	saddr.bit, \$saddr16	2
DBNZ	saddr.bit, \$saddr16	2

Notes 1. When only the source (saddr-S), only the destination (saddr-D), or both are in the SFR area

2. When either or both are in the SFR area

(b) To access the following SFR area by an instruction having operand sfr or sfrp

Correction value = number of clocks increased per access × access count

Accessed SFR	No. of Clocks Increased per Access	
	At Read	At Write
MM	n/0 ^{Note}	
PWC	0/1 ^{Note}	

Note At normal fetch/high-speed fetch

Remark n is the wait count specified by the PWC register.

(c) Branch instruction and CALL instruction

A branch or CALL instruction clears the instruction queue. Before the instruction following the branch instruction is executed, the additional clocks shown below are required:

In the high-speed fetch mode: 7 to 8 clocks

In the normal fetch mode : 7 to $10 + n$ clocks

The number of clocks to be added depends on when op-code fetch is executed and varies in the range shown above.

CHAPTER 20 CAUTIONS

This chapter collects the cautions described in each chapter. Read this chapter before developing application products. The number enclosed in parentheses represents the page on which the caution is initially described.

20.1 Cautions for CHAPTER 2 PIN FUNCTIONS

- (1) When the $\overline{\text{RESET}}$ signal is input, pins P00-P07 (port 0), P10-P17 (port 1), P30-P36 (port 3), and P80-P85 (port 8) are set to the input mode (output high impedance). The contents of the output latch becomes indefinite. (p.17-19, 22)
- (2) When the $\overline{\text{RESET}}$ signal is input, pins P40-P47 (port 4), P50-P57 (port 5), and P90-P93 (port 9) are set to the input mode (output high impedance) in either of port mode or external memory expansion mode. The contents of the output latch becomes indefinite. (p.20, 21, 23)
- (3) Connect the MODE0 and MODE1 pins directly to V_{DD} or V_{SS} . (p.23)
- (4) The $\mu\text{PD78P368A}$ cannot be set in the ROM-less mode. (p.23)

20.2 Cautions for CHAPTER 3 CPU ARCHITECTURE

- (1) The $\mu\text{PD78P368A}$ cannot be set in the ROM-less mode. (p.33)
- (2) When a word access to the main RAM area (FE00H to FEFFH) (containing stack handling) is executed, addresses specified in operands are limited to even addresses. (p.30-33, 46, 56-59)
- (3) When a word access to the main RAM area (FE00H-FEFFFH) (containing stack handling) is executed, the access operation varies, depending on whether the reference address is even or odd. Therefore, if an access to an even address and an access to an odd address are mixed, an error is caused. Specify only even reference addresses. To execute a 16-bit data transfer instruction, specify even addresses in operands. If odd addresses are specified, an error occurs in the assembler package (RAM78K3). (p.35)
- (4) Do not make a word access across the peripheral RAM area and main RAM area. (p.35)
- (5) Unmapped addresses of the special function register cannot be accessed (except the external access area). (p.39)
- (6) The vector tables for the $\overline{\text{RESET}}$ input, BRK instruction, and op-code trap interrupt are fixed to 0000H, 003EH, and 003CH, respectively. They are not affected by the TPF. (p.46)
- (7) Write 0 or 1 into any bit of the special function register (SFR) correctly whenever it is predetermined to be so. (p.50)
- (8) Do not write data into the register which is only used for data reading. Writing data into such registers may result in an error. (p.50)

- (9) The SFR area addresses (FF00H to FFFFH) to which a special function register is not assigned cannot be accessed (except the external access area). Accessing these addresses may result in an error. (p.50)
- (10) When the read data is used as byte data, take care of the undefined bit before use. (p.50)
- (11) The timer out register (TOUT), serial transmission shift register (TXS) are write-only registers. Do not read from them. (p.50)
- (12) Bits 0, 1 and 4 of the serial bus interface control register (SBIC) are write-only bits. If these bits are read, the value read is 0. (p.50)

20.3 Cautions for CHAPTER 5 PORT FUNCTIONS

- (1) In the μ PD78365A and in the ROM-less mode of μ PD78363A, 78366A, and 78368A, ports 4, 5, and 9 (lower 2 bits) do not function as ports. (p.65)
- (2) The pins functioning as input pins in the control mode may operate erroneously if the corresponding bits of the port mode register are rewritten during the operation. Therefore, write into the port mode register when the system is initialized, etc. Do not dynamically rewrite the bits during operation. (p.69)
- (3) Bits 7 to 1 of the port read control register (PRDC) are fixed to "0" by hardware. Even if "1" is written to them, they remain "0". (p.72)
- (4) If the PRDC register is in the pin access mode (PRDC0 = 1), no bit manipulation instruction for a port operates normally. After a port check is completed, be sure to reset the mode to the normal mode (PRDC0 = 0). (p.73)
- (5) If an interrupt occurs when the PRDC register is in the pin access mode, a bit manipulation instruction may be executed in the same mode. This will cause an error. Before starting a check, be sure to set the DI state. In addition, do not use macro services that manipulate ports. (p.73)
- (6) Non-maskable interrupts are unavoidable if the PRDC register is in the pin access mode. So, the following provisions should be made in the program as required: (p.73)
 - The non-maskable interrupt routine is to perform no port manipulation.
 - The level of PRDC.0 is to be saved at the start of the nonmaskable interrupt routine, then is restored when control is returned.
- (7) The pin access mode is a function to access the pin status of an output port. If a pin set in the input port mode (PMXn = 1) is read in the pin access mode (PRDC0 = 1), "0" is read regardless of the input level. (p.73)
- (8) Use the CHKL and CHKLA instructions only when the PRDC0 bit of the PRDC register is set to 0 (normal mode). (p.74)
- (9) In the case of those input/output port pins that are set to the input port mode, the results of the CHKL or CHKLA instructions always match no matter whether they are set in the port mode or control mode. In the case of the dedicated input port, because it is not provided with output latch, the input pin level is read when the CHKL or CHKLA instruction is executed. Thus, the CHKL or CHKLA instruction is actually invalid for the dedicated input port and these instructions should not be used. (p.74)

- (10) If the CHKL or CHKLA instruction is executed with port 4 set to the input port mode or extended mode, a mismatch may be generated (a mismatch is generated if the pin level changes in execution of the CHKL or CHKLA instruction). Therefore, while port 4 is set to the input port mode or extended mode, do not execute these instructions. (p.74)
- (11) Set control output pins to the input mode before executing the CHKL or CHKLA instruction to check the output level of the pin of a port where control and port output pins are used together. (The output level of a control pin cannot be checked with the CHKL or CHKLA instruction because the output level varies asynchronously.) (p.74)
- (12) Bits 3, 4 and 7 of the memory expansion mode register (MM) are fixed to "0" by hardware. Even if "1" is written to them, they remain "0". (p.85)
- (13) Invalid combinations are specified as "Setting prohibited" in format of MM register. Never write these combinations. (p.85)
- (14) The P20/NMI pin does not contain a pull-up resistor on hardware. Therefore, even if PUO2 is set to 1, no internal pull-up resistor is contained in the P20/NMI pin. (p.86)
- (15) To emulate the μ PD78366A with the IE-78350-R, the internal pull-up resistors of ports 1, 4, 5, and 9 are invalid even if the PUO1, PUO4, PUO5, and PUO9 bits of the pull-up resistor option register (PUOL, PUOH) registers are set to "1". To use the pull-up resistor, set the corresponding bit to "1" to share the software between the IE-78350-R and μ PD78366A, and connect an external pull-up resistor. (p.87)
- (16) Bits 7 and 6 of the PUOL register and bits 7 to 2 of the PUOH register are fixed to "0" by hardware. Even if "1" is written to them, they remain "0". (p.87)

20.4 Cautions for CHAPTER 6 CLOCK GENERATOR

- (1) When using an external clock, do not set the STOP mode. (p.89)
- (2) When using a system clock oscillation circuit, wire as follows to prevent adverse influence from wiring capacitance: (p.90)
 - Keep the wiring length as short as possible.
 - Do not cross the wiring with any other signal lines. Do not route the wiring in the vicinity of line through which a high alternating current flows.
 - Always keep the ground potential for the capacitor in the oscillation circuit at the same potential as Vss. Do not ground the capacitor to a ground pattern through which a high current flows.
 - Do not extract any signal from the oscillation circuit.
- (3) Make sure that a load such as wiring capacitance is not connected to the X2 pin when inputting an external clock. (p.90)

20.5 Cautions for CHAPTER 7 REAL-TIME PULSE UNIT

- (1) Data of the timer out register (TOUT) is output to port 8 if port 8 is in control mode (TO00 to TO05 outputs) and TM0 is in the general-purpose interval timer mode or PWM mode 1. Otherwise, the TOUT data is not output to port 8 and it is therefore impossible to read the TOUT contents. (p.101)
- (2) Specification of the ALVTO bit of timer unit mode register 0 (TUM0) is only valid for pins set to the control mode (TO00 to TO05) by the port 8 mode control register (PMC8), and when TM0 is in PWM mode 0 (symmetrical triangular wave, asymmetrical triangular wave, toothed wave). (p.104)
- (3) Specification of the ICME bit of the TUM0 register is only valid when TM0 is in the general-purpose interval timer mode (TMOD02 = 0). (p.104)
- (4) It is prohibited to change bits RMOD, TMOD02 to TMOD00 and ALVTO of the TUM0 register during the operation of TM0 (CE0 = 1) and DTM0 to DTM2 (CED = 1). (p.104)
- (5) After the valid edge of NMI has been generated, the valid edge generation signal is retained for the duration of about 20 system clocks. If the TODIS0 bit is changed from "0" to "1" during this period, the TO00 through TO05 pins are set to OFF by the NMI that has been already generated.
It is recommended that the TODIS0 bit be manipulated in the NMI routine except when the system is initialized. (p.105)
- (6) Bit 6 of the timer control register 0 (TMC0) is fixed to "0" by hardware. Even if "1" is written, it remains "0". (p.107)
- (7) It is prohibited to change bits B3TR, PRM02 to PRM00 of the TMC0 register during the operation of TM0 (CE0 = 1). (p.107)
- (8) If CE0 is set to 0 (TM0 stop), the \bar{U}/D flag is cleared to 0 (p.107).
- (9) The CED bit of the timer control register 1 (TMC1) controls the three dead time timers (DTM0-DTM2). (p.108)
- (10) Bits 6 to 4 and 2 of the TMC1 register are fixed to "0" by hardware. Even if "1" is written, they remain "0". (p.108, 142)
- (11) It is prohibited to change the CED bit of the TMC1 register during the operation of TM0 (CE0 = 1). (p.108)
- (12) Bit 1 of the INTM0 register is fixed to "0" by hardware. Even if "1" is written, it remains "0". (p.109 160)
- (13) Setting of CM03 = 0000H is prohibited. (p.111, 113, 120, 129, 137)
- (14) If such values are set that the active width of positive phase or negative phase becomes "0" or "minus" according to the above calculation methods, pins TO00 to TO05 output a waveform with "0" active width and fixed at the inactive level. (p.115, 122, 131)

- (15) When timer 0 is in PWM mode 0 (toothed modulation), in the 1st cycle, even if CM00 to CM02 match, F/F does not change. Therefore, the positive phase (TO00, TO02 and TO04) of the output remains at the inactive level and the negative phase (TO01, TO03 and TO05) remains at the active level. The PWM waveform can be output from the 2nd cycle. The active level is set by the ALVTO bit of the TUM0 register. (p.129)
- (16) SBUF0 to SBUF5 and MBUF0 to MBUF5 are 8-bit access registers. However, only the lower 6 bits are output to TO00 to TO05. The higher 2 bits are ignored (fixed to "0" by hardware). (p.137)
- (17) It is prohibited to change the PRM11 and PRM10 bits of the TMC1 register during the operation of TM1 (CE1 = 1). (p.142)
- (18) To use CC20 as a compare register (CCM2 = 1), INTP3 cannot be used as an external interrupt pin. (p.147)
- (19) The CC20 incorporates the compare register and capture register separately on hardware. Either of the two can be selected by setting timer control register 2 (TMC2). Writing is only possible to the compare register. Reading of contents is possible from either register which is selected. (p.147)
- (20) Bits 6 and 5 of the TMC2 register are fixed to "0" by hardware. Even if "1" is written, they remain "0". (p.147)
- (21) It is prohibited to change the CCM2, CLR2 and PRM22 to PRM20 bits of the TMC2 register during the operation of TM2 (CE2 = 1). (p.147)
- (22) Bits 7 to 4 of the INTM1 register are fixed to "0" by hardware. Even if "1" is written, they remain "0". (p.148, 160)
- (23) INTCC20 and INTP3 share the interrupt vector table. (p.151)
- (24) To use CC30 as a compare register (CCM3 = 1), INTP0 cannot be used as an external interrupt pin. (p.159)
- (25) The CC30 incorporates the compare register and capture register separately on hardware. Either of the two can be selected by the setting of timer control register 3 (TMC3). Writing is only possible to the compare register. Reading of contents is possible from either register which is selected. (p.159)
- (26) Bits 6 and 5 of the TMC3 register are fixed to "0" by hardware. Even if "1" is written, they remain "0". (p.159)
- (27) It is prohibited to change the CCM3, CLR3 and PRM32 to PRM30 bits of the TMC3 register during the operation of TM3 (CE3 = 1). (p.159)
- (28) Bits 7 to 2 of the SMPC1 register are fixed to "0" by hardware. Even if "1" is written, they remain "0". (p.162)
- (29) INTCC30 and INTP0 share the interrupt vector table. (p.164)
- (30) The specification by the bits 0, 1, and 4-6 of the timer unit mode register 1 (TUM1) are valid only when CMD = 1. The specification of the bits 2 and 3 are valid only when CMD = 0. (p.174)
- (31) If the operation of TM4 is set in mode 4 (specified by the TMC4 register), the specification (made by the TUM1 register) of the valid edge of the TIUD pin is invalid. (p.174, 179, 186, 191)

- (32) It is prohibited to change each bit of the TUM1 register during the operation of TM4 (CE4 = 1). (p.174)
- (33) The specification of the \bar{U}/D bit of timer control register 4 (TMC4) is valid only when CMD = 1. (p.176)
- (34) It is prohibited to change the ENMD, \bar{U}/D , \bar{I}/E , PRM41 and PRM40 bits of timer control register 4 (TMC4) during the operation of TM4 (CE4 = 1). (p.176)
- (35) The TMC4 register's manipulation of the \bar{U}/D bit varies depending on the setting of the CMD bit of the TUM1 register as follows. (p.178)
- When CMD = 0,
- The \bar{U}/D bit is fixed to "0" by hardware. Even if "1" is written, it remains "0". If this bit is read, the value read is always "0".
- When CMD = 1 and the internal clock is selected,
- The \bar{U}/D bit is write-enabled. If reading operation is applied to it, the written value is read.
- When CMD = 1 and the external clock is selected,
- Writing to the \bar{U}/D bit is not possible by hardware. If reading operation is applied to it, the up/down status of TM4 is read.
- (36) While TM4 is stopped (CE4 = 0), even if "1" is written to the \bar{U}/D bit, it remains "0" as it is not possible to rewrite it. If it is desirable to allow TM4 perform down operation from the beginning, simultaneously set (1) the CE4 bit and \bar{U}/D bit of the TMC4 register by the MOV instruction. (p.178, 185)
- (37) Setting ENMD = 1 is prohibited in the UDC mode. (p.183)
- (38) If the TIUD pin edge and TCUD pin edge are input simultaneously in mode 4 of TM4, TM4 continues counting while maintaining the same up/down operation immediately before the input. (p.191)
- (39) The real-time output port register (RTP) is an 8-bit access register. When the RTP register is read, the data set in the RTP register is placed in the lower 4 bits. The higher 4 bits are fixed to "0" by hardware, and therefore "0" is read. (p.195)
- (40) Bits 7, 6, 3 and 2 of the real-time output port mode register (RTPM) are fixed to "0" by hardware. Even if "1" is written to them, they remain "0". (p.196)
- (41) The higher 4 bits of the RTP register have no latch circuit and are fixed to "0". Therefore, even if "1" is written to the higher 4 bits, they remain "0". (p.197)

20.6 Cautions for CHAPTER 8 A/D CONVERTER

- (1) Connect a capacitor to the analog input pins (ANI0-ANI7) and reference voltage input pin (AV_{REF}) to prevent malfunctioning due to noise. (p.201)
- (2) Do not apply a voltage exceeding the range of AV_{SS} to AV_{DD} to the pins used as the input pins of the A/D converter. (p.201, 209)
- (3) Observe the ratings of the voltage input to ANI0-ANI7. If a voltage exceeding V_{DD} or falling below V_{SS} (even if the voltage is in the range of the absolute maximum ratings) is input to these pins, the conversion value of the channel is undefined. In addition, the conversion value of the other channels may also be affected. (p.202)

- (4) The A/D converter of the μ PD78366A cannot stop conversion. Therefore, when data has been set to the ADM register, conversion operation continues in the set operation mode, until the contents of the ADM register are rewritten. (p.206)
- (5) When performing branch processing directly using the values resulting from the A/D conversion, if a program is created that branches only when the conversion result reaches a specific value, the conversion result may not reach that specific value due to the effect of a conversion error and the program may not be able to branch according to the prescribed routine. Therefore, create a program that will branch when the conversion result is within the range of the overall error. (p.212)
- (6) To execute the select operation by using the software trigger, and to start A/D conversion in the scan mode again, the A/D conversion executed before the software trigger was input is resumed. (p.219, 221)
- (7) To execute the select operation by using the external trigger or interrupt trigger and to start A/D conversion in the scan mode again, the A/D conversion executed before the external trigger or interrupt trigger was input is resumed. (p.219, 221)

20.7 Cautions for CHAPTER 9 ASYNCHRONOUS SERIAL INTERFACE

- (1) Bit 7 of the asynchronous serial interface mode register (ASIM) is fixed to "1" by hardware. Even if "0" is written to it, it remains "1". (p.229, 233, 235, 243)
- (2) Usually, the transmission end interrupt (INTST) occurs when the transmit shift register (TXS) becomes empty. However, the interrupt does not occur even if the transmission shift register becomes empty due to the RESET input. (p.240, 241)
- (3) Data written to the TXS register is invalid while a transmission operation is in progress, until INTST is generated. (p.240)
- (4) Be sure to read the receive buffer (RXB) even when a reception error has occurred. Otherwise, an overrun error occurs when the next data is received, and the reception error status persists. (p.243, 247)
- (5) The contents of the asynchronous serial interface status register (ASIS) are reset (to 0) when the receive buffer (RXB) is read or when the next data is received. To identify the nature of the error, be sure to read the ASIS register before reading the receive buffer (RXB).
If the received data is transferred to memory using a macro service, the reception buffer (RXB) is read during the reception of serial data, and therefore the ASIS register is reset (0). Thus, it is not possible to know more than the fact that an error has occurred. Be sure to check if there is no problem concerning this point before using the register.
An error can be detected when the reception error interrupt request flag (SERIF) is set (1) or by acknowledgment of a reception error interrupt (INTSER). (p.247)
- (6) Bits 7 to 3 of the ASIS register are fixed to "0" by hardware. Even if "1" is written to them, they remain "0". (p.247)

20.8 Cautions for CHAPTER 10 CLOCKED SERIAL INTERFACE

- (1) Selection of a serial clock is performed in asynchronous with the serial clock. If the serial clock is changed while communication is in progress, a serial clock of undefined width may be output. Do not change the serial clock during communication. (p.255)
- (2) Since the serial data bus pin (SB0 or SB1) serves as an open-drain output pin in the SBI mode, the serial data bus line is wired-ORed. Therefore, a pull-up resistor must be connected to the serial data bus line. (p.272)
- (3) To exchange the master with a slave in the SBI mode, the input mode and output mode of $\overline{\text{SCK}}$ of the master and slave is switched asynchronously. Therefore, a pull-up resistor must also be connected to $\overline{\text{SCK}}$. (p.272)
- (4) The transition of the SB0 (SB1) line (transition from low to high or high to low) while the $\overline{\text{SCK}}$ line is high in the SBI mode is recognized as a bus release signal or a command signal. If the timing of changes on the bus shifts due to the influence of board capacitance, etc., a bus release signal (or a command signal) may be detected even while data is being transmitted. Exercise care when routing the wiring. (p.274)
- (5) Do not manipulate the RELT and CMDT bits of the serial bus interface control register (SBIC) during transmission/reception. (p.282)
- (6) Be sure to specify a pin and serial clock before setting the CTXE and CRXE bits in the SBI mode. (p.286)

20.9 Cautions for CHAPTER 11 PWM SIGNAL OUTPUT FUNCTION

- (1) To output PWM signal, be sure to set CNTE bit of PWM control register 1 (PWMC1) to 1. (p.296)
- (2) The PWMC1 register is a control register common to PWM0 and PWM1. (p.296)
- (3) Bits 4 to 7 of the PWMC1 register are fixed to "0" by hardware. Even if "1" is written to them, they remain "0". (p.296)
- (4) A byte access/bit access can be made to the low-order part of the PWM0/PWM1 register, but cannot be made to the high-order part. (p.296)
- (5) Bits 12 to 15 of the PWM0/PWM1 registers are fixed to "0" by hardware. Even if "1" is written to them, they remain "0". (p.296)

20.10 Cautions for CHAPTER 12 WATCHDOG TIMER

- (1) Data can be written into the watchdog timer mode register (WDM) only by a dedicated instruction (MOV WDM, #byte). (p.301)
- (2) Set the priority of interrupt requests by the WDM register at the time of initialization of the application system such as initialization of the stack pointer, and do not dynamically change it in execution of the program. (p.301)
- (3) Once it is set (1), the RUN bit of the WDM register cannot be reset to 0 by software. (p.301)
- (4) The count clock is not reset even when the watchdog timer is cleared by setting the RUN bit of the WDM register to 1. (p.301)

- (5) If a watchdog timer interrupt and NMI interrupt are generated simultaneously when $PRC = 1$ ($INTWDT > NMI$), execute the watchdog timer interrupt service routine after executing the first 1 instruction of the NMI interrupt service routine. Therefore, when used with the $PRC = 1$ setting, use an NOP instruction as the first instruction of the NMI interrupt service routine. (p.301)
- (6) Bits 6, 5 and 0 of the WDM register are fixed to “0” by hardware. Even if “1” is written to them, they remain “0”. (p.301)
- (7) Be sure to write 0 to bit 3 of the WDM register. (p.301)
- (8) Immediately after the power is turned on, the watchdog timer output pin (\overline{WDTO}) may go low for a period of up to 32 clocks. (p.302)
- (9) If the watchdog timer is operating as a timer for the oscillation stabilization time after release of the STOP mode, the \overline{WDTO} pin remains at the inactive level even if an overflow occurs. (p.302)

20.11 Cautions for CHAPTER 13 INTERRUPT FUNCTION

- (1) Bits 3 and 2 of the interrupt control register are fixed to “0” by hardware. Even if “1” is written to them, they remain “0”. (p.311, 312)
- (2) Bits 6 to 0 of the interrupt mode control register (IMC) are fixed to “0” by hardware. Even if “1” is written to them, they remain “0”. (p.315)
- (3) The in-service priority register (ISPR) is a read-only register. Writing to it may cause misoperation. (p.316)
- (4) Bits 7 to 4 of the ISPR register are fixed to “0” by hardware. Even if “1” is written to them, they remain “0”. (p.316)
- (5) A macro service request is acknowledged and serviced even in a non-maskable interrupt service routine. If it is desirable not to execute a macro service in the non-maskable interrupt service routine, manipulate the mask flag register during the non-maskable interrupt service routine to prevent any macro service from being generated. (p.321)
- (6) Be sure to use the RETI instruction for restoring from the non-maskable interrupt. Acknowledgment of subsequent interrupts otherwise would not be normally carried out by other instructions. (p.321)

- (7) A non-maskable interrupt always is acknowledged except when another non-maskable interrupt is being serviced (unless a non-maskable interrupt request with higher priority is generated while a non-maskable interrupt with lower priority is being serviced) and except for a certain period after execution of a specific instruction shown in 13.9. Therefore, a non-maskable interrupt is also acknowledged when the stack pointer value is undefined, especially after release of a reset, etc. In this case, depending on the stack pointer value, the program counter (PC) or program status word (PSW) may be written to the address where writing to the special function register is disabled (refer to **Table 3-4** in **3.2.3 Special function register (SFR)**). This will cause the CPU to deadlock, or an unexpected signal to be output from the pin or PC or PSW to be written to the address where no RAM is mounted, in which case the program cannot return normally from the non-maskable interrupt service routine to the main routine and it goes into an inadvertent loop. Therefore, be sure to program as follows after release of RESET. (p.321)

```

CSEG  AT 0
DW    STRT
STRT:
MOVW SP, #imm16

```

- (8) If a maskable interrupt is acknowledged by a vectored interrupt, be sure to restore the program by the RETI instruction. Operation of subsequent interrupts otherwise would not be normally carried out by other instructions. (p.324)
- (9) Be sure to use the RETCS instruction for returning from the interrupt by context switching. Operation of subsequent interrupts otherwise would not be normally carried out by other instructions. (p.325)
- (10) Do not use the RETI instruction to restore from a software interrupt executed by the BRK instruction. (p.332)
- (11) If the context switching function is activated by executing the BRKCS instruction, resetting (0) the ISPR register bits while the RETCS instruction may destroy the interrupt nesting control.
Be sure to use the RETCSB instruction to restore from the processing activated by the BRKCS instruction. (p.334)
- (12) If data is transmitted by UART, using the macro service, a vectored interrupt request is generated two times (p.341).
- (13) Word buffers must be placed at even addresses in the block transfer mode. (p.347)
- (14) Word buffers must be placed at even addresses in the block transfer mode (with a memory pointer). (p.349)
- (15) MEM.PTR must be placed at an even address in the block transfer mode (with a memory pointer). (p.349)
- (16) Don't set 00H in the MSC in the data difference mode. (p.351)
- (17) Buffers must be placed at even addresses in the data difference mode. (p.351)
- (18) The "previous value" must be initialized to dummy data in advance in the data difference mode. (p.351)
- (19) The SFRP can specify 16-bit SFRs only in the data difference mode. (p.351)

20.12 Cautions for CHAPTER 14 STANDBY FUNCTION

- (1) After the SBF flag in the standby control register (STBC) is read, set it to 1. Software can then discriminate a power-on reset from release of the STOP or HALT mode. (p.358)
- (2) To use the external clock, do not set the STP bit of the standby control register (STBC) to 1. (p.359)
- (3) If the interrupt request flag ($\times\times IF$) is set to 1 and the interrupt is not masked ($\times\times MK = 0$), the system does not enter the HALT mode. When macro service processing ($\times\times ISM = 1$) is performed, the system enters the HALT mode after the macro service terminates. (p.360)
- (4) Data can be written into the watchdog timer mode register (WDM) only by a dedicated instruction (MOV WDM, #byte). (p.365)
- (5) The priority of interrupt requests should be set by the WDM register at initialization of the application system such as initialization of the stack pointer, and should not be dynamically changed during execution of the program. (p.365)
- (6) Once it is set to 1, the RUN bit of the WDM register cannot be reset to 0 by software. (p.365)
- (7) The count clock is not reset even when the watchdog timer is cleared by setting the RUN bit of the WDM register to 1. (p.365)
- (8) If a watchdog timer interrupt and NMI interrupt are generated simultaneously with $PRC = 1$ ($INTWDT > NMI$), after the first 1 instruction of the NMI interrupt service routine is executed, the watchdog timer interrupt service routine is executed. Therefore, when used with $PRC = 1$, the first instruction of the NMI interrupt service routine should be an NOP instruction. (p.365)
- (9) Bits 6, 5 and 0 of the WDM register are fixed to "0" by hardware. Even if "1" is written to them, they remain "0". (p.365)
- (10) Be sure to write 0 to bit 3 of the WDM register. (p.365)

20.13 Cautions for CHAPTER 15 RESET FUNCTION

- (1) When \overline{RESET} is active, all pins except \overline{WDTO} , AV_{REF} , AV_{DD} , AV_{SS} , V_{DD} , V_{SS} , X1, and X2 go into the high-impedance state. (p.373)
- (2) When memory is expanded externally with RAM, attach pull-up resistors to the P90/ \overline{RD} pin and P91/ \overline{WR} pin. If these pins may go into the high-impedance state, the contents of the external RAM may be lost. Or signal collision on the address/data bus may damage the input/output circuit. (p.373)

20.14 Cautions for CHAPTER 16 BUS INTERFACE FUNCTION

- (1) The internal memory (ROM, RAM) capacity of the μ PD78P368A can be changed by setting the memory expansion mode register (MM). (p.382)
- (2) Bits 3, 4 and 7 of the MM register are fixed to "0" by hardware. Even if "1" is written to them, they remain "0". (p.385)
- (3) Invalid combinations are specified as "Setting prohibited" in format of MM register. Never write these combinations. (p.385)
- (4) The number of cycles shown in Figure 16-6 is when no address wait cycle is appended. If an address wait cycle is appended, one cycle must be added. (p.386)
- (5) Instruction fetch from and data access to the peripheral RAM area (F700H-FDFFH, in case of the μ PD78363A FC00H-FDFFH) are enabled, but wait specification by setting the PWC register is invalid. The peripheral RAM area operates with a 16-bit bus. Instruction fetch becomes high-speed fetch. (p.386)
- (6) Instructions cannot be fetched from the main RAM area (FE00H-FEFFFH). Wait specification by setting the PWC register when data is accessed is invalid. The main RAM area is accessed in a 16-bit units. (For bus cycles, special two bus cycles are started.) (p.386)
- (7) To make a word access to the main RAM area (containing stack handling), only even addresses can be specified in operands. (p.386)
- (8) The internal ROM area operates with 16-bit bus regardless how the PWC register is set. Wait specification can be set in the PWC register. (p.386)
- (9) To access external memory, do not specify high-speed fetch. (p.386)
- (10) External SFR area (FFD0H-FFDFH) wait specification is set in the PWC7 and PWC6 bits. (p.386)
- (11) To access the internal memory in the high-speed fetch mode, set 0 in the AW0 and AW1 bits of the PWC register. (p.387)
- (12) Bits 8-13 of the PWC register are fixed to "0" by hardware. Even if "1" is written to them, they remain "0". (p.387)

20.15 Cautions for CHAPTER 19 INSTRUCTION EXECUTION RATE

- (1) When a word access to the main RAM area (FE00H to FEFFFH) (containing stack handling) is executed, addresses specified in operands are limited to even addresses. (p.421)
- (2) The PWC register specifies the mode of an area, irrespective of whether the area is in internal memory or external memory. After reset, the whole space is set in the normal fetch cycle mode. (p.422)

[MEMO]

APPENDIX A DIFFERENCE BETWEEN μ PD78366A AND μ PD78328

The following table shows the differences between the μ PD78366A subseries (μ PD78363A, 78365A, 78366A, 78368A, and 78P368A) and the μ PD78328 subseries (μ PD78327, 78328, 78P328):

Function List (1/2)

Part Number		μPD78363A	μPD78365A	μPD78366A	μPD78368A	μPD78P368A
Parameter						
Minimum instruction execution time		125 ns (internal clock: 16 MHz, external clock: 8 MHz)				
Internal memory	ROM	24K bytes	–	32K bytes	48K bytes	–
	PROM	–	–	–	–	48K bytes
	RAM	768K bytes	2K bytes			
Memory space		64K bytes (externally expandable)				
General-purpose registers		8-bit × 16 × 8 banks				
Basic instructions		115				
Instruction set		<ul style="list-style-type: none">• 16-bit transfer/operation• Multiplication/division (16 bits × 16 bits, 32 bits ÷ 16 bits)• Bit manipulation• String• Sum-of-products operation (16 bits × 16 bits + 32 bits)• Relative operation				
I/O line	Input	49	14 (8 multiplexed with analog inputs)			
	I/O	49	31	49		
Real-time pulse unit		<ul style="list-style-type: none">• 16-bit timer × 5• 16-bit compare register × 7• 16-bit capture register × 3• 16-bit capture/compare register × 2• Two output modes selectable<ul style="list-style-type: none">Mode 0 set/reset output: 6 channelsMode 1 buffer output: 6 channels• 16-bit resolution PWM output: 1 channel				
Real-time output port		4 (buffer output in 4-bit units)				
PWM unit		8-/9-/10-/12-bit resolution variable PWM output: 2 channels				
A/D converter		10-bit resolution, 8 channels				
Serial interface		With dedicated baud rate generator <ul style="list-style-type: none">UART (with pin select function): 1 channelClock-synchronized serial interface/SBI: 1 channel				
Interrupt function		<ul style="list-style-type: none">• External: 6, internal: 14 (2 multiplexed with external sources)• Four priority levels selectable via software• Three service modes selectable (vectored interrupt/macro service/context switching)				
Test		–				
PLL control circuit		Available (External 8 MHz → Internal 16 MHz)				
Package	w/o window	80-pin plastic QFP (14 × 20 mm)				
	w/window	–				80-pin ceramic WQFN
Others		<ul style="list-style-type: none">• Watchdog timer incorporated• Standby function (HALT mode, STOP mode)				

Function List (2/2)

Part Number		μ PD78327	μ PD78328	μ PD78P328
Parameter				
Minimum instruction execution time		250 ns (internal clock: 8 MHz, external clock: 16 MHz)		
Internal memory	ROM	—	16K bytes	—
	PROM	—	—	16K bytes
	RAM	512 bytes		
Memory space		64K bytes (externally expandable)		
General-purpose registers		8-bit \times 16 \times 8 banks		
Basic instructions		111		
Instruction set		<ul style="list-style-type: none"> • 16-bit transfer/operation • Multiplication/division (16 bits \times 16 bits, 32 bits \div 16 bits) • Bit manipulation • String 		
I/O line	Input	11 (8 multiplexed with analog inputs)		
	I/O	23	41	
Real-time pulse unit		<ul style="list-style-type: none"> • 16-bit timer \times 3 • 16-bit compare register \times 14 • 16-bit capture/compare register \times 1 • Two output modes selectable Mode 0 set/reset output: 6 channels, toggle output: 1 channel Mode 1 buffer output: 8 channels		
Real-time output port		4/8 (buffer output in 4-/8-bit units)		
PWM unit		8-bit resolution PWM output: 1 channel		
A/D converter		10-bit resolution, 8 channels		
Serial interface		With dedicated baud rate generator UART: 1 channel Clocked serial interface/SBI: 1 channel		
Interrupt function		<ul style="list-style-type: none"> • External: 4, internal: 17 • Three priority levels selectable via software • Three service modes selectable (vectored interrupt/macro service/context switching)		
Test		Internal: 1		
PLL control circuit		None		
Package	w/o window	<ul style="list-style-type: none"> • 64-pin plastic shrink DIP • 64-pin plastic QFP (14 \times 20 mm) 		
	w/window	—		64-pin ceramic shrink DIP with window
Others		<ul style="list-style-type: none"> • Watchdog timer • Standby function (HALT mode, STOP mode) 		

[MEMO]

APPENDIX B TOOLS

B.1 Development Tools

The following tools are provided for developing a system that uses the μ PD78366A:

Language Processor

78K/III series relocatable assembler (RA78K3)	Relocatable assembler that can be commonly used with 78K/III series. Provided with macro function, this assembler enhances development efficiency. Structured assembler that explicitly describes program control structure is also supplied to improve program productivity and maintainability.		
	Host machine	OS	Part number (Product name)
	PC-9800 series	MS-DOS™	3.5"2HD
			5"2HD
	IBM PC/AT and its compatibles	PC DOS™	3.5"2HC
			5"2HC
	HP9000 series 700™	HP-UX™	DAT
	SPARCstation™	SunOS™	Cartridge tape
	NEWS™	NEWS-OS™	(QIC-24)
78K/III series C compiler (CC78K3)	C compiler that can be commonly used with 78K/III series, and that converts program written in C language into object code that can be executed by microcontroller. When using this compiler, 78K/III series relocatable assembler (RA78K3) is necessary.		
	Host machine	OS	Part number (Product name)
	PC-9800 series	MS-DOS	3.5"2HD
			5"2HD
	IBM PC/AT and its compatibles	PC DOS	3.5"2HC
			5"2HC
	HP9000 series 700	HP-UX	DAT
	SPARCstation	SunOS	Cartridge tape
	NEWS	NEWS-OS	(QIC-24)

Remark The operation of the relocatable assembler and C compiler are guaranteed with the above host machines and OSs only.

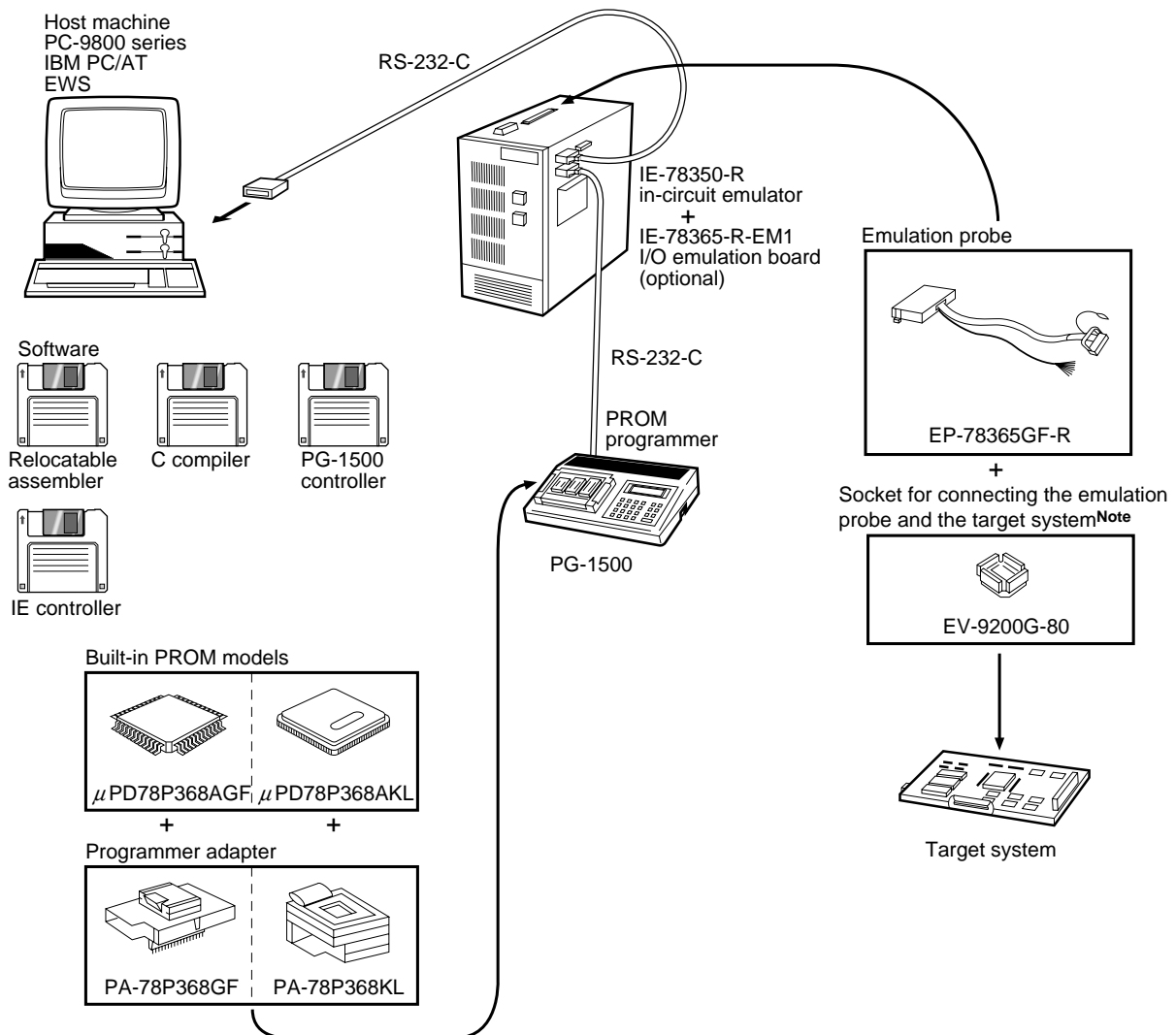
PROM Writing Tools

Hardware	PG-1500	PROM programmer that can program single-chip microcontroller with PROM in standalone mode or under control of host machine when connected with supplied accessory board and optional programmer adapter. It can also program representative PROMs including 256K-bit models and 4M-bit models.			
	PA-78P368GF PA-78P368KL	PROM programmer adapter to write program to μ PD78P368A on general-purpose PROM programmer such as PG-1500. PA-78P368GF ... for μ PD78P368AGF PA-78P368KL ... for μ PD78P368AKL			
Software	PG-1500 controller	Connects PG-1500 to host machine with serial and parallel interfaces and controls PG-1500 on host machine.			
		Host machine			Part number (Product name)
			OS	Distributed media	
		PC-9800 series	MS-DOS	3.5"2HD	μ S5A13PG1500
				5"2HD	μ S5A10PG1500
		IBM PC/AT and its compatibles	PC DOS	3.5"2HD	μ S7B13PG1500
				5"2HC	μ S7B10PG1500

Debugging Tools (When using IE controller)

Hardware	IE-78350-R	In-circuit emulator connected to host system to develop and debug application system.			
	IE-78365-R-EM1	I/O emulation board to emulate peripheral functions of μ PD78366A such as I/O ports.			
	IE-78365GF-R	Emulation probe that connects IE-78350-R to target system. One conversion socket, EV-9200G-80, that is used to connect target system is supplied.			
	EV-9200G-80				
Software	IE-78350-R control program (IE controller)	Program that controls IE-78350-R on host machine. Can execute commands automatically, to enhance debugging efficiency.			
		Host machine		Part number (Product name)	
			OS		Distributed media
		PC-9800 series	MS-DOS	3.5"2HD	μ S5A13IE78365A
				5"2HD	μ S5A10IE78365A
		IBM PC/AT and its compatibles	PC DOS	3.5"2HC	μ S7B13IE78365A
				5"2HC	μ S7B10IE78365A

Remark The operation of PG-1500 controller and IE controller are guaranteed with the above host machine and OS only.

Configuration of development tools (When using IE controller)

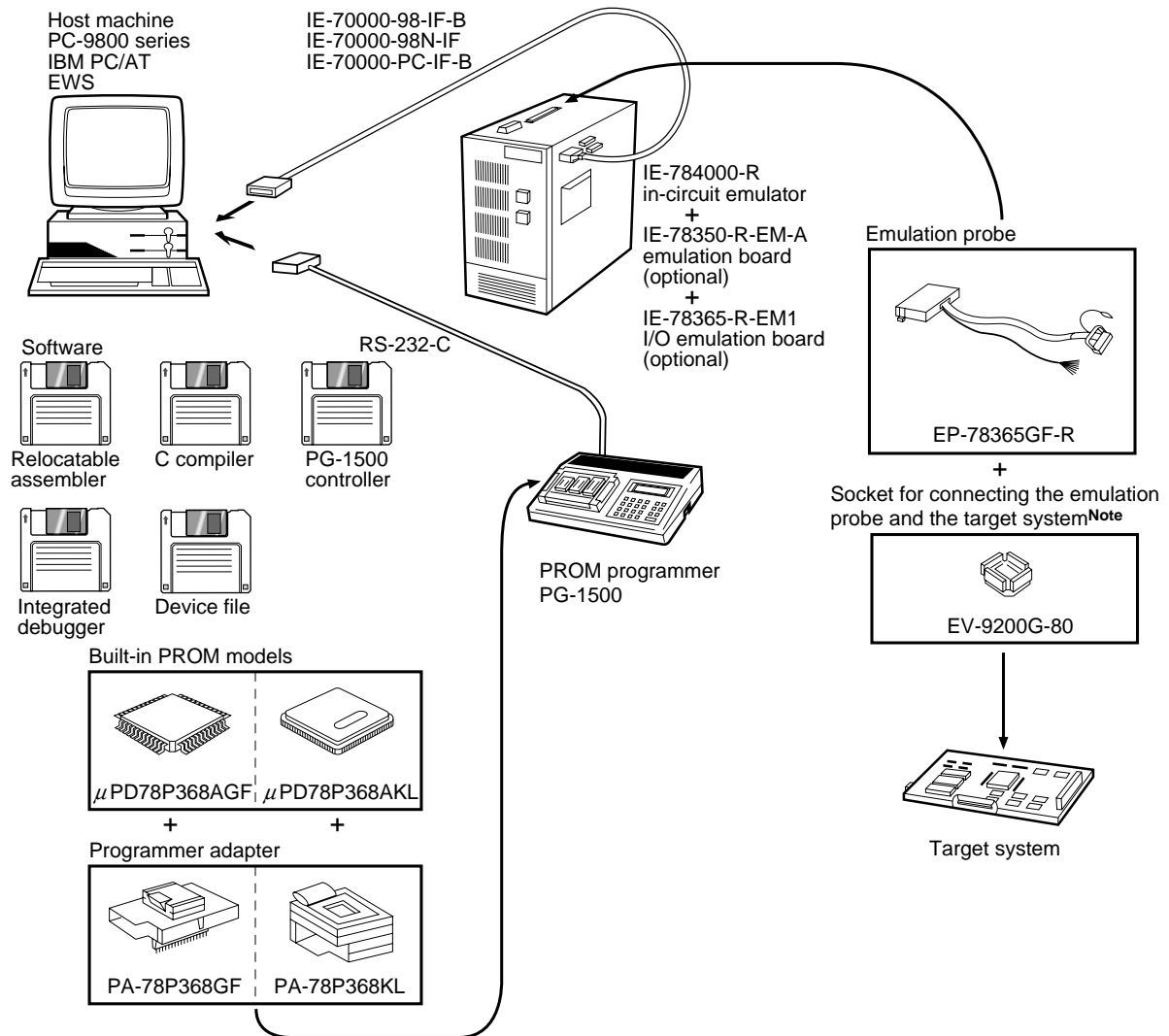
Note The socket is supplied with the emulation probe.

- Remarks** 1. The PG-1500 can be directly connected to the host machine via the RS-232-C interface.
2. In this figure 3.5-inch FD represents distributed media of software.

Debugging Tools (When using integrated debugger)

Hardware	IE-784000-R		In-circuit emulator that can be used for development and debugging of the application system. It performs debugging with host machine connected to it.			
	IE-78350-R-EM-A		Emulation board to perform emulation of peripheral functions such as target device input/output ports			
	IE-78365-R-EM1		I/O emulation board to perform emulation of peripheral functions such as target device input/output ports			
	EP78365GF-R		Emulation probe to connect IE-784000-R to target system. It comes with one conversion socket EV-9200G-80 used for connection with target system.			
	EV-9200G-80					
	IE-70000-98-IF-B		Interface adapter used when PC-9800 series (except notebook personal computer) is used as host machine			
	IE-70000-98N-IF		Interface adapter and cable used when PC-9800 series notebook personal computer is used as host machine			
	IE-70000-PC-IF-B		Interface adapter used when IBM PC/AT is used as host machine			
	IE-78000-R-SV3		Interface adapter and cable used when EWS is used as host machine			
Software	Integrated debugger (ID78K3)		Program to control 78K/III series in-circuit emulator. Used in combination with device file (DF78365). It can perform source-program level debugging for programs written in C language, structured assembly language, and assembly language. With its capability of displaying various types of information simultaneously by dividing the host machine screen, efficient debugging is possible.			
			Host machine		Part number (Product name)	
				OS	Distributed media	
			PC-9800 series	MS-DOS+Windows™	3.5"2HD	μSAA13ID78K3
					5"2HD	μSAA10ID78K3
			IBM PC/AT and its compatibles (Japanese Windows)	PC DOS+Windows	3.5"2HC	μSAB13ID78K3
					5"2HC	μSAB10ID78K3
			IBM PC/AT and its compatibles (English Windows)		3.5"2HC	μSBB13ID78K3
					5"2HC	μSBB10ID78K3
			Device file (DF78365)		File that contains device-specific information. Used in combination with assembler (RA78K3), C compiler (CC78K3) and integrated debugger (ID78K3).	
	Host machine				Part number (Product name)	
		OS			Distributed media	
	PC-9800 series	MS-DOS			3.5"2HD	μS5A13DF78365
					5"2HD	μS5A10DF78365
	IBM PC/AT and its compatibles	PC DOS			3.5"2HC	μS7B13DF78365
					5"2HC	μS7B10DF78365

Remark Operation of the integrated debugger and device file is guaranteed only on the host machines and operating systems quoted above.

Configuration of development tools (When using integrated debugger)

Note A socket is provided with the emulation probe.

Remarks 1. Desk top-type PC represents host machine in this figure.
2. 3.5-inch FD represents the distributed media of software in this figure.

B.2 Built-In Software

The following built-in software is provided for efficient program development and maintenance.

Real-time Operation System

Real-time OS (RX78K/III) ^{Note}	The RX78K/III is intended to provide a multitask environment for a control field where real-time operation is required. The CPU idle time can be assigned to other processing for performance improvement of the entire system. The RX78K/III provides system call conforming to the μ ITRON specifications. The RX78K/III package provides a tool (configurator) to create information tables and RX78K/III nucleus.		
	Host machine		Part number (Product name)
		OS	
	PC-9800 series	MS-DOS	3.5"2HD
			5"2HD
	IBM PC/AT and its compatibles	PC DOS	3.5"2HC
			5"2HC

Note Under development

Caution To purchase the real time operation system, enter on purchase application form and make a written agreement for use before purchasing it.

Remark The RA78K/III assembler package (option) is required to use the RX78K3 Real-time operation system.

Fuzzy Inference Development Support System

Fuzzy knowledge data preparation tool	A program which supports editing and simulation of fuzzy knowledge data (fuzzy rules and membership functions).			
	Host machine			Part number (Product name)
	OS	Distributed media		
	PC-9800 series	MS-DOS	3.5"2HD	μ S5A13FE9000
			5"2HD	μ S5A10FE9000
	IBM PC/AT and its compatibles	PC DOS+Windows	3.5"2HC	μ S7B13FE9200
5"2HC			μ S7B10FE9200	
Translator (FT78K3) ^{Note}	A program which converts fuzzy knowledge data prepared with the fuzzy knowledge data preparation tool into an assembler source program for RA78K3.			
	Host machine			Part number (Product name)
	OS	Distributed media		
	PC-9800 series	MS-DOS	3.5"2HD	μ S5A13FT78K3
			5"2HD	μ S5A10FT78K3
	IBM PC/AT and its compatibles	PC DOS	3.5"2HC	μ S7B13FT78K3
5"2HC			μ S7B10FT78K3	
Fuzzy inference module (FI78K/III) ^{Note}	A program which executes fuzzy inference. It is linked with the fuzzy knowledge data converted by the translator, thereby executing fuzzy inference.			
	Host machine			Part number (Product name)
	OS	Distributed media		
	PC-9800 series	MS-DOS	3.5"2HD	μ S5A13FI78K3
			5"2HD	μ S5A10FI78K3
	IBM PC/AT and its compatibles	PC DOS	3.5"2HC	μ S7B13FI78K3
5"2HC			μ S7B10FI78K3	
Fuzzy inference debugger (FD78K/III)	Support software for evaluating or adjusting fuzzy knowledge data at the hardware level with in-circuit emulator.			
	Host machine			Part number (Product name)
	OS	Distributed media		
	PC-9800 series	MS-DOS	3.5"2HD	μ S5A13FD78K3
			5"2HD	μ S5A10FD78K3
	IBM PC/AT and its compatibles	PC DOS	3.5"2HC	μ S7B13FD78K3
5"2HC			μ S7B10FD78K3	

Note Under development

[MEMO]

APPENDIX C REGISTER INDEX**C.1 Register Index (In Alphabetical Order with Respect to Register Name)****[A]**

A/D conversion result register: ADCR ... 207
A/D converter mode register: ADM ... 203
Asynchronous serial interface mode register: ASIM ... 226
Asynchronous serial interface status register: ASIS ... 226

[B]

Baud rate generator compare register: BRG ... 236, 256
Baud rate generator control register: BRGC ... 236, 256
Buffer register CM00-CM03: BFCM00-BFCM03 ... 100

[C]

Capture register 20: CT20 ... 145
Capture register 30: CT30 ... 157
Capture register 31: CT31 ... 157
Capture/compare register 20: CC20 ... 145
Capture/compare register 30: CC30 ... 157
Clocked serial interface mode register: CSIM ... 250
Compare register 00: CM00 ... 100
Compare register 01: CM01 ... 100
Compare register 02: CM02 ... 100
Compare register 03: CM03 ... 100
Compare register 10: CM10 ... 141
Compare register 40: CM40 ... 172
Compare register 41: CM41 ... 172
CPU control word: CCW ... 46

[E]

External interrupt mode register 0: INTM0 ... 109, 160
External interrupt mode register 1: INTM1 ... 148, 160

[I]

In-service priority register: ISPR ... 307, 316
Interrupt control register (INTAD): ADIC ... 307, 309
Interrupt mask flag register: MK0L, MK0H ... 307, 313
Interrupt mode control register: IMC ... 307, 315

[M]

Master buffer register: MBUF0-MBUF5 ... 101
Memory expansion mode register: MM ... 85, 384

[P]

Port 0: P0 ... 76
Port 0 mode control register: PMC0 ... 82
Port 0 mode register: PM0 ... 77
Port 1: P1 ... 76
Port 1 mode register: PM1 ... 77
Port 2: P2 ... 76
Port 3: P3 ... 76
Port 3 mode control register: PMC3 ... 83, 230, 252
Port 3 mode register: PM3 ... 78, 231, 253
Port 4: P4 ... 76
Port 5: P5 ... 76
Port 5 mode register: PM5 ... 78
Port 7: P7 ... 76
Port 8: P8 ... 76
Port 8 mode register: PM8 ... 78
Port 8 mode control register: PMC8 ... 84
Port 9: P9 ... 76
Port 9 mode register: PM9 ... 79
Port read control register: PRDC ... 72
Programmable wait control register: PWC ... 386
Pull-up resistor option register H: PUOH ... 87
Pull-up resistor option register L: PUOL ... 87
PWM buffer register: PWM0, PWM1 ... 296
PWM control register 0: PWMC0 ... 295
PWM control register 1: PWMC1 ... 296

[R]

Real-time output port mode register: RTPM ... 196
Real-time output port register: RTP ... 197
Receive buffer: RXB ... 227
Reload register: DTIME ... 99

[S]

Sampling control register 0: SMPC0 ... 149, 161
Sampling control register 1: SMPC1 ... 162
Serial bus interface control register: SBIC ... 250
Shift register: SIO ... 250
Slave buffer register: SBUF0-SBUF5 ... 101
Standby control register: STBC ... 358

[T]

Timer 0: TM0 ... 98
Timer 1: TM1 ... 141
Timer 2: TM2 ... 145
Timer 3: TM3 ... 157
Timer 4: TM4 ... 172
Timer control register 0: TMC0 ... 106
Timer control register 1: TMC1 ... 108, 142
Timer control register 2: TMC2 ... 146
Timer control register 3: TMC3 ... 158
Timer control register 4: TMC4 ... 175
Timer out register: TOUT ... 101
Timer unit mode register 0: TUM0 ... 102
Timer unit mode register 1: TUM1 ... 173
Transmit shift register: TXS ... 227

[W]

Watchdog timer mode register: WDM ... 300, 365

C.2 Register Index (In Alphabetical Order with Respect to Register Symbol)**[A]**

ADCR0 : A/D conversion result register 0 ... 207
 ADCR1 : A/D conversion result register 1 ... 207
 ADCR2 : A/D conversion result register 2 ... 207
 ADCR3 : A/D conversion result register 3 ... 207
 ADCR4 : A/D conversion result register 4 ... 207
 ADCR5 : A/D conversion result register 5 ... 207
 ADCR6 : A/D conversion result register 6 ... 207
 ADCR7 : A/D conversion result register 7 ... 207
 ADCR0H : A/D conversion result register 0H ... 208
 ADCR1H : A/D conversion result register 1H ... 208
 ADCR2H : A/D conversion result register 2H ... 208
 ADCR3H : A/D conversion result register 3H ... 208
 ADCR4H : A/D conversion result register 4H ... 208
 ADCR5H : A/D conversion result register 5H ... 208
 ADCR6H : A/D conversion result register 6H ... 208
 ADCR7H : A/D conversion result register 7H ... 208
 ADIC : Interrupt control register (INTAD) ... 307, 312
 ADM : A/D converter mode register ... 203
 ASIM : Asynchronous serial interface mode register ... 226
 ASIS : Asynchronous serial interface status register ... 226

[B]

BFCM00 : Buffer register CM00 ... 100
 BFCM01 : Buffer register CM01 ... 100
 BFCM02 : Buffer register CM02 ... 100
 BFCM03 : Buffer register CM03 ... 100
 BRG : Baud rate generator compare register ... 236, 256
 BRGC : Baud rate generator control register ... 236, 256

[C]

CC20 : Capture/compare register 20 ... 145
 CC30 : Capture/compare register 30 ... 157
 CCW : CPU control word ... 46
 CM00 : Compare register 00 ... 100
 CM01 : Compare register 01 ... 100
 CM02 : Compare register 02 ... 100
 CM03 : Compare register 03 ... 100
 CM10 : Compare register 10 ... 141
 CM40 : Compare register 40 ... 172
 CM41 : Compare register 41 ... 172
 CMIC03 : Interrupt control register (INTCM03) ... 307, 311
 CMIC10 : Interrupt control register (INTCM10) ... 307, 312

CMIC40 : Interrupt control register (INTCM40) ... 307, 312
 CMIC41 : Interrupt control register (INTCM41) ... 307, 312
 CSIIC : Interrupt control register (INTCSI) ... 307, 312
 CSIM : Clocked serial interface mode register ... 250
 CT20 : Capture register 20 ... 145
 CT30 : Capture register 30 ... 157
 CT31 : Capture register 31 ... 157

[D]

DTIME : Reload register ... 99

[I]

IMC : Interrupt mode control register ... 307, 315
 INTM0 : External interrupt mode register 0 ... 109, 160
 INTM1 : External interrupt mode register 1 ... 148, 160
 ISPR : In-service priority register ... 307, 316

[M]

MBUF0 : Master buffer register 0 ... 101
 MBUF1 : Master buffer register 1 ... 101
 MBUF2 : Master buffer register 2 ... 101
 MBUF3 : Master buffer register 3 ... 101
 MBUF4 : Master buffer register 4 ... 101
 MBUF5 : Master buffer register 5 ... 101
 MK0H : Interrupt mask flag register ... 307, 313
 MK0L : Interrupt mask flag register ... 307, 313
 MM : Memory expansion mode register ... 85, 384

[O]

OVIC3 : Interrupt control register (INTOV3) ... 307, 311

[P]

PIC0 : Interrupt control register (INTP0/INTCC30) ... 307, 311
 PIC1 : Interrupt control register (INTP1) ... 307, 311
 PIC2 : Interrupt control register (INTP2) ... 307, 311
 PIC3 : Interrupt control register (INTP3/INTCC20) ... 307, 311
 PIC4 : Interrupt control register (INTP4) ... 307, 311
 PM0 : Port 0 mode register ... 77
 PM1 : Port 1 mode register ... 77
 PM3 : Port 3 mode register ... 78, 231, 253
 PM5 : Port 5 mode register ... 78
 PM8 : Port 8 mode register ... 78
 PM9 : Port 9 mode register ... 79

PMC0 : Port 0 mode control register ... 82
 PMC3 : Port 3 mode control register ... 83, 230, 252
 PMC8 : Port 8 mode control register ... 84
 PRDC : Port read control register ... 72
 PUOH : Pull-up resistor option register H ... 87
 PUOL : Pull-up resistor option register L ... 87
 PWC : Programmable wait control register ... 386
 PWM0 : PWM buffer register 0 ... 296
 PWM1 : PWM buffer register 1 ... 296
 PWMC0 : PWM control register 0 ... 295
 PWMC1 : PWM control register 1 ... 296

[R]

RTP : Real-time output port register ... 197
 RTPM : Real-time output port mode register ... 196
 RXB : Receive buffer ... 227

[S]

SBIC : Serial bus interface control register ... 250
 SBUF0 : Slave buffer register 0 ... 101
 SBUF1 : Slave buffer register 1 ... 101
 SBUF2 : Slave buffer register 2 ... 101
 SBUF3 : Slave buffer register 3 ... 101
 SBUF4 : Slave buffer register 4 ... 101
 SBUF5 : Slave buffer register 5 ... 101
 SERIC : Interrupt control register (INTSER) ... 307, 312
 SIO : Shift register ... 250
 SMPC0 : Sampling control register 0 ... 149, 161
 SMPC1 : Sampling control register 1 ... 162
 SRIC : Interrupt control register (INTSR) ... 307, 312
 STBC : Standby control register ... 358
 STIC : Interrupt control register (INTST) ... 307, 312

[T]

TM0 : Timer 0 ... 98
 TM1 : Timer 1 ... 141
 TM2 : Timer 2 ... 145
 TM3 : Timer 3 ... 157
 TM4 : Timer 4 ... 172
 TMC0 : Timer control register 0 ... 106
 TMC1 : Timer control register 1 ... 108, 142
 TMC2 : Timer control register 2 ... 146
 TMC3 : Timer control register 3 ... 158
 TMC4 : Timer control register 4 ... 175
 TMIC0 : Interrupt control register (INTTM0) ... 307, 311

TOUT : Timer out register ... 101
TUM0 : Timer unit mode register 0 ... 102
TUM1 : Timer unit mode register 1 ... 173
TXS : Transmit shift register ... 227

[W]

WDM : Watchdog timer mode register ... 300, 365

[MEMO]

APPENDIX D FUNCTION INDEX

μPD78366A operation mode setting ... 11, 23

[A]

Acknowledgement of interrupt

Maskable interrupt ... 322

Non-maskable interrupt ... 318

Op-code trap interrupt ... 335

Software interrupt ... 332

Address wait control ... 387

Addressing

Based addressing ... 56, 57, 58, 59

Based indexed addressing ... 56, 57, 58, 59

Direct addressing ... 56, 57, 58, 59

Implied addressing ... 47, 59

Register addressing ... 47, 56, 57, 58, 59

Register indirect addressing ... 56, 57, 58, 59

Short direct addressing ... 49, 57, 58, 59, 60

Special function register (SFR) addressing ... 49, 56, 57, 58, 59, 60

[B]

Basic operation of A/D converter ... 209

Baud rate setting

Asynchronous serial interface ... 234, 238

Clocked serial interface ... 254, 258

Bus cycle wait control ... 387

[C]

Calculating the number of instruction execution clocks ... 429

Capture operation

Timer 2 (TM2) ... 152

Timer 3 (TM3) ... 165

Clear operation control

Timer 0 (TM0) ... 103

Timer 2 (TM2) ... 147

Timer 3 (TM3) ... 159

Timer 4 (TM4) ... 176
Timer 4 (TM4) (Clear operation by TCLRUD) ... 174

Compare operation

Timer 1 (TM1) ... 143
Timer 2 (TM2) ... 152
Timer 3 (TM3) ... 165
Timer 4 (TM4) ... 172

Context switching

Activates ____ ... 324, 332
Returns from ____ ... 325, 334
Specifies ____ handling ... 309

Controlling and detecting status of serial bus ... 280

[D]

Data access ... 425
Data transfer control from BFCM03 to CM03 ... 106

[E]

External device expansion ... 377, 383

[F]

Free running operation

Timer 2 (TM2) ... 151
Timer 3 (TM3) ... 164
Timer 4 (TM4) ... 181

[H]

HALT mode

Releasing ____ ... 361
Setting ____ ... 360

[I]

Interrupt

Holding the priority level ... 316
Nesting ... 326
Nesting control ... 315
Returning from non-maskable interrupt ... 321
Specifying interrupt request enable/disable ... 310, 313, 314
Specifying priority of maskable interrupt ... 309, 326
Specifying priority of non-maskable interrupt ... 301, 318, 365

Interval operation

- Timer 0 (TM0) ... 111
- Timer 1 (TM1) ... 143
- Timer 2 (TM2) ... 151
- Timer 3 (TM3) ... 164
- Timer 4 (TM4) ... 181

[M]

Macro service

- Counter mode operation specification ... 345
- Mode setting ... 342
- Specification of block transfer mode operation ... 346
- Specification of block transfer mode (with a memory pointer) operation ... 348
- Specification of processing ... 309
- Specifying channel address ... 342
- Terminating operation ... 341

[O]

Operation control

- Baud rate generator (BRG) ... 237, 259
- Counter for PWM output control ... 296
- Dead time timers (DTM0-DTM2) ... 108
- PWM modulation ... 196
- PWM output ... 295
- Timer 0 (TM0) ... 106
- Timer 1 (TM1) ... 142
- Timer 2 (TM2) ... 147
- Timer 3 (TM3) ... 159
- Timer 4 (TM4) ... 176
- Watchdog timer (WDT) ... 301, 365

Operation of sampling circuit

- Timer 2 (TM2) ... 154
- Timer 3 (TM3) ... 169

Op-code fetch ... 422

Output driver off function ... 105

- NMI control ... 103
- Software control ... 103

[P]

Port output data checking ... 72

Processing of unused pins ... 26

PROM

Procedure for reading ... 397

Procedure for writing (Byte program mode) ... 394

Procedure for writing (Page program mode) ... 391

Specification of operation mode for programming ... 25, 390

Specification of programming mode ... 25, 389

PWM output operation

PWM0, PWM1 ... 297

Timer 4 (TM4) ... 192

[R]

Reading of A/D conversion result ... 207, 208

Real-time output ... 195

Reception

Asynchronous serial interface ... 242

Detect error ... 246

SBI mode ... 286

Three-wire serial I/O mode ... 266

Reset

Release ____ ... 373

System reset ... 373

[S]

Selecting A/D conversion time ... 204

Selecting pins for serial communication

Asynchronous serial interface ... 230

Clocked serial interface ... 252

Selecting serial clock

Asynchronous serial interface ... 234

Clocked serial interface ... 254

Selection of analog input ... 204

Setting serial data format ... 232

Specifying active level of output pin

PWM0 and PWM1 pins ... 295

TO00-TO05 pins ... 103

TO40 pin ... 174

Specifying capture trigger

Capture/compare register 20 (CC20) ... 148

Capture register 20 (CT20) ... 148

Specifying control mode of pins

P00-P07 pins ... 82

P05-P07 pins ... 174

P30-P36 pins ... 83, 230, 252

P40-P47, P50-P57 pins ... 80, 85, 385

P80-P85 pins ... 84

P90,P91 pins ... 81

Specifying count clock

PWM output control counter ... 296

Timer 0 (TM0) ... 106

Timer 1 (TM1) ... 142

Timer 2 (TM2) ... 147

Timer 3 (TM3) ... 159

Timer 4 (TM4) ... 176

TMBRG ... 237, 259

Watchdog timer (WDT) ... 301, 365

Specifying counter bits for PWM output control ... 295

Specifying general-purpose register ... 43, 47

Specifying internal memory capacity ... 85, 385

Specifying I/O mode of pin

P00-P07 pins ... 77

P10-P17 pins ... 77

P30-P36 pins ... 78, 231, 253

P50-P57 pins ... 78

P80-P85 pins ... 78

P90-P93 pins ... 79

Specifying operation mode

A/D converter ... 204

Capture/compare register 20 (CC20) ... 147

Capture/compare register 30 (CC30) ... 159

Clocked serial interface ... 263, 273

SBI mode ... 273

Timer 0 (TM0) ... 103

Timer 2 (TM2) ... 147

Timer 3 (TM3) ... 159

Timer 4 (TM4) ... 174

Specifying operation of output pin

TO40 Pin ... 174

Specifying pull-up resistor ... 86

Specifying sampling clock

Timer 2 (TM2) ... 149

Timer 3 (TM3) ... 161, 162

Specifying trigger signal

A/D converter (mixed mode) ... 204

Real-time output port ... 196

Specifying up/down count operation ... 176

Specifying valid edge of external signal

INTP0-INTP2 ... 160

INTP3 ... 148, 160

INTP4 ... 160

NMI ... 109, 160

TCLRUD ... 174

TIUD ... 174

Stack pointer (SP) ... 46

STOP mode

STOP mode releasing ... 364

STOP mode setting ... 363

Switching pins for serial communication

Asynchronous serial interface ... 228

[T]

Timer 0 (TM0) operation

General-purpose interval timer mode (RTP output) ... 111

PWM mode 0 (asymmetric triangular wave modulation) ... 119

PWM mode 0 (symmetric triangular wave modulation) ... 112

PWM mode 0 (toothed wave modulation) ... 129

PWM mode 1 (buffer output) ... 136

Timer 4 (TM4) operation

General-purpose timer mode ... 181

UDC mode (external clock operation) ... 186

UDC mode (internal clock operation) ... 185

Timer basic operation

Timer 0 (TM0) ... 110

Timer 1 (TM1) ... 143

Timer 2 (TM2) ... 150

Timer 3 (TM3) ... 163

Timer 4 (TM4) ... 180

Transmission

Asynchronous serial interface ... 240

SBI mode ... 286

Three-wire serial I/O mode ... 264

Transmission/reception

SBI mode ... 286

Three-wire serial I/O mode ... 268

[V]

Vectored interrupt

Returning from the interrupt service ... 306, 324

Specifying interrupt service ... 309, 310

[W]

Wake-up function ... 290

Write by dedicated instruction

Standby control register (STBC) ... 358

Watchdog timer mode register (WDM) ... 300

Write specification to CM00 to CM02 ... 106

[MEMO]

APPENDIX E REVISION HISTORY

Version	Major Revision from the Previous Version	Applied to:
2nd	The μ PD78363A, 78365A, 78366A, and 78P368A have been developed.	Throughout
	3.2.3 Special function register (SFR) Table 3-4 Special Function Registers Bit manipulation instruction possible has been changed to impossible in the item Manipulatable Bit Unit of P2, P7, ASIS.	CHAPTER 3 CPU ARCHITECTURE
	5.1 Hardware Configuration Table 5-1. Read Operation in Control Mode has been added.	CHAPTER 5 PORT FUNCTION
	Setting preset operation of timer 4 has been disabled because timer 4 may misoperate if underflow of TM4 and rewriting CM40, or timer clear operation by TCLRUD are in contention in the preset operation mode.	CHAPTER 7 REAL-TIME PULSE UNIT
	The formula of baud rate has been changed in 9.5.2 Specific baud rate setting .	CHAPTER 9 ASYNCHRONOUS SERIAL INTERFACE
	9.8 Transmitting/receiving Data Using Macro Service has been added.	
3rd	The formula of baud rate has been changed in 10.3.2 Specific baud rate setting .	CHAPTER 10 Clocked SERIAL INTERFACE
	The μ PD78368A has been added as a target device. The μ PD78P368AKL-S has been developed.	Throught
	8.2 A/D Converter Made Register (ADM) The conversion time by setting the FR bit has been changed.	CHAPTER 8 A/D CONVERTER
	10.6.1 SBI data format A caution has been added regarding the bus release signal and command signal.	CHAPTER 10 CLOCKED SERIAL INTERFACE

[MEMO]

Facsimile Message

From:

Name

Company

Tel.

FAX

Address

Although NEC has taken all possible steps to ensure that the documentation supplied to our customers is complete, bug free and up-to-date, we readily accept that errors may occur. Despite all the care and precautions we've taken, you may encounter problems in the documentation. Please complete this form whenever you'd like to report errors or suggest improvements to us.

Thank you for your kind support.

North America

NEC Electronics Inc.
Corporate Communications Dept.
Fax: 1-800-729-9288
1-408-588-6130

Hong Kong, Philippines, Oceania

NEC Electronics Hong Kong Ltd.
Fax: +852-2886-9022/9044

Asian Nations except Philippines

NEC Electronics Singapore Pte. Ltd.
Fax: +65-250-3583

Europe

NEC Electronics (Europe) GmbH
Technical Documentation Dept.
Fax: +49-211-6503-274

Korea

NEC Electronics Hong Kong Ltd.
Seoul Branch
Fax: 02-528-4411

Japan

NEC Corporation
Semiconductor Solution Engineering Division
Technical Information Support Dept.
Fax: 044-548-7900

South America

NEC do Brasil S.A.
Fax: +55-11-889-1689

Taiwan

NEC Electronics Taiwan Ltd.
Fax: 02-719-5951

I would like to report the following error/make the following suggestion:

Document title: _____

Document number: _____ Page number: _____

If possible, please fax the referenced page or drawing.

Document Rating	Excellent	Good	Acceptable	Poor
Clarity	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Technical Accuracy	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Organization	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>