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.

Preliminary User's Manual

μPD784955 Subseries

16-Bit Single-Chip Microcontrollers

Hardware

μPD784953
μPD784955
μPD78F4956

Document No. U12833EJ2V0UM00 (2nd edition)
Date Published August 1998 N CP(K)

Printed in Japan

[MEMO]

NOTES FOR CMOS DEVICES

① PRECAUTION AGAINST ESD FOR SEMICONDUCTORS

Note:

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

② HANDLING OF UNUSED INPUT PINS FOR CMOS

Note:

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

③ STATUS BEFORE INITIALIZATION OF MOS DEVICES

Note:

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

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

Licence not needed	: μPD78F4956-8BT
The customer must judge the need for licence	: μPD784953-×××-8BT, 784955-×××-8BT

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

The information contained in this document is being issued in advance of the production cycle for the device. The parameters for the device may change before final production or NEC Corporation, at its own discretion, may withdraw the device prior to its production.

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.

M5 96.5

Regional Information

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

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

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

NEC Electronics Inc. (U.S.)
Santa Clara, California
Tel: 408-588-6000
800-366-9782
Fax: 408-588-6130
800-729-9288

NEC Electronics (Germany) GmbH
Duesseldorf, Germany
Tel: 0211-65 03 02
Fax: 0211-65 03 490

NEC Electronics (UK) Ltd.
Milton Keynes, UK
Tel: 01908-691-133
Fax: 01908-670-290

NEC Electronics Italiana s.r.1.
Milano, Italy
Tel: 02-66 75 41
Fax: 02-66 75 42 99

NEC Electronics (Germany) GmbH
Benelux Office
Eindhoven, The Netherlands
Tel: 040-2445845
Fax: 040-2444580

NEC Electronics (France) S.A.
Velizy-Villacoublay, France
Tel: 01-30-67 58 00
Fax: 01-30-67 58 99

NEC Electronics (France) S.A.
Spain Office
Madrid, Spain
Tel: 01-504-2787
Fax: 01-504-2860

NEC Electronics (Germany) GmbH
Scandinavia Office
Taeby, Sweden
Tel: 08-63 80 820
Fax: 08-63 80 388

NEC Electronics Hong Kong Ltd.
Hong Kong
Tel: 2886-9318
Fax: 2886-9022/9044

NEC Electronics Hong Kong Ltd.
Seoul Branch
Seoul, Korea
Tel: 02-528-0303
Fax: 02-528-4411

NEC Electronics Singapore Pte. Ltd.
United Square, Singapore 1130
Tel: 65-253-8311
Fax: 65-250-3583

NEC Electronics Taiwan Ltd.
Taipei, Taiwan
Tel: 02-719-2377
Fax: 02-719-5951

NEC do Brasil S.A.
Cumbica-Guarulhos-SP, Brasil
Tel: 011-6465-6810
Fax: 011-6465-6829

J98. 2

Major Revisions in This Edition (1/2)

Page	Description
INTRODUCTION	Addition and modification of document numbers of related documents
	CHAPTER 1 OVERVIEW
p. 32	Deletion of μPD78F4943 Subseries and addition of μPD784937 Subseries in **78K/IV Series Product Development Diagram**
p. 37	Deletion of the block of external bus in **1.4 Block Diagram**
	CHAPTER 2 PIN FUNCTIONS
p. 46	Modification of **Table 2-2 Operating Modes of Port 1**
p. 47	Modification of **Table 2-3 Operating Modes of Port 2**
p. 49	Modification of **Table 2-4 Operating Modes of Port 3**
	CHAPTER 3 CPU ARCHITECTURE
p. 58	Modification of internal ROM area in **Figure 3-3 μPD78F4956 Memory Map**
p. 65	Deletion of **3.4.3 External SFR area**
	CHAPTER 4 CLOCK GENERATOR
p. 91	Modification of **Figure 4-1 Block Diagram of Clock Generator**
p. 98	Modification of system clock frequency in **4.5 Clock Generator Operations** f_{XX}→f_{CLK}
	CHAPTER 5 PORT FUNCTIONS
p. 112	Modification of description in **(2) Pull-up resistor option registers (PU0 to PU3, PU9, PUO)** in **5.3 Control Registers**
	CHAPTER 6 REAL-TIME OUTPUT FUNCTIONS
p. 116	Modification of **Figure 6-1 Block Diagram of Real-Time Output Port**
p. 119	Modification of the name of external interrupt trigger in **Figure 6-4 Format of Real-Time Output Port Control Register 0 (RTPC0)** INTP3→INTP3TRG
p. 120	Modification of **Figure 6-5 Format of PWM Modulation Control Register 0 (PWMC0)**
p. 127	Modification of the name of external interrupt trigger in **Figure 6-12 Format of Real-Time Output Port Control Register 1 (RTPC1)** INTP5→INTP5TRG
p. 128	Modification of **Figure 6-13 Format of PWM Modulation Control Register 1 (PWMC1)**
p. 135	Modification of **6.6 Cautions**
	CHAPTER 14 8-BIT TIMER/COUNTER 6
p. 236	Addition of Caution 3 in **Figure 14-2 Format of Timer Mode Control Register 6 (TMC6)**
p. 240	Deletion of Caution 2 in **14.4.3 Free running operation of TM6 (PWM output)**
p. 242	Deletion of **(3) TM6 read out during timer operation** in **14.5 Cautions**
	CHAPTER 17 A/D CONVERTER
p. 256	Modification of **Figure 17-1 A/D Converter Block Diagram**
p. 261	Deletion of Caution in **17.4.1 Basic operations of A/D converter**

Major Revisions in This Edition (2/2)

Page	Description
	CHAPTER 19 ASYNCHRONOUS SERIAL INTERFACE
p. 274	Modification of **Figure 19-1 Block Diagram in Asynchronous Serial Interface Mode**
p. 285	Modification of an expression of baud rate
p. 285	Modification of **Table 19-2 Relationship between 5-Bit Counter Source Clock and m Value**
p. 289	Modification of Caution in the case of UART transmission
p. 292	Deletion of the description of selection of external clock in **19.2.3 Standby mode operation**
	CHAPTER 20 3-WIRE SERIAL I/O MODE
p. 294	Modification of **Figure 20-1 Block Diagram of Clocked Serial Interface (3-Wire Serial I/O Mode)**
	CHAPTER 21 EDGE DETECTION FUNCTION
p. 299	Modification of pins with edge detection function P00 to P07→P00
	CHAPTER 22 INTERRUPT FUNCTIONS
p. 342	Addition of reserved word to **Figure 22-20 Macro Service Control Word Format**
	APPENDIX B DEVELOPMENT TOOLS
p. 450	Deletion of **Note** Under development
p. 469	Addition of **APPENDIX E REVISION HISTORY**

The mark ★ shows major revised points.

[MEMO]

INTRODUCTION

Readers

This manual explains the functions of the μPD784955 Subseries to engineers who will design application systems.

Purpose

This manual describes the hardware functions of the μPD784955 Subseries.

Organization

The μPD784955 Subseries user's manual is organized into two volumes, the hardware volume (this manual) and the instruction volume.

Hardware

Pin functions
Internal block functions
Interrupts
Other on-chip peripheral functions

Instructions

CPU functions
Addressing
Instruction set

There are Cautions associated with using this product.
Be sure to read the Cautions in the text of each chapter and summarized at the end of each chapter.

How to read this manual

Reading this manual requires general knowledge about electronics, logic circuits, and microcontrollers.

- **If there are no particular differences in the function**
  In this manual, the μPD784955 is explained as a representative product. If this manual is to be used as the user's manual for the μPD784953 or 78F4956, the μPD784955 should be replaced with the μPD784953 or 78F4956.

- **If there are differences in the function**
  Each product name is presented and described separately.

- **To understand the overall function**
  → Read following the table of contents.

- **To debug when the operation is unusual**
  → Since the cautions are summarized at the end of each chapter, see the cautions associated with the function.

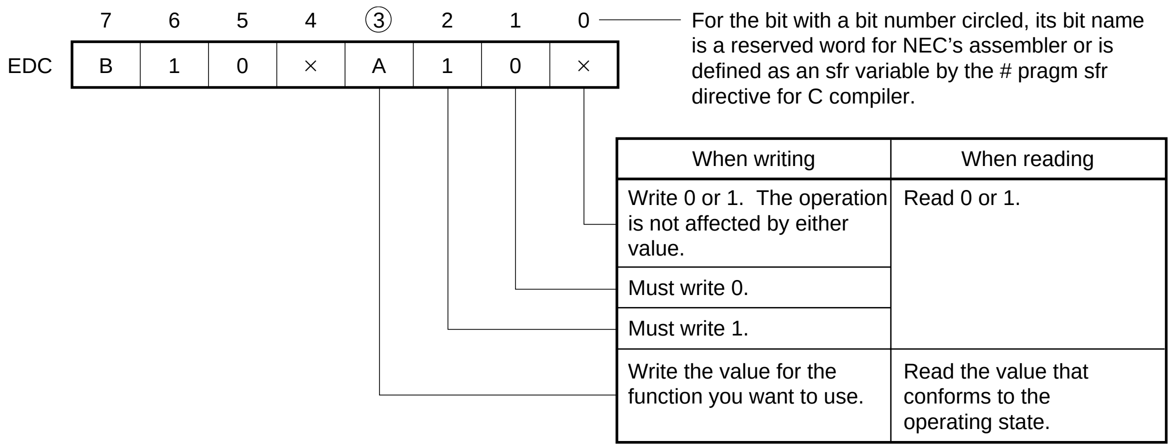
- **Since the cautions are summarized at the end of each chapter, see the cautions associated with the function.**
  → See **APPENDIX D REGISTER INDEX.**

- **For detailed explanations of the instruction functions**
  → Refer to the other manual **78K/IV Series User's Manual – Instructions (U10905E).**

- **For explanations of the application examples of the functions**
  → Refer to the application notes.

Conventions	Data significance	:	Higher digits on the left and lower digits on the right
	Active low representation	:	××× (overscore over pin or signal name)
	Note	:	Footnote for item marked with **Note** in the text
	Caution	:	Information requiring particular attention
	Remark	:	Supplementary information
	Numerical representation	:	Binary ... ×××× or ××××B Decimal ... ×××× Hexadecimal ... ××××H

Register notation



For the bit with a bit number circled, its bit name is a reserved word for NEC's assembler or is defined as an sfr variable by the # pragm sfr directive for C compiler.

When writing	When reading
Write 0 or 1. The operation is not affected by either value.	Read 0 or 1.
Must write 0.	
Must write 1.	
Write the value for the function you want to use.	Read the value that conforms to the operating state.

Never write a combination of codes that have "Setting Prohibited" written in the register description in this manual.

Characters that are confused : 0 (zero), O (capital o)
: 1 (one), l (letter l), I (capital i)

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

Documents related to device

Document Name	Document No.	
	Japanese	English
μPD784953, 784955 Preliminary Product Information	U12830J	U12830E
μPD78F4956 Preliminary Product Information	U12831J	U12831E
μPD784955 Subseries Special Function Register Table	U12832J	To be prepared
μPD784955 Subseries User's Manual, Hardware	U12833J	This document
78K/IV Series Application Note, Software Basics	U10095J	U10095E
78K/IV Series User's Manual, Instructions	U10905J	U10905E
78K/IV Series Instruction Table	U10594J	—
78K/IV Series Instruction Set	U10595J	—

Documents related to development tools (User's Manuals)

Document Name		Document No.	
		Japanese	English
RA78K4 Assembler Package	Operation	U11334J	U11334E
	Language	U11162J	U11162E
RA78K4 Structured Assembler Preprocessor		U11743J	U11743E
CC78K4 C Compiler	Operation	U11572J	U11572E
	Language	U11571J	U11571E
CC78K Series Library Source File		U12322J	U12322E
IE-78K4-NS		U13856J	Under preparation
IE-784000-R		U12903J	EEU-1534
IE-784956-NS-EM1		U13856J	To be prepared
SM78K4 System Simulator, Windows™ Based	Reference	U10093J	U10093E
SM78K Series System Simulator	External parts user open interface specification	U10092J	U10092E
ID78K4-NS Integrated Debugger	Reference	U12796J	U12796E
ID78K4 Integrated Debugger, Windows Based	Reference	U10440J	U10440E
ID78K4 Integrated Debugger HP-UX™, SunOS™, NEWS-OS™ Based	Reference	U11960J	U11960E

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

Documents related to embedded software (User's Manual)

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

Other documents

Document Name	Document No.	
	Japanese	English
IC Package Manual	C10943X	
Semiconductor Device Mounting Technology Manual	C10535J	C10535E
Quality Grades on NEC Semiconductor Devices	C11531J	C11531E
NEC Semiconductor Device Reliability/Quality Control System	C10983J	C10983E
Guide to Prevent Damage for Semiconductor Devices by Electrostatic Discharge (ESD)	C11892J	C11892E
Guide to Quality Assurance for Semiconductor Devices	–	MEI-1202
Microcomputer Product Series Guide	U11416J	–

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

CONTENTS

LIST OF FIGURES (1/7)

LIST OF FIGURES (2/7)

Figure No. Title Page

LIST OF FIGURES (3/7)

LIST OF FIGURES (4/7)

LIST OF FIGURES (5/7)

LIST OF FIGURES (6/7)

LIST OF FIGURES (7/7)

LIST OF TABLES (1/3)

LIST OF TABLES (2/3)

LIST OF TABLES (3/3)

CHAPTER 1 OVERVIEW

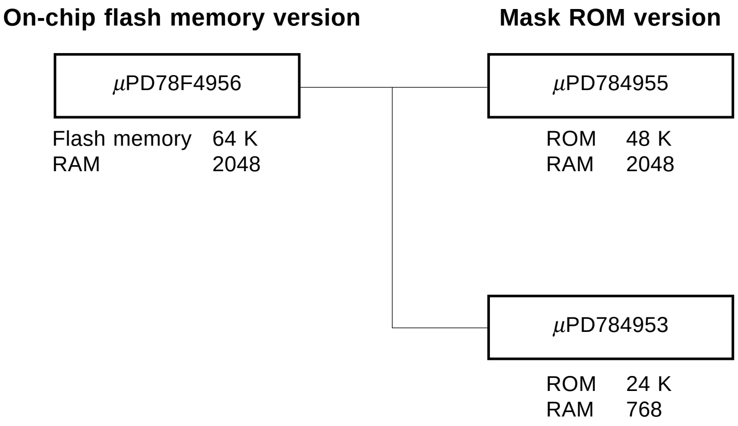
The μPD784955 Subseries is an 80-pin microcontroller of the 78K/IV Series that is suitable for performing specific-purpose inverter control. The 78K/IV Series are 16-bit single-chip microcontroller that are comprised of a high-performance CPU.

The μPD784955 has a 48-Kbyte mask ROM and 2048-byte RAM on chip. Also, it has a high-function timer/counter, 8-bit A/D converter and 2-channel independent serial interface on chip.

The μPD784953 is the same as the μPD784955 only with a 24-Kbyte mask ROM and 768-byte RAM.

The μPD78F4956 replaces the mask ROM of μPD784955 with a 64-Kbyte flash memory.

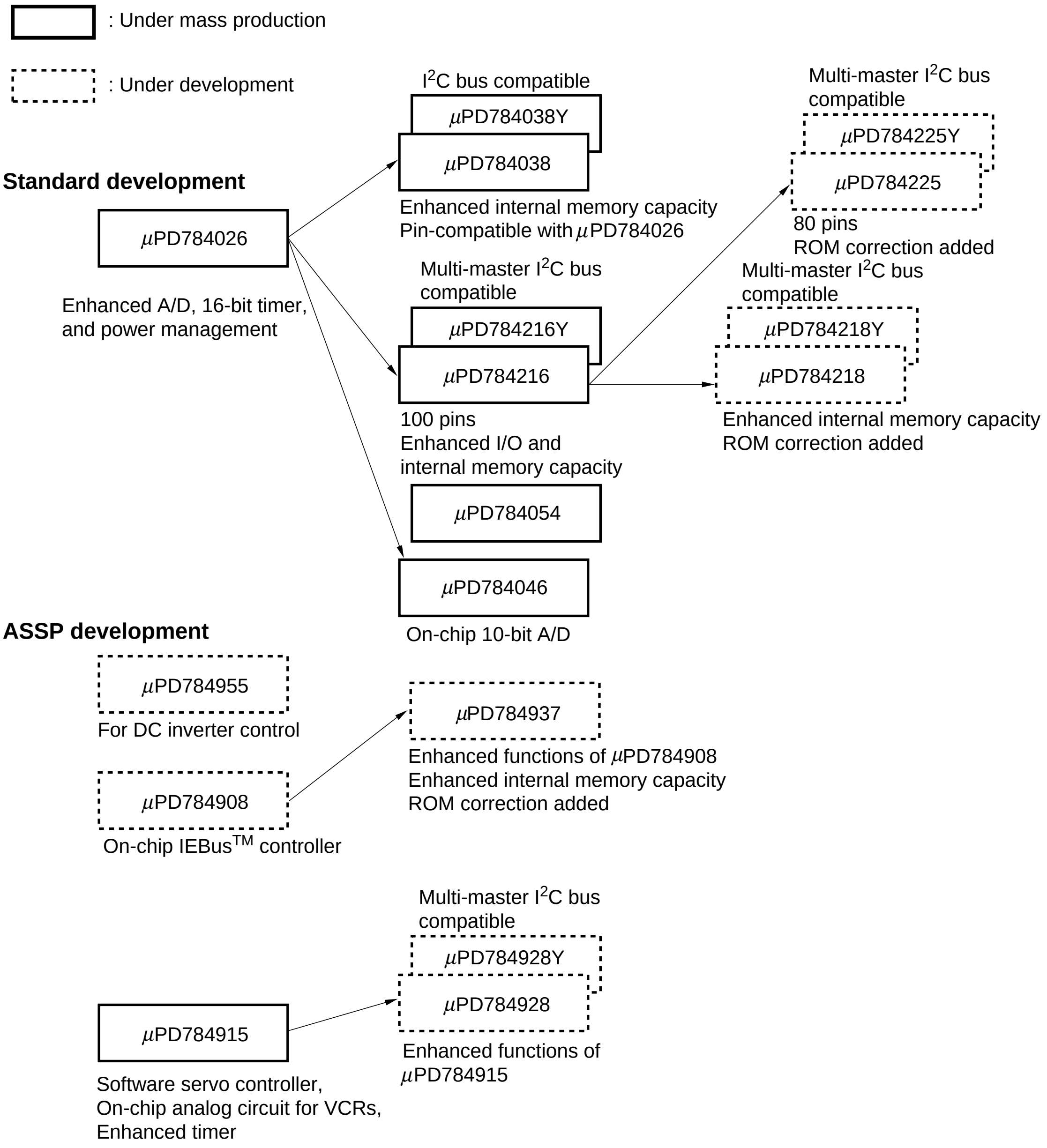
The relationship among the products are shown below.



These products can be applied to the following field:

- Motor control for inverter air-conditioners, etc.

★ 78K/IV Series Product Development Diagram


: Under mass production
: Under development
Standard development
μPD784026
Enhanced A/D, 16-bit timer, and power management
I²C bus compatible
μPD784038Y
μPD784038
Enhanced internal memory capacity
Pin-compatible with μPD784026
Multi-master I²C bus compatible
μPD784216Y
μPD784216
100 pins
Enhanced I/O and internal memory capacity
μPD784054
μPD784046
On-chip 10-bit A/D
Multi-master I²C bus compatible
μPD784225Y
μPD784225
80 pins
ROM correction added
Multi-master I²C bus compatible
μPD784218Y
μPD784218
Enhanced internal memory capacity
ROM correction added
ASSP development
μPD784955
For DC inverter control
μPD784908
On-chip IEBus™ controller
μPD784937
Enhanced functions of μPD784908
Enhanced internal memory capacity
ROM correction added
Multi-master I²C bus compatible
μPD784928Y
μPD784928
Enhanced functions of μPD784915
μPD784915
Software servo controller,
On-chip analog circuit for VCRs,
Enhanced timer

1.1 Features

- 78K/IV Series (16-bit CPU core)
- Minimum instruction execution time: 160 ns (f_{CLK} = 12.5-MHz operation)
- Instruction set suited for control applications
- On-chip memory:

Mask ROM	48 Kbytes (μPD784955)
	24 Kbytes (μPD784953)
Flash memory	64 Kbytes (μPD78F4956)
RAM	2,048 bytes (μPD784955, 78F4956)
	768 bytes (μPD784953)

- I/O ports: 67
  - Software programmable pullup resistors: 59 inputs
  - LED direct drive possible: 32 outputs
- Timer/counter: 16-bit timer/counter × 6 units
  8-bit timer/counter × 2 units
- Watchdog timer: 1 channel
- Serial interfaces: 2 channels
  - UART: 1 channel (on-chip baud rate generator)
  - CSI (3-wire serial I/O): 1 channel
- Real-time output function: 6-bit resolution × 2 channels
- A/D converter: 8-bit resolution × 8 channels
- Interrupt controller (4-level priority)
  - Vectored interrupt/macro service/context switching
- Standby function
  - HALT, STOP, IDLE modes
- Power supply voltage: V_{DD} = 4.5 to 5.5 V

1.2 Ordering Information

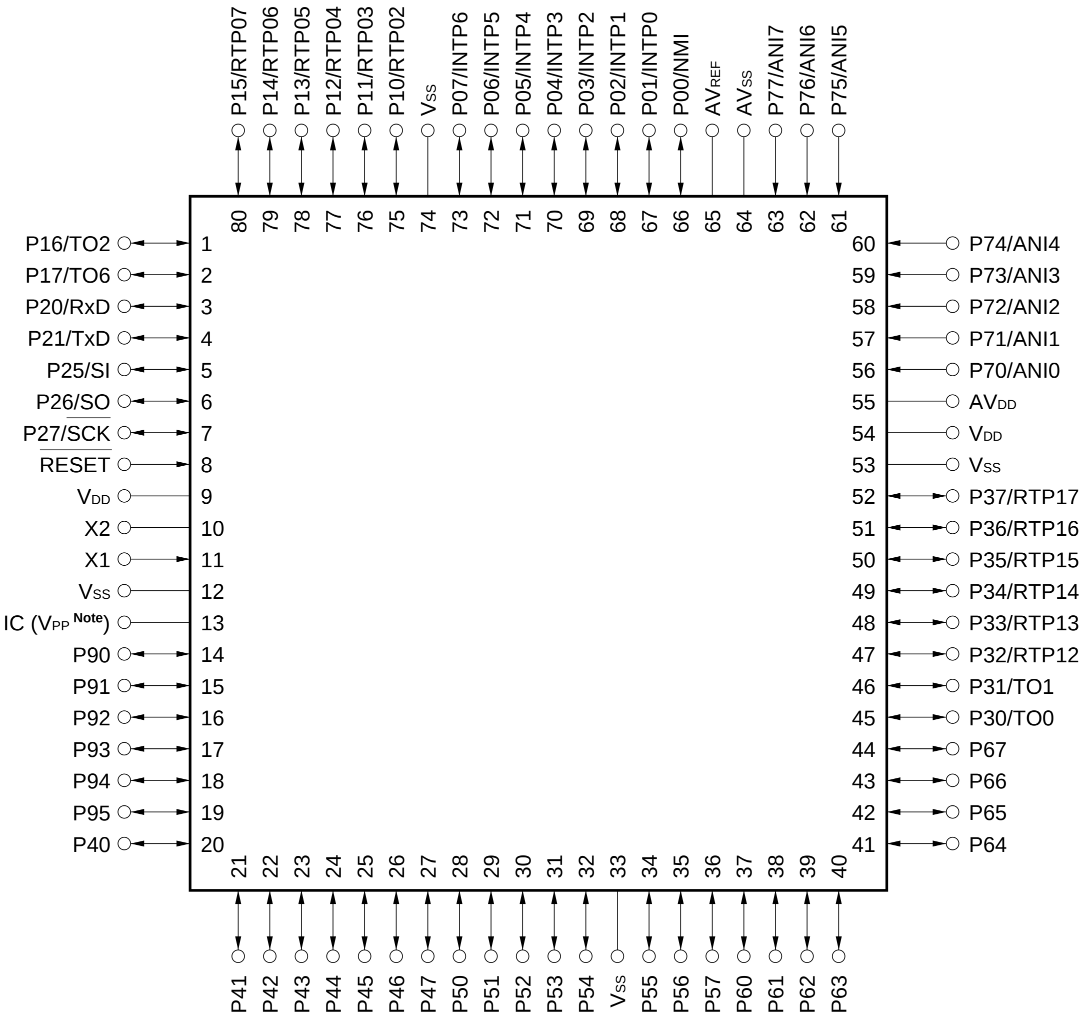
Part Number	Package	Internal ROM
μPD784953GC-××× -8BT	80-pin plastic QFP (14 × 14 mm)	Mask ROM
μPD784955GC-××× -8BT	80-pin plastic QFP (14 × 14 mm)	Mask ROM
μPD78F4956GC-8BT	80-pin plastic QFP (14 × 14 mm)	Flash memory

Remark ××× indicates ROM code suffix.

1.3 Pin Configuration (Top View)

- **80-pin plastic QFP (14 × 14 mm)**

  μPD784953GC-×××-8BT, 784955GC-×××-8BT, 78F4956GC-8BT



Note The V_{PP} pin applies to the μPD78F4956 only.

Cautions 1. Connect the IC (Internally Connected) pin to V_{SS} directly.

2. Connect the AV_{DD} pin to V_{DD}.

3. Connect the AV_{SS} pin to V_{SS}.

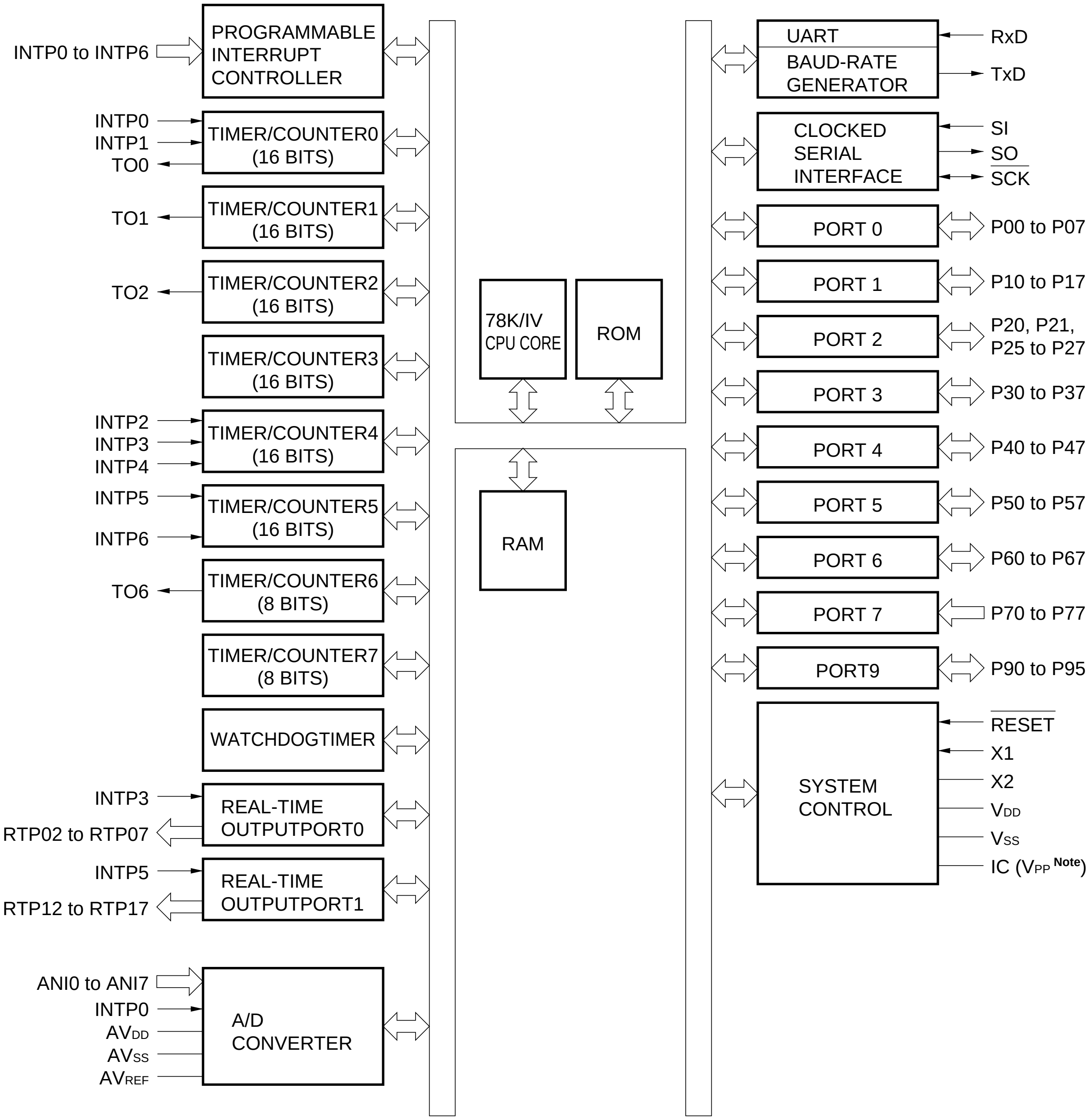
ANI0 to ANI7	: Analog Input
AV$_{DD}$	: Analog Power Supply
AV$_{SS}$	: Analog Ground
AV$_{REF}$	: Analog Reference Voltage
IC	: Internally Connected
INTP0 to INTP6	: Interrupt from Peripherals
NMI	: Non-maskable Interrupt
P00 to P07	: Port 0
P10 to P17	: Port 1
P20, P21, P25 to P27	: Port 2
P30 to P37	: Port 3
P40 to P47	: Port 4
P50 to P57	: Port 5
P60 to P67	: Port 6
P70 to P77	: Port 7
P90 to P95	: Port 9

$\overline{\text{RESET}}$	: Reset
RTP02 to RTP07	: Real-time Port 0
RTP12 to RTP17	: Real-time Port 1
RxD	: Receive Data
$\overline{\text{SCK}}$	: Serial Clock
SI	: Serial Input
SO	: Serial Output
TO0 to TO2, TO6	: Timer Output
TxD	: Transmit Data
V$_{DD}$	: Power Supply
V$_{PP}$**Note**	: Programming Power Supply
V$_{SS}$	: Ground
X1, X2	: Crystal 1, 2

Note The V$_{PP}$ pin applies to the μPD78F4956 only.

★

1.4 Block Diagram



Note The V_{PP} pin applies to the μPD78F4956 only.

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

1.5 Function List

(1/2)

Product Name Item		μPD784953	μPD784955	μPD78F4956
No. of basic instructions (mnemonics)		113		
General registers		8 bits × 16 registers × 8 banks or 16 bits × 8 registers × 8 banks (memory mapping)		
Minimum instruction execution time		• 160 ns (f_{CLK} = 12.5-MHz operation)		
Internal memory	ROM	24 Kbytes (mask ROM)	48 Kbytes (mask ROM)	64 Kbytes (flash memory)
	RAM	768 bytes	2,048 bytes	
I/O ports	Total	67		
	CMOS inputs	8		
	CMOS I/Os	59		
Pins with added functions[Note]	Pins with pull-up resistors	59		
	LED direct drive outputs	32		
Real-time output ports		6 bits × 2		
Timer/counters		16-bit timer/counter 0: Timer register × 1 Capture/compare register × 2 Pulse output possible • PWM output		
		16-bit timer/counter 1: Timer register × 1 Compare register × 2 Pulse output possible • PWM output		
		16-bit timer/counter 2: Timer register × 1 Compare register × 2 Pulse output possible • PWM output		
		16-bit timer/counter 3: Timer register × 1 Compare register × 2		
		16-bit timer/counter 4: Timer register × 1 Capture/compare register × 3		
		16-bit timer/counter 5: Timer register × 1 Compare register × 1 Capture/compare register × 2		
		8-bit timer/counter 6: Timer register × 1 Compare register × 1 Pulse output possible • PWM output		
		8-bit timer/counter 7: Timer register × 1 Compare register × 1		

Note These added functions are valid when these pins are used as I/O pins.

(2/2)

Product Name Item		μPD784953	μPD784955	μPD78F4956
Serial interfaces		• UART: 1 channel (on-chip baud rate generator) • CSI (3-wire serial I/O): 1 channel		
A/D converter		8-bit resolution × 8 channels		
Watchdog timer		1 channel		
Standby function		• HALT, STOP, IDLE modes		
Interrupts	Hardware sources	28 (external: 8 (with internal 2), internal 22)		
	Software sources	BRK instruction, BRKCS instruction, operand error		
	Non-maskable	Internal: 1, external: 1		
	Maskable	Internal: 20, external: 7		
		• 4-level programmable priority • 3 processing mode: Vectored interrupt/macro service/context switching		
Power supply voltage		V_{DD} = 4.5 to 5.5 V		
Package		80-pin plastic QFP (14 × 14 mm)		

1.6 Differences Between μPD784955 Subseries Products

The only difference between the μPD784953 and 784955 lies in the internal memory capacity.

The μPD78F4956 is provided with a 64-Kbyte flash memory instead of the mask ROM of the above products. These differences are summarized in Table 1-1.

Table 1-1. Differences among Products of μPD784955 Subseries

Product Name Item	μPD784953	μPD784955	μPD78F4956
Internal ROM	24 Kbytes (mask ROM)	48 Kbytes (mask ROM)	64 Kbytes (flash memory)
Internal RAM	768 bytes	2,048 bytes	
Internal memory size switching register (IMS)	No		Yes
IC pin	Yes		No
V_{PP} pin	No		Yes
Electrical specifications and recommended soldering conditions	Refer to the respective product data sheet.		

CHAPTER 2 PIN FUNCTIONS

2.1 Pin Function List

(1) Port pins (1/2)

Pin Name	I/O	Alternate Function	Function
P00	I/O	NMI	Port 0 (P0) : • 8-bit I/O port • Can be set in input or output mode bit-wise. • Pins set in input mode can be connected to internal pull-up resistors by software bit-wise.
P01		INTP0	
P02		INTP1	
P03		INTP2	
P04		INTP3	
P05		INTP4	
P06		INTP5	
P07		INTP6	
P10	I/O	RTP02	Port 1 (P1) : • 8-bit I/O port • Can be set in input or output mode bit-wise. • Pins set in input mode can be connected to internal pull-up resistors by software bit-wise. • Direct LED drive capability
P11		RTP03	
P12		RTP04	
P13		RTP05	
P14		RTP06	
P15		RTP07	
P16		TO2	
P17		TO6	
P20	I/O	RxD	Port 2 (P2) : • 5-bit I/O port • Can be set in input or output mode bit-wise. • Pins set in input mode can be connected to internal pull-up resistors by software bit-wise.
P21		TxD	
P25		SI	
P26		SO	
P27		$\overline{\text{SCK}}$	
P30	I/O	TO0	Port 3 (P3) : • 8-bit I/O port • Can be set in input or output mode bit-wise. • Pins set in input mode can be connected to internal pull-up resistors by software bit-wise. • Direct LED drive capability
P31		TO1	
P32		RTP12	
P33		RTP13	
P34		RTP14	
P35		RTP15	
P36		RTP16	
P37		RTP17	
P40 to P47	I/O	—	Port 4 (P4) : • 8-bit I/O port • Can be set in input or output mode bit-wise. • All pins set in input mode can be connected to internal pull-up resistors by software. • Direct LED drive capability

(1) Port pins (2/2)

Pin Name	I/O	Alternate Function	Function
P50 to P57	I/O	—	Port 5 (P5) : • 8-bit I/O port • Can be set in input or output mode bit-wise. • All pins set in input mode can be connected to internal pull-up resistors by software. • Direct LED drive capability
P60 to P67	I/O	—	Port 6 (P6) : • 8-bit I/O port • Can be set in input or output mode bit-wise. • All pins set in input mode can be connected to internal pull-up resistors by software.
P70 to P77	Input	ANI0 to ANI7	Port 7 (P7) : • 8-bit input only port
P90 to P95	I/O	—	Port 9 (P9) : • 6-bit I/O port • Can be set in input or output mode bit-wise. • Pins set in input mode can be connected to internal pull-up resistor by software bit-wise.

(2) Non-port pins

Pin Name	I/O	Alternate Functions	Function
NMI	Input	P00	Non-maskable interrupt request input
INTP0		P01	External interrupt request input
INTP1		P02	
INTP2		P03	
INTP3		P04	
INTP4		P05	
INTP5		P06	
INTP6		P07	
RTP02 to RTP07	Output	P10 to P15	Real-time output port that outputs data synchronized to the trigger
TO2		P16	16-bit timer output (can also be used as 16-bit PWM output)
TO6		P17	8-bit timer output (can also be used as 8-bit PWM output)
RxD	Input	P20	Serial data input (UART)
TxD	Output	P21	Serial data output (UART)
SI	Input	P25	Serial data input (3-wire serial I/O)
SO	Output	P26	Serial data output (3-wire serial I/O)
$\overline{\text{SCK}}$	I/O	P27	Serial clock input/output (3-wire serial I/O)
TO0	Output	P30	16-bit timer output (can also be used as 16-bit PWM output)
TO1		P31	
RTP12 to RTP17		P32 to P37	Real-time output port that outputs data in synchronization with trigger
ANI0 to ANI7	Input	P70 to P77	Analog voltage input for A/D converter
$\overline{\text{RESET}}$		—	System reset input
X1			To connect system clock oscillation crystal
X2	—		
AV_{REF}			Reference voltage applied to D/A converter
AV_{DD}			Positive power supply to A/D converter. Connect to V_{DD}.
AV_{SS}			Ground for A/D converter. Connect to V_{SS}.
V_{DD}			Positive power supply
V_{SS}			GND potential
IC[Note 1]			Directly connect to V_{SS} (IC test pin).
V_{PP}[Note 2]			Flash memory programming mode setting High-voltage application pin during program write/verify

Notes 1. Only products with mask ROM have an IC pin.

2. Only μPD78F4956 has a V_{PP} pin.

2.2 Explanation of Pin Functions

2.2.1 Normal operating mode

(1) P00 to P07 (Port 0) .... 3-state I/O

Port 0 is an 8-bit I/O port with output latch. With the port mode register (PM0) it is possible to specify input or output in 1-bit units. Each pin contains a software programmable pull-up resistor. Besides operating as an I/O pin, it also operates as a control signal input pin such as an external interrupt signal pin. (See **Table 2-1. Operating Modes of Port 0**.) There is also Schmitt-trigger input to prevent malfunction of all 8 pins due to noise. It becomes an input port (outputs high-impedance state) by $\overline{\text{RESET}}$ input, and the contents of the output latch become undefined.

Table 2-1. Operating Modes of Port 0

Pin Name	Function
P00	Input port / NMI input [Note]
P01	Input port / INTP0 input / CR01 capture trigger input / Count clock of 16-bit timer/counter 0
P02	Input port / INTP1 input / CR00 capture trigger input / Count clock of 16-bit timer/counter 0
P03	Input port / INTP2 input / CR41 capture trigger input
P04	Input port / INTP3 input / CR40 capture trigger input / Count clock of 16-bit timer/counter 4 / Trigger signal of real-time output port
P05	Input port / INTP4 input / CR42 capture trigger input
P06	Input port / INTP5 input / CR51 capture trigger input / Count clock of 16-bit timer/counter 5 / Trigger signal of real-time output port
P07	Input port / INTP6 input / CR50 capture trigger input / Count clock of 16-bit timer/counter 5

Note NMI input is received regardless of interrupt enabled/disabled state.

(a) Function as a port pin

Regardless of the input mode/output mode specification, it is possible to connect a pull-up resistor in 1-bit units using pull-up resistor option register 0 (PU0). Even if its alternate function is used, it can always be used to read or test the pin level.

(b) Function as a control-signal input pin

(i) NMI (Non-maskable interrupt)

This is an external non-maskable interrupt request input pin. It is possible to specify rising-edge detection or falling-edge detection by using the external interrupt rising edge enable register (EGP0) or external interrupt falling edge enable register (EGN0).

(ii) INTP0 to INTP6 (Interrupt from Peripherals)

This is an external interrupt request input pin. An interrupt occurs when the valid edge specified by prescaler-mode registers 0, 4 and 5 (PRM0, 4, 5) is detected at pins INTP0 to INTP6. (Refer to **CHAPTER 8 16-BIT TIMER/COUNTER 0, CHAPTER 12 16-BIT TIMER/COUNTER 4,** and **CHAPTER 13 16-BIT TIMER/COUNTER 5.**)

Also, INTP0 to INTP6 are used as external trigger input pins for various functions.

- INTP0 .... Capture trigger input pin for 16-bit timer/counter 0,
  External count clock input pin for 16-bit timer/counter 0,
- INTP1 .... Capture trigger input pin for 16-bit timer/counter 0
  External count clock input pin for 16-bit timer/counter 0,
- INTP2 .... Capture trigger input pin for 16-bit timer/counter 4,
  External count clock input pin for 16-bit timer/counter 4
- INTP3 .... Capture trigger input pin for 16-bit timer/counter 4,
  External count clock input pin for 16-bit timer/counter 4,
  Trigger input pin for real-time output port
- INTP4 .... Capture trigger input pin for 16-bit timer/counter 4,
  External count clock input pin for 16-bit timer/counter 4
- INTP5 .... Capture trigger input pin for 16-bit timer/counter 5,
  External count clock input pin for 16-bit timer/counter 5,
  Trigger input pin for real-time output port
- INTP6 .... Capture trigger input pin for 16-bit timer/counter 5,
  External count clock input pin for 16-bit timer/counter 5

(2) P10 to P17 (Port 1) .... 3-state I/O

Port 1 is an 8-bit I/O port with output latch. It is possible to specify input or output in 1-bit units using the port 1 mode register (PM1). Regardless of whether the input mode or output mode is specified, it is possible to connect a pull-up resistor in 1-bit units by using the pull-up resistor option register (PU1).

As a 6-bit real-time output port, P10 to P15 can output the contents of real-time output buffer register 0 (RTBL0, RTBH0) at an arbitrary interval time. Selection as a normal output port or as a real-time output port is done using real-time output port control register 0 (RTPC0). Pins P16 and P17 can also function as timer-output pins.

It is also possible to drive the LED directly.

Port 1 becomes an input port (outputs high-impedance state) at $\overline{\text{RESET}}$ input, and the contents of the output latch become undefined.

★ **Table 2-2. Operating Modes of Port 1**

(n = 0 to 7, m = 2 to 7)

Pin Name	PM1n = 1	PM1n = 0			
		RTPM0m = 1	TOE2 = 1	TOE6 = 1	Other than on left
P10	I	RTP02	O	O	O
P11	I	RTP03	O	O	O
P12	I	RTP04	O	O	O
P13	I	RTP05	O	O	O
P14	I	RTP06	O	O	O
P15	I	RTP07	O	O	O
P16	I	O	TO2 Output	O	O
P17	I	O	O	TO6 Output	O

Cautions 1. RTPM0m is bits 2 to 7 of the real-time port mode register (RTPM0n).
TOE2 is bit 0 of the timer output control register 2 (TOC2).
TOE6 is bit 0 of the timer mode control register 6 (TMC6).

2. When using port 1 as an alternate function pin (PM1n = 0), set P1n (output latch) to “0”.

3. When using P10 to P15 as output ports, set bit 7 (RTPOE0) of the real-time output port control register 0 (RTPC0) to “0” (real-time output operation disable), and set bits 1 to 7 of the PWM modulation control register 0 (PWMC0) to “0” (PWM modulation operation disable and real-time output level inversion disable).

4. When using P16 and P17 as output ports, set TOE2 and TOE6 to “0” (timer output disable).

Remark I: input port, O: output port

(3) P20, P21, P25 to P27 (Port 2) ...... 3-state I/O

Port 2 is a 5-bit I/O port with latch. It is possible to specify input or output in 1-bit units using the port 2 mode register (PM2). Regardless of whether input or output is specified, it is possible to connect a pull-up resistor in 1-bit units by using pull-up resistor option register 2 (PU2).

Besides functioning as an I/O port, it can also function as a control signal pin. Also, the operating mode can be specified in bit units as shown in Table 2-3 using the asynchronous serial interface mode register 1 (ASIM1) or the serial operating mode register 0 (CSIM0).

Even if its alternate function is used, it can always be used to read or test the pin level.

Port 2 becomes an input port (outputs high-impedance state) at $\overline{\text{RESET}}$ input, and the contents of the output latch become undefined.

★ **Table 2-3. Operating Modes of Port 2**

(n = 0 to 7, m = 2 to 7)

Pin Name	PM2n = 1			PM2n = 0		
	RXE = 1	CSIE0 = 1, MODE0 = 0/1	Other than on left	TXE = 1	CSIE0 = 1, MODE0 = 0	Other than on left
P20	RXD	I	I	O	O	O
P21	I	I	I	O	O	O
P25	I	I	I	O	O	O
P26	I	I	I	O	O	O
P27	I	$\overline{\text{SCK}}$[Note]	I	O	$\overline{\text{SCK}}$[Note]	O

Note The $\overline{\text{SCK}}$ pin functions with specification of bit 0 and bit 1 of CSIM0 (selection of clock), independently of the specification of bit 2 (serial transfer operating mode) of the serial operating mode register 0 (CSIM0). To use the $\overline{\text{SCK}}$ pin as external clock (SCL00, SCL01 = 00B), set CSIE0 = 1 and PM27 = 1 (input mode). To use the $\overline{\text{SCK}}$ pin as internal clock (SCL00, SCL01 = except 00B), set CSIE0 = 1 and PM27 = 0 (output mode).

Cautions 1. RXE1 and TXE1 are bits 6 and 7 of the asynchronous serial interface mode register 1 (ASIM1). SCL00, SCL01, MODE0 and CSIE0 are bits 0, 1, 2 and 7 of the serial operating mode register 0 (CSIM0).

2. When used as alternate function pin (only output) (PM2n = 0), set P2n (output latch) to “0”.

3. When using P20, P21 and P25 to 27 as output ports, set RXE1 and TXE1 to “0” (UART operation disable), and set bits 6 and 7 of CSIM0 to “0” (SIO operation disable).

Remark I: input port, O: output port

(a) Port mode

For ports, input or output can be specified in 1-bit units by PM2.

(b) Serial transfer operation mode

It is possible to set the pins as control pins in 1-bit units by setting PM2, ASIM1 and CSIM1.

(i) RxD (Receive Data)

RxD is a serial data input pin for the asynchronous serial interface.

(ii) TxD (Transmit Data)

TxD is a serial data output pin for the asynchronous serial interface.

(iii) SI (Serial Input)

SI is a serial data input pin (for 3-wire serial I/O mode).

(iv) SO (Serial Output)

SO is a serial data output pin (for 3-wire serial I/O mode).

(v) $\overline{\text{SCK}}$ (Serial Clock)

$\overline{\text{SCK}}$ is a serial clock I/O pin (for 3-wire serial I/O mode).

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

Port 3 is an 8-bit I/O port with output latch. It is possible to specify input or output in 1-bit units by using the port 3 mode register (PM3). Regardless of whether input or output is specified, it is possible to connect a pull-up resistor in 1-bit units by using pull-up resistor option register 3 (PU3). Each pin incorporates software-programmable pull-up resistor.

As a real-time output port, P32 to P37 can output the contents of real-time output buffer register 1 (RTBL1, RTBH1) in arbitrary time intervals. Selection as a normal output port or as real-time output is done by using the real-time output port control register 1 (RTPC1). Also, P30 and P31 can function as timer output pins.

Also, the LED can be driven directly.

Port 3 becomes an input port (outputs high-impedance state) at $\overline{\text{RESET}}$ input, and the contents of the output latch become undefined.

★ **Table 2-4. Operating Modes of Port 3**

(n = 0 to 7, m = 2 to 7)

Pin Name	PM3n = 1	PM3n = 0			
		TOE0 = 1	TOE1 = 1	RTPM1m = 1	Other than on left
P30	I	TO0 Output	O	O	O
P31	I	O	TO1 Output	O	O
P32	I	O	O	RTP12	O
P33	I	O	O	RTP13	O
P34	I	O	O	RTP14	O
P35	I	O	O	RTP15	O
P36	I	O	O	RTP16	O
P37	I	O	O	RTP17	O

Cautions 1. TOE0 is bit 0 of the timer output control register 0 (TOC0).
TOE1 is bit 0 of the timer output control register 1 (TOC1).
RTPM1m is bits 2 to 7 of the real-time output port mode register 1 (RTPM1).

2. When using as alternate function pin (PM3n = 0), set P3n (output latch) to “0”.

3. When using P30 and P31 as output port, set TOE0 and TOE1 to “0” (disable timer output).

4. When using P32 to P37 as output port, set bit 7 (RTPOE1) of the real-time output port control register 1 (RTPC1) to “0” (real-time output operation disable), and set bits 1 to 7 of PWM modulation control register 1 (PWMC1) to “0” (PWM modulation operation disable and real-time output level inversion disable).

Remark I: input port, O: output port

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

Port 4 is an 8-bit I/O port with output latch. It is possible to specify input or output in 1-bit units using the port 4 mode register (PM4). It is possible to connect a pull-up resistor in 8-bit units only in the input mode by using the pull-up resistor option register (PUO).

Also, the LED can be driven directly.

Port 4 becomes an input port (outputs high-impedance state) at $\overline{\text{RESET}}$ input, and the contents of the output latch become undefined.

(6) P50 to P55 (Port 5) ......3-state I/O

Port 5 is an 8-bit I/O port with output latch. It is possible to specify input or output in 1-bit units using the port 5 mode register (PM5). It is possible to connect a pull-up resistor in 8-bit units only in the input mode by using the pull-up resistor option register (PUO).

Also, the LED can be driven directly.

Port 5 becomes an input port (outputs high-impedance state) at $\overline{\text{RESET}}$ input, and the contents of the output latch become undefined.

(7) P60 to P67 (Port 6) ......3-state I/O

Port 6 is an 8-bit I/O port with output latch. It is possible to specify input or output in 1-bit units using the port 4 mode register (PM6). It is possible to connect a pull-up resistor in 8-bit units only in the input mode by using the pull-up resistor option register (PUO).

Also, the LED can be driven directly.

Port 6 becomes an input port (outputs high-impedance state) at $\overline{\text{RESET}}$ input, and the contents of the output latch become undefined.

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

Port 7 is an 8-bit input-only pin. Besides operating as an input port, these pins can also operate as analog-input pins (ANI0 to ANI7) for the A/D converter. Regardless of dual-pin operation, reading or testing the pin level is always possible. These pins do not incorporate pull-up resistors.

(a) Port mode

Functions as an 8-bit input-only pin.

(b) Control mode

Operates as analog-input pins (ANI0 to ANI7) for the A/D converter. The values read by the pins specified for analog input are undefined.

(9) P90 to P95 (Port 9) ......3-state I/O

Port 9 is a 6-bit I/O port with output latch. It is possible to specify input or output in 1-bit units using the port 9 mode register (PM9). Regardless of whether input or output is specified, it is possible to connect a pull-up resistor in 1-bit units by using the pull-up resistor option register (PUO).

Each pin incorporates a software-programmable pull-up resistor.

Port 9 becomes an input port (outputs high-impedance state) at $\overline{\text{RESET}}$ input, and the contents of the output latch become undefined.

(10) AV_{REF}

This is a reference-voltage-input pin for the A/D converter.

(11) AV_{DD}

This is the analog power-supply pin for the A/D converter. Even when the A/D converter is not used, apply the same potential as the V_{DD} pin.

(12) AV_{SS}

This pin is the ground-potential pin for the A/D converter. Even when the A/D converter is not used, apply the same potential as the Vss pin.

(13) $\overline{\text{RESET}}$

This is the active-low system reset input pin.

(14) X1, X2

These pins are connected to the crystal resonator for generating the system clock.

When an external clock is supplied, it should be input to the pin X1, and the inverted signal should be input to pin X2.

(15) V_{DD}

V_{DD} is the positive power-supply pin.

(16) V_{SS}

V_{SS} is the ground-potential pin.

(17) V_{PP} (μPD78F4956 only)

This is the high-voltage application pin for flash-memory programming mode settings and program write/verify.

(18) IC (mask ROM version only)

This pin is for testing the IC. Connect directly to V_{SS}.

2.3 Pin I/O Circuit and Recommended Connections of Unused Pins

Table 2-5 shows the pin I/O circuit types and the recommended connections of unused pins.
See Figure 2-1 for each type of I/O circuit.

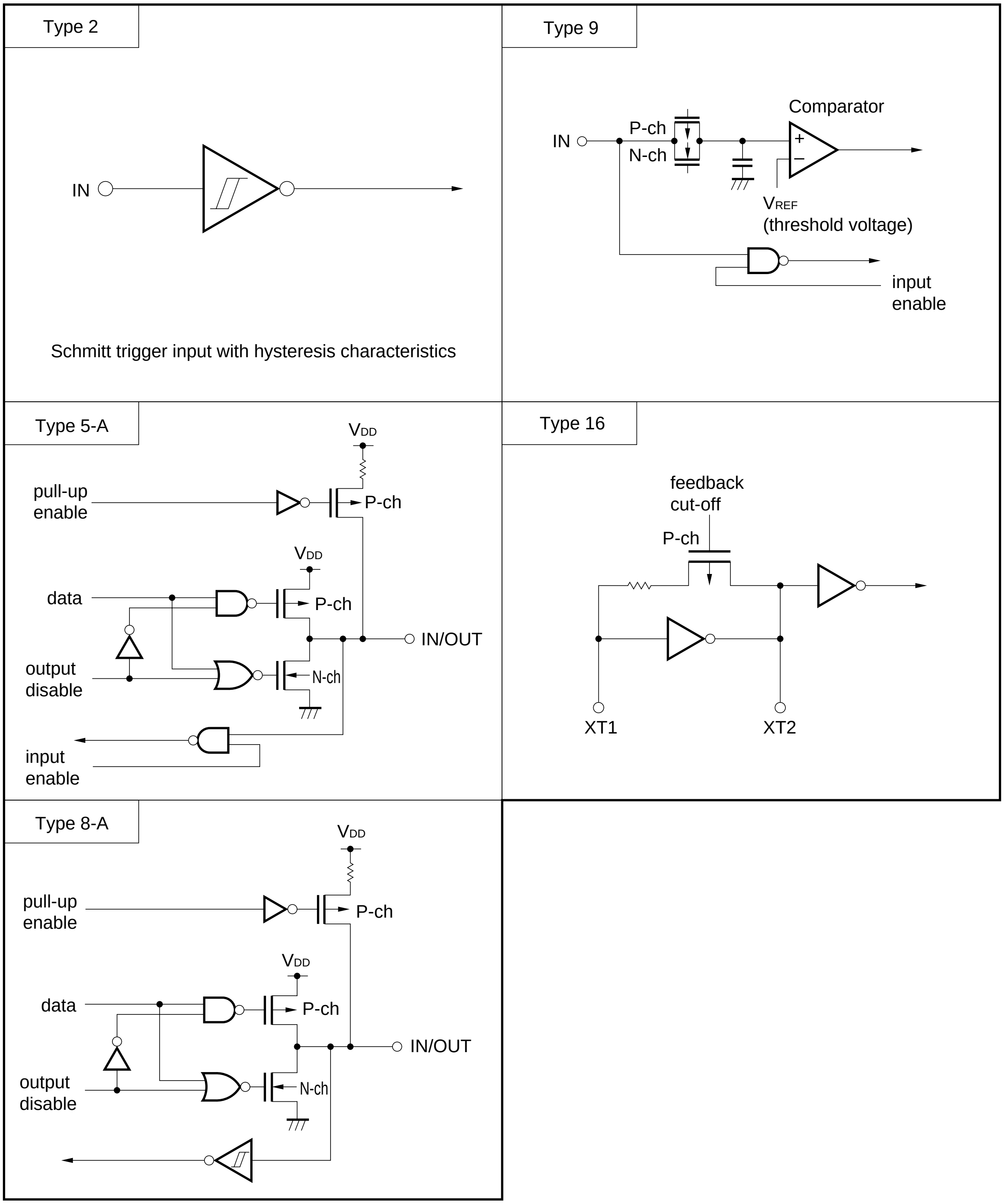
Table 2-5. I/O Circuit Type for Each Pin and Recommended Connections of Unused Pins

Pin Name	I/O Circuit Type	I/O	Recommended Connection When Unused
P00/NMI	8-A	I/O	When input: Independently connect to V_{SS} or V_{DD} via a resistor. When output: Leave open.
P01/INTP0			
P02/INTP1			
P03/INTP2			
P04/INTP3			
P05/INTP4			
P06/INTP5			
P07/INTP6			
P10/RTP02 to P15/RTP07	5-A		
P16/TO2			
P17/TO6			
P20/RxD	8-A		
P21/TxD	5-A		
P25/SI	8-A		
P26/SO	5-A		
P27/$\overline{SCK}$	8-A		
P30/TO0	5-A		
P31/TO1			
P32/RTP12 to P36/RTP17			
P40 to P47			
P50 to P57			
P60 to P67			
P70/ANI0 to P77/ANI7	9	Input	Connect to V_{SS} or V_{DD}.
P90 to P95	5-A	I/O	During input: Independently connect to V_{SS} or V_{DD} via a resistor. During output: Leave open.
X1	16	Input	Connect to V_{SS}.
X2		—	Leave open.
$\overline{RESET}$	2	Input	—
AV_{DD}	—	—	Connect to V_{DD}.
AV_{REF}			Connect to V_{SS}.
AV_{SS}			
IC			Connect to V_{SS} directly.
V_{PP} Note			

Note The V_{PP} pin applies to the μPD78F4956 only.

Remark A unified numbering system for I/O circuit type is used for the entire 78K Series. Since, for a given product, only some of these circuit types are available on-chip, the ones listed may not be numbered sequentially.

Figure 2-1. Pin I/O Circuit



[MEMO]

CHAPTER 3 CPU ARCHITECTURE

3.1 Memory Space

The μPD784955 is provided with the function that selects mapping of the internal data area (containing special-function registers and internal RAM) by using the LOCATION instruction. The LOCATION instruction must be executed after reset is cleared, and cannot be used more than once.

The program after reset must be as shown below.

```
RETVCT     CSEG     AT 0
           DW       RSTSTRT

             to

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

(1) When the LOCATION 0 instruction is executed

- **Internal memory**

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

Product Name	Internal Data Area	Internal ROM Area
μPD784953	0FC00H to 0FFFFH	00000H to 05FFFH
μPD784955	0F700H to 0FFFFH	00000H to 0BFFFH
μPD78F4956	0F700H to 0FFFFH	00000H to 0F6FFH

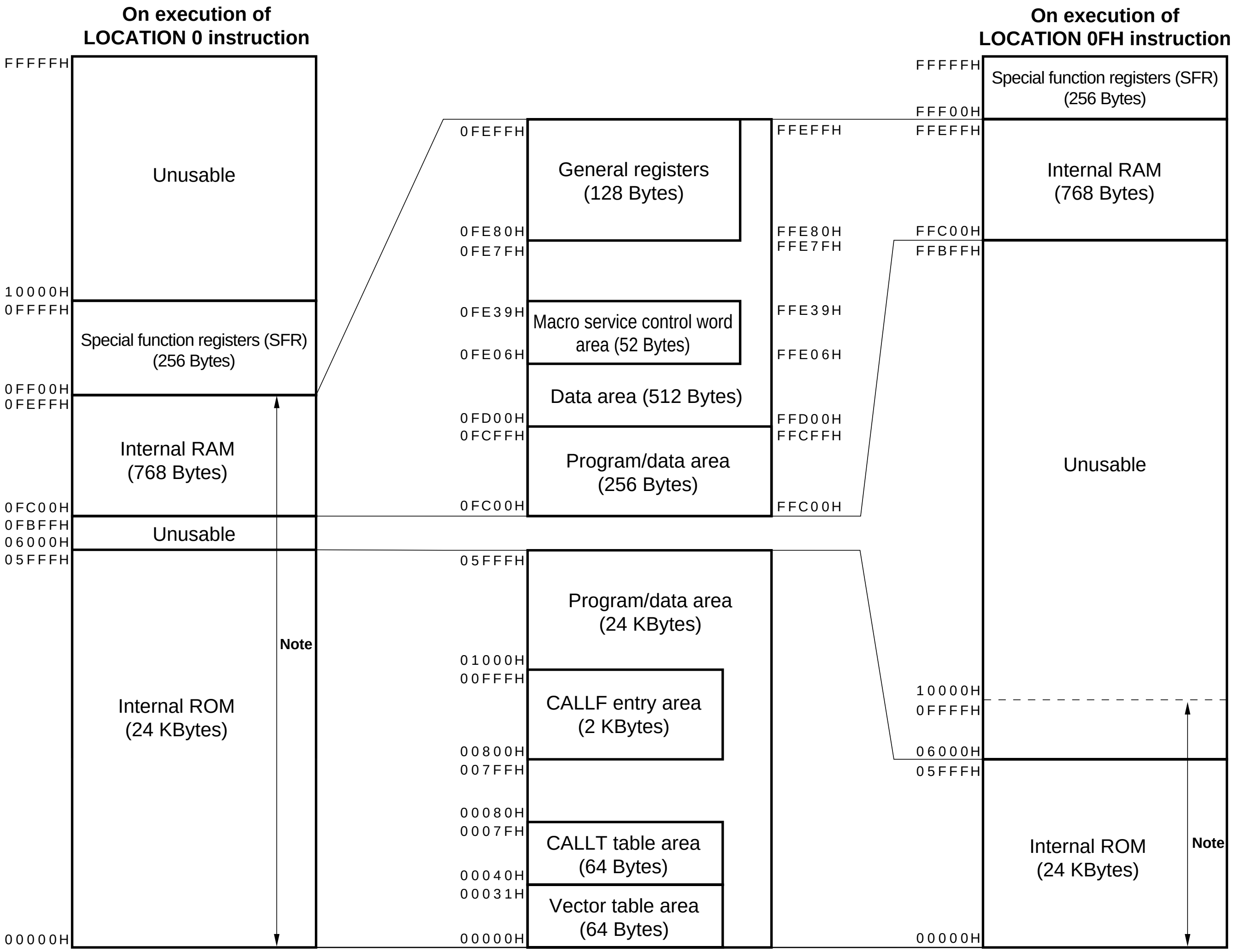
(2) When the LOCATION 0FH instruction is executed

- **Internal memory**

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

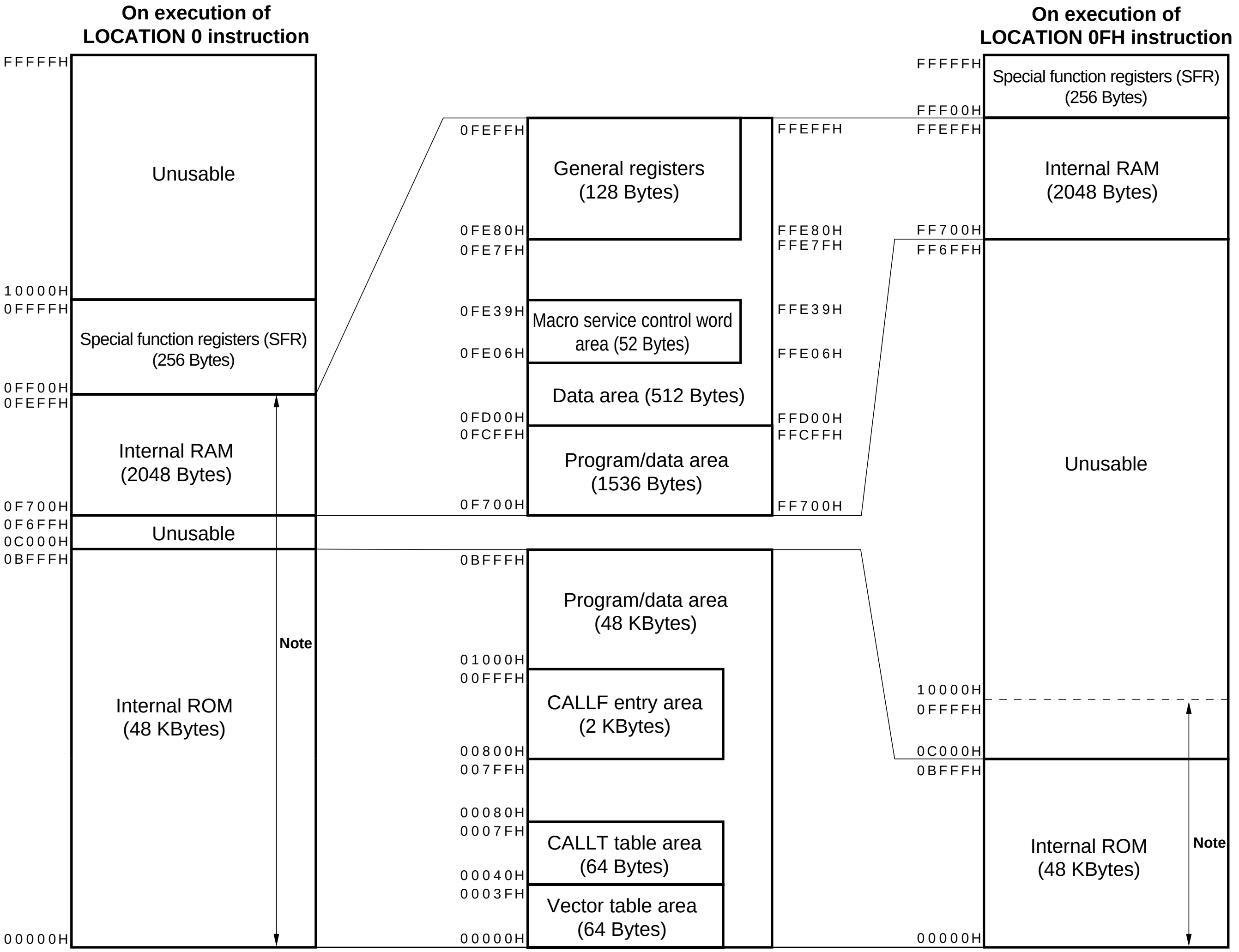
Product Name	Internal Data Area	Internal ROM Area
μPD784953	FFC00H to FFFFFH	00000H to 05FFFH
μPD784955	FF700H to FFFFFH	00000H to 0BFFFH
★ μPD78F4956	FF700H to FFFFFH	00000H to 0FFFFH

Figure 3-1. μPD784953 Memory Map



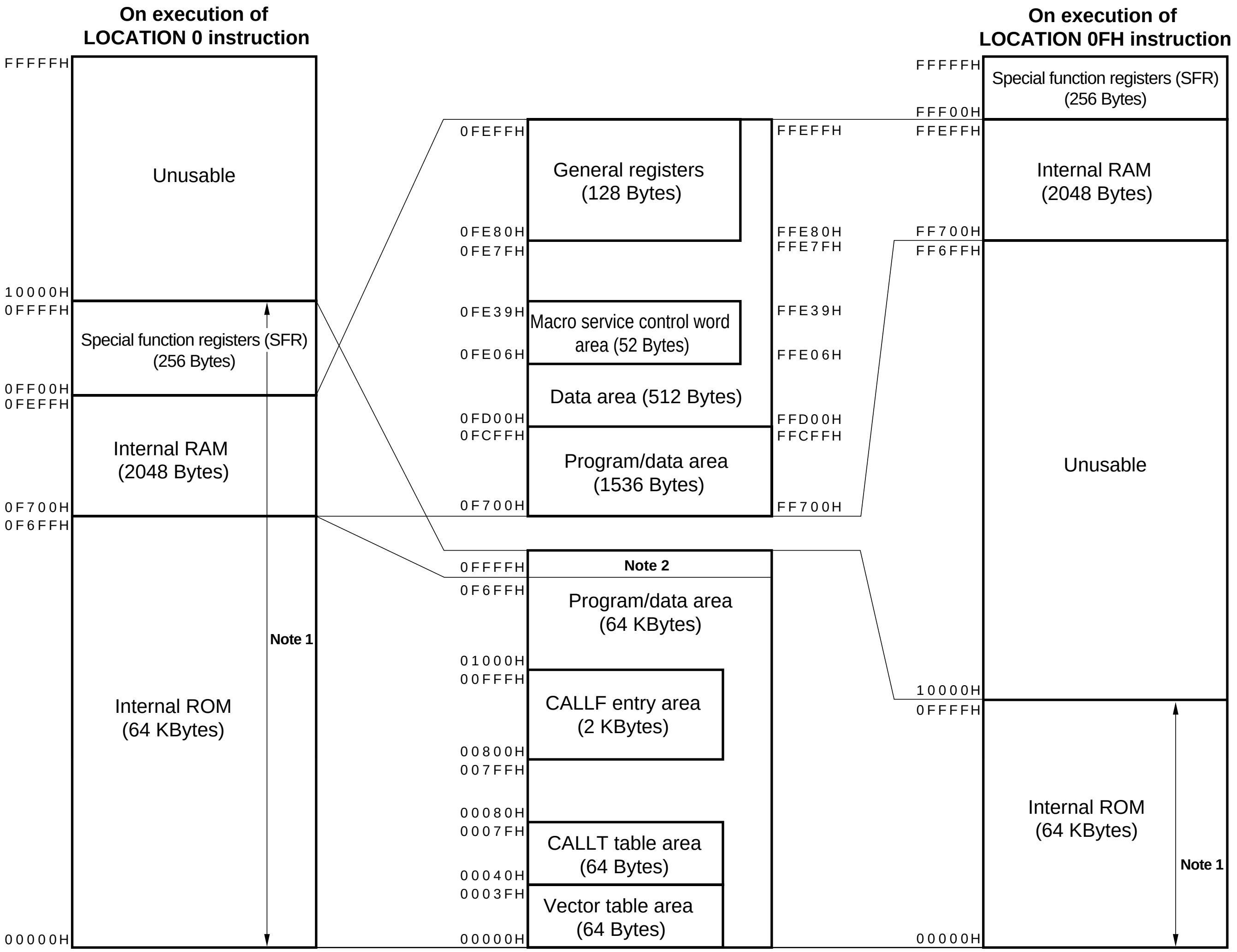
Note Base area and entry area for reset or interrupt. However, the internal RAM area is not used as a reset entry area.

Figure 3-2. μPD784955 Memory Map



Note Base area and entry area for reset or interrupt. However, the internal RAM area is not used as a reset entry area.

★ **Figure 3-3. μPD78F4956 Memory Map**



Notes 1. Base area and entry area for reset or interrupt. However, the internal RAM area is not used as a reset entry area.

2. 2304 Bytes in this area can be used as internal ROM only when executing the LOCATION 0FH instruction.

3.2 Internal ROM Area

μPD784955 has on-chip ROM that can store programs and table data.

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

Part Number	Internal ROM	Address Space	
		LOCATION 0 Instruction	LOCATION 0FH Instruction
μPD784953	24 K × 8 bits	00000H to 05FFFH	00000H to 05FFFH
μPD784955	48 K × 8 bits	00000H to 0BFFFH	00000H to 0BFFFH
★ μPD78F4956	64 K × 8 bits	00000H to 0F6FFH	00000H to 0FFFFH

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

3.3 Base Area

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

- Entry address for reset
- Entry address for interrupt
- Entry address for CALLT instruction
- 16-bit immediate addressing mode (instruction address addressing)
- 16-bit direct addressing mode
- 16-bit register addressing mode (instruction address addressing)
- 16-bit register indirect addressing mode
- Short direct 16-bit memory indirect addressing mode

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

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

3.3.1 Vector table area

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

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

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

Table 3-1. Vector Table Address

Interrupt Source	Vector Table Address	Interrupt Source	Vector Table Address
BRK instruction	003EH	INTTM20	001CH
Operand error	003CH	INTTM21	001EH
$\overline{\text{RESET}}$ (reset input)	0000H	INTTM30	0020H
NMI	0002H	INTTM31	0022H
INTWDT	0004H	INTTM40	0024H
INTP0	0006H	INTTM42	0026H
INTP1	0008H	INTTM50	0028H
INTP2/INTTM41	000AH	INTTM52	002AH
INTP3	000CH	INTTM6	002CH
INTP4	000EH	INTTM7	002EH
INTP5/INTTM51	0010H	INTSER1	0030H
INTP6	0012H	INTSR1	0032H
INTTM00	0014H	INTST1	0034H
INTTM01	0016H	INTCSI0	0036H
INTTM10	0018H	INTAD	0038H
INTTM11	001AH		

3.3.2 CALLT instruction table area

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

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

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

3.3.3 CALLF instruction entry area

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

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

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

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

3.4 Internal Data Area

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

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

Example

```
RSTVCT    CSEG AT 0
          DW     RSTSTRT

           to

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

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

3.4.1 Internal RAM area

The µPD784955 has an on-chip general-purpose static RAM.

This area has the following configuration.

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

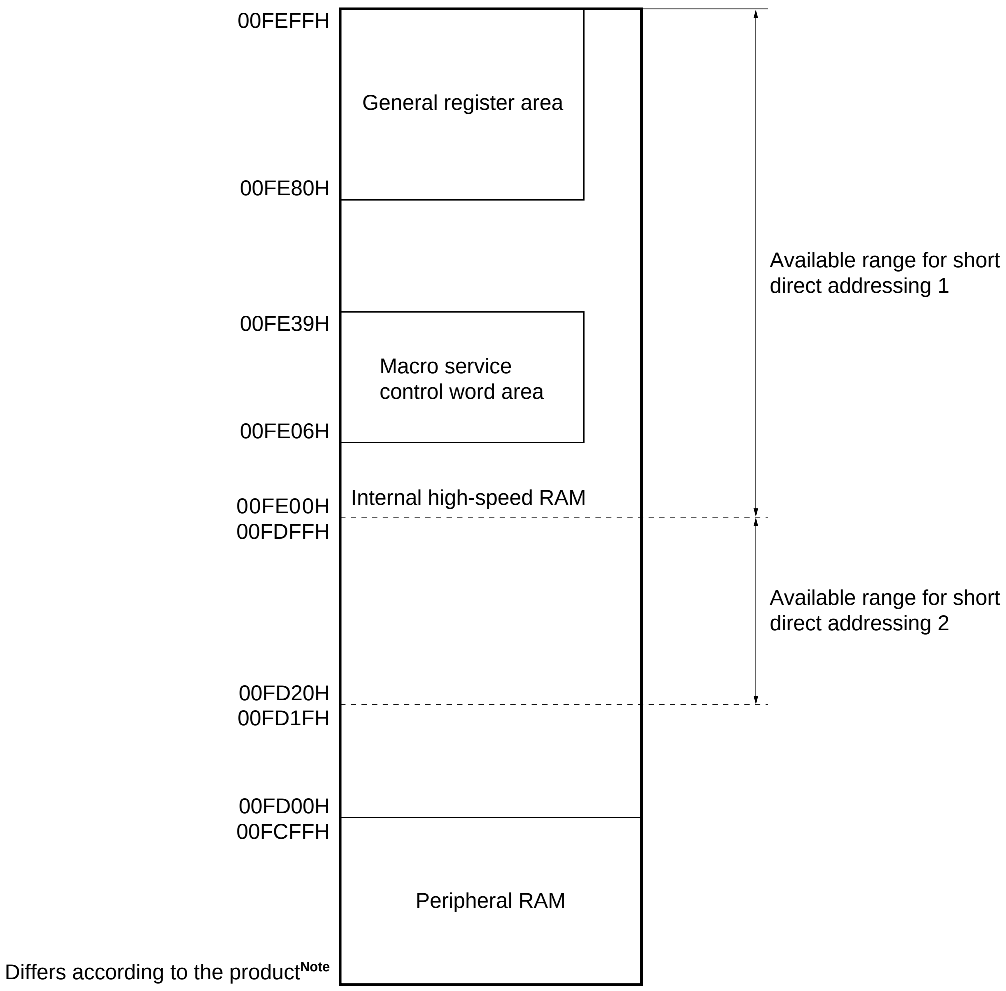
Table 3-2. Internal RAM Area

Internal RAM Part Number	Internal RAM Area		
		Peripheral RAM: PRAM	Internal High-speed RAM: IRAM
µPD784953	768 bytes (0FC00H to 0FEFFH)	256 bytes (0FC00H to 0FCFFH)	512 bytes (0FD00H to 0FEFFH)
µPD784955	2,048 bytes (0F700H to 0FEFFH)	1,536 bytes (0F700H to 0FCFFH)	
µPD78F4956			

Remark The addresses in the table are the values when the LOCATION 0 instruction is executed. When the LOCATION 0FH instruction is executed, 0F0000H must be added to the above values.

Figure 3-4 shows the internal RAM memory map.

Figure 3-4. Internal RAM Memory Map


00FEFFH
General register area
00FE80H
Available range for short direct addressing 1
00FE39H
Macro service control word area
00FE06H
Internal high-speed RAM
00FE00H
00FDFFH
Available range for short direct addressing 2
00FD20H
00FD1FH
00FD00H
00FCFFH
Peripheral RAM
Differs according to the productNote

Note μPD784953	:	00FC00H
μPD784955, 78F4956	:	00F700H

Remark The addresses in the figure are the values when the LOCATION 0 instruction is executed. When the LOCATION 0FH instruction is executed, 0F0000H must be added to the above values.

(1) Internal high-speed RAM (IRAM)

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

A program cannot be fetched from IRAM. If a program is fetched from an address that is mapped by IRAM, the CPU goes into an inadvertent loop.

The following areas are reserved in IRAM.

• General register area	: FE80H to FEFFH
• Macro service control word area	: FE06H to FE39H
• Macro service channel area	: FE00H to FEFFH (The address is set by a macro service control word.)

When the reserved function is not used for each area, the area can be used as normal data memory.

Remark The addresses in this text are the addresses when the LOCATION 0 instruction is executed. When the LOCATION 0FH instruction is executed, 0F0000H must be added to the values in this text.

(2) Peripheral RAM (PRAM)

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

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

3.4.2 Special function register (SFR) area

The special function register (SFR) of the on-chip peripheral hardware is mapped to the area from 0FF00H to 0FFFFH (see **Figures 3-1** to **3-3**).

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

Remark The addresses in this text are the addresses only when the LOCATION 0 instruction is executed. If the LOCATION 0FH instruction is executed, 0F0000H must be added to the values in the text.

3.5 μPD78F4956 Memory Mapping

The μPD78F4956 has a 64-Kbyte flash memory and 2,048-byte internal RAM.

The μPD78F4956 has a function (memory size switching function) so that a part of the internal memory is not used by the software.

The size of the memory can be switched using the internal memory switching register (IMS).

Based on the IMS setting, the memory mapping can be the same as that of the mask ROM versions having a different internal memory (ROM/RAM) size.

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

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

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

Address: 0FFFCH After Reset: FFH W

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

ROM1	ROM0	Internal ROM Capacity Selections
0	0	24 Kbytes
0	1	Setting prohibited
1	0	48 Kbytes
1	1	64 Kbytes

RAM1	RAM0	Internal RAM Capacity Selections
0	0	768 bytes
0	1	Setting prohibited
1	0	
1	1	2,048 bytes

Caution Mask ROM versions (μPD784953, 784955) do not have an IMS.

Table 3-3 shows the IMS settings to have the same memory map as the mask ROM versions.

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

Target Mask ROM Version	IMS Settings
μPD784953	CCH
μPD784955	EFH

3.6 Control Registers

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

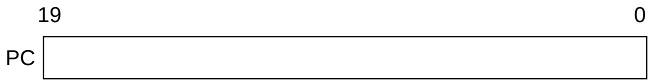
3.6.1 Program counter (PC)

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

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

$\overline{\text{RESET}}$ input sets the low-order 16 bits of the PC to the 16-bit data at addresses 0 and 1, and 0000 in the high-order four bits of the PC.

Figure 3-6. Program Counter (PC) Format



3.6.2 Program status word (PSW)

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

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

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

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

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

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

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

Each flag is described below.

(1) Carry flag (CY)

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

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

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

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

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

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

- **Parity flag action**

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

- **Overflow flag action**

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

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

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

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

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

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

(3) Interrupt request enable flag (IE)

This flag controls the CPU interrupt request acceptance.

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

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

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

(4) Auxiliary carry flag (AC)

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

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

(5) Register set selection flag (RSS)

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

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

(6) Zero flag (Z)

This flag indicates that the operation result is 0.

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

(7) Sign flag (S)

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

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

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

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

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

Table 3-4. Register Bank Selection

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

(9) User flag (UF)

This flag is set and reset by a user program and can be used in program control.

3.6.3 Using the RSS bit

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

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

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

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

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

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

(1) Using the RSS bit

- Registers used in instructions where the A, X, B, C, and AX registers are directly described in the operand column of the operation list (see **26.2**)

- Registers that are implicitly specified in instructions that use the A, AX, B, and C registers by implied addressing

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

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

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

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

The registers used in other cases always become the same registers regardless of the contents of the RSS bit. For registers A, X, B, C, AX, and BC in NEC assembler RA78K4, instruction code is generated for any register described by name or for registers set by an RSS pseudo instruction in the assembler.
When the RSS bit is set or reset, always specify an RSS pseudo instruction immediately before (or immediately after) that instruction (see the following examples).

<Program examples>

- When RSS = 0

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

- When RSS = 1

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

(2) Generation of instruction code in the RA78K4

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

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

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

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

- If R0 to R7 and RP0 to RP4 are specified in r, r', rp, and rp' in the operand column, an instruction code that conforms to the specification is output. (Instruction code that directly describes A or AX in the operand column is not output.)

- The A, B, and C registers that are used in indexed addressing and based indexed addressing cannot be described as R1, R3, R2, or R5, R7, R6.

(3) Usage Cautions

Switching the RSS bit obtains the same effect as holding two register sets. However, be careful and write the program so that implicit descriptions in the program and dynamically changing the RSS bit during program execution always agree.
Also, since a program with RSS = 1 cannot be used in a program that uses context switching, the portability of the program becomes poor. Furthermore, since different registers having the same name are used, the readability of the program worsens, and debugging becomes difficult. Therefore, when RSS = 1 must be used, write the program while taking these problems into consideration.
A register that does not have the RSS bit set can be accessed by specifying the absolute name.

3.6.4 Stack pointer (SP)

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

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

SP is accessed by special instructions.

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

Example Initializing SP

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

Figure 3-8. Stack Pointer (SP) Format

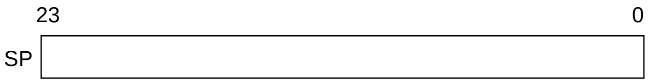


Figure 3-9. Data Saved to the Stack

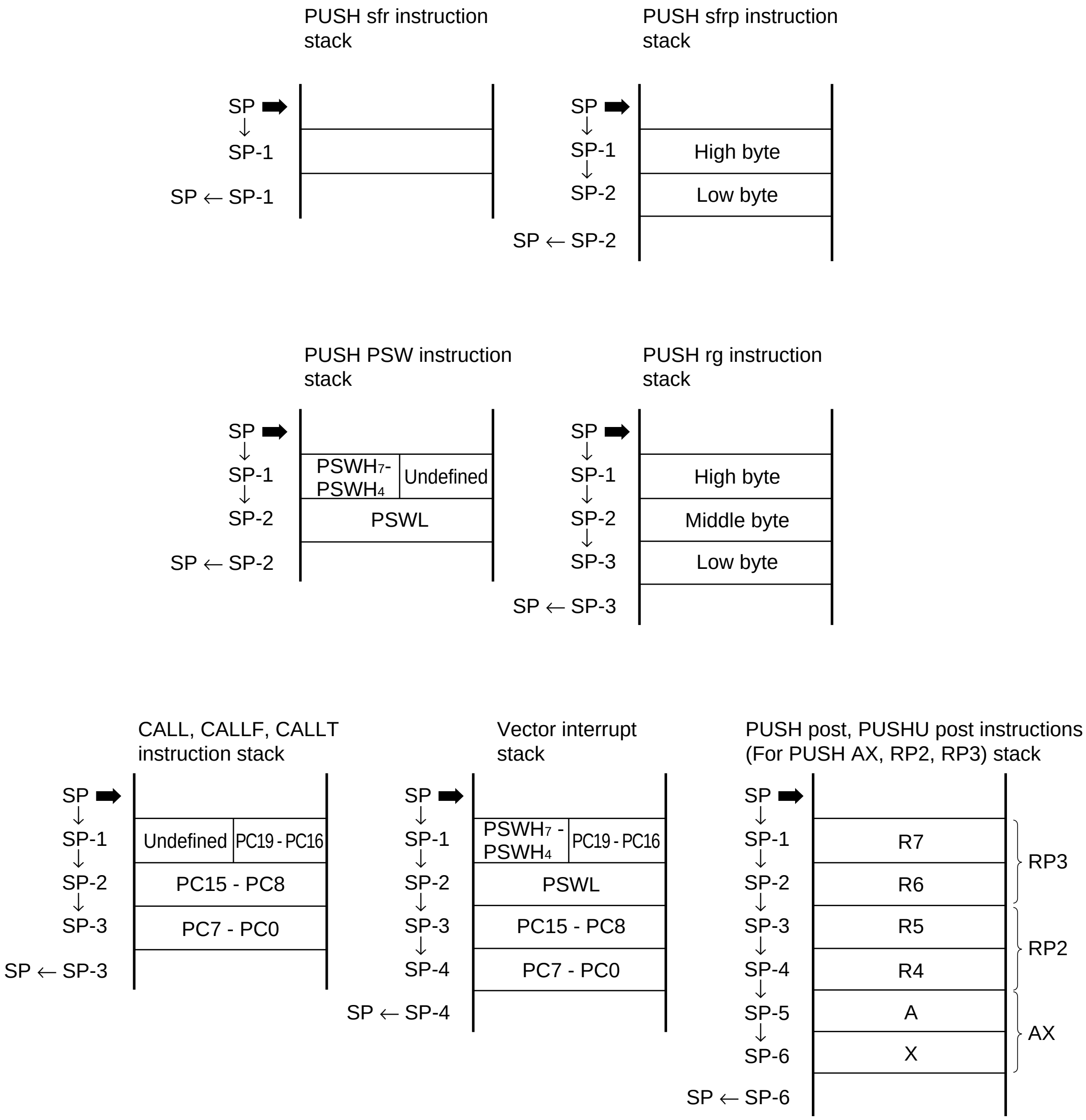
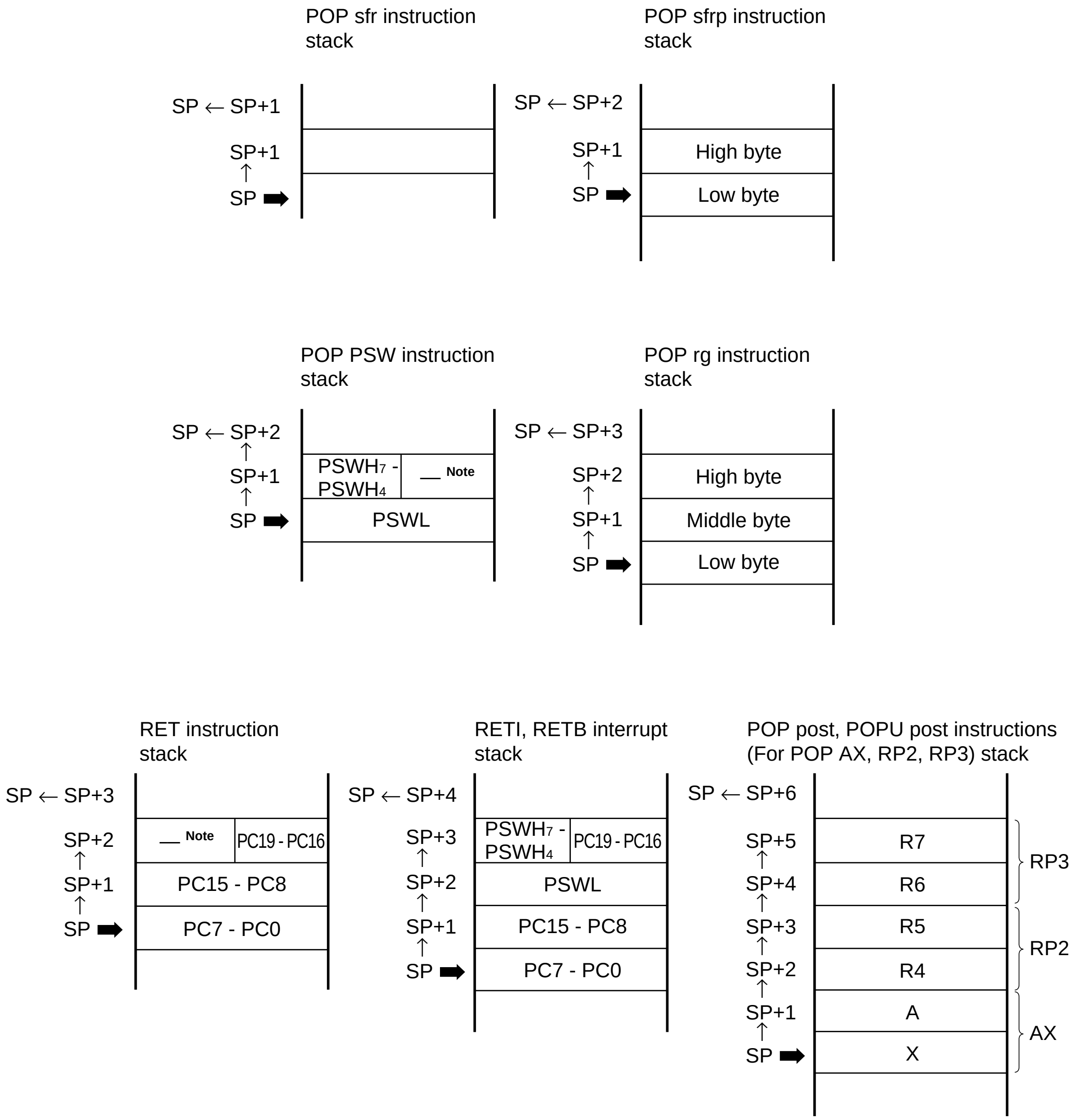


Figure 3-10. Data Restored from the Stack



Note This 4-bit data is ignored.

Cautions 1. In stack addressing, the entire 1-Mbyte space can be accessed, but the stack cannot be guaranteed in the SFR area and internal ROM area.

2. The stack pointer (SP) becomes undefined when $\overline{\text{RESET}}$ is input. In addition, even when SP is in the undefined state, non-maskable interrupts can be accepted. Therefore, when the SP is in the undefined state immediately after the reset is cleared and a request for a non-maskable interrupt is generated, unexpected actions sometimes occur. To avoid this danger, always specify the following in the program after clearing a reset.

```
RSTVCT     CSEG  AT 0
           DW    RSTSTRT

            to

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

3.6.5 Memory expansion mode register (MM)

MM is an 8-bit register that controls the internal fetch cycle. It is set using a 1-bit or 8-bit memory manipulation instruction.

Figure 3-11 shows the format of MM.

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

Figure 3-11. Format of the Memory Expansion Mode Register (MM)

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

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

IFCH	Internal ROM Fetch
0	Normal fetch (Fetches at 6 cycles = 1 byte instruction.)
0	Hi-speed fetch (Fetches at 2 cycles = 1 word (2 bytes) instruction.)

Caution After reset, it should be set to 80H.

3.7 General Registers

3.7.1 Configuration

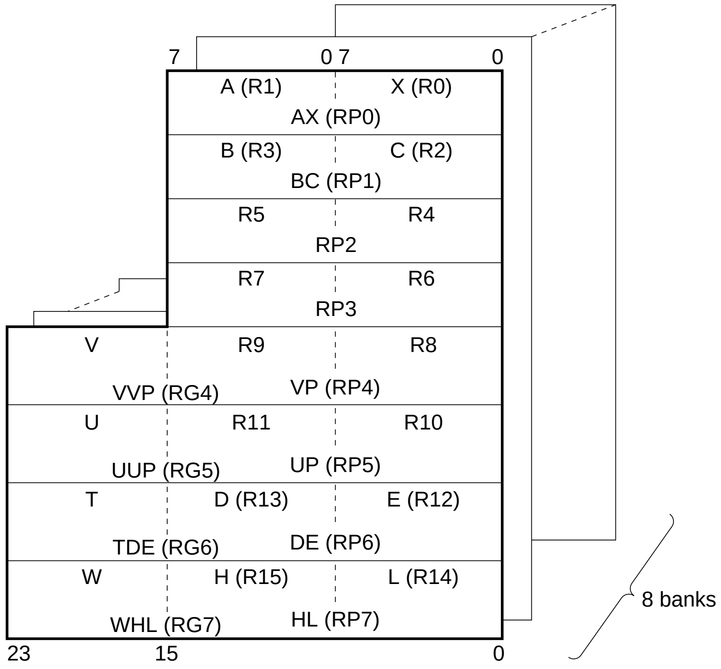
There are sixteen 8-bit general registers. In addition, two 8-bit general registers can be combined and used as a 16-bit general register. Furthermore, four of the 16-bit general registers are combined with an 8-bit register for address expansion and used as a 24-bit address specification register.

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

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

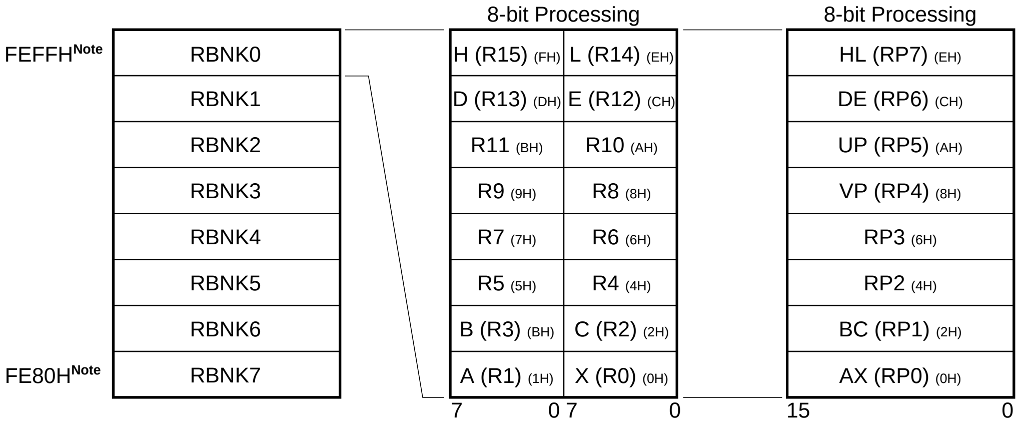
$\overline{\text{RESET}}$ input selects register bank 0. In addition, the register banks that are used in an executing program can be verified by reading the register bank selection flags (RBS0, RBS1, RBS2) in the PSW.

Figure 3-12. General Register Format



Remark The parentheses enclose the absolute names.

Figure 3-13. General Register Addresses



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

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

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

Example When register bank 2 is specified

```
        :
PUSH PSW
SEL RB2   ⎫
        : ⎬ Operation in register bank 2
        : ⎪
POP PSW   ⎭
        :
        :   Operation in original register bank
```

3.7.2 Functions

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

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

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

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

Each register has the unique functions shown below.

A (R1):

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

X (R0):

- This register can store bit data.

AX (RP0):

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

AXDE:

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

B (R3):

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

C (R2):

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

RP2:

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

RP3:

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

VVP (RG4):

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

UUP (RG5):

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

DE (RP6), HL (RP7):

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

TDE (RG6):

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

WHL (RG7):

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

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

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

(a) 8-bit registers

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

(b) 16-bit registers

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

(c) 24-bit registers

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

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

Remark R8 to R11 do not have function names.

3.8 Special Function Registers (SFRs)

These registers are assigned special functions such as the mode register and control register of the internal peripheral hardware and are mapped to the 256-byte area from 0FF00H to 0FFFFH[Note].

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

Caution In this area, do not access an address that is not allocated by an SFR. If erroneously accessed, the μPD784955 enters the deadlock state. The deadlock state is released only by reset input.

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

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

Table 3-6. Special Function Register (SFR) List (1/5)

Address Note 1	Name of Special Function Register (SFR)	Symbol	R/W	Bit Manipulation Unit			After Reset
				1 bit	8 bits	16 bits	
0FF00H	Port 0	P0	R/W	○	○	—	00H[Note 2]
0FF01H	Port 1	P1		○	○	—	
0FF02H	Port 2	P2		○	○	—	
0FF03H	Port 3	P3		○	○	—	
0FF04H	Port 4	P4		○	○	—	
0FF05H	Port 5	P5		○	○	—	
0FF06H	Port 6	P6		○	○	—	
0FF07H	Port 7	P7	R	○	○	—	
0FF09H	Port 9	P9	R/W	○	○	—	
0FF10H	16-bit timer register 0	TM0	R	—	—	○	0000H
0FF11H							
0FF12H	16-bit capture/compare register 00 (16-bit timer/counter)	CR00	R/W	—	—	○	Undefined
0FF13H							
0FF14H	16-bit capture/compare register 01 (16-bit timer/counter)	CR01		—	—	○	
0FF15H							
0FF16H	Capture/compare control register 0	CRC0		○	○	—	00H
0FF18H	16-bit timer mode control register 0	TMC0		○	○	—	
0FF1AH	Timer output control register 0	TOC0		○	○	—	
0FF1CH	Prescaler mode register 0	PRM0		○	○	—	
0FF20H	Port 0 mode register	PM0		○	○	—	FFH
0FF21H	Port 1 mode register	PM1		○	○	—	
0FF22H	Port 2 mode register	PM2		○	○	—	
0FF23H	Port 3 mode register	PM3		○	○	—	
0FF24H	Port 4 mode register	PM4		○	○	—	
0FF25H	Port 5 mode register	PM5		○	○	—	
0FF26H	Port 6 mode register	PM6		○	○	—	
0FF29H	Port 9 mode register	PM9		○	○	—	
0FF30H	Pull-up resistor option register 0	PU0		○	○	—	00H
0FF31H	Pull-up resistor option register 1	PU1		○	○	—	
0FF32H	Pull-up resistor option register 2	PU2		○	○	—	
0FF33H	Pull-up resistor option register 3	PU3		○	○	—	
0FF39H	Pull-up resistor option register 9	PU9		○	○	—	
0FF3CH	16-bit compare register 10	CR10		—	—	○	Undefined
0FF3DH							

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

2. Because each port is initialized to input mode at reset, “00H” is not actually read. The output latch is initialized to “0”.

Table 3-6. Special Function Register (SFR) List (2/5)

Address Note	Name of Special Function Register (SFR)	Symbol	R/W	Bit Manipulation Unit			After Reset
				1 bit	8 bits	16 bits	
0FF3EH	16-bit compare register 11	CR11	R/W	—	—	○	Undefined
0FF3FH							
0FF42H	16-bit compare register 20	CR20		—	—	○	
0FF43H							
0FF4EH	Pull-up resistor option register	PUO		○	○	—	0000H
0FF50H	8-bit timer register 6	TM6		—	○	—	
0FF51H	8-bit timer register 7	TM7		—	○	—	
0FF52H	8-bit compare register 6	CR6		—	○	—	Undefined
0FF53H	8-bit compare register 7	CR7		—	○	—	
0FF54H	Timer mode control register 6	TMC6		○	○	—	00H
0FF55H	Timer mode control register 7	TMC7		○	○	—	
0FF56H	Timer clock select register 6	TCL6		○	○	—	
0FF57H	Timer clock select register 7	TCL7		○	○	—	
0FF60H	16-bit timer register 1	TM1	R	—	—	○	0000H
0FF61H							
0FF62H	16-bit timer register 2	TM2		—	—	○	
0FF63H							
0FF64H	16-bit timer register 3	TM3		—	—	○	
0FF65H							
0FF66H	16-bit timer register 4	TM4		—	—	○	
0FF67H							
0FF68H	16-bit timer register 5	TM5		—	—	○	
0FF69H							
0FF6BH	16-bit timer mode control register 1	TMC1	R/W	○	○	—	00H
0FF6CH	16-bit timer mode control register 2	TMC2		○	○	—	
0FF6DH	16-bit timer mode control register 3	TMC3		○	○	—	
0FF6EH	16-bit timer mode control register 4	TMC4		○	○	—	
0FF6FH	16-bit timer mode control register 5	TMC5		○	○	—	
0FF70H	Asynchronous serial interface mode register 1	ASIM1		○	○	—	
0FF72H	Asynchronous serial interface status register 1	ASIS1	R	○	○	—	
0FF74H	Transmission shift register 1	TXS1	W	—	○	—	FFH
	Reception buffer register 1	RXB1	R	—	○	—	
0FF76H	Baud rate generator control register 1	BRGC1	R/W	○	○	—	00H
0FF78H	16-bit compare register 21	CR21		—	—	○	Undefined
0FF79H							
0FF7BH	Timer output control register 1	TOC1		○	○	—	00H

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

Table 3-6. Special Function Register (SFR) List (3/5)

Address Note	Name of Special Function Register (SFR)	Symbol		R/W	Bit Manipulation Unit			After Reset
					1 bit	8 bits	16 bits	
0FF7CH	Timer output control register 2	TOC2		R/W	○	○	—	00H
0FF7DH	Capture/compare control register 4	CRC4			○	○	—	
0FF7EH	Capture/compare control register 5	CRC5			○	○	—	
0FF80H	A/D converter mode register 0	ADM0			○	○	—	
0FF81H	Analog input channel specification register 0	ADS0			○	○	—	
0FF83H	A/D conversion result register 0	ADCR0		R	—	○	—	Undefined
0FF85H	Prescaler mode register 1	PRM1		R/W	○	○	—	00H
0FF86H	Prescaler mode register 2	PRM2			○	○	—	
0FF87H	Prescaler mode register 3	PRM3			○	○	—	
0FF88H	Prescaler mode register 4	PRM4			○	○	—	
0FF89H	Prescaler mode register 5	PRM5			○	○	—	
0FF8AH	16-bit compare register 30	CR30			—	—	○	Undefined
0FF8BH								
0FF8EH	16-bit compare register 31	CR31			—	—	○	
0FF8FH								
0FF90H	Serial operating mode register 0	CSIM0			○	○	—	00H
0FF94H	Serial I/O shift register 0	SIO0			—	○	—	
0FF96H	Real-time output buffer register L0	RTBL0			—	○	—	
0FF97H	Real-time output buffer register H0	RTBH0			—	○	—	
0FF98H	Real-time output port mode register 0	RTPM0			○	○	—	
0FF99H	Real-time output port control register 0	RTPC0			○	○	—	
0FF9AH	Real-time output buffer register L1	RTBL1			—	○	—	
0FF9BH	Real-time output buffer register H1	RTBH1			—	○	—	
0FF9CH	Real-time output port mode register 1	RTPM1			○	○	—	
0FF9DH	Real-time output port control register 1	RTPC1			○	○	—	
0FFA0H	External interrupt rising edge enable register	EGP0			○	○	—	
0FFA2H	External interrupt falling edge enable register	EGN0			○	○	—	
0FFA4H	PWM modulation control register 0	PWMC0			○	○	—	
0FFA5H	PWM modulation control register 1	PWMC1			○	○	—	
0FFA6H	PWM modulation buffer register 0	BFPWMC0			○	○	—	
0FFA7H	PWM modulation buffer register 1	BFPWMC1			○	○	—	
0FFA8H	In-service priority register	ISPR		R	○	○	—	
0FFAAH	Interrupt mode control register	IMC		R/W	○	○	—	80H
0FFACH	Interrupt mask flag register 0L	MK0L	MK0		○	○	○	FFFFH
0FFADH	Interrupt mask flag register 0H	MK0H			○	○		

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

Table 3-6. Special Function Register (SFR) List (4/5)

Address Note	Name of Special Function Register (SFR)	Symbol		R/W	Bit Manipulation Unit			After Reset
					1 bit	8 bits	16 bits	
0FFAEH	Interrupt mask flag register 1L	MK1L	MK1	R/W	○	○	○	FFFFH
0FFAFH	Interrupt mask flag register 1H	MK1H			○	○		
0FFB0H	16-bit capture/compare register 40	CR40			—	—	○	Undefined
0FFB1H					—	—	○	
0FFB2H	16-bit capture/compare register 41	CR41			—	—	○	
0FFB3H					—	—	○	
0FFB4H	16-bit capture/compare register 42	CR42			—	—	○	
0FFB5H					—	—	○	
0FFB6H	16-bit capture/compare register 50	CR50			—	—	○	
0FFB7H					—	—	○	
0FFB8H	16-bit capture/compare register 51	CR51			—	—	○	
0FFB9H					—	—	○	
0FFBEH	16-bit compare register 52	CR52			—	—	○	
0FFBFH					—	—	○	
0FFC0H	Standby control register	STBC			—	○	—	00H
0FFC2H	Watchdog timer mode register	WDM			—	○	—	
0FFC4H	Memory expansion mode register	MM			○	○	—	20H
0FFCFH	Oscillation stabilizing time selection register	OSTS			○	○	—	00H
0FFE0H	Interrupt control register (INTP0)	PIC0			○	○	—	43H
0FFE1H	Interrupt control register (INTP1)	PIC1			○	○	—	
0FFE2H	Interrupt control register (INTTM41/INTP2)	PIC2			○	○	—	
0FFE3H	Interrupt control register (INTP3)	PIC3			○	○	—	
0FFE4H	Interrupt control register (INTP4)	PIC4			○	○	—	
0FFE5H	Interrupt control register (INTTM51/INTP5)	PIC5			○	○	—	
0FFE6H	Interrupt control register (INTP6)	PIC6			○	○	—	
0FFE7H	Interrupt control register (INTTM00)	TMIC00			○	○	—	
0FFE8H	Interrupt control register (INTTM01)	TMIC01			○	○	—	
0FFE9H	Interrupt control register (INTTM10)	TMIC10			○	○	—	
0FFEAH	Interrupt control register (INTTM11)	TMIC11			○	○	—	
0FFEBH	Interrupt control register (INTTM20)	TMIC20			○	○	—	
0FFECH	Interrupt control register (INTTM21)	TMIC21			○	○	—	
0FFEDH	Interrupt control register (INTTM30)	TMIC30			○	○	—	
0FFEEH	Interrupt control register (INTTM31)	TMIC31			○	○	—	
0FFEFH	Interrupt control register (INTTM40)	TMIC40			○	○	—	

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

Table 3-6. Special Function Register (SFR) List (5/5)

Address Note 1	Name of Special Function Register (SFR)	Symbol	R/W	Bit Manipulation Unit			After Reset
				1 bit	8 bits	16 bits	
0FFF0H	Interrupt control register (INTTM42)	TMIC42	R/W	○	○	—	43H
0FFF1H	Interrupt control register (INTTM50)	TMIC50		○	○	—	
0FFF2H	Interrupt control register (INTTM52)	TMIC52		○	○	—	
0FFF3H	Interrupt control register (INTTM6)	TMIC6		○	○	—	
0FFF4H	Interrupt control register (INTTM7)	TMIC7		○	○	—	
0FFF5H	Interrupt control register (INTSER1)	SERIC1		○	○	—	
0FFF6H	Interrupt control register (INTSR1)	SRIC1		○	○	—	
0FFF7H	Interrupt control register (INTST1)	STIC1		○	○	—	
0FFF8H	Interrupt control register (INTCSI0)	CSIIC0		○	○	—	
0FFF9H	Interrupt control register (INTAD)	ADIC		○	○	—	
0FFFCH	Internal memory size switching register Note 2	IMS	W	—	○	—	FFH

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

2. Only for the *μ*PD78F4956

3.9 Cautions

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

(2) Special function register (SFR)
Do not access an address that is allocated to an SFR in the area from 0FF00H to 0FFFFH[Note]. If mistakenly accessed, the μPD784955 enters the deadlock state. The deadlock state is released only by $\overline{\text{RESET}}$ input.

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

(3) Stack pointer (SP) operation
Although the entire 1-Mbyte space can be accessed by stack addressing, the stack cannot be guaranteed in the SFR area and the internal ROM area.

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

```
RSTVCT     CSEG  AT 0
           DW    RSTSTRT

            to

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

[MEMO]

CHAPTER 4 CLOCK GENERATOR

4.1 Functions

The clock generator is the circuit that generates the clock that is supplied to the CPU and peripheral hardware. It oscillates at the frequency of 12.5 MHz. Oscillation is stopped by executing the STOP instruction or by setting standby control register (STBC).

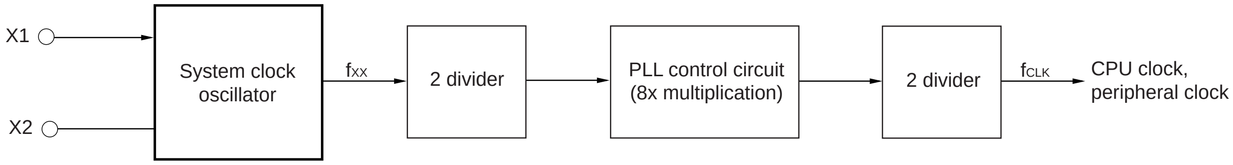
4.2 Configuration

The clock generator consists of the following hardware.

Table 4-1. Clock Generator Configuration

Item	Configuration
Control register	Standby control register (STBC) Oscillation stabilization time specification register (OSTS)

★ **Figure 4-1. Block Diagram of Clock Generator**



X1
X2
System clock oscillator
fXX
2 divider
PLL control circuit (8x multiplication)
2 divider
fCLK
CPU clock, peripheral clock

Remark f_{XX} : Oscillation frequency

f_{CLK} : Internal system clock frequency ($f_{CLK} = 2 \bullet f_{XX}$)

4.3 Control Register

(1) Standby control register (STBC)

This register is used to set the standby mode. For the details of the standby mode, refer to **CHAPTER 23 STANDBY FUNCTION**.

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

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

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

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

Figure 4-2 shows the format of STBC.

Figure 4-2. Standby Control Register (STBC) Format

Address: 0FFC0H After Reset: 00H R/W

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

STP	HLT	Operation Specification Flag
0	0	Normal operation mode
0	1	HALT mode (Automatically cleared upon cancellation of HALT mode)
1	0	STOP mode (Automatically cleared upon cancellation of STOP mode)
1	1	IDLE mode (Automatically cleared upon cancellation of IDLE mode)

Caution After the standby instruction (after standby is cleared), a NOP instruction should be executed three times. If there is contention between execution of the standby instruction and an interrupt request, the standby instruction is not executed, and the interrupt is received after several instructions following the standby instruction have been executed. Instructions that are executed before the interrupt is received are instructions that start within a maximum of 6 blocks after execution of the standby instruction.

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

(2) Oscillation stabilization time specification register (OSTS)

OSTS is a register that specifies operation of the oscillator.
OSTS is set by a 1-bit or 8-bit memory manipulation instruction.
$\overline{\text{RESET}}$ input sets OSTS to 00H.

Figure 4-3. Oscillation Stabilization Time Specification Register (OSTS) Format

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

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

OSTS1	OSTS0	Oscillation Stabilization Time
0	0	$2^{19}/f_{CLK}$ (65.5 ms)
0	1	$2^{18}/f_{CLK}$ (32.8 ms)
1	0	$2^{17}/f_{CLK}$ (16.4 ms)
1	1	$2^{16}/f_{CLK}$ (8.2 ms)

Remark Figures in parentheses apply to operation with f_{CLK} = 8 MHz.

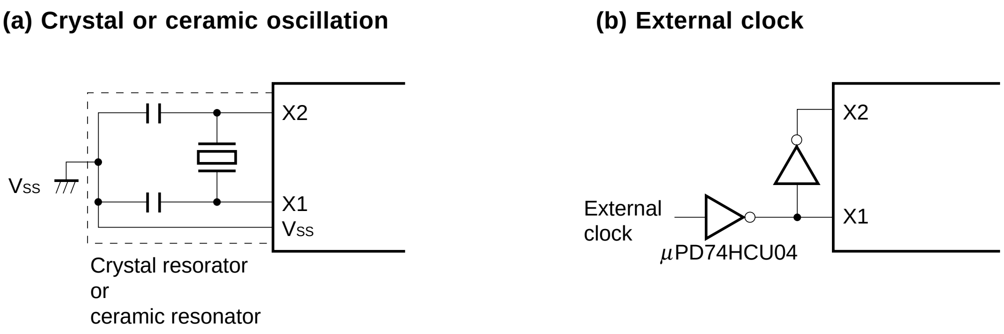
4.4 System Clock Oscillator

The system clock oscillator oscillates with a crystal resonator or a ceramic resonator connected to the X1 and X2 pins.

External clocks can be input to the system clock oscillator. In this case, input a clock signal to the X1 pin and an anti-phase clock signal to the X2 pin.

Figure 4-4 shows an external circuit of the system clock oscillator.

Figure 4-4. External Circuit of System Clock Oscillator



(a) Crystal or ceramic oscillation

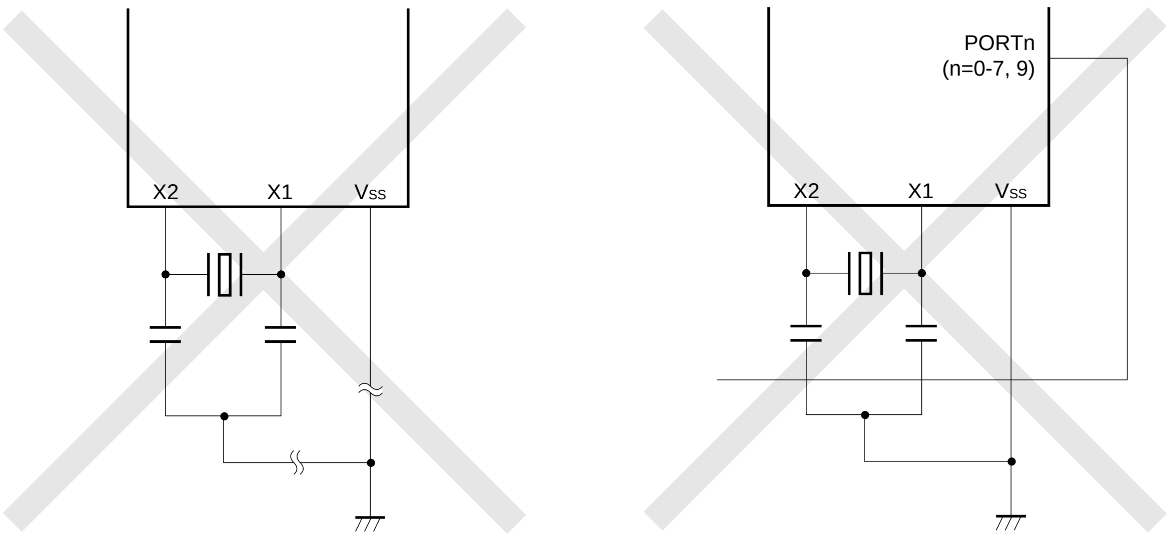
(b) External clock

Caution When using a system clock oscillator, carry out wiring in the broken line area in Figure 4-4 to prevent any effects from wiring capacities.

- **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 a line through which a high alternating current flows.**
- **Always keep the ground of the capacitor of the oscillation circuit at the same potential as V_{SS1}. Do not ground the capacitor to a ground pattern through which a high current flows.**
- **Do not fetch signals from the oscillation circuit.**

Figure 4-5 shows examples of oscillators that are connected incorrectly.

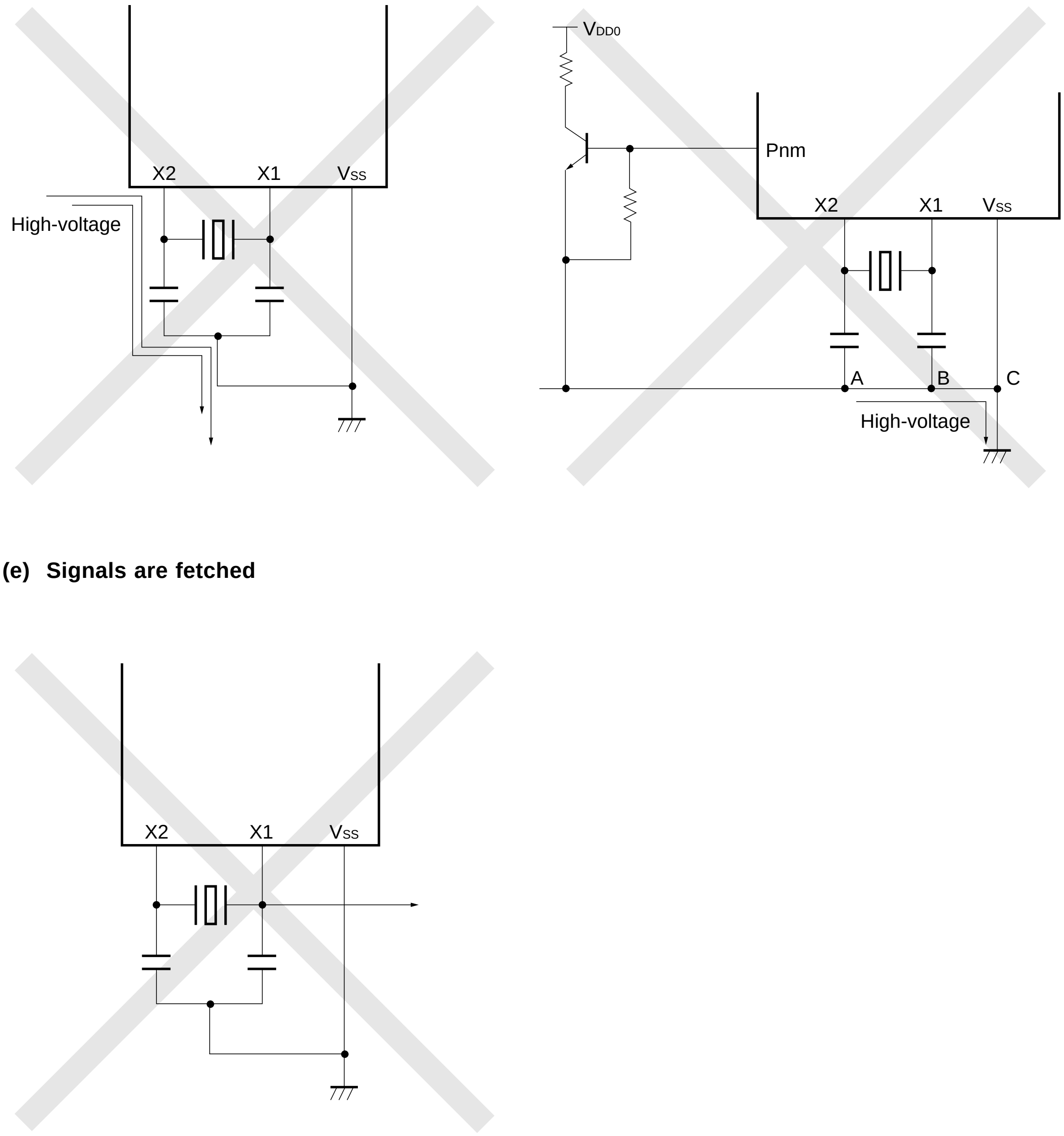
Figure 4-5. Examples of Oscillator Connected Incorrectly (1/2)



(a) Wiring of connection circuits is too long

(b) Signal conductors intersect each other

Figure 4-5. Examples of Oscillator Connected Incorrectly (2/2)



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

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

(e) Signals are fetched

4.4.1 Frequency divider

The frequency divider divides the main system clock oscillator output (f_{XX}) and generates various clocks.

4.5 Clock Generator Operations

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

★
- System clock (f_{CLK})
- CPU clock (f_{CPU})
- Clock to peripheral hardware

Operation of the clock generator is set by the standby control register (STBC).

System clock oscillation stops while low level is applied to the $\overline{\text{RESET}}$ pin.

CHAPTER 5 PORT FUNCTIONS

5.1 Digital Input/Output Port

The ports shown in Figure 5-1 are provided to make various control operations possible. Table 5-1 shows the function of each port. Ports 0 through 6, 9 can be connected to internal pull-up resistors by software when inputting.

Figure 5-1. Port Configuration

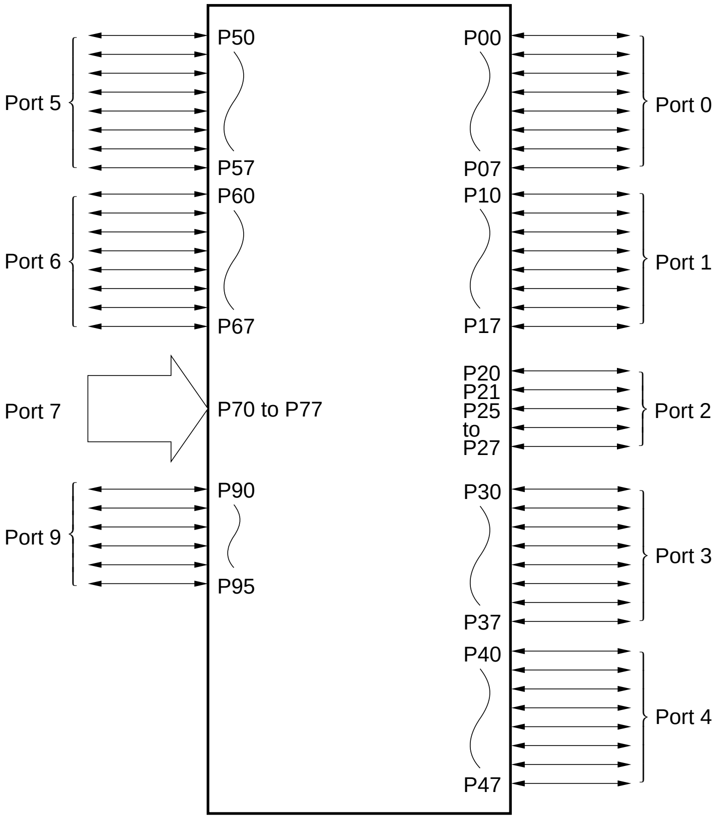


Table 5-1. Port Functions

Port	Pin Name	Function	Specification of Software Pull-Up Resistor
Port 0	P00 to P07	Can be specified for input or output bit-wise	Specifiable bit-wise
Port 1	P10 to P17	Can be specified for input or output bit-wise	Specifiable bit-wise
Port 2	P20, P21, P25 to P27	Can be specified for input or output bit-wise	Specifiable bit-wise
Port 3	P30 to P37	Can be specified for input or output bit-wise	Specifiable bit-wise
Port 4	P40 to P47	Can be specified for input or output bit-wise	Specifiable for each port in a batch
Port 5	P50 to P57	Can be specified for input or output bit-wise	Specifiable for each port in a batch
Port 6	P60 to P67	Can be specified for input or output bit-wise	Specifiable for each port in a batch
Port 7	P70 to P77	Input port	—
Port 9	P90 to P95	Can be specified for input or output bit-wise	Specifiable bit-wise

5.2 Port Configuration

Ports consist of the following hardware:

Table 5-2. Port Configuration

Item	Configuration
Control register	Port mode register (PMm: m = 0 to 6, 9) Pull-up resistor option register (PUO, PUm: m = 0 to 3, 9)
Port	Total: 67 ports (8 inputs, 59 inputs/outputs)
Pull-up resistor	Total: 59 (software control)

5.2.1 Port 0

Port 0 is a 8-bit input/output port with output latch. The P00 to P07 pins can specify the input mode/output mode in 1-bit units with the port 0 mode register. A pull-up resistor can be connected to the P00 to P07 pins via the pull-up resistor option register 0, regardless of whether the input mode or output mode is specified.

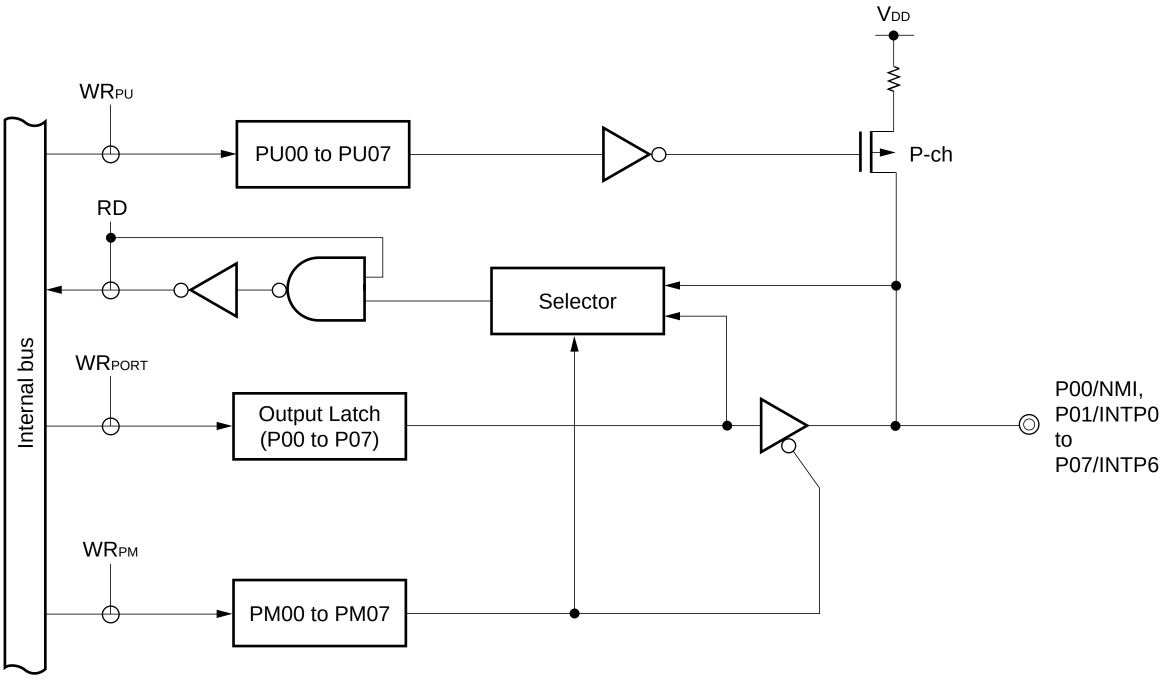
Port 0 also supports external interrupt request input as an alternate function.

$\overline{\text{RESET}}$ input sets port 0 to the input mode.

Figure 5-2 shows the block diagram of port 0.

Caution Because port 0 also serves for external interrupt request input, the interrupt request flag is set by specifying the port function output mode and changing the output level. Thus, when the output mode is used, set the interrupt mask flag to 1.

Figure 5-2. Block Diagram of P00 to P07



PU : Pull-up resistor option register
PM : Port mode register
RD : Port 0 read signal
WR : Port 0 write signal

5.2.2 Port 1

Port 3 is an 8-bit input/output port with output latch. The P10 to P17 pins can specify the input mode/output mode in 1-bit units with the port 1 mode register. A pull-up resistor can be connected to the P10 to P17 pins via the pull-up resistor option register 1, regardless of whether the input mode or output mode is specified.

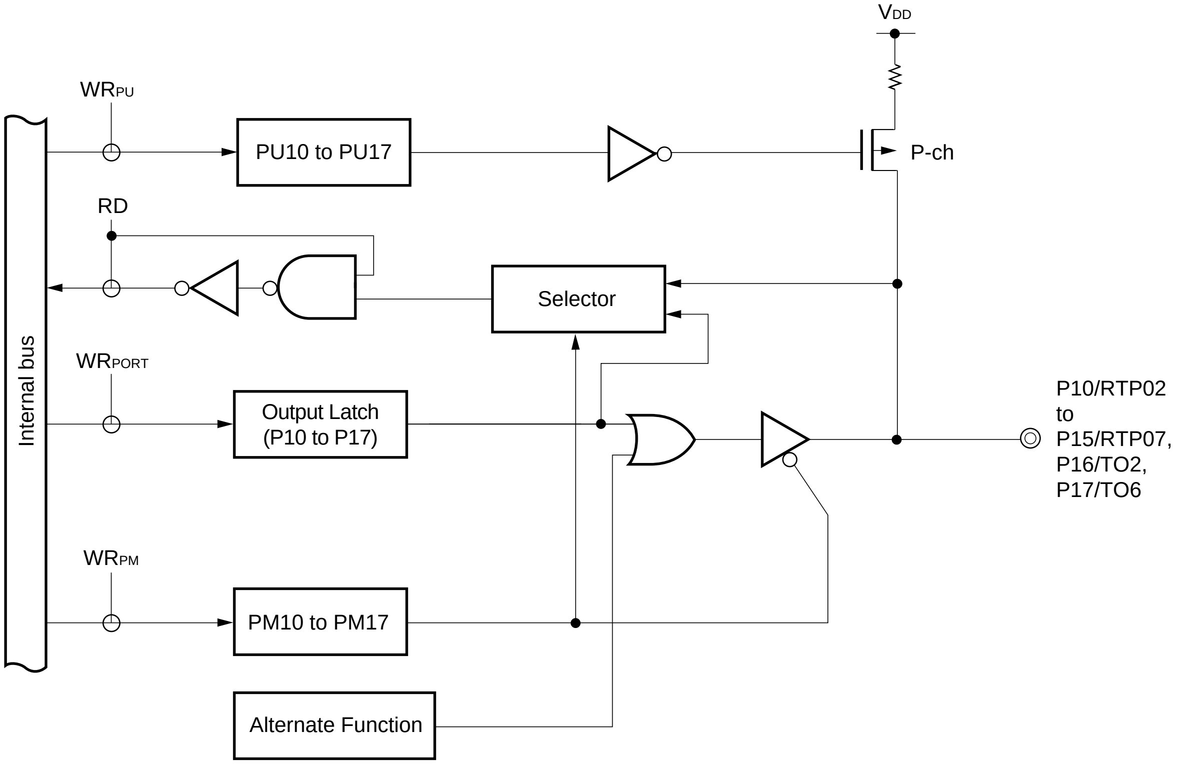
Port 1 also supports real-time output and timer output as alternate functions.

Port 1 can drive the LED directly.

$\overline{\text{RESET}}$ input sets port 1 to the input mode.

Figure 5-3 shows a block diagram of port 1.

Figure 5-3. Block Diagram of P10 to P17



PU : Pull-up resistor option register
PM : Port mode register
RD : Port 1 read signal
WR : Port 1 write signal

5.2.3 Port 2

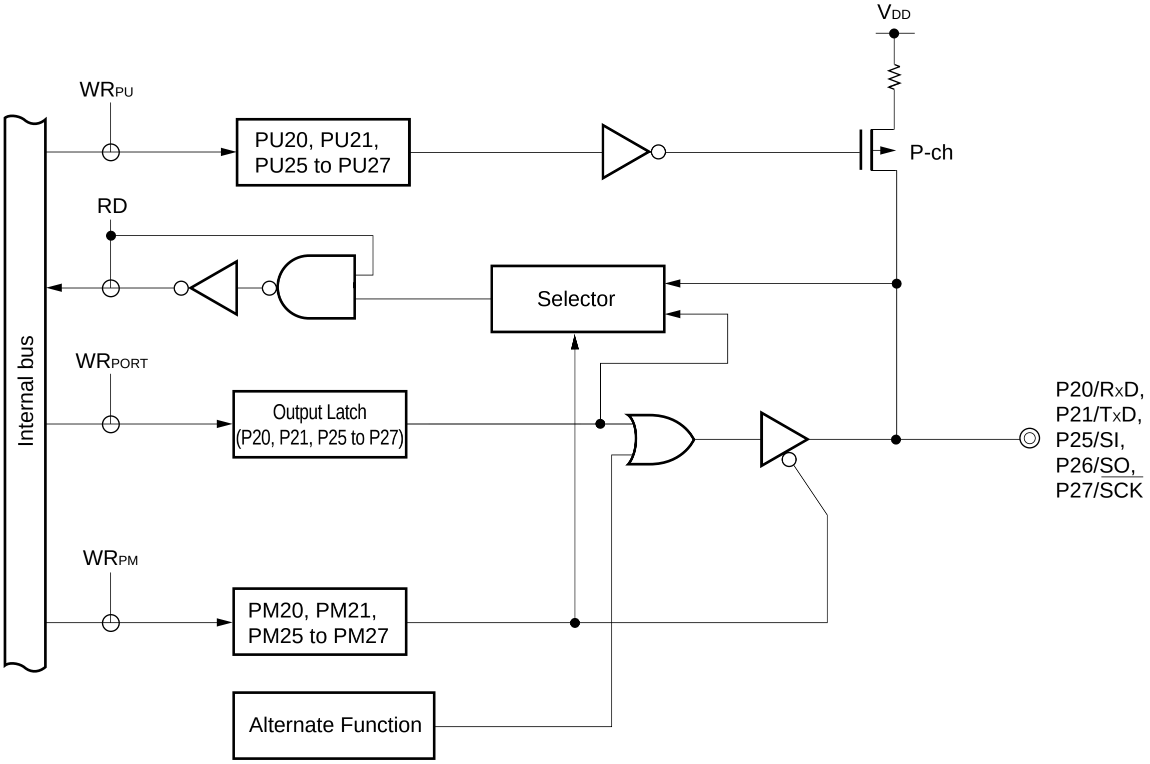
Port 2 is an 5-bit input/output port with output latch. P20, P21, P25 to P27 pins can specify the input mode/output mode in 1-bit units with the port 2 mode register. A pull-up resistor can be connected to the P20, P21, P25 to P27 pins via the pull-up resistor option register 2, regardless of whether the input mode or output mode is specified.

Port 2 also supports serial interface data input/output as alternate functions.

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

Figure 5-4 shows a block diagram of port 2.

Figure 5-4. Block Diagram of P20, P21, P25 to P27



PU : Pull-up resistor option register

PM : Port mode register

RD : Port 2 read signal

WR : Port 2 write signal

5.2.4 Port 3

Port 3 is an 8-bit input/output port with output latch. The P30 to P37 pins can specify the input mode/output mode in 1-bit units with the port 1 mode register. A pull-up resistor can be connected to the P30 to P37 pins via the pull-up resistor option register 1, regardless of whether the input mode or output mode is specified.

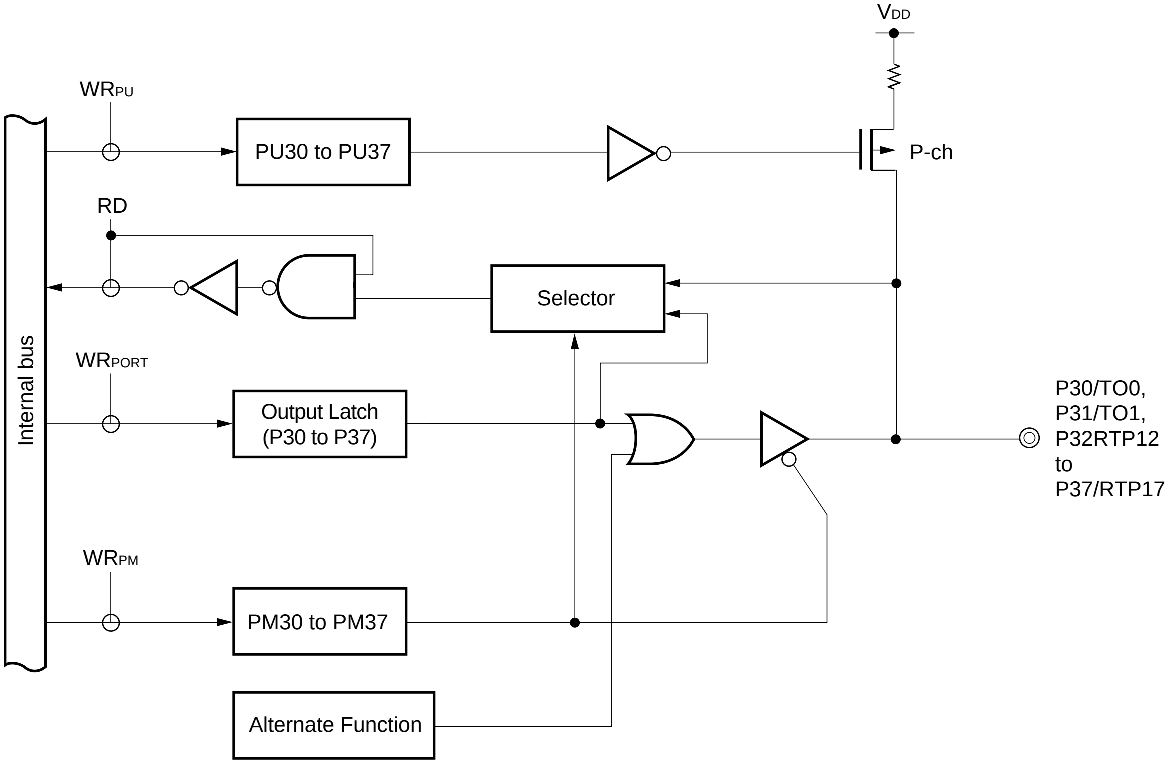
Port 3 also supports real-time output and timer output as alternate functions.

Port 3 can drive the LED directly.

$\overline{\text{RESET}}$ input sets port 3 to the input mode.

Figure 5-5 shows a block diagram of port 3.

Figure 5-5. Block Diagram of P30 to P37



PU : Pull-up resistor option register

PM : Port mode register

RD : Port 3 read signal

WR : Port 3 write signal

5.2.5 Port 4

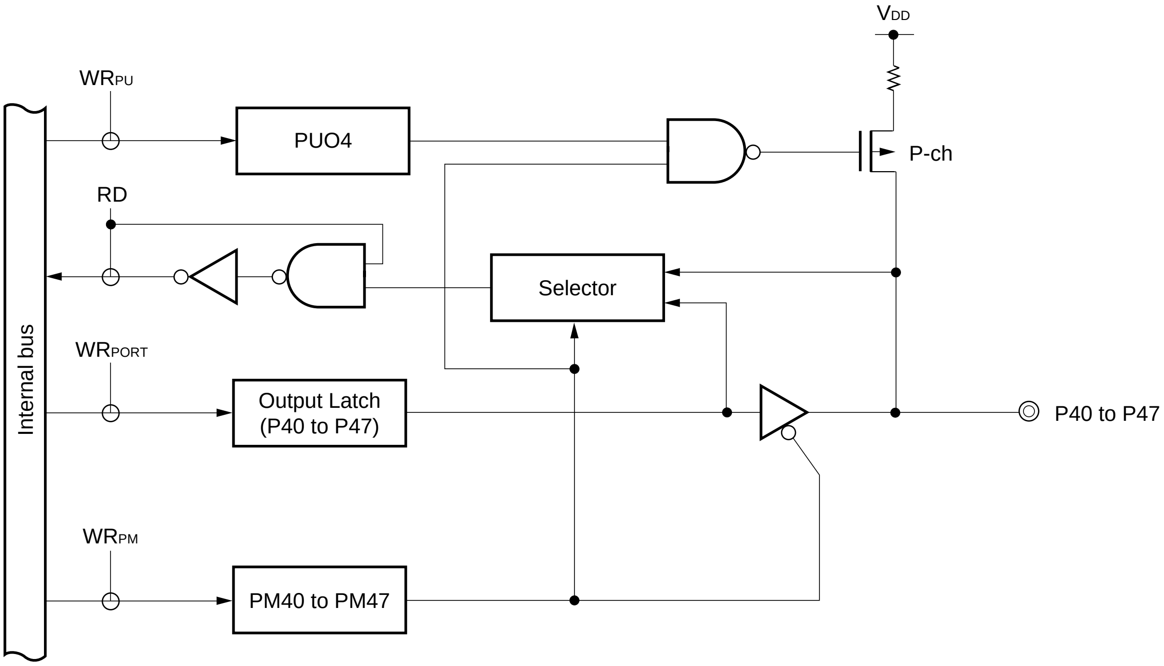
Port 4 is an 8-bit input/output port with output latch. The P40 to P47 pins can specify the input mode/output mode in 1-bit units with the port 4 mode register. When the P40 to P47 pins are used as input ports, a pull-up resistor can be connected to them in 8-bit units with bit 4 (PUO4) of the pull-up resistor option register.

Port 4 can drive LED directly.

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

Figure 5-6 shows a block diagram of port 4.

Figure 5-6. Block Diagram of P40 to P47



PUO	:	Pull-up resistor option register
PM	:	Port mode register
RD	:	Port 4 read signal
WR	:	Port 4 write signal

5.2.6 Port 5

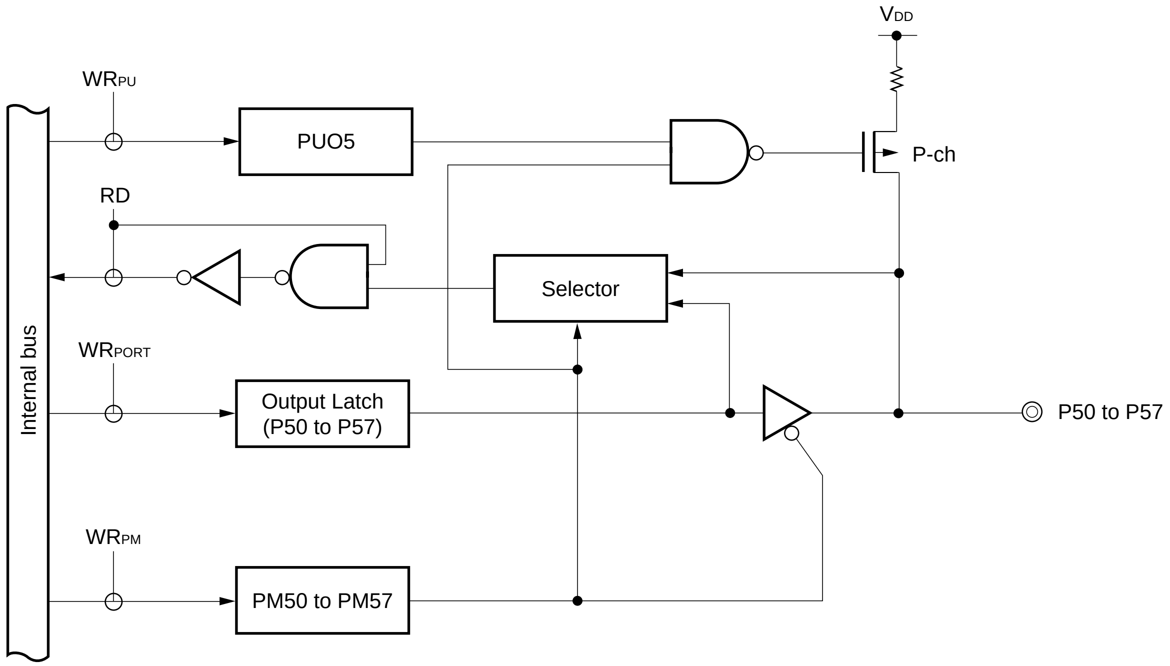
Port 5 is an 8-bit input/output port with output latch. The P50 to P57 pins can specify the input mode/output mode in 1-bit units with the port 5 mode register. When the P50 to P57 pins are used as input ports, a pull-up resistor can be connected to them in 8-bit units with bit 5 (PUO5) of the pull-up resistor option register.

Port 5 can drive LEDs directly.

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

Figure 5-7 shows a block diagram of port 5.

Figure 5-7. Block Diagram of P50 to P57



PUO	:	Pull-up resistor option register
PM	:	Port mode register
RD	:	Port 5 read signal
WR	:	Port 5 write signal

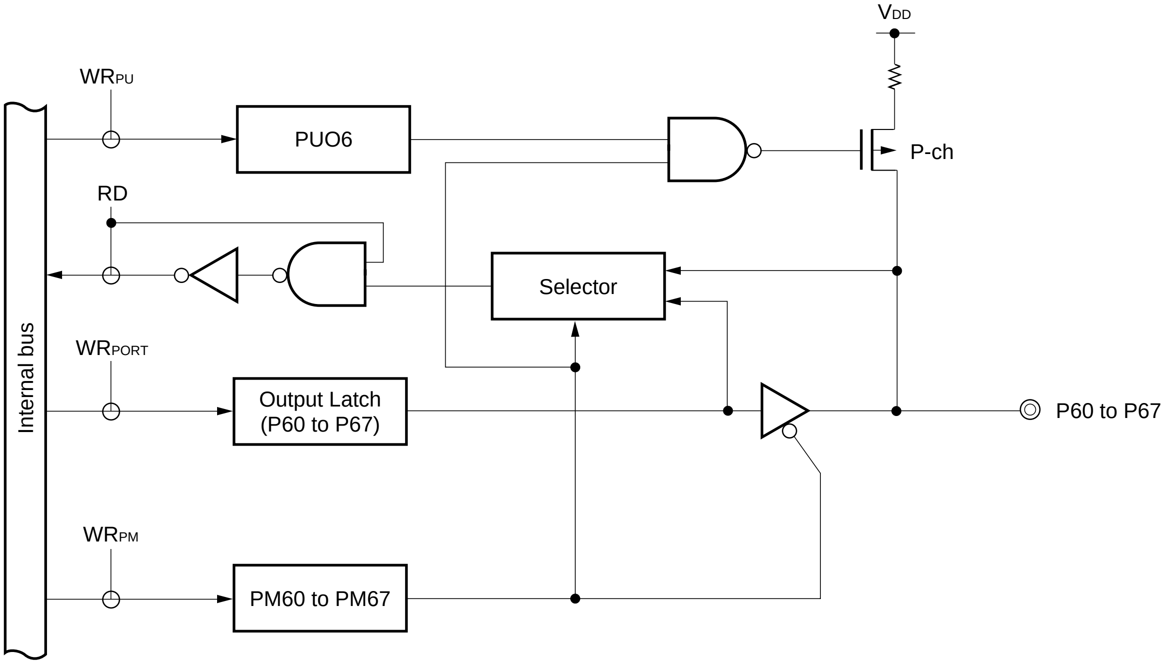
5.2.7 Port 6

Port 6 is an 8-bit input/output port with output latch. The P60 to P67 pins can specify the input mode/output mode in 1-bit units with the port 6 mode register. When pins P60 to P67 are used as input ports, a pull-up resistor can be connected to them in 8-bit units with bit 6 (PUO6) of the pull-up resistor option register.

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

Figure 5-8 shows block diagrams of port 6.

Figure 5-8. Block Diagram of P60 to P67



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

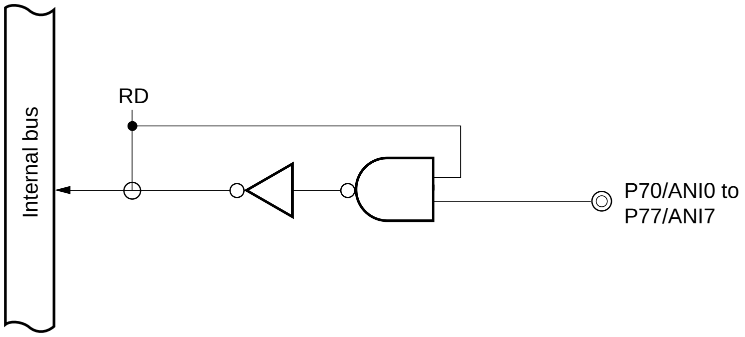
5.2.8 Port 7

This is an 8-bit input-only port with no on-chip pull-up resistor.

Port 7 supports A/D converter analog input as an alternate function.

Figure 5-9 shows a block diagram of port 7.

Figure 5-9. Block Diagram of P70 to P77



RD : Port 7 read signal

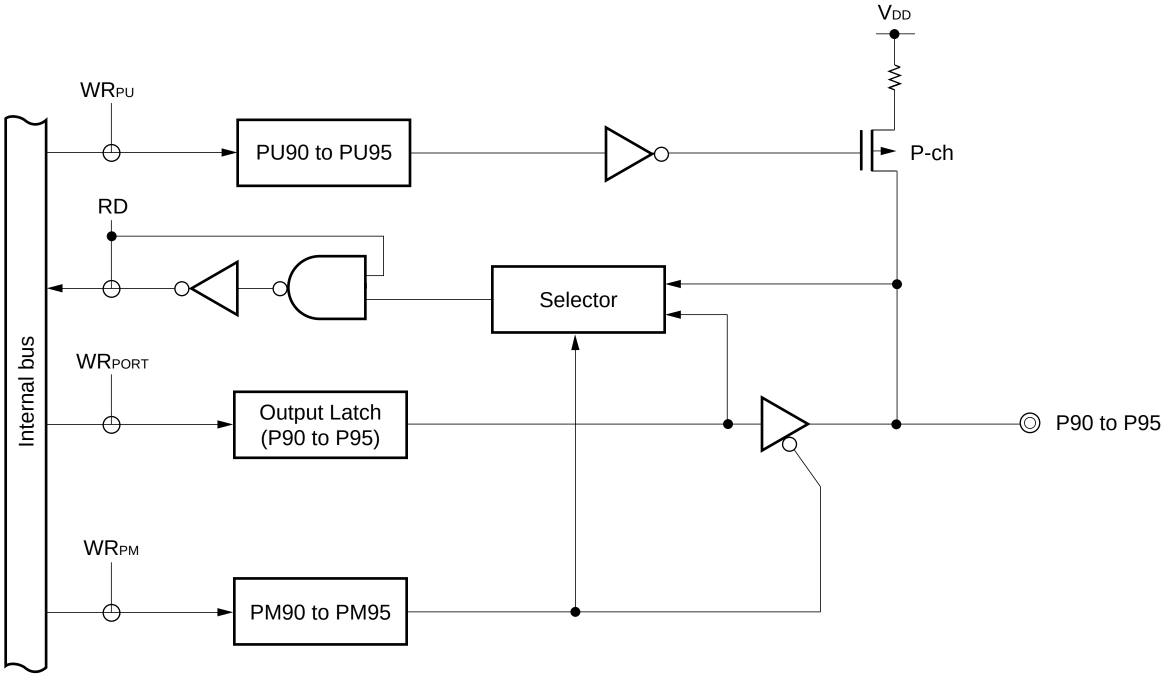
5.2.9 Port 9

This is an 6-bit input/output port with output latch. Input mode/output mode can be specified in 1-bit units with the port 6 mode register. A pull-up resistor can be connected to the P90 to P95 pins via the pull-up resistor option register 9, regardless of whether the input mode or output mode is specified.

$\overline{\text{RESET}}$ input sets port 9 to the input mode.

Figure 5-10 shows a block diagram of port 9.

Figure 5-10. Block Diagram of P90 to P95



PU : Pull-up resistor option register

PM : Port mode register

RD : Port 9 read signal

WR : Port 9 write signal

5.3 Control Registers

The following two types of registers control the ports.

- Port mode registers (PM0 to PM6, PM9)
- Pull-up resistor option registers (PU0 to PU3, PU9, PUO)

(1) Port mode registers (PM0 to PM6, PM9)

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

PM0 to PM6 and PM9 are set with a 1-bit or 8-bit memory manipulation instruction, respectively.

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

When port pins are used as alternate function pins, set the port mode registers and output latches according to Table 5-3.

Caution As port 0 has an alternative function as external interrupt request input, specifying the port function output mode and changing the output level sets the interrupt request flag. When the output mode is used, therefore, the interrupt mask flag should be set to 1 beforehand.

Table 5-3. Port Mode Register and Output Latch Settings When Using Alternate Functions

Pin Name	Alternate Function		PM××	P××
	Name	I/O		
P00	NMI	Input	1	×
P01 to P07	INTP0 to INTP6	Input	1	×
P10 to P15	RTP02 to RTP07	Output	0	0
P16	TO2	Output	0	0
P17	TO6	Output	0	0
P20	RxD	Input	1	×
P21	TxD	Output	0	0
P25	SI	Input	1	×
P26	SO	Output	0	0
P27	$\overline{\text{SCK}}$	Input	1	×
		Output	0	0
P30	TO0	Output	0	0
P31	TO1	Output	0	0
P32 to P37	RTP12 to RTP17	Output	0	0
P70 to P77	ANI0, ANI1	Input	—	

Remark × : don't care (setting is not required)

— : Port mode register and output latch do not exist

PM×× : Port mode register

P×× : Port output latch

Figure 5-11. Port Mode Register Format

Address: 0FF20H to 0FF26H, 0FF29H After Reset: FFH R/W

Symbol	7	6	5	4	3	2	1	0
PM0	PM07	PM06	PM05	PM04	PM03	PM02	PM01	PM00
PM1	PM17	PM16	PM15	PM14	PM13	PM12	PM11	PM10
PM2	PM27	PM26	PM25	1	1	1	PM21	PM20
PM3	PM37	PM36	PM35	PM34	PM33	PM32	PM31	PM30
PM4	PM47	PM46	PM45	PM44	PM43	PM42	PM41	PM40
PM5	PM57	PM56	PM55	PM54	PM53	PM52	PM51	PM50
PM6	PM67	PM66	PM65	PM64	PM63	PM62	PM61	PM60
PM9	1	1	PM95	PM94	PM93	PM92	PM91	PM90

PMxn	Pxn Pin I/O Mode Specification
	x = 0, 1, 3 to 6: n = 0 to 7
	x = 2: n = 0, 1, 5 to 7
	x = 9: n = 0 to 5

0	Output mode (output buffer ON)
1	Input mode (output buffer OFF)

★ (2) Pull-up resistor option registers (PU0 to PU3, PU9, PUO)

These registers are used to set whether or not to use an internal pull-up resistor at each port in 1-bit or 8-bit units. PUn (n = 0 to 3, 9) can specify the pull-up resistor connection of each port pin, whether in input mode or in output mode. PUO can specify the pull-up resistor connection of ports 4, 5, and 6 only in input mode. Pull-up resistors are connected irrespective of whether an alternate function is used.

These registers are set by a 1-bit or 8-bit memory manipulation instruction.

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

Cautions 1. Particular care is required with the circuit configuration when connecting a pull-up resistor, because of variations in the resistance values of pull-up resistors.

2. Port 7 does not incorporate a pull-up resistor.

Figure 5-12. Pull-Up Resistor Option Register Format

Address: 0FF30H to 0FF33H, 0FF39H After Reset: 00H R/W

Symbol	⑦	⑥	⑤	④	③	②	①	⓪
PU0	PU07	PU06	PU05	PU04	PU03	PU02	PU01	PU00
PU1	PU17	PU16	PU15	PU14	PU13	PU12	PU11	PU10
PU2	PU27	PU26	PU25	0	0	0	PU21	PU20
PU3	PU37	PU36	PU35	PU34	PU33	PU32	PU31	PU30
PU9	0	0	PU95	PU94	PU93	PU92	PU91	PU90

PUxn	Pxn Pin Pull-Up Resistor Specification
	(x = 0, 1, 3: n = 0 to 7
	x = 2: n = 0, 1, 5 to 7
	x = 9: n = 0 to 5)

0	No pull-up resistor connection
1	Pull-up resistor connection

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

Symbol	7	⑥	⑤	④	3	2	1	0
PUO	0	PUO6	PUO5	PUO4	0	0	0	0

PUOn	Port n Pull-Up Resistor Specification (n = 4 to 6)
0	No pull-up resistor connection
1	Pull-up resistor connection

5.4 Operations

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

5.4.1 Writing to input/output port

(1) Output mode

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

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

(2) Input mode

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

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

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

5.4.2 Reading from input/output port

(1) Output mode

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

(2) Input mode

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

5.4.3 Operations on input/output port

(1) Output mode

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

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

(2) Input mode

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

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

CHAPTER 6 REAL-TIME OUTPUT FUNCTIONS

6.1 Functions

The real-time output function transfers preset data in the real-time output buffer register to the output latch by hardware synchronized to the generation of a timer interrupt or an external interrupt and outputs it off the chip. Also, the pins for output off the chip are called the real-time output port. The real-time output port has two channels; RTP0 and RTP1.

Since jitter-free signals can be output by using the real-time output port, the operation is optimized for the control such as stepping motors.

The port mode or real-time output mode is selectable in 1-bit units.

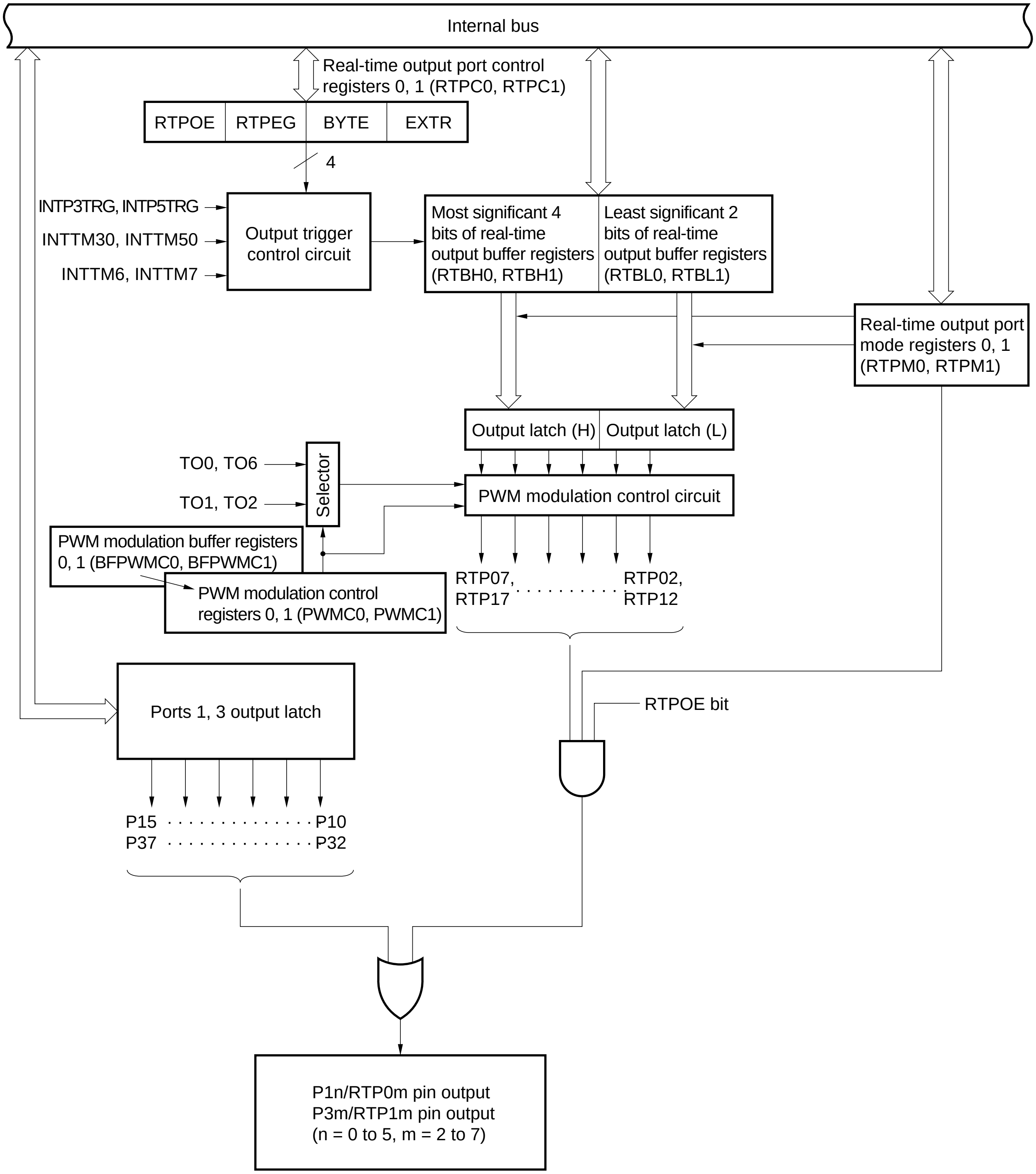
6.2 Configuration

The real-time output port consists of the following hardware.

Table 6-1. Configuration of Real-Time Output Ports 0, 1

Item	Structure
Registers	Real-time output buffer registers 0, 1 (RTBL0, RTBL1, RTBH0, RTBH1) PWM modulation buffer registers 0, 1 (BFPWMC0, BFPWMC1)
Control registers	Real-time output port mode registers 0, 1 (RTPM0, RTPM1) Real-time output port control registers 0, 1 (RTPC0, RTPC1) PWM modulation control registers 0, 1 (PWMC0, PWMC1)

★ **Figure 6-1. Block Diagram of Real-Time Output Port**


Internal bus
Real-time output port control registers 0, 1 (RTPC0, RTPC1)
RTPOE
RTPEG
BYTE
EXTR
4
INTP3TRG, INTP5TRG
INTTM30, INTTM50
INTTM6, INTTM7
Output trigger control circuit
Most significant 4 bits of real-time output buffer registers (RTBH0, RTBH1)
Least significant 2 bits of real-time output buffer registers (RTBL0, RTBL1)
Real-time output port mode registers 0, 1 (RTPM0, RTPM1)
Output latch (H)
Output latch (L)
TO0, TO6
TO1, TO2
Selector
PWM modulation control circuit
PWM modulation buffer registers 0, 1 (BFPWMC0, BFPWMC1)
PWM modulation control registers 0, 1 (PWMC0, PWMC1)
RTP07, RTP17
RTP02, RTP12
Ports 1, 3 output latch
RTPOE bit
P15
P10
P37
P32
P1n/RTP0m pin output
P3m/RTP1m pin output
(n = 0 to 5, m = 2 to 7)

- **Real-time output buffer registers 0, 1 (RTBL0, RTBH0, RTBL1, RTBH1)**

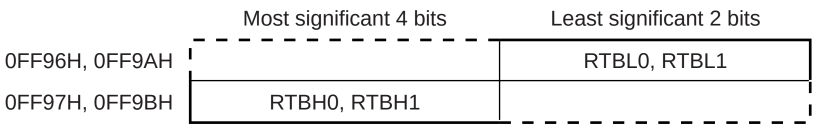
  These 4-bit registers save the output data beforehand. RTBL0, RTBH0, RTBL1, RTBH1 are mapped to independent addresses in the special function register (SFR) as shown in Figure 6-2.

  When the 4 bits × 1 channel and 2 bits × 1 channel operating mode is specified, RTBL0, RTBH0, RTBL1, RTBH1 can be independently set with data. In addition, if the addresses of both RTBL0, RTBH0, RTBL1, RTBH1 are specified, the data in both registers can be read in a batch.

  When the 6 bits × 1 channel operating mode is specified, writing 6-bit data to either RTBL0, RTBH0, RTBL1, RTBH1 can set data in either register. In addition, if the addresses of either RTBL0, RTBH0, RTBL1, RTBH1 are specified, the data in both can be read in a batch.

  Table 6-2 lists the operations for manipulating RTBL0, RTBH0, RTBL1, RTBH1.

Figure 6-2. Configuration of Real-Time Output Buffer Registers 0, 1



	Most significant 4 bits	Least significant 2 bits
0FF96H, 0FF9AH		RTBL0, RTBL1
0FF97H, 0FF9BH	RTBH0, RTBH1	

Table 6-2. Operation for Manipulating Real-Time Output Buffer Registers 0, 1

Operating Mode	Manipulated Register	Reading [Note 1]		Writing [Note 2]	
		Most significant 4 bits	Least significant 2 bits	Most significant 4 bits	Least significant 2 bits
4 bits × 1 channel	RTBL0, RTBL1	RTBH0, RTBH1	RTBL0, RTBL1	Invalid	RTBL0, RTBL1
2 bits × 1 channel	RTBH0, RTBH1	RTBH0, RTBH1	RTBL0, RTBL1	RTBH0, RTBH1	Invalid
6 bits × 1 channel	RTBL0, RTBL1	RTBH0, RTBH1	RTBL0, RTBL1	RTBH0, RTBH1	RTBL0, RTBL1
	RTBH0, RTBH1	RTBH0, RTBH1	RTBL0, RTBL1	RTBH0, RTBH1	RTBL0, RTBL1

Notes 1. Only the bits specified in the real-time output port mode can be read. When the bits set in the bits set in the port mode are read, zeros are read.

2. After setting the real-time output port, set the output data in RTBL0, RTBH0, RTBL1, RTBH1 until the real-time output trigger is generated.

6.3 Real-Time Output Port 0 (RTP0)

6.3.1 Control registers

The real-time output port 0 is controlled by the following three registers.

- Real-time output port mode register 0 (RTPM0)
- Real-time output port control register 0 (RTPC0)
- PWM modulation control register 0 (PWMC0)

(1) Real-time output port mode register 0 (RTPM0)

This register sets the real-time output port mode and port mode selections in bit-wise.
RTPM0 is set by a 1-bit or 8-bit memory manipulation instruction.
$\overline{\text{RESET}}$ input sets RTPM0 to 00H.

Figure 6-3. Format of Real-Time Output Port Mode Register 0 (RTPM0)

Address: 0FF98H After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
RTPM0	RTPM07	RTPM06	RTPM05	RTPM04	RTPM03	RTPM02	0	0

RTPM0m	Real-time Output Port 0 Selection (m = 2 to 7)
0	Port mode
1	Real-time output mode

Caution When used as a real-time output port 0, set the port for real-time output in the output mode.

(2) Real-time output port control register 0 (RTPC0)

This register sets the operating mode and output trigger of the real-time output port 0.

Table 6-3 shows the relationships between the operating modes and output triggers of the real-time output port 0.

RTPC0 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets RTPC0 to 00H.

★ **Figure 6-4. Format of Real-Time Output Port Control Register 0 (RTPC0)**

Address: 0FF99H After Reset: 00H R/W

Symbol	⑦	⑥	⑤	④	3	2	1	0
RTPC0	RTPOE0	RTPEG0	BYTE0	EXTR0	0	0	0	0

RTPOE0	Real-time Output Port 0 Operation Control
0	Operation disabled
1	Operation enabled Note

RTPEG0	INTP3TRG Valid Edge Setting
0	Falling edge
1	Rising edge

BYTE0	Real-time Output Port 0 Operation Mode
0	4 bits × 1 channel (upper side), 2 bits × 1 channel (lower side)
1	8 bits × 1 channels

EXTR0	Real-time Output Control by INTP3TRG
0	Do not set INTP3TRG as a real-time output trigger
1	Set INTP3TRG as a real-time output trigger

Note When real-time output operation is enabled (RTPOE0 = 1), the values of the real-time output buffer registers H0 and L0 (RTBH0, RTBL0) are transferred to the output latch of real-time output port 0.

Table 6-3. Operating Modes and Output Triggers of Real-Time Output Port 0

<table>
<tr><th>BYTE0</th><th>EXTR0</th><th>Operating Mode</th><th>RTBH0 → Port Output</th><th>RTBL0 → Port Output</th></tr>
<tr><td>0</td><td>0</td><td rowspan="2">4 bits × 1 channel
2 bits × 1 channel</td><td>INTTM6 (internal)</td><td>INTTM30 (internal)</td></tr>
<tr><td>0</td><td>1</td><td colspan="2">INTTM30 (internal)</td></tr>
<tr><td>1</td><td>0</td><td rowspan="2">6 bits × 1 channel</td><td>INTTM30 (internal)</td><td>INTP3TRG (external)</td></tr>
<tr><td>1</td><td>1</td><td colspan="2">INTP3TRG (external)</td></tr>
</table>

(3) PWM modulation control register 0 (PWMC0)

PWMC0 is a register which performs real-time output modulation control and specifies the output level of real-time output port 0.

PWMC0 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets PWMC0 to 00H.

★ **Figure 6-5. Format of PWM Modulation Control Register 0 (PWMC0)**

Address: 0FFA4H After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
PWMC0	PWMC07	PWMC06	PWMC05	PWMC04	PWMC03	PWMC02	INVRTP0	SELPWM0

PWMC0n[Note 1]	PWM Modulation Operation Specification (n = 2 to 7)
0	Disable
1	Enable

INVRTP0[Note 2]	Real-time Output Port 0 Output Level Specification
0	Inversion disabled
1	Inversion enabled

SELPWM0[Note 3]	PWM Signal Specification
0	Timer output (TO0) of 16-bit timer/counter 0 (TM0)
1	Timer output (TO1) of 16-bit timer/counter 1 (TM1)

Notes **1.** PWMC0n (bits 2 to 7) specifies enable/disable for PWM modulation operation of real-time output pins. It is possible to set PWMC0n for each pin independently.

2. INVRTP0 (bit 1) enables/disables inversion operation of output level in real time. When INVRTP0 is set, the inversion level of the value being set to real-time output buffer register 0 (RTBH0, RTBL0) is output. When performing PWM modulation, the output level after PWM modulation is also inverted.

3. SELPWM0 (bit 0) specifies the PWM signal. At real-time output port 0 (RTP0), TO0 (timer output of TM0) and TO1 (timer output of TM1) output PWM signals.

(4) PWM modulation buffer register 0 (BFPWMC0)

BFPWMC0 is a register that is synchronized to the real-time output transfer signal, and transfers data (upper 6 bits only) to PWMC0.

BFPWMC0 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets BFPWMC0 to 00H.

Figure 6-6. Format of PWM Modulation Buffer Register 0 (BFPWMC0)

Address: 0FFA6H After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
BFPWMC0	BFPWMC07	BFPWMC06	BFPWMC05	BFPWMC04	BFPWMC03	BFPWMC02	0	0

Transfer of data (upper 6 bits) from BFPWMC0 to PWM modulation control register 0 (PWMC0) is performed in synchronization with transfer of data from real-time output buffer register 0 to the output latch. There are the following transfer operations.

- From register RTBH0 to the output latch (H): Transfers the upper 4 bits.
- From register RTBL0 to the output latch (L): Transfers the lower 2 bits.
- Transfer of upper and lower 6 bits at the same time

The transfer operation is specified by bits 4 and 5 (EXTR0, BYTE0) of real-time output port control register 0 (RTPC0). It is specified as shown in Table 6-4.

Table 6-4. Data Transfer from PWM Modulation Buffer Register 0 (BFPWMC0) to PWM Modulation Control Register 0 (PWMC0)

BYTE0	EXTR0	Transfer of upper 4 bits	Transfer of lower 2 bits
0	0	4-bit transfer from (BFPWMC07 to BFPWMC04) to (PWMC07 to PWMC04) at the INTTM6 transfer trigger	2-bit transfer from BFPWMC03 and BFPWMC02 to PWMC03 and PWMC02 at the INTTM30 transfer trigger
0	1	6-bit transfer from (BFPWMC07 to BFPWMC02) to (PWMC07 to PWMC02) at the INTTM30 transfer trigger	
1	0	4-bit transfer from (BFPWMC07 to BFPWMC04) to (PWMC07 to PWMC04) at the INTTM30 transfer trigger	2-bit transfer from BFPWMC03 and BFPWNMC02 to PWMC03 and PWMC02 at the INTP transfer trigger
1	1	6-bit transfer from (BFPWMC07 to BFPWMC02) to (PWMC07 to PWMC02) at the INTP transfer trigger	

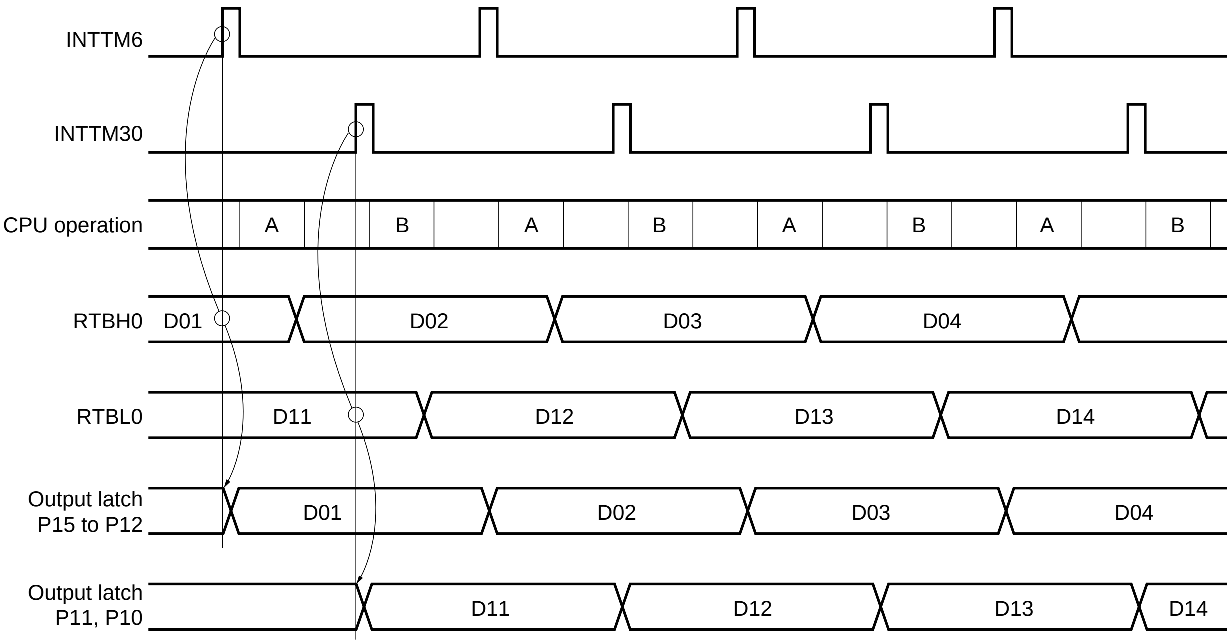
Caution Transferring data from BFPWMC0 to PWMC0 can only be performed for the upper 6 bits, and cannot be performed for the lower 2 bits.

6.3.2 Operation

When real-time output is enabled by bit 7 (RTPOE0) = 1 in the real-time output port control register 0 (RTPC0), data in the real-time output buffer register 0 (RTBH0, RTBL0) are transferred to the output latch synchronized to the generation of the selected transfer trigger (set by EXTR0 and BYTE0 **Note**). Based on the setting of the real-time output port mode register 0 (RTPM0), only the transferred data for the bits specified in the real-time output port are output from bits RTP02 to RTP07. A port set in the port mode by RTPM0 can be used as a general-purpose I/O port.

Note EXTR0: Bit 4 of the real-time output port control register 0 (RTPC0)
BYTE0: Bit 5 of the real-time output port control register 0 (RTPC0)

Figure 6-7. Example of the Operation Timing of Real-Time Output Port 0 (EXTR0 = 0, BYTE0 = 0)



A: Software processing by INTTM6 (RTBH0 write)
B: Software processing by INTTM30 (RTBL0 write)

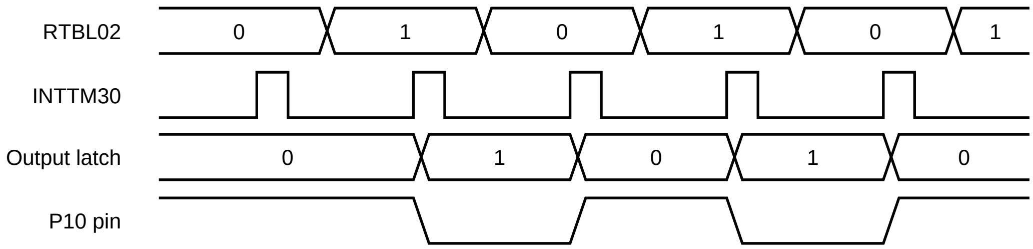
6.3.3 PWM modulation control

For real-time output port 0 (RTP0), PWM modulation, which takes the OR logic of the PWM signal (TO0 or TO1) in the real-time output, is possible. Also, by enabling inversion of the real-time output level, it is possible to generate a pulse waveform whose level is the inverted value of the value set in registers RTBH0 and RTBL0.

(1) Inversion of RTP0 real-time output

By setting bit 1 (INVRTP0) of PWM modulation control register 0 (PWMC0) to “1”, level whose value is the inverted value of the value set in the real-time output buffer registers (RTBH0, RTBL0) is output. An example of the operation is shown in Figure 6-8.

Figure 6-8. Example of PWM Output Level Inversion (RTP0)



Remark INTTM30 is the transfer trigger.

(2) PWM Modulation Operation of RTP0

When pins P10 to P15 are used in the real-time output mode (RTP02 to RTP07 output mode), PWM modulation can be performed the output pattern of each pin. When PWM modulation is performed, the signal which takes the OR logic value between the signal transferred to the output latch from RTBH0 and RTBL0 and the PWM signal (TO0 or TO1), is output to pins P10 to P15. Also, it is possible to perform PWM modulation for each pin independently. An example of the PWM modulation operation is shown in Figure 6-9.

Figure 6-9. Example of PWM Modulation Operation (RTP0)

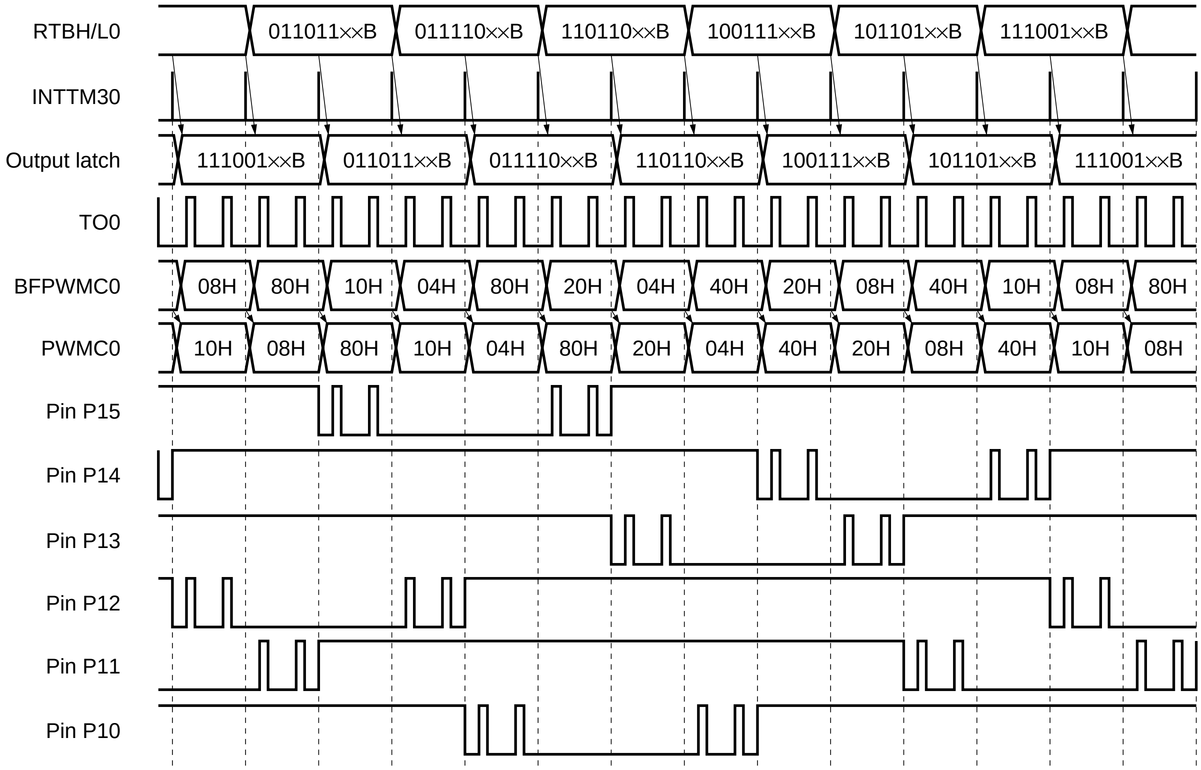
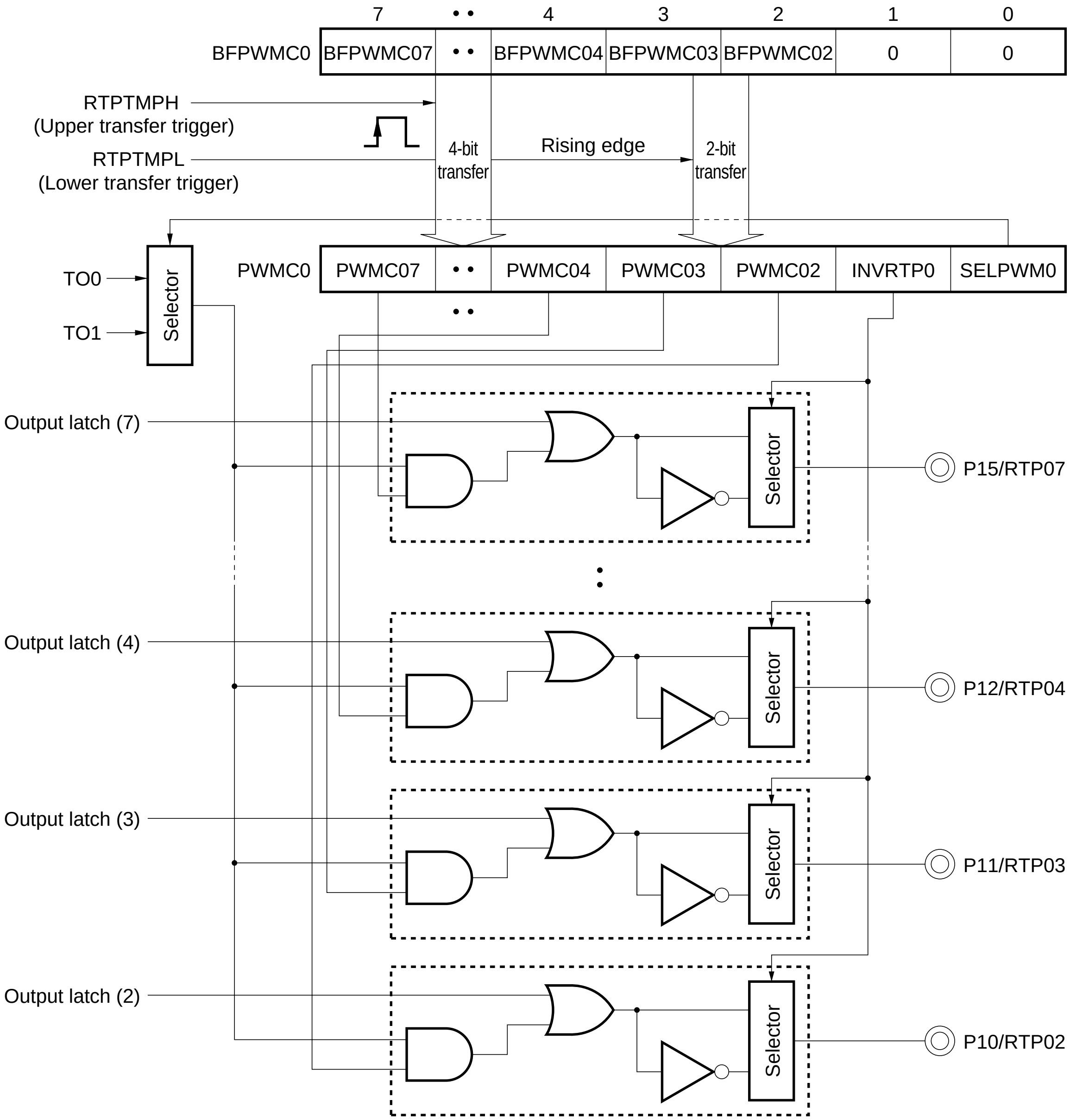


Figure 6-10. Configuration of the PWM Modulation Control Circuit (RTP0)



6.4 Real-Time Output Port 1 (RTP1)

6.4.1 Control registers

The real-time output port 1 is controlled by the following three registers.

- Real-time output port mode register 1 (RTPM1)
- Real-time output port control register 1 (RTPC1)
- PWM modulation control register 1 (PWMC1)

(1) Real-time output port mode register 1 (RTPM1)

This register sets the real-time output port mode and port mode selections bit-wise.

RTPM1 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets RTPM1 to 00H.

Figure 6-11. Format of Real-Time Output Port Mode Register 1 (RTPM1)

Address: 0FF9CH After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
RTPM1	RTPM17	RTPM16	RTPM15	RTPM14	RTPM13	RTPM12	0	0

RTPM0m	Real-time Output Port 1 Selection (m = 2 to 7)
0	Port mode
1	Real-time output mode

Caution When used as a real-time output port 1, set the port for real-time output to the output mode.

(2) Real-time output port control register 1 (RTPC1)

This register sets the operating mode and output trigger of the real-time output port 1.

Table 6-5 shows the relationships between the operating modes and output triggers of the real-time output port 1.

RTPC1 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets RTPC1 to 00H.

★ **Figure 6-12. Format of Real-Time Output Port Control Register 1 (RTPC1)**

Address: 0FF9BH After Reset: 00H R/W

Symbol	⑦	⑥	⑤	④	3	2	1	0
RTPC1	RTPOE1	RTPEG1	BYTE1	EXTR1	0	0	0	0

RTPOE1	Real-time Output Port 1 Operation Control
0	Operation disabled
1	Operation enabled Note

RTPEG1	INTP5TRG Valid Edge Setting
0	Falling edge
1	Rising edge

BYTE1	Real-time Output Port 1 Operation Mode
0	4 bits × 1 channel (upper side), 2 bits × 1 channel (lower side)
1	8 bits × 1 channels

EXTR1	Real-time Output Control by INTP5TRG
0	Do not set INTP5TRG as a real-time output trigger
1	Set INTP5TRG as a real-time output trigger

Note When real-time output operation is enabled (RTPOE1 = 1), the values of the real-time output buffer registers H0 and L0 (RTBH1, RTBL1) are transferred to the output latch of real-time output port 0.

Table 6-5. Operating Modes and Output Triggers of Real-Time Output Port 1

<table>
<tr><th>BYTE1</th><th>EXTR1</th><th>Operating Mode</th><th>RTBH1 → Port Output</th><th>RTBL1 → Port Output</th></tr>
<tr><td>0</td><td>0</td><td rowspan="2">4 bits × 1 channel
2 bits × 1 channel</td><td>INTTM7 (internal)</td><td>INTTM50 (internal)</td></tr>
<tr><td>0</td><td>1</td><td colspan="2">INTTM50 (internal)</td></tr>
<tr><td>1</td><td>0</td><td rowspan="2">6 bits × 1 channel</td><td>INTTM50 (internal)</td><td>INTP5TRG (external)</td></tr>
<tr><td>1</td><td>1</td><td colspan="2">INTP5TRG (external)</td></tr>
</table>

(3) PWM modulation control register 1 (PWMC1)

PWMC1 is a register which performs real-time output modulation control and specifies the output level of real-time output port 1.

PWMC1 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets PWMC1 to 00H.

★ **Figure 6-13. Format of PWM Modulation Control Register 1 (PWMC1)**

Address: 0FFA5H After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
PWMC1	PWMC17	PWMC16	PWMC15	PWMC14	PWMC13	PWMC12	INVRTP1	SELPWM1

PWMC1n Note 1	PWM Modulation Operation Specification (n = 2 to 7)
0	Disable
1	Enable

INVRTP1 Note 2	Real-Time Output Port 1 Output Level Specification
0	Inversion disabled
1	Inversion enabled

SELPWM1 Note 3	PWM Signal Specification
0	Timer output (TO2) of 16-bit timer/counter 2 (TM2)
1	Timer output (TO6) of 8-bit timer/counter 6 (TM6)

Notes **1.** PWMC1n (bits 2 to 7) specifies enable/disable for PWM modulation of real-time output pins. It is possible to set PWMC1n for each pin independently.

2. INVRTP1 (bit 1) enables/disables inversion of output level in real time. When INVRTP1 is set, the inversion level of the value being set to real-time output buffer register 1 (RTBH1, RTBL1) is output. When performing PWM modulation, the output level after PWM modulation is also inverted.

3. SELPWM1 (bit 0) specifies the PWM signal. At real-time output port 1 (RTP1), TO2 (timer output of TM2) and TO6 (timer output of TM6) output PWM signals.

(4) PWM modulation buffer register 1 (BFPWMC1)

BFPWMC1 is a register that is synchronized to the real-time output transfer signal, and transfers data (upper 6 bits only) to PWMC1.

BFPWMC1 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets BFPWMC1 to 00H.

Figure 6-14. Format of PWM Modulation Buffer Register 1 (BFPWMC1)

Address: 0FFA7H After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
BFPWMC1	BFPWMC17	BFPWMC16	BFPWMC15	BFPWMC14	BFPWMC13	BFPWMC12	0	0

Transfer of data (upper 6 bits) from BFPWMC1 to PWM modulation control register 1 (PWMC1) is performed in synchronization with transfer of data from real-time output buffer register 1 to the output latch. There are the following three transfer operations.

- From register RTBH1 to the output latch (H): Transfers the upper 4 bits.
- From register RTBL1 to the output latch (L): Transfers the lower 2 bits.
- Transfer of upper and lower 6 bits at the same time

The transfer operation is specified by bits 4 and 5 (EXTR1, BYTE1) of real-time output port control register 1 (RTPC1). It is specified as shown in Table 6-6.

Table 6-6. Data Transfer from PWM Modulation Buffer Register 1 (BFPWMC1) to PWM Modulation Control Register 1 (PWMC1)

BYTE1	EXTR1	Transfer of upper 4 bits	Transfer of lower 2 bits
0	0	4-bit transfer from (BFPWMC17 to BFPWMC14) to (PWMC17 to PWMC14) at the INTTM7 transfer trigger	2-bit transfer from BFPWMC13 and BFPWMC12 to PWMC13 and PWMC12 at the INTTM50 transfer trigger
0	1	6-bit transfer from (BFPWMC17 to BFPWMC12) to (PWMC17 to PWMC12) at the INTTM50 transfer trigger	
1	0	4-bit transfer from (BFPWMC17 to BFPWMC14) to (PWMC17 to PWMC14) at the INTTM50 transfer trigger	2-bit transfer from BFPWMC13 and BFPWMC12 to PWMC13 and PWMC12 at the INTP5 transfer trigger
1	1	6-bit transfer from (BFPWMC17 to BFPWMC12) to (PWMC17 to PWMC12) at the INTP5 transfer trigger	

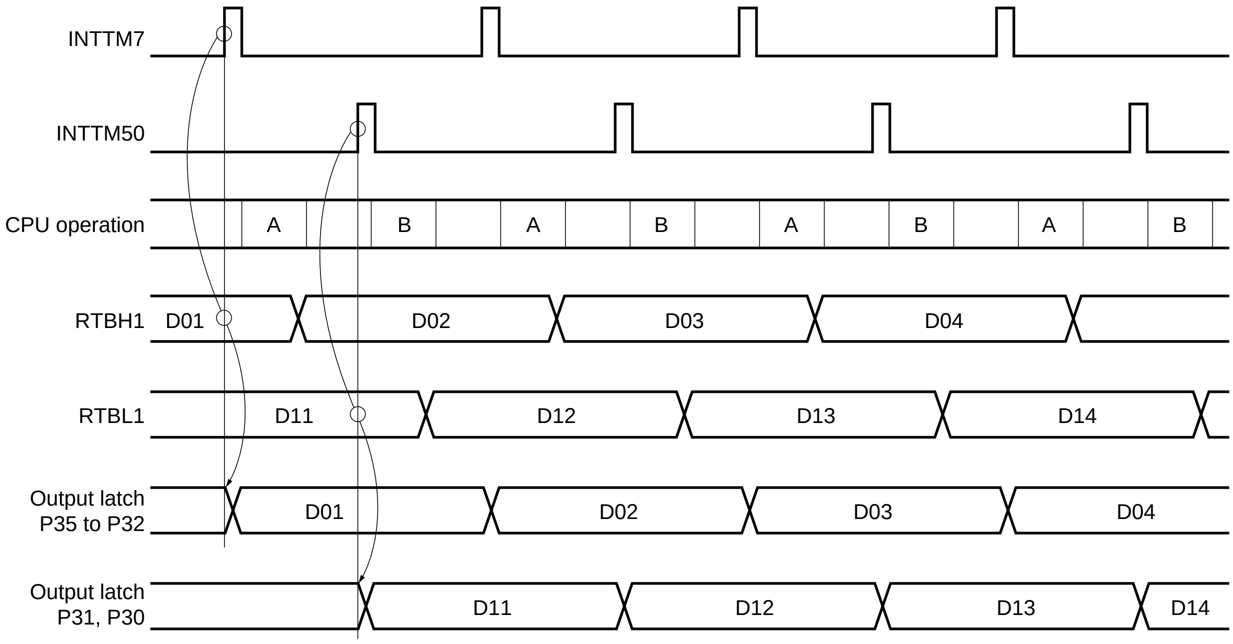
Caution Transferring data from BFPWMC1 to PWMC1 can only be performed for the upper 6 bits, and cannot be performed for the lower 2 bits.

6.4.2 Operation

When real-time output is enabled by bit 7 (RTPOE1) = 1 in the real-time output port control register 1 (RTPC1), data in the real-time output buffer register 1 (RTBH1, RTBL1) are transferred to the output latch synchronized to the generation of the selected transfer trigger (set by EXTR1 and BYTE1 **Note**). Based on the setting of the real-time output port mode register 1 (RTPM1), only the transferred data for the bits specified in the real-time output port are output from bits RTP0 to RTP7. A port set in the port mode by RTPM1 can be used as a general-purpose I/O port.

Note EXTR1: Bit 4 of the real-time output port control register 1 (RTPC1)
BYTE1: Bit 5 of the real-time output port control register 1 (RTPC1)

Figure 6-15. Example of the Operation Timing of Real-Time Output Port 1 (EXTR1 = 0, BYTE1 = 0)



A: Software processing by INTTM7 (RTBH1 write)
B: Software processing by INTTM50 (RTBL1 write)

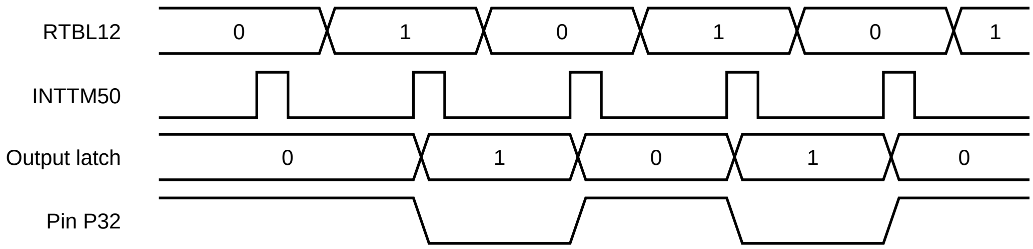
6.4.3 PWM modulation control

For real-time output port 1 (RTP1), PWM modulation, which takes the OR logic of the PWM signal (TO2 or TO6) in the real-time output, is possible. Also, by enabling inversion of the real-time output level, it is possible to generate a pulse waveform whose level is the inverted value of the value set in registers RTBH1 and RTBL1.

(1) Inversion of RTP1 real-time output

By setting bit 1 (INVRTP1) of PWM modulation control register 1 (PWMC1) to “1”, level whose value is the inverted value of the value set in the real-time output buffer registers (RTBH1, RTBL1) is output. An example of the operation is shown in Figure 6-16.

Figure 6-16. Example of PWM Output Level Inversion (RTP1)



Remark INTTM50 is the transfer trigger.

(2) PWM Modulation Operation of RTP1

When pins P32 to P37 are used in the real-time output mode (RTP12 to RTP17 output mode), PWM modulation can be performed to the output pattern of each pin. When PWM modulation is performed, the signal which takes the OR logic value between the signal transferred to the output latch from RTBH1 and RTBL1 and the PWM signal (TO2 or TO6), is output to pins P32 to P37. Also, it is possible to perform PWM modulation for each pin independently. An example of the PWM modulation operation is shown in Figure 6-17.

Figure 6-17. Example of PWM Modulation Operation (RTP1)

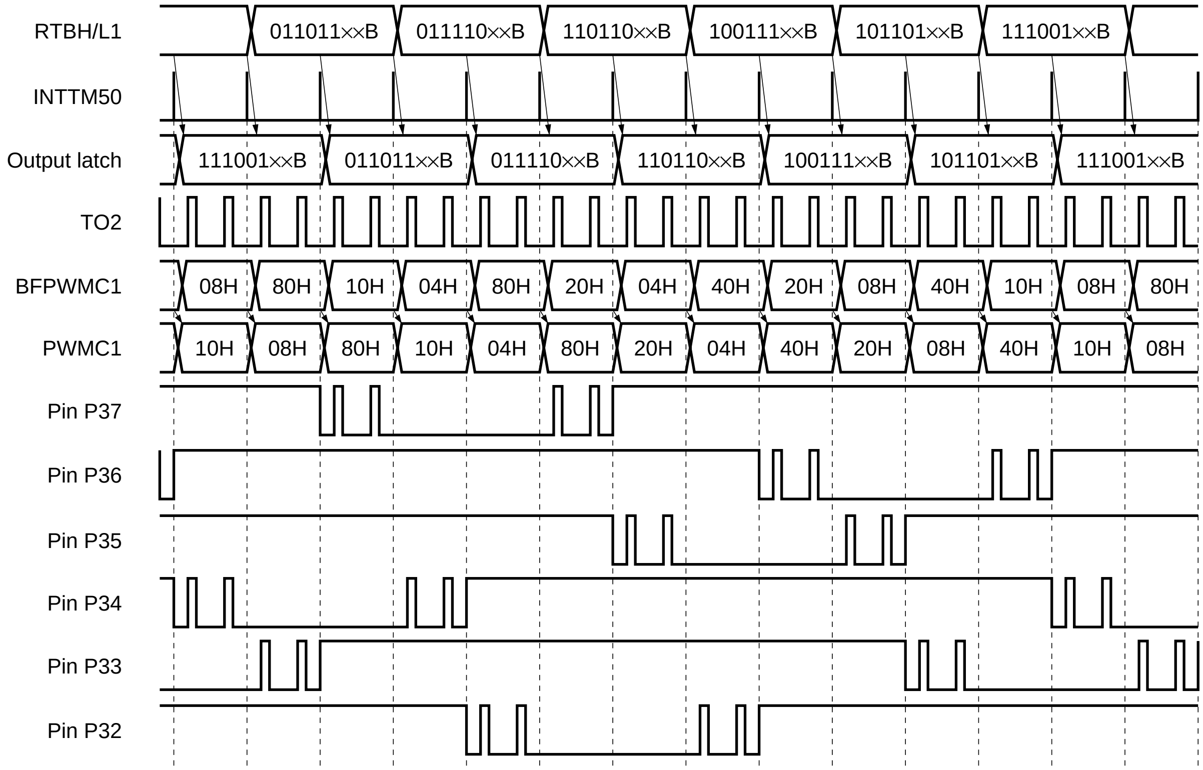
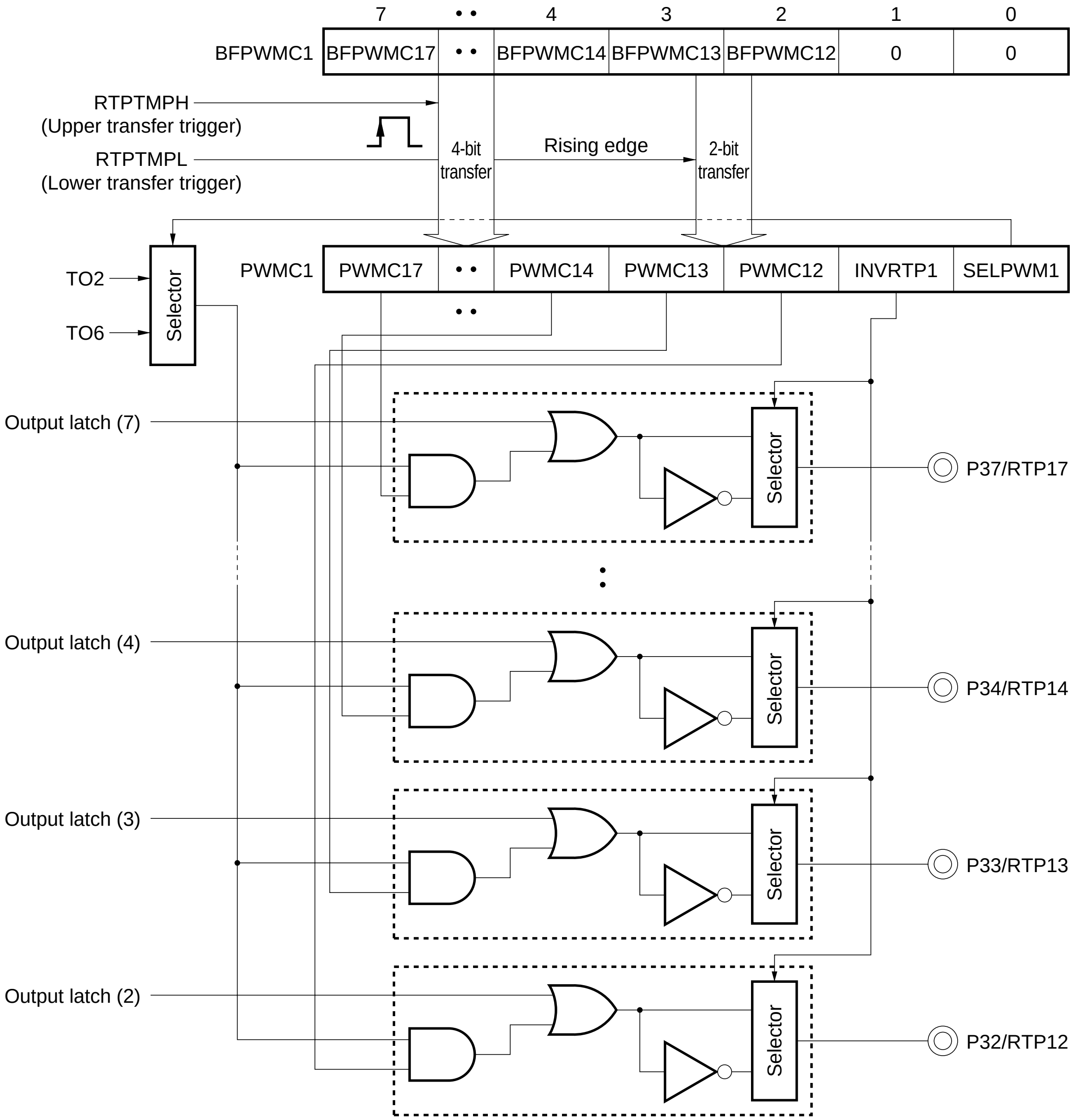


Figure 6-18. Configuration of the PWM Modulation Control Circuit (RTP1)



6.5 Using this Function

(1) Disabling the real-time output operation

Set bit 7 (RTPOE0, RTPOE1) = 0 in the real-time output port control registers 0, 1 (RTPC0, RTPC1).

(2) Initial Setting

- Sets the initial value in the output latch of the port.
- Specifies the real-time output port mode or port mode in bit units.
  Sets real-time output port mode registers 0 and 1 (RTPM0, RTPM1).
- Selects the transfer trigger.
  Sets bits 4, 5 and 6 of RTPC0 and RTPC1 (EXTR0, EXTR1, BYTE0, BYTE1, RTPEG0, RTPEG1).
- Sets initial values for real-time output buffer registers 0 and 1 (RTBL0, RTBH0, RTBL1, RTBH1) that are the same as those of the output latch of the port.

(3) Enabling real-time output operation

RTPOE0, RTPOE1 = 1

When operation is enabled, the values of RTBL0, RTBH0, RTBL1 and RTBH1 are latched to the output latch of real-time output ports 0 and 1 (RTP0, RTP1).

(4) Until the selected transfer trigger is generated, the output latch of the port is “0”, and sets the next output in RTBL0, RTBH0, RTBL1 and RTBH1.

The values output by the real-time output operation are “OR” values of the port’s output latch and real-time output ports 0 and 1 (see **Figure 6-1 Block Diagram of Real-Time Output Port**). From the time the real-time output operation is enabled until the transfer trigger is generated, the output latch of the port should be set to “0”.

(5) By interrupt processing of the selected trigger, the next real-time output values are set in order for RTBL0, RTBH0, RTBL1 and RTBH1.

6.6 Cautions

(1) For the initial setting, set bit 7 (RTPOE0, RTPOE1) in the real-time output port control registers 0 and 1 (RTPC0, RTPC1) to 0 to disable the real-time output operation.

(2) When the real-time output operation is disabled (RTPOE0, RTPOE1 = 0) once, always set the same initial value as in the output latch in the real-time output buffer registers (RTBL0, RTBH0, RTBL1, RTBH1) before enabling real-time output (RTPOE0, RTPOE1 = 0 → 1).

(3) Operation is not guaranteed when there is contention of the following signals.

- Contention between the real-time output port mode/port mode switch (bit 7 (RTPOE0, RTPOE1) of real-time output port control register 0 and 1 (RTPC0, TRTPC1)) and the real-time output transfer trigger.
- Contention between writing to real-time output buffer registers 0 and 1 (RTBH0, RTBL0, RTBH1, RTBL1) in the real-time output port mode and the real-time output transfer trigger.
- Contention between writing to PWM modulation buffer registers 0 and 1 (BFPWMC0, BFPWMC1) in the real-time output port mode and the real-time output transfer trigger.
- ★ Contention between reading from PWM modulation control registers 0 and 1 (PWMC0, PWMC1) in the real-time output port mode and the real-time output transfer trigger.

(4) Before switching the port mode and real-time output port mode (bit 7 (RTPOE0, RTPOE1) of RTPC0 and RTPC1), initial values should be set for real-time output buffer registers 0 and 1 (RTBH0, RTBL0, RTBH1 and RTBL1), PWM modulation control registers 0 and 1 (PWMC0, PWMC1), and PWM modulation buffer registers 0 and 1 (BFPWMC0, BFPWMC1).

(5) The real-time output latch cannot be read or written directly.

(6) RTP02 to RTP07 and RTP12 to RTP17 are taken to be the external-pin output of the "OR" between output latches of the output ports for each bit.

(7) During real-time output operation, the output latches of pins P10 to P15 and P32 to P37 should be set to "0".

(8) While setting the real-time output port mode, writing to PWM modulation control register 0 and 1 (PWMC0 and PWMC1) is prohibited except for setting initial values.

(9) If pins P10/RTP02 to P15/RTP07 and P32/RTP12 to P37/RTP17 are used as output ports (P10 to P15, P32 to P37), bits 7 (RTPOE0, RTPOE1) of RTPC0 and RTPC1 should be set to "0" (prohibiting real-time output), bits 2 to 7 (PWMC02 to PWMC07) of PWMC0 and PWMC1 should be set to "0" (prohibiting PWM modulation), and bits 1 (INVRTP0, INVRTP1) of BFPWMC0 and BFPWMC1 should be set to "0" (prohibiting inversion of the real-time output level).

[MEMO]

CHAPTER 7 TIMER/COUNTER OVERVIEW

There are six units of on-chip 16-bit timer/counters and two units of on-chip 8-bit timer/counters.
Since a total of 21 interrupt requests is supported, these timer/counters can function as eight units of timer/counters.

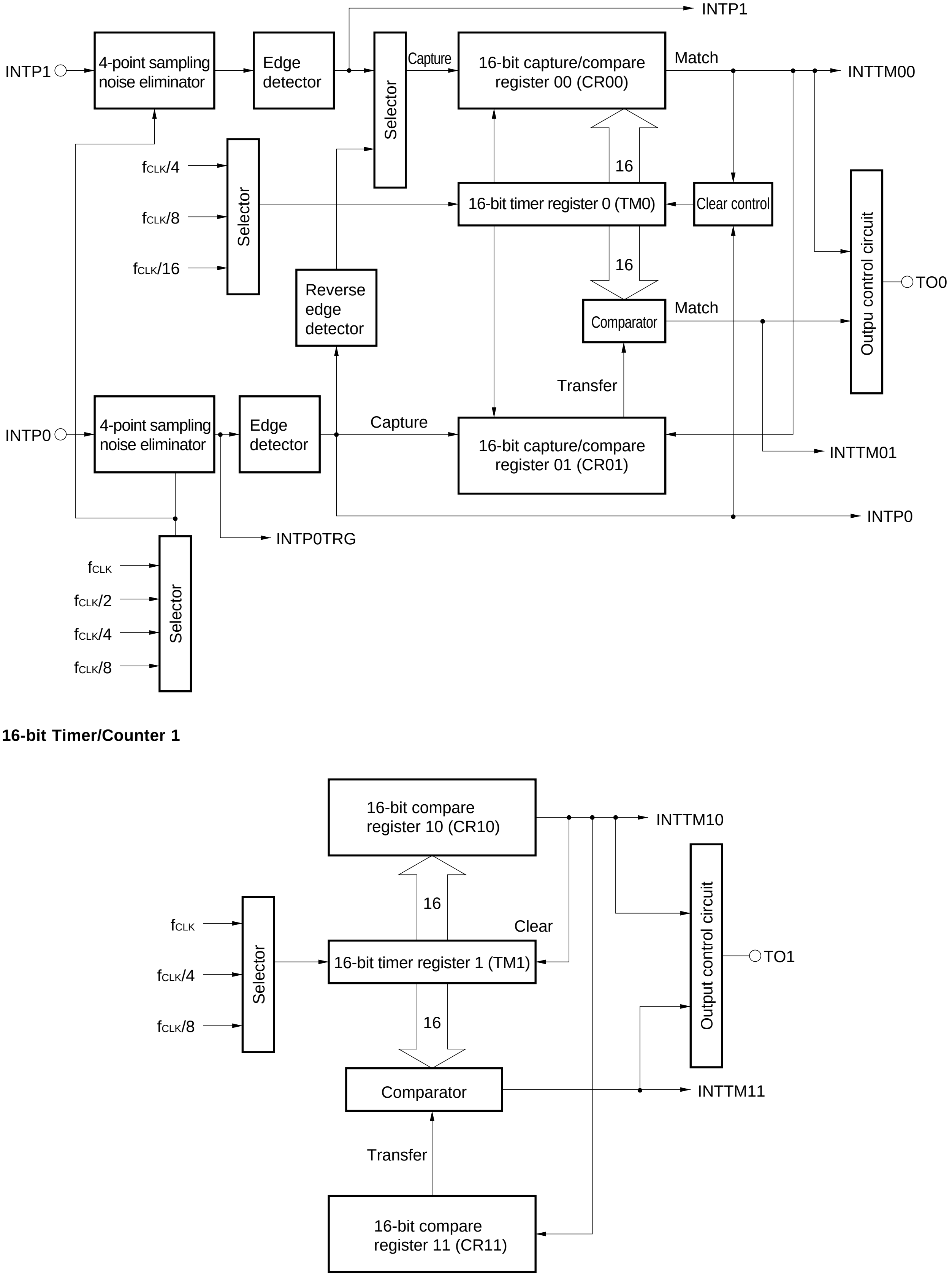
Table 7-1. Timer/Counter Operation

Item	Name	16-bit Timer/ counter 0	16-bit Timer/ counter 1	16-bit Timer/ counter 2	16-bit Timer/ counter 3	16-bit Timer/ counter 4	16-bit Timer/ counter 5
Count width	8 bits	—	—	—	—	—	—
	16 bits	√	√	√	√	√	√
Operating mode	Interval timer	1 ch	1 ch	1 ch	1 ch	1 ch	1 ch
	External event counter	—	—	—	—	—	—
Function	Timer output	1 ch	1 ch	1 ch	—	—	—
	PWM output	√	√	√	—	—	—
	Pulse width measurement	2 inputs	—	—	—	3 inputs	2 inputs
	No. of interrupt requests	4	2	2	2	5	4

Item	Name	8-bit Timer/ counter 6	8-bit Timer/ counter 7
Count width	8 bits	√	√
	16 bits	√	
Operating mode	Interval timer	1 ch	1 ch
	External event counter	—	—
Function	Timer output	1 ch	—
	PWM output	√	—
	Pulse width measurement	—	—
	No. of interrupt requests	1	1

Figure 7-1. Timer/Counter Block Diagram (1/5)

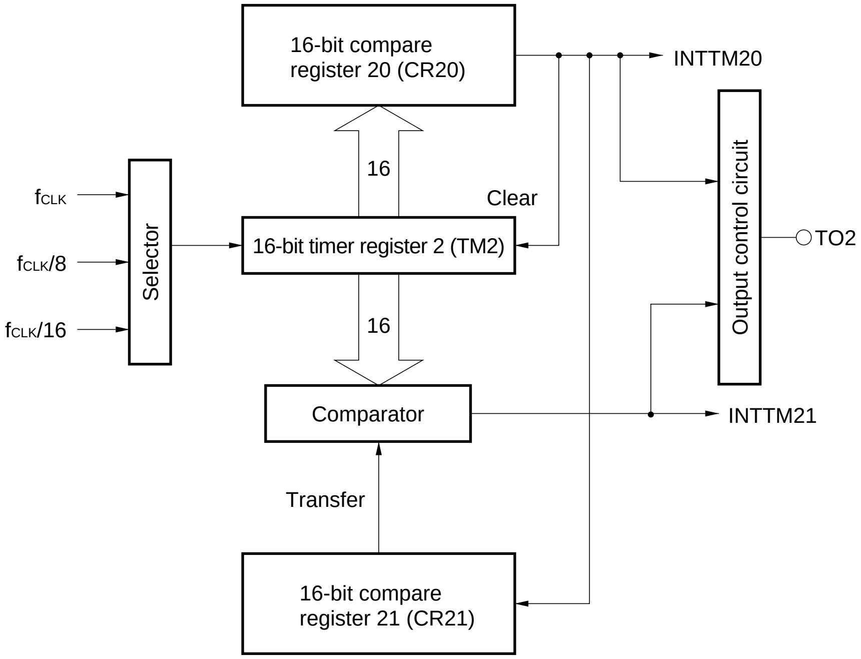
16-bit Timer/Counter 0



16-bit Timer/Counter 1

Figure 7-1. Timer/Counter Block Diagram (2/5)

16-bit Timer/Counter 2


16-bit compare register 20 (CR20)
INTTM20
16
Clear
Output control circuit
fCLK
fCLK/8
fCLK/16
Selector
16-bit timer register 2 (TM2)
TO2
16
Comparator
INTTM21
Transfer
16-bit compare register 21 (CR21)

16-bit Timer/Counter 3

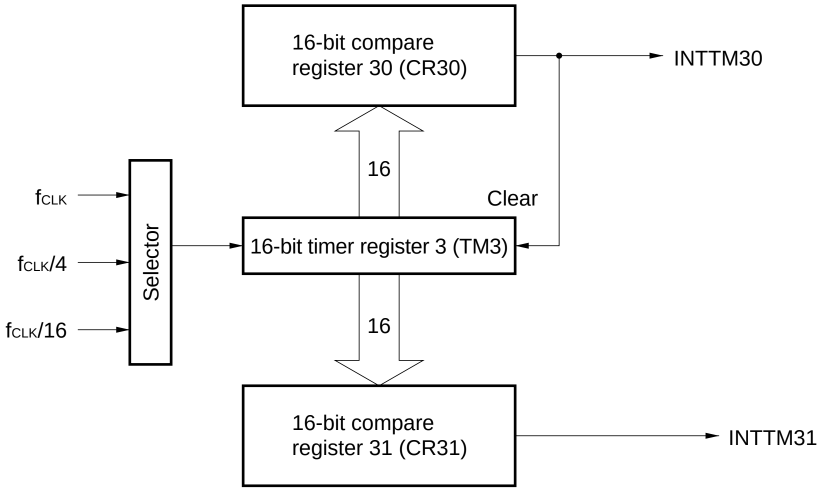

16-bit compare register 30 (CR30)
INTTM30
16
Clear
fCLK
fCLK/4
fCLK/16
Selector
16-bit timer register 3 (TM3)
16
16-bit compare register 31 (CR31)
INTTM31

Figure 7-1. Timer/Counter Block Diagram (3/5)

16-bit Timer/Counters 4

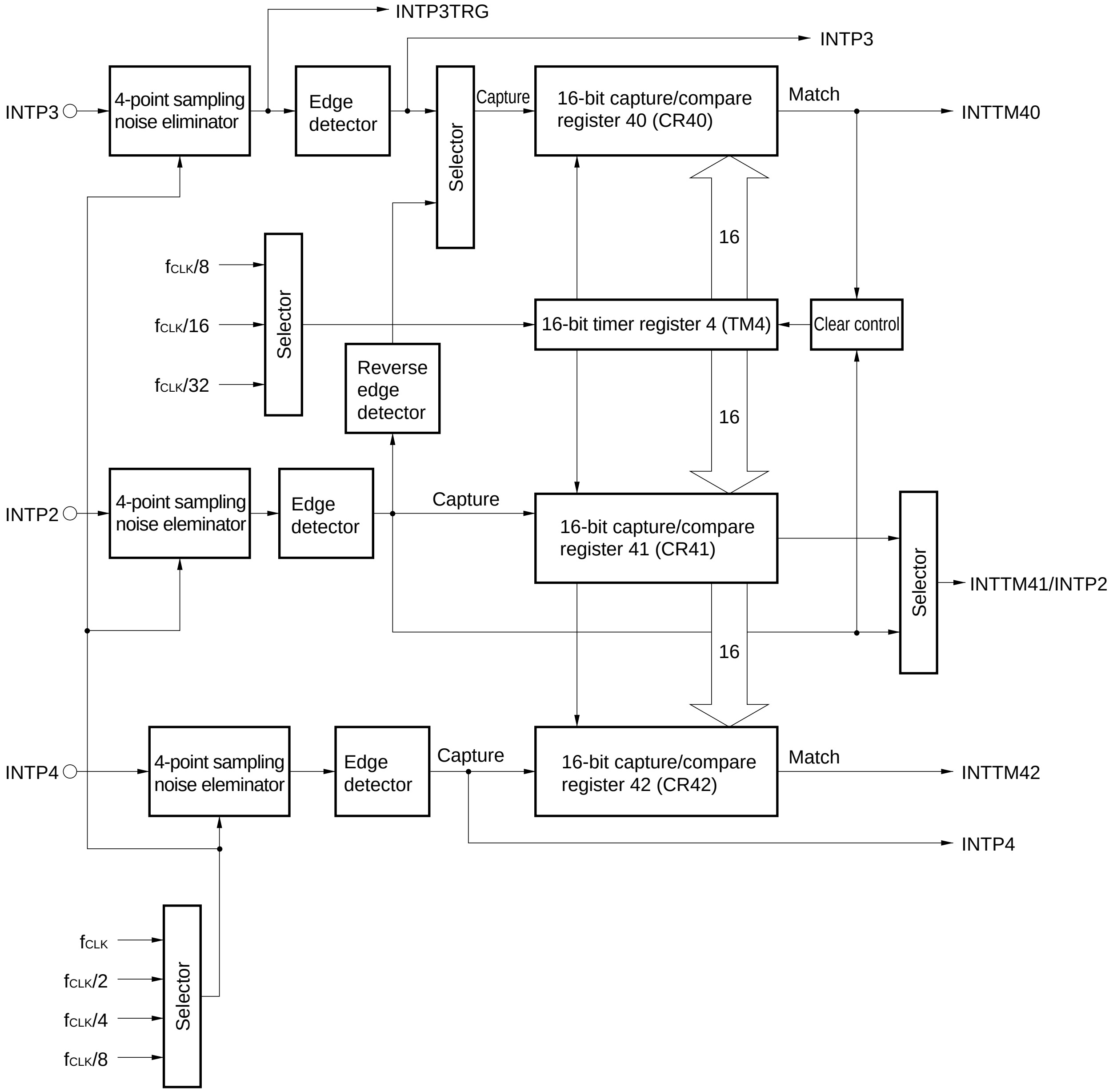
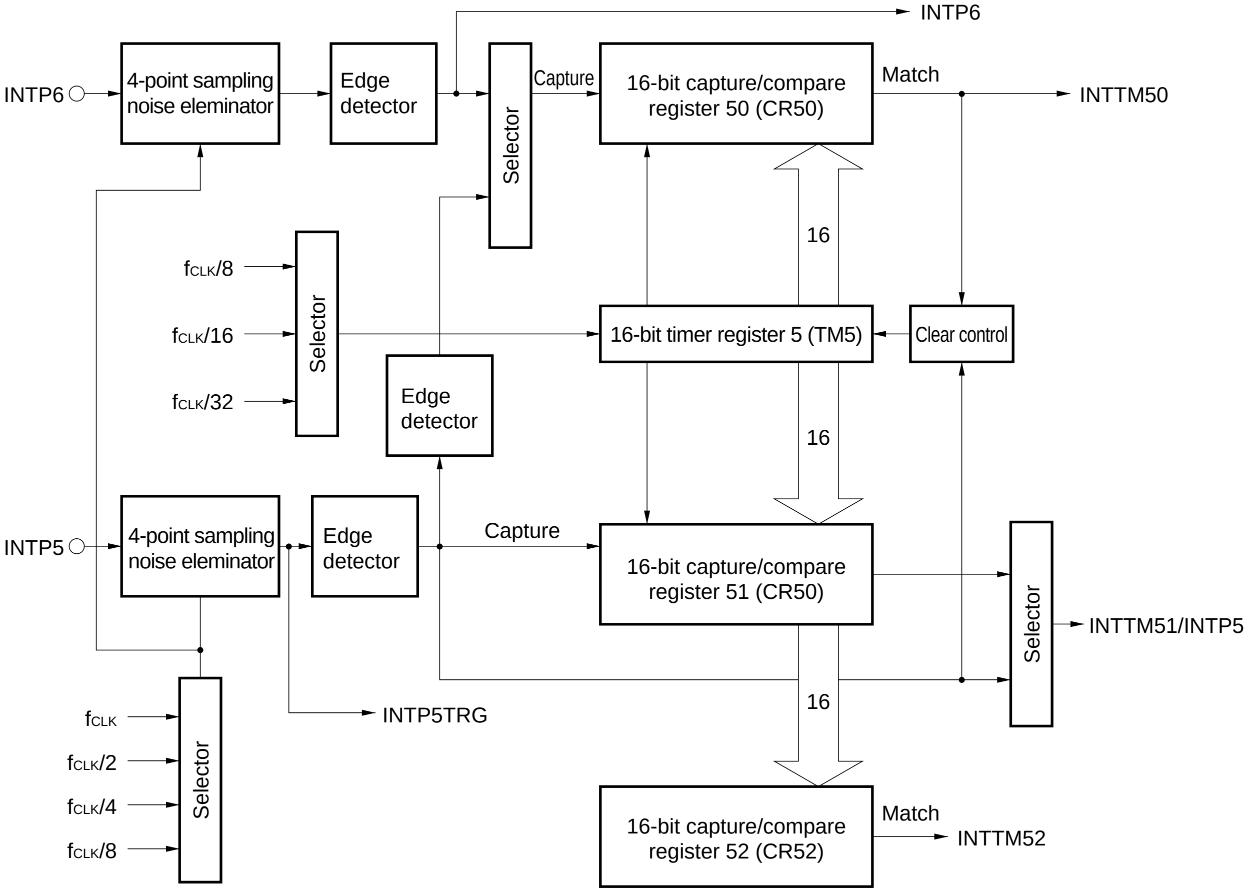

INTP3TRG
INTP3
INTP3
4-point sampling noise eliminator
Edge detector
Selector
Capture
16-bit capture/compare register 40 (CR40)
Match
INTTM40
16
fCLK/8
fCLK/16
fCLK/32
Selector
16-bit timer register 4 (TM4)
Clear control
Reverse edge detector
16
INTP2
4-point sampling noise eleminator
Edge detector
Capture
16-bit capture/compare register 41 (CR41)
Selector
INTTM41/INTP2
16
INTP4
4-point sampling noise eleminator
Edge detector
Capture
16-bit capture/compare register 42 (CR42)
Match
INTTM42
INTP4
fCLK
fCLK/2
fCLK/4
fCLK/8
Selector

Figure 7-1. Timer/Counter Block Diagram (4/5)

16-bit Timer/Counters 5


INTP6
INTP6
4-point sampling noise eleminator
Edge detector
Selector
Capture
16-bit capture/compare register 50 (CR50)
Match
INTTM50
16
fCLK/8
fCLK/16
fCLK/32
Selector
16-bit timer register 5 (TM5)
Clear control
Edge detector
16
INTP5
4-point sampling noise eleminator
Edge detector
Capture
16-bit capture/compare register 51 (CR50)
Selector
INTTM51/INTP5
INTP5TRG
16
fCLK
fCLK/2
fCLK/4
fCLK/8
Selector
16-bit capture/compare register 52 (CR52)
Match
INTTM52

8-bit Timer/Counters 6

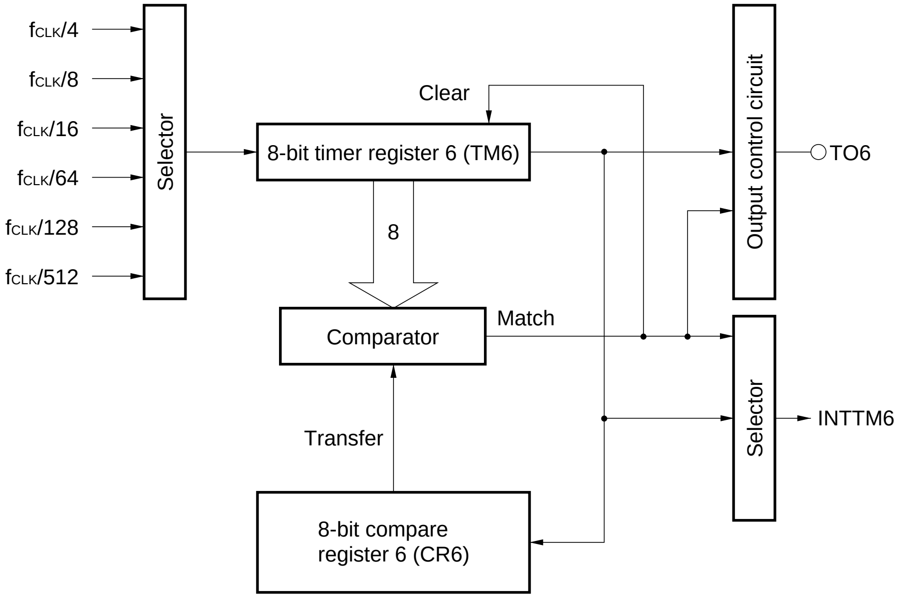
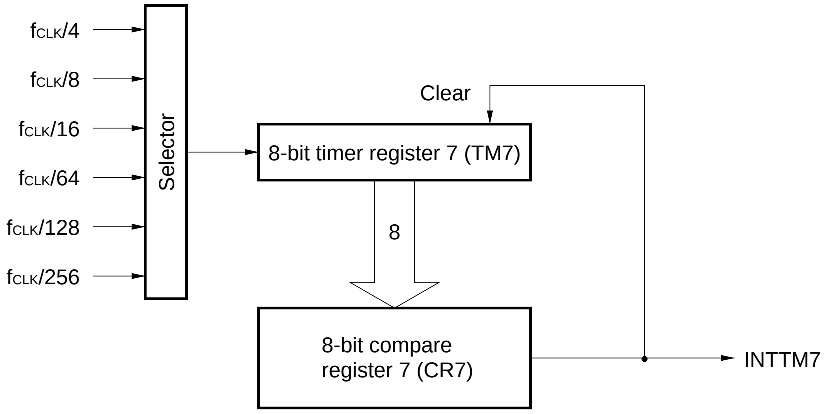


Figure 7-1. Timer/Counter Block Diagram (5/5)

8-bit Timer/Counters 7



CHAPTER 8 16-BIT TIMER/COUNTER 0

8.1 Function

16-bit timer/counter 0 (TM0) has the following functions:

- Interval timer
  An interrupt request is generated at an arbitrary time interval that was set in advance.
- PWM output

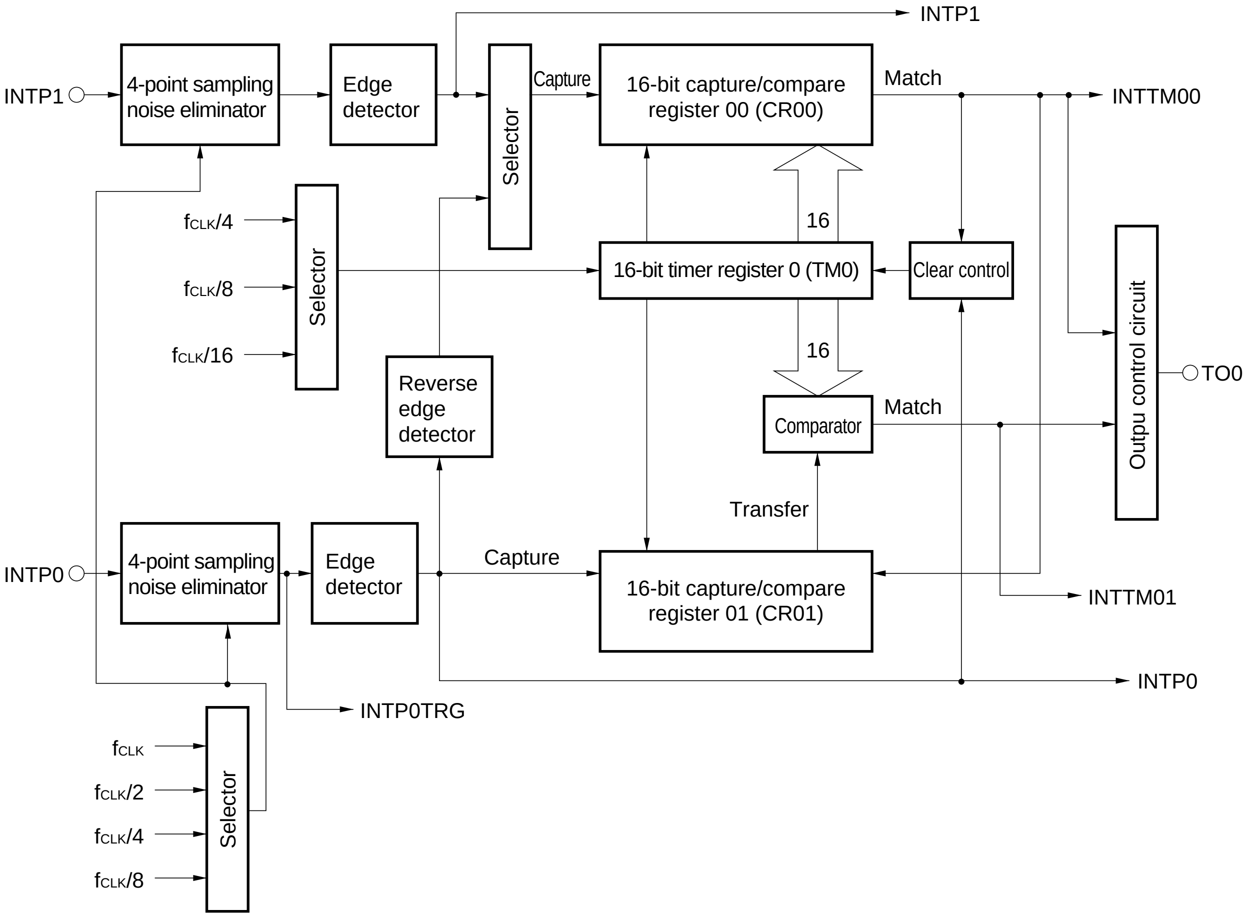
8.2 Configuration

16-bit timer/counter 0 (TM0) consists of the following hardware:

Table 8-1. Configuration of 16-Bit Timer/Counter 0 (TM0)

Item	Configuration
Timer register	16 bits × 1 (TM0)
Register	Capture/compare register: 16 bits × 2 (CR00, CR01)
Timer output	1 (TO0)
Control register	16-bit timer mode control register 0 (TMC0) Capture/compare control register 0 (CRC0) Timer output control register 0 (TOC0) Prescaler mode register 0 (PRM0)

Figure 8-1. Block Diagram of 16-Bit Timer/Counter 0 (TM0)



(1) 16-bit timer register 0 (TM0)

TM0 is a 16-bit free-running or interval timer that counts the count pulse. The count is incremented in synchronization with the rise of the input clock.

In the following cases the count value becomes 0000H.

<1> $\overline{\text{RESET}}$ input

<2> When bits 2 and 3 (TMC02, TMC03) of the 16-bit timer mode control register 0 (TMC0) are cleared.

<3> When the INTP0 valid edge input in the clear and start mode is input at the INTP0 valid edge.

<4> When 16-bit capture/compare register 00 (CR00) match, and when TM0 and CR00 match in the clear and start mode.

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

CR00 is a 16-bit register with combined capture-register and compare-register functions.

Depending on the value of bit 0 (CRC00) of the capture/compare control register 0 (CRC0) sets whether the register is used as a capture register or compare register.

- **Using CR00 as a compare register**

  Constantly compares the value set on CR00 and the count value of 16-bit timer register 0 (TM0) and if they match, generates an interrupt request (INTTM00). When TM0 is set for interval-timer operation, it can also be used as a register for containing the interval time.

- **Using CR00 as a capture register**

  The valid edge of pin INTP0 or pin INTP1 can be selected as a capture trigger, and is set by bits 4 and 5 (ES00, ES01) and bits 6 and 7 (ES10, ES11) of prescaler mode register 0 (PRM0).

  If the valid edge of pin INTP0 is specified as the capture trigger, setting is as shown in Table 8-2, and if the valid edge of pin INTP1 is specified as the capture trigger, then setting is as shown in Table 8-3.

Table 8-2. Valid Edge of Pin INTP0 and CR00 Capture Trigger

ES01	ES00	Valid Edge of INTP0 Pin	Capture Trigger of CR00	Capture Trigger of CR01
0	0	Falling edge	Rising edge	Falling edge
0	1	Rising edge	Falling edge	Rising edge
1	0	Setting prohibited	Setting prohibited	Setting prohibited
1	1	Both rising and falling edges	No capture operation	Both rising and falling edges

Table 8-3. Valid Edge of Pin INTP1 and CR00 Capture Trigger

ES11	ES10	Valid Edge of INTP1 Pin	Capture Trigger of CR00
0	0	Falling edge	Falling edge
0	1	Rising edge	Rising edge
1	0	Setting prohibited	Setting prohibited
1	1	Both rising and falling edges	Both rising and falling edges

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

$\overline{\text{RESET}}$ input makes CR00 undefined.

Caution When switching CR00 from the capture mode to the compare mode, the value of CR00 becomes the last captured value. Also, when switching from the compare mode to the capture mode, the value of CR00 becomes the last value that was set in the compare register.

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

CR01 is a 16-bit register with combined capture-register and compare-register functions.

Depending on the value of bit 2 (CRC02) of the capture/compare control register 0 (CRC0) sets whether the register is used as a capture register or compare register.

- **Using CR01 as a compare register**
  Constantly compares the value set on CR01 and the count value of 16-bit timer register 0 (TM0) and if they match, generates an interrupt request (INTTM01). When TM0 is set for interval-timer operation, it can also be used as a register for containing the interval time.

- **Using CR01 as a capture register**
  The valid edge of pin INTP0 can be selected as a capture trigger, and is set by bits 4 and 5 (ES00, ES01) of prescaler mode register 0 (PRM0).
  If the valid edge of pin INTP0 is specified as the capture trigger, setting is as shown in Table 8-4.

Table 8-4. Valid Edge of Pin INTP0 and CR01 Capture Trigger

ES01	ES00	Valid Edge of INTP0 Pin	Capture Trigger of CR01
0	0	Falling edge	Falling edge
0	1	Rising edge	Rising edge
1	0	Setting prohibited	Setting prohibited
1	1	Both rising and falling edges	Both rising and falling edges

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

$\overline{\text{RESET}}$ input makes CR00 undefined.

Caution When switching CR01 from the capture mode to the compare mode, the value of CR01 becomes the last captured value. Also, when switching from the compare mode to the capture mode, the value of CR01 becomes the last value that was set in the compare register.

8.3 Control Register

The following four types of registers control 16-bit timer/counter 0 (TM0).

- 16-bit timer mode control register 0 (TMC0)
- Capture/compare control register 0 (CRC0)
- Timer output control register 0 (TOC0)
- Prescaler mode register 0 (PRM0)

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

This register specifies the operation mode of the 16-bit timer; and the clear mode, output timing, and overflow detection of the 16-bit timer register 0 (TM0).

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

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

Caution TM0 starts operating when values other than "0" and "0" (operation stop mode) are set for bits 2 and 3 (TMC02, TMC03) of TMC0, respectively. To stop operation, set TMC02 and TMC03 to "0" and "0", respectively.

Figure 8-2. Format of 16-Bit Timer Mode Control Register 0 (TMC0)

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

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

TMC03	TMC02	Selection of Operating Mode and Clear Mode	Generation of Interrupt
0	0	Operation stop (TM0 is cleared to 0).	Does not generate.
0	1	Free running mode	Generates on coincidence between TM0 and CR00.
1	0	Clears and starts at valid edge of INTP0.	
1	1	Clears and starts on coincidence between TM0 and CR00.	

0VF0	Detection of Overflow of TM0
0	Does not overflow.
1	Overflows.

Caution Switch to the clear mode after stopping timer operation (setting bits 2 and 3 (TMC02 and TMC03) of 16-bit timer mode control register 0 (TMC0) to "0" and "0", respectively).
The valid edge of INTP0 is set by prescaler mode register 0 (PRM0).

Remark INTP0: input pin of 16-bit timer/counter 0 (TM0)
TM0 : 16-bit timer register 0
CR00 : 16-bit capture/compare register 00

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

This register controls the operation of the capture/compare registers 00, 01 (CR00 and CR01).

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

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

Figure 8-3. Format of Capture/Compare Control Register 0 (CRC0)

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

Symbol	7	6	5	4	3	2	1	0
CRC0	SMPC01	SMPC00	0	0	0	CRC02	CRC01	CRC00

SMPC01	SMPC00	Selection of Sampling Clock
0	0	f_{CLK}
0	1	$f_{CLK}/2$
1	0	$f_{CLK}/4$
1	1	$f_{CLK}/8$

CRC02	Selection of Operation Mode of CR01
0	Operates as compare register.
1	Operates as capture register.

CRC01	Selection of Capture Trigger of CR00
0	Captured at valid edge of INTP1.
1	Captured in reverse phase of valid edge of INTP0.

CRC00	Selection of Operation Mode of CR00
0	Operates as compare register.
1	Operates as capture register.

Cautions 1. CRC0 should be set after stopping timer operation.

2. Do not specify CR00 as a capture register when the clear and start mode when TM0 and CR00 match is selected by 16-bit timer mode control register 0 (TMC0).

(3) Timer output control register 0 (TOC0)

This is a register for controlling operation of the 16-bit timer/counter 0 (TM0) output control circuit. It sets or resets the R-S type flip-flop (ALV0), and enables or disables TM0 timer output.

TOC0 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets TOC0 to 00H.

Figure 8-4. Format of Timer Output Control Register 0 (TOC0)

Address: 0FF1AH After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	⓪
TOC0	0	0	0	0	0	0	ALV0	TOE0

ALV0	Specify of Active Level
0	Active level “0” (low)
1	Active level “1” (high)

TOE0	Output Control of 16-Bit Timer/Counter 0 (TM0)
0	Disables output (output is set to 0 level).
1	Enables output.

Caution TOC0 should be set after TM0 timer operation has been stopped.

Remark Setting and resetting of the timer output is controlled by INTTM00 (set signal) and INTTM01 (reset signal).

(4) Prescaler mode register 0 (PRM0)

Prescaler mode register 0 (PRM0) is a register that sets the count clock of 16-bit timer register 0 (TM0), and the valid edges of INTP0 and INTP1 input.

PRM0 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets PRM0 to 00H.

Figure 8-5. Format of Prescaler Mode Register 0 (PRM0)

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

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

ES11	ES10	Selection of Valid Edge of INTP1
0	0	Falling edge
0	1	Rising edge
1	0	Setting prohibited
1	1	Both falling and rising edges

ES01	ES00	Selection of Valid Edge of INTP0
0	0	Falling edge
0	1	Rising edge
1	0	Setting prohibited
1	1	Both falling and rising edges

PRM01	PRM00	Selection of Count Clock
0	0	$f_{CLK}/4$ (2 MHz)
0	1	$f_{CLK}/8$ (1 MHz)
1	0	$f_{CLK}/16$ (500 kHz)
1	1	Setting prohibited

Cautions 1. If the valid edges of INTP0 and INTP1 are set for the count clock, do not set the clear and start mode or the capture trigger with the valid edges of INTP0 and INTP1.

2. PRM0 should be set after timer operation has been stopped.

Remark Figures in parentheses apply to operation with f_{CLK} = 8 MHz

8.4 Operation

8.4.1 Basic operation of TM0

16-bit timer/counter 0 (TM0) is a 16-bit free running or interval timer that counts the count pulse. Its count is incremented in synchronization with the rise of the input clock.

All of the bits of TM0 are cleared (0) by $\overline{\text{RESET}}$ input, and the count operation stops.

Enabling or disabling the count operation is controlled by bits 2 and 3 (TMC02, TMC03) of 16-bit timer mode control register 0 (TMC0). If TMC02 and TMC03 are set to an operation mode other than "0" and "0" the count operation starts, and when they are reset (TMC02 and TMC03 are set to "0" and "0"), TM0 is cleared and the count operation stops.

Also, the count value becomes 0000H.

TM0 changes from 0000H to 0001H at the first count clock input after the count-start setting.

TM0 continues operating as is even if the same operating mode is set during operation and the timer is not cleared.

Count does not stop even during the TM0 read period.

Figure 8-6. Basic Operation Timing of TM0

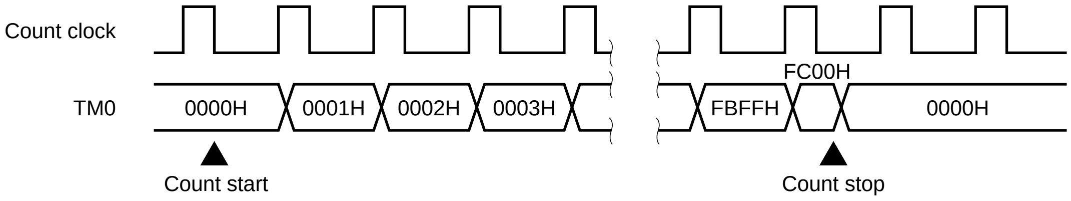
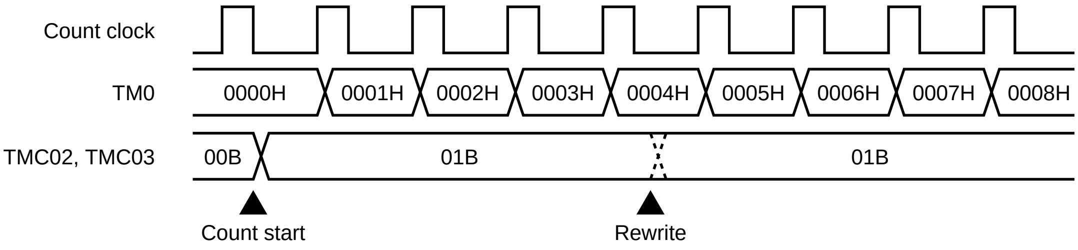


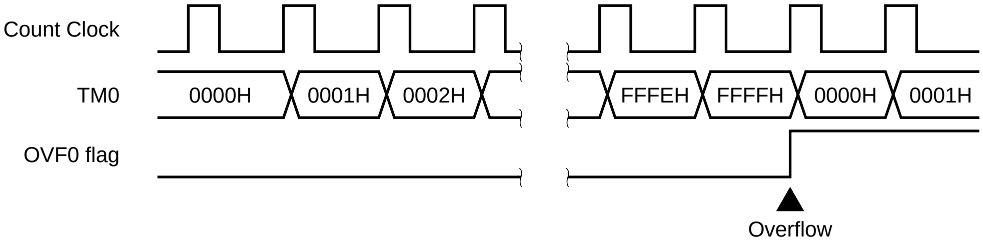
Figure 8-7. Timing for Rewriting to TMC02 and TMC03 (Free Running Mode)



8.4.2 Free running operation of TM0

If bits 2 and 3 (TMC02, TMC03) of 16-bit timer mode control register 0 (TMC0) are set to “1” and “0”, respectively, TM0 performs free running operation. If TM0 has a full count by FFFFH, bit 0 of TMC0 (OVF0) is set to “1” at the next count clock, and TM0 is cleared. Counting continues after that, and it is possible to clear OVF0 using an instruction.

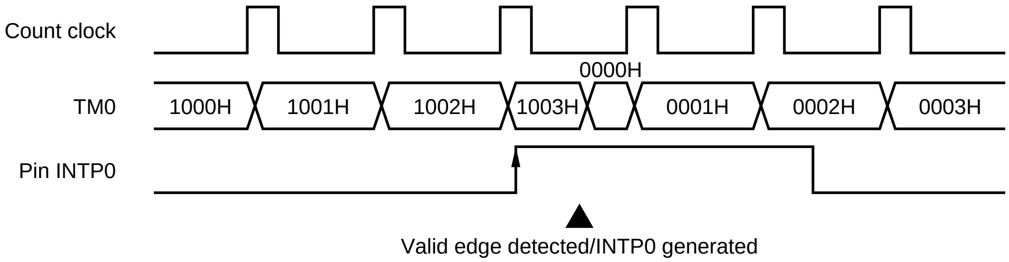
Figure 8-8. Free Running Mode Operation Timing of TM0



8.4.3 Clear and start operation of TM0 at valid edge of INTP0

If bits 2 and 3 (TMC02, TMC03) of 16-bit timer mode control register 0 (TMC0) are set to “0” and “1”, respectively, TM0 is set to the clear and start mode at input of the valid edge of INTP0. When the valid edge of INTP0 is input (interrupt request signal: INTP0 is generated), TM0 is cleared (0000H) and becomes 0001H on the next count clock. Count continues after that.

Figure 8-9. Clear and Start Mode Operation Timing of TM0 at Input of the Valid Edge of INTP0



8.4.4 Clear and start operation when TM0 and CR00 match

If bits 2 and 3 (TMC02, TMC03) of 16-bit timer mode control register 0 (TMC0) are set to “1” and “1”, respectively, TM0 is set to the clear and start mode when 16-bit capture/compare register 00 (CR00) match. When TM0 and CR00 match, an interrupt request signal (INTTM00) is generated at the next count clock, and TM0 is cleared (0000H).

Figure 8-10. Clear and Start Mode Operation Timing When TM0 and CR00 Match (CR00 ≠ 0000H)

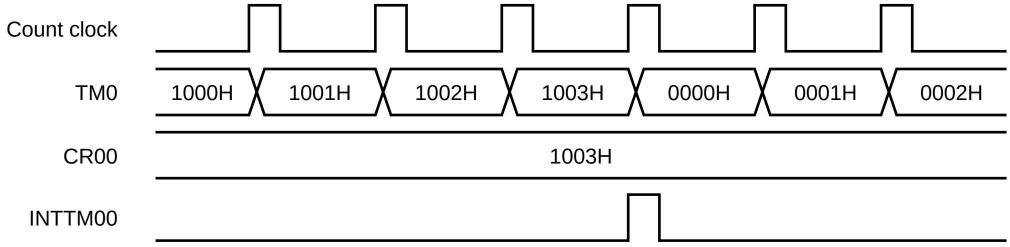
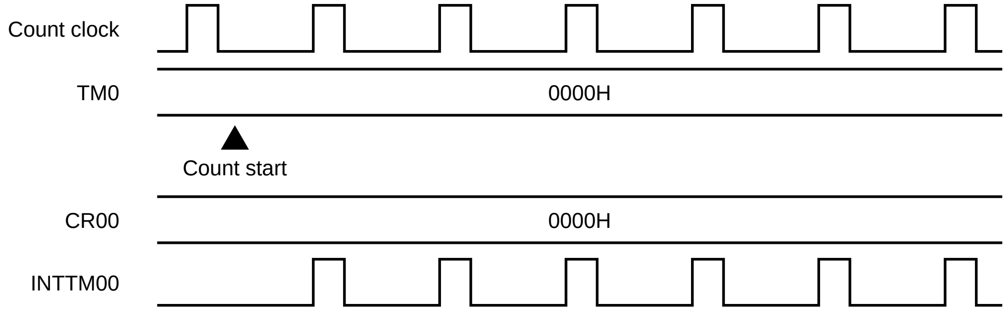


Figure 8-11. Clear and Start Mode Operation Timing When TM0 and CR00 Match (CR00 = 0000H)



Caution CR00 should be set to the compare mode.

Remark Interval period = (CR00 + 1) × TM0 count-clock rate

8.4.5 Operation as 16-bit PWM output

By setting bit 0 (TOE0) of timer output control register 0 (TOC0) to “1”, it operates as PWM output.

<Setting method>

<1> Specify the active level (bit 1 (ALV0) of TOC0) of timer 0 output (TO0), and enable TO0 output (set bit 0 (TOE0) of TOC0 to “1”).

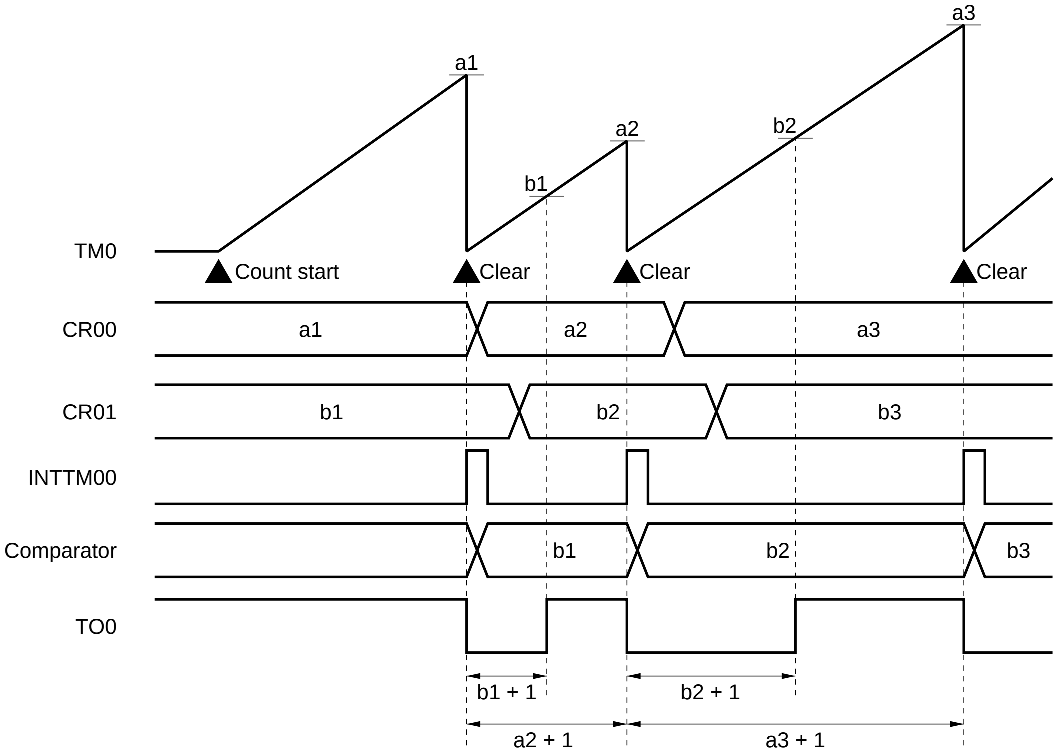
<2> Set 16-bit capture/compare registers 00 and 01 (CR00, CR01) to the compare mode (set bits 0 and 2 (CRC00, CRC02) of capture/compare control register 0 (CRC0) to “0”).

<3> Set the interval period in CR00, and set the active-level width in CR01.

<4> The count clock is selected by bits 0 and 1 (PRM00, PRM01) of prescaler mode register 0 (PRM0).

<5> By setting bits 2 and 3 (TMC02, TMC03) of 16-bit timer mode control register 0 (TMC0) to “1”, the count operation starts, and the PWM signal is output from pin TO0.

Figure 8-12. Example of PWM Output of TO0



Caution If CR00 = CR01 is set, TO0 outputs inactive level ($\overline{\text{ALV0}}$).
If CR00 < CR01 is set, TO0 outputs active level (ALV0).

Remark CR00, CR01: Compare mode, CR00 > CR01, Active level: “0”

8.4.6 Capture operation of TM0

If bits 0 and 2 (CRC00, CRC02) of capture/compare control register 0 (CRC0) are set to “1”, 16-bit capture/compare registers 00 and 01 (CR00, CR01) are set to the capture mode. When the capture trigger is input, the value of TM0 is captured to CR00 and CR01.

Figure 8-13. Capture Operation Timing (Free Running Mode)

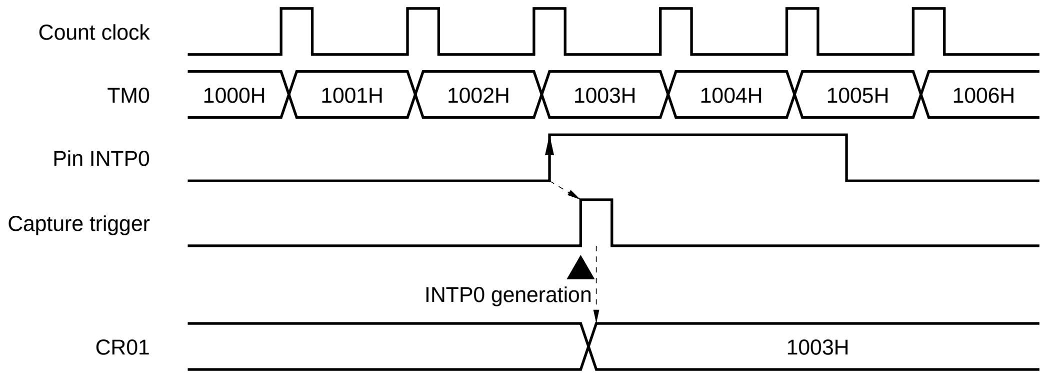
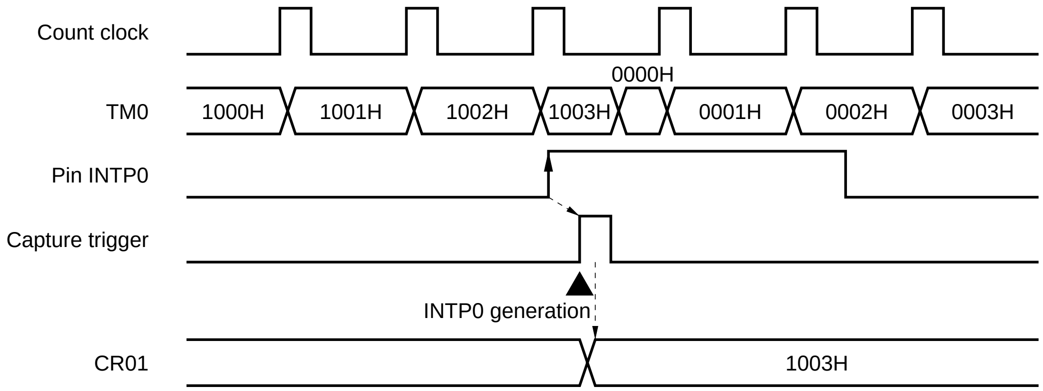


Figure 8-14. Capture Operation Timing (Clear and Start Mode at INTP0 Valid Edge Input)



8.4.7 Pulse width measurement operation

(1) Pulse width measurement (both rising and falling edge)

It is possible to use 16-bit timer register 0 (TM0) to measure the pulse width of the signal input to pins INTP0/ P01 and INTP1/P02. The width from edge to edge is measured.

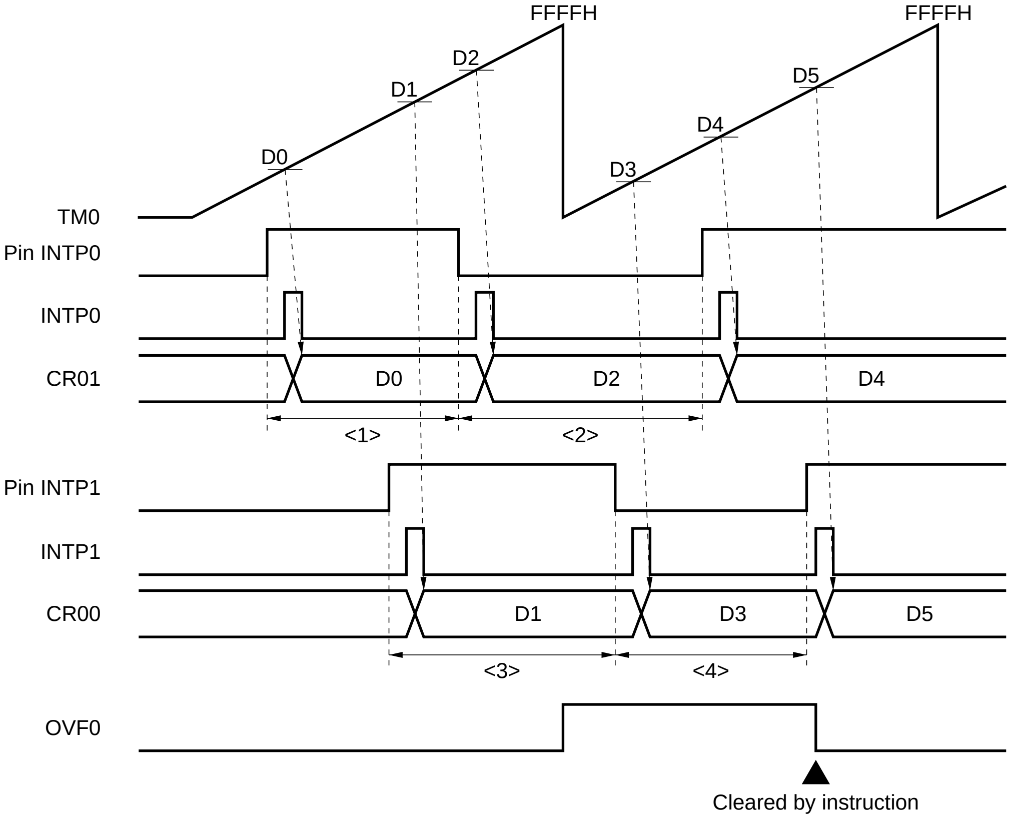
<1> Set 16-bit capture/compare registers 00 and 01 (CR00, CR01) to the capture mode (set bits 0 and 2 (CRC00, CRC02) of capture/compare control register 0 (CRC0) to “1”).

<2> Set the CR00 capture trigger in INTP1 (set bit 1 (CRC01) of CRC0 to “0”).

<3> Set both edges of INTP0 and INTP1 as valid edges (set bits 4 to 7 (ES00, ES01, ES10, ES11) of prescaler mode register 0 (PRM0) to “1”).

<4> Set the free running mode (set bits 2 and 3 (TMC02, TMC03) of 16-bit timer mode control register 0 (TMC0) to “1” and “0”, respectively).

Figure 8-15. Pulse Width Measurement Timing (When Both Edges Are Specified)



[Measurement Method]

- The CR01 and OVF0 (bit 0 of TMC0) flags are read in INTP0 interrupt processing.
  <1> is (D2 – D0) × count-clock rate.
  <2> is (10000H – D2 + D4) × count-clock rate.
- The CR00 and OVF0 flags are read in INTP1 interrupt processing.
  <3> is (10000H – D1 + D3) × count-clock rate.
  <4> is (D5 – D3) × count-clock rate.

Remark Dn: TM0 count value (n = 0, 1, 2, ...)

(2) Pulse width measurement (rising edge)

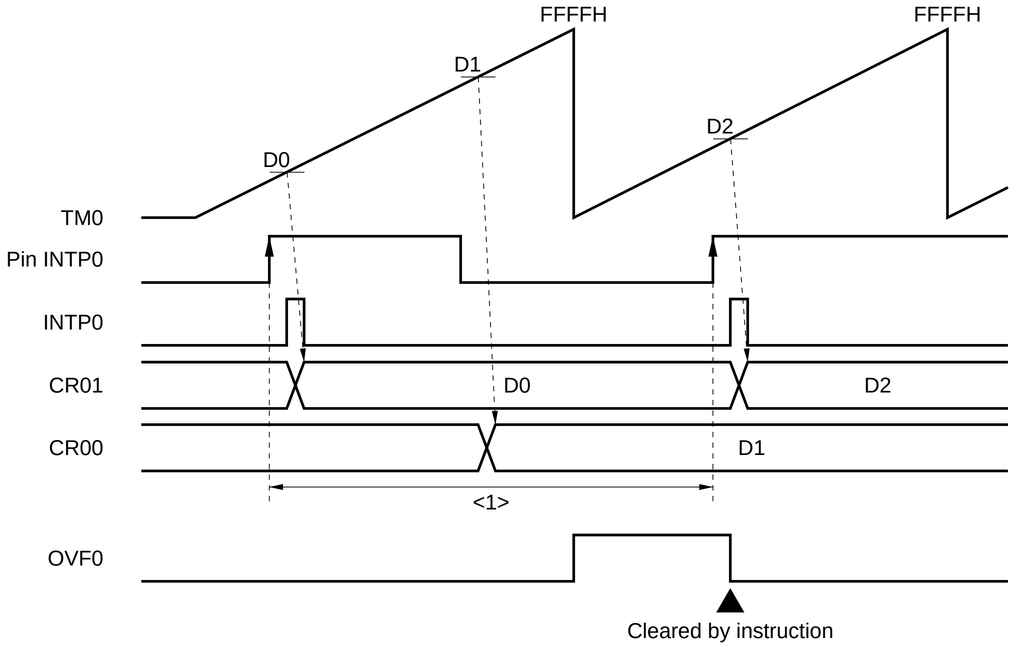
It is possible to use 16-bit timer register 0 (TM0) to measure the width of the pulse input to pin INTP0/P01. The width from edge to edge is measured.

<1> Set 16-bit capture/compare control register 00 and 01 (CR00, CR01) to the capture mode (set bits 0 and 2 (CRC00, CRC02) of capture/compare control register 0 (CRC0) to “1”).

<2> Set the opposite edge of INTP0 as the CR00 capture trigger (set bit 1 (CRC01) of CRC0 to “1”).

<3> Set the free-running mode (set bits 2 and 3 (TMC02, TMC03) of 16-bit timer mode control register 0 (TMC0) to “1” and “0”, respectively).

Figure 8-16. Pulse Width Measurement Timing (When Rising Edge Is Specified)



[Measurement Method]

- The CR01 and OVF0 (bit 0 of TMC0) flags are read in INTP0 interrupt processing.
  <1> is (10000H – D0 + D2) × count-clock rate.

Caution CR00 is captured at the opposite edge of INTP0, however, an interrupt request signal (INTP0) is not generated at that time. INTP0 (request signal) is only generated when the specified valid edge is detected.

Remark Dn: TM0 count value (n = 0, 1, 2, ...)

(3) Pulse width measurement (falling edge)

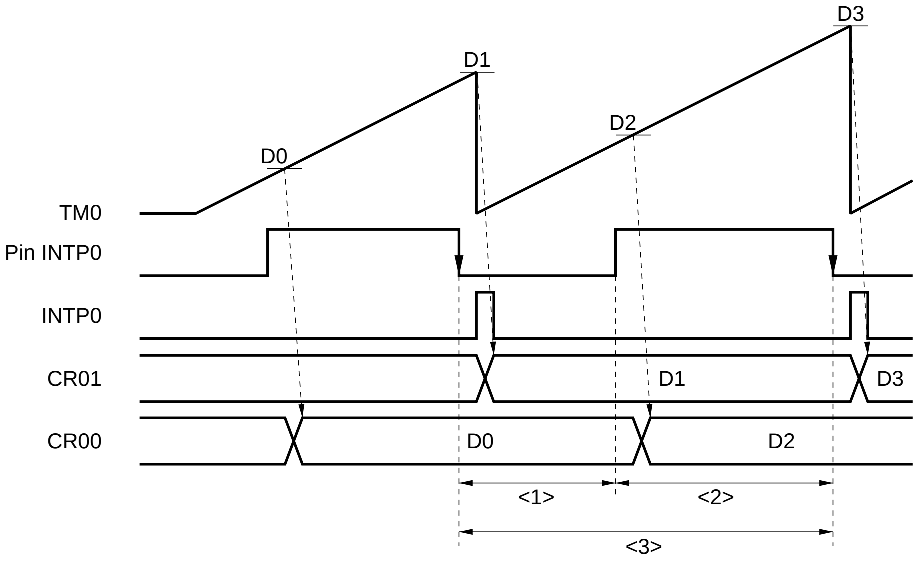
It is possible to use 16-bit timer register 0 (TM0) to measure the width of the pulse input to pin INTP0/P01. The high width and low width are individually measured.

<1> Set 16-bit capture/compare control registers 00 and 01 (CR00, CR01) to the capture mode (set bits 0 and 2 (CRC00, CRC02) of capture/compare control register 0 (CRC0) to “1”).

<2> Set the opposite edge of INTP0 as the CR00 capture trigger (set bit 1 (CRC01) of CRC0 to “1”).

<3> Set the clear and start mode at the valid edge input of INTP0 (set bits 2 and 3 (TMC02, TMC03) of 16-bit timer mode control register 0 (TMC0) to “0” and “1”, respectively).

Figure 8-17. Pulse Width Measurement Timing (When Falling Edge Is Specified)



[Measurement Method]

- CR00 and CR01 are read by INTP0 interrupt processing.
  <1> Low width is D2 × count-clock rate.
  <2> High width is (D3 – D2) × count-clock rate.
  <3> 1 period is D3 × count-clock rate.
  However, TM0 must be corrected when there is overflow.

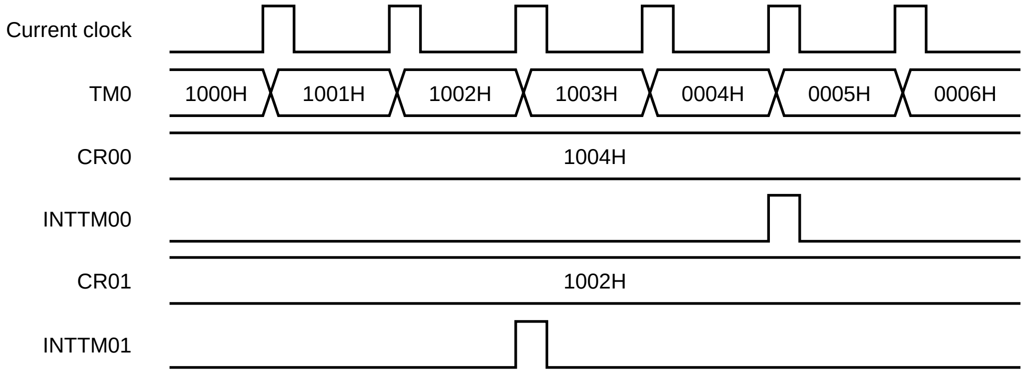
Caution CR00 is captured at the opposite edge of INTP0, however, an interrupt request signal (INTP0) is not generated at that time. INTP0 (request signal) is only generated when the specified valid edge is detected.

Remark Dn: TM0 count value (n = 0, 1, 2, ...)

8.4.8 Compare operation of TM0

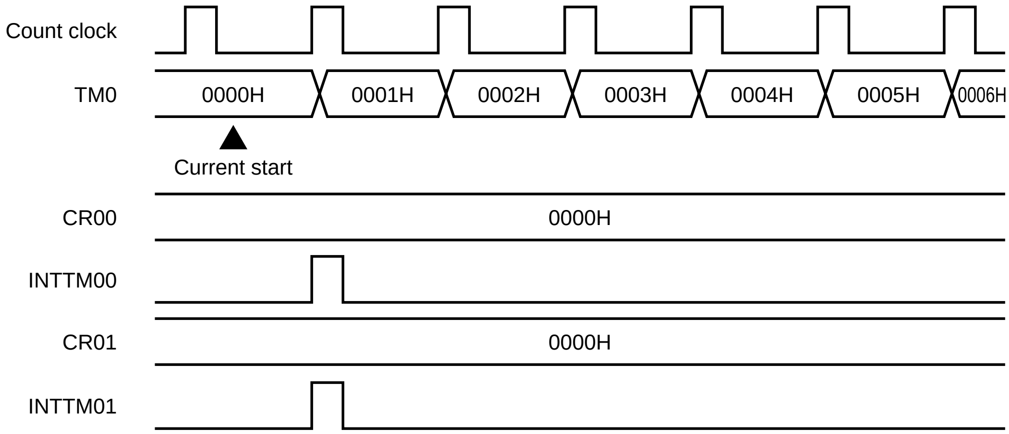
Set bits 0 and 2 (CRC00, CRC02) of capture/compare control register 0 (CRC0) are set to “0”, and set 16-bit capture/compare registers 00 and 01 (CR00, CR01) to the compare mode. If 16-bit timer register 0 (TM0) matches CR00 or CR01, an interrupt request signal (INTTM00 or INTTM01) is generated at the next count clock.

Figure 8-18. Compare Operation Timing of TM0 (CR00, CR01 ≠ 0000H)



Caution The operating mode of TM0 shown in Figure 8-18 is a mode other than the clear and start mode when TM0 and CR00 match. In the case of the clear and start mode when TM0 and CR00 match, TM0 is cleared at the next count clock after TM0 and CR00 match. (See Figure 8-10. Clear and Start Mode Operation Timing When TM0 and CR00 Match (CR00 ≠ 0000H).) TM0 is not cleared if it matches with CR01.

Figure 8-19. Compare Operation Timing of TM0 (CR00, CR01 = 0000H)

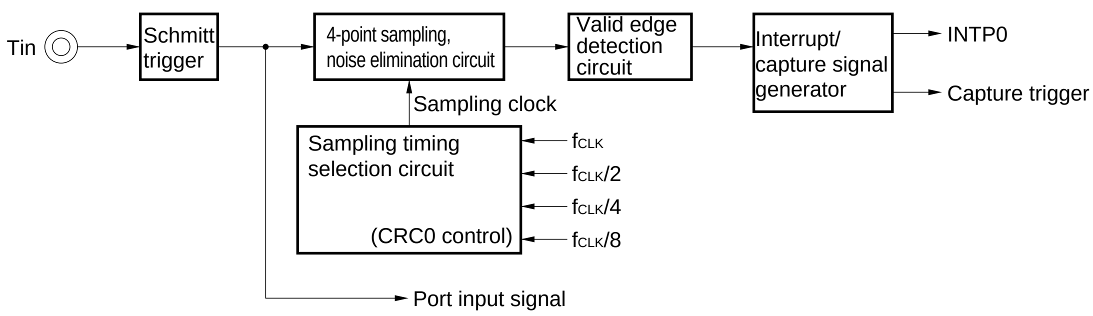


Caution The operating mode of TM0 shown in Figure 8-19 is a mode other than the clear and start mode when TM0 and CR00 match. In the case of the clear and start mode when TM0 and CR00 match, TM0 remains 0000H. (See Figure 8-11. Clear and Start Mode Operation Timing When TM0 and CR00 Match (CR00 = 0000H).)

8.4.9 Noise elimination circuit

The noise elimination circuit of 16-bit timer/counter 0 (TM0) performs sampling at four point at the timing specified by bits 6 and 7 (SMPC00, SMPC01) of capture/compare control register 0 (CRC0). It performs sampling in succession, and if the same level is detected four times in a row, that level is fetched.

Figure 8-20. INTP0 Block Diagram


Tin
Schmitt trigger
4-point sampling, noise elimination circuit
Valid edge detection circuit
Interrupt/ capture signal generator
INTP0
Capture trigger
Sampling clock
Sampling timing selection circuit
(CRC0 control)
fCLK
fCLK/2
fCLK/4
fCLK/8
Port input signal

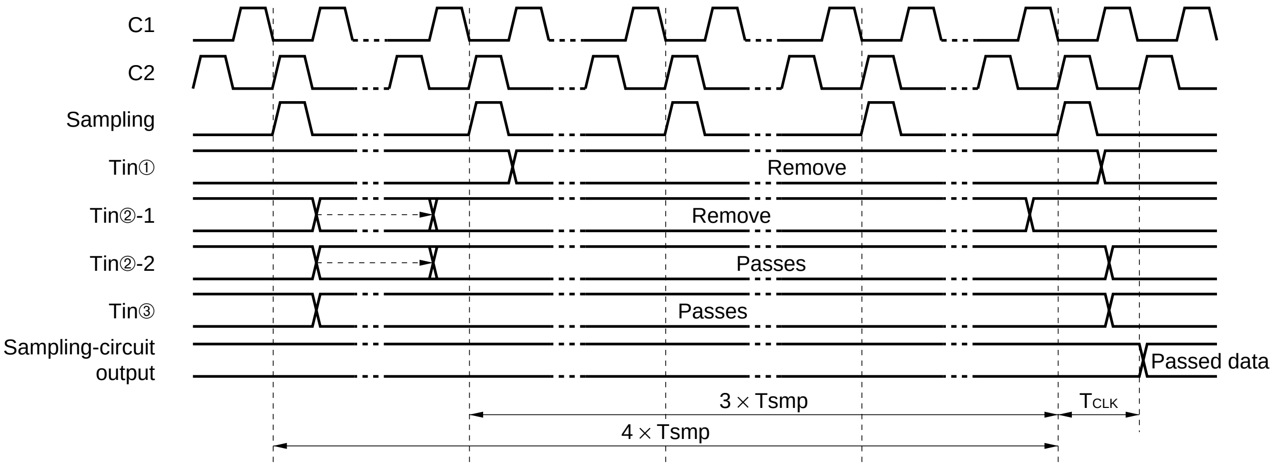
[Sampling timing]

Tin: Width of INTP0 pin input signal, Tsmp: Sampling timing,
C1, C2: System clock, TCLK: System-clock rate (= 1/fCLK)

<1> Tin ≤ (3 × Tsmp) ..... Removed as noise.
<2> (3 × Tsmp) < Tin < (4 × Tsmp) .... May be removed as noise, or may pass as a valid signal.
<3> Tin ≥ (4 × Tsmp) .... Passes as a valid signal.

Caution In order to ensure that valid signals pass, input a signal whose width is 4 × Tsmp.

Figure 8-21. Sampling Timing Diagram



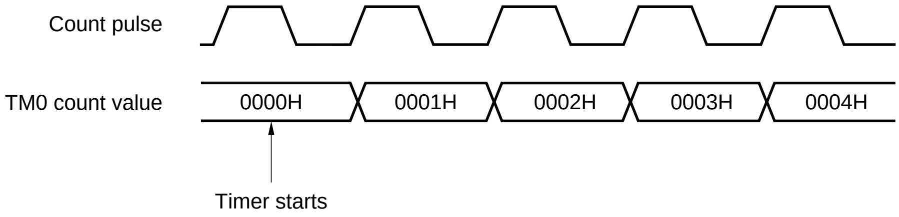
Remark The time when the pin level (Tin) passes the sampling circuit may vary by 1 × Tsmp at (3 × Tsmp + TCLK) to (4 × Tsmp + TCLK).

★ 8.5 Cautions

(1) Error at timer start

After the timer starts, the time until a uniform signal is generated may have a maximum error of 1 clock. This is because the start of 16-bit timer register 0 (TM0) is asynchronous with respect to the count pulse.

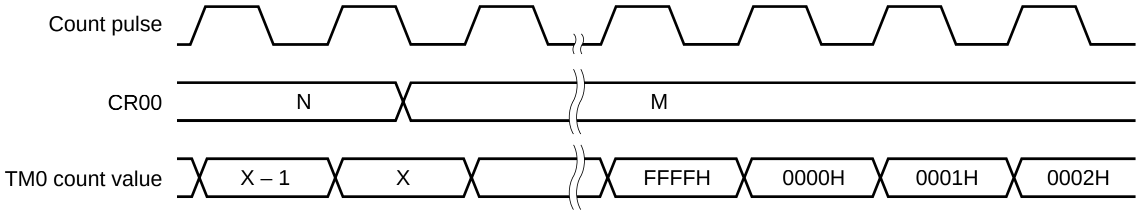
Figure 8-22. Start Timing of 16-Bit Timer Register 0



(2) Operation after changes in the compare register during timer count operation

If the value of 16-bit capture/compare register 00 (CR00) after changing is less than the value of 16-bit timer register 0 (TM0), TM0 continues counting, and when an overflow occurs, it starts over from 0. Also, if the value of CR00 after changing (M) is less than the value before changing (N), the timer must be restarted after CR00 changes.

Figure 8-23. Timing After Changing Compare Register during Timer Count Operation



Remark N > X > M

(3) Valid edge setting

The valid edge of pin INTP0/P01 should be set after setting bits 2 and 3 (TMC02, TMC03) of 16-bit timer mode control register 0 (TMC0) to “0” and “0”, respectively, and after timer operation stops. The valid edge is set using bits 4 and 5 (ES00, ES01) of prescaler mode register 0 (PRM0).

[MEMO]

9.1 Functions

16-bit timer/counter 1 (TM1) has the following two modes.

- Interval timer
  An interrupt request is generated at an arbitrary time interval that was set in advance.
- PWM output

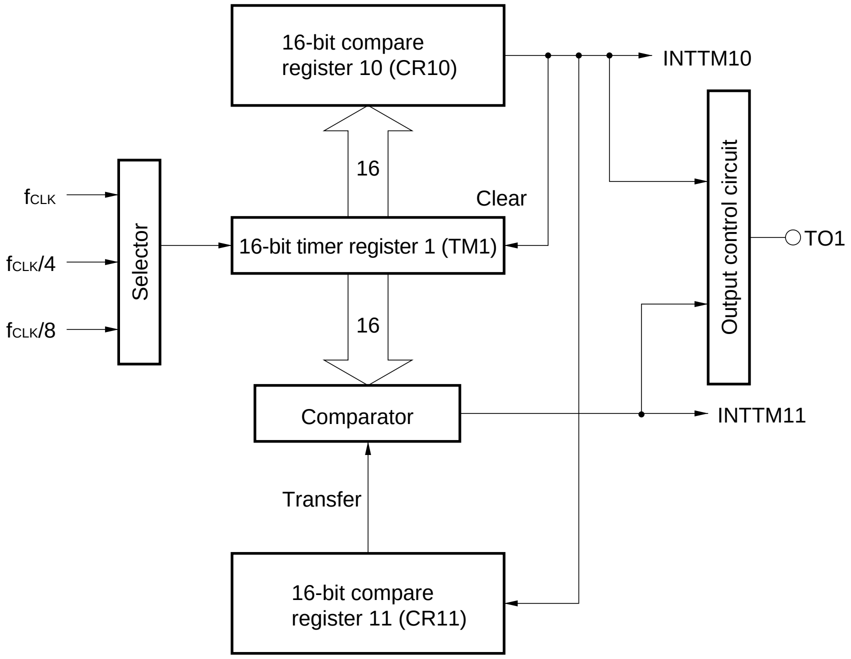
9.2 Configuration

16-bit timer/counter 1 (TM1) is comprised of the following hardware.

Table 9-1. 16-Bit Timer/Counter 1 (TM1) Configuration

Item	Configuration
Timer register	16-bit × 1 (TM1)
Register	Compare registers: 16-bit × 2 (CR10, CR11)
Timer output	1 (TO1)
Control register	16-bit timer mode control register 1 (TMC1) Timer output control register 1 (TOC1) Prescaler mode register 1 (PRM1)

Figure 9-1. Block Diagram of 16-Bit Timer/Counter 1 (TM1)



(1) 16-bit timer register 1 (TM1)

TM1 is a 16-bit free running or interval timer that counts the count pulse. The count is incremented in synchronization with the rise of the input clock.

In the following cases the count value becomes 0000H.

<1> $\overline{\text{RESET}}$ input

<2> When bits 2 and 3 (TMC12, TMC13) of 16-bit timer mode control register 1 (TMC1) are cleared.

<3> When TM1 and CR10 match in the clear and start mode at the match of 16-bit compare register 10 (CR10).

(2) 16-bit compare register 10 (CR10)

This register constantly compares the value set for CR10 with the count value of 16-bit timer register 1 (TM1), and if they match, it generates an interrupt request (INTTM10). If TM1 is set as an interval timer, it can also be used as a register for holding the interval time.

CR10 is set by a 16-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input makes CR10 undefined.

(3) 16-bit compare register 11 (CR11)

This register constantly compares the value set for CR11 with the count value of 16-bit timer register 1 (TM1), and if they match, it generates an interrupt request (INTTM11). If TM1 is set as an interval timer, it can also be used as a register for holding the interval time.

CR11 is set by a 16-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input makes CR11 undefined.

9.3 Control Registers

The following three registers control 16-bit timer/counter 1 (TM1).

- 16-bit timer mode control register 1 (TMC1)
- Timer output control register 1 (TOC1)
- Prescaler mode register 1 (PRM1)

(1) 16-bit timer mode control register 1 (TMC1)

TMC1 is a register for detecting the setting and overflow of the clear mode of 16-bit timer register 1 (TM1).
TMC1 is set by a 1-bit or 8-bit memory manipulation instruction.
$\overline{\text{RESET}}$ input sets TMC1 to 00H.

Caution TM1 starts operating when bits 2 and 3 (TMC12, TMC13) of TMC1 are set to values other than "0" and "0" (operation stop mode), respectively. To stop operation, set TMC12 and TMC13 to "0" and "0", respectively.

Figure 9-2. Format of the 16-Bit Timer Mode Control Register 1 (TMC1)

Address: 0FF6BH After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	⓪
TMC1	0	0	0	0	TMC13	TMC12	0	OVF1

TMC13	TMC12	Selection of Operating Mode and Clear Mode	Generation of Interrupt
0	0	Operation stop (TM1 is cleared to 0).	Does not generate
0	1	Free running mode.	Generates on coincidence between TM1 and CR10.
1	0	Setting prohibited	
1	1	Clears and starts on coincidence between TM1 and CR10.	

OVF1	Detection of Overflow of TM1
0	Does not overflow
1	Overflow

Caution TMC1 should be set only after TM1 timer operation has stopped.

(2) Timer output control register 1 (TOC1)

This is a register for controlling the operation of the 16-bit timer/counter 1 (TM1) output control circuit. It sets or resets the R-S type flip-flop (ALV1), enables or disables output inversion, and enables or disables TM1 timer output.

TOC1 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets TOC1 to 00H.

Figure 9-3. Format of Timer Output Control Register 1 (TOC1)

Address: 0FF7BH After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	⓪
TOC1	0	0	0	0	0	0	ALV1	TOE1

ALV1	Specify of Active Level
0	Active level “0” (low)
1	Active level “1” (high)

TOE1	Output Control of 16-Bit Timer/Counter 1 (TM1)
0	Disable output (output is set to 0 level)
1	Enable output

Cautions 1. TOC1 should be set after TM1 timer operation has been stopped.

2. If PWM is to be output from TM1, bits 2 and 3 (TMC12, TMC13) of 16-bit timer mode control register 1 (TMC1) should be set to “1” and “1”, respectively. If TMC12 and TMC13 are set to “0” and “0”, respectively, TO1 becomes inactive.

Remark Setting and resetting of the timer output is controlled by INTTM10 (set signal) and INTTM11 (reset signal).

(3) Prescaler mode register 1 (PRM1)

Prescaler mode register 1 (PRM1) is a register for specifying the count clock of 16-bit timer register 1 (TM1).

PRM1 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets PRM1 to 00H.

Figure 9-4. Format of the Prescaler Mode Register 1 (PRM1)

Address: 0FF85H After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
PRM1	0	0	0	0	0	0	PRM11	PRM10

PRM11	PRM10	Count Clock Selection
0	0	f_{CLK} (8 MHz)
0	1	$f_{CLK}/4$ (2 MHz)
1	0	$f_{CLK}/2$ (1 MHz)
1	1	Setting prohibited

Caution PRM1 should be set after TM1 timer operation has stopped.

Remark Figures in parentheses apply to operation with f_{CLK} = 8 MHz.

9.4 Operation

9.4.1 Basic operation of TM1

16-bit timer/counter 1 (TM1) is a 16-bit free-running or interval timer that counts the count pulse. Its count is incremented in synchronization with the rise of the input clock.

All of the bits of TM1 are cleared (0) by $\overline{\text{RESET}}$ input, and the count operation stops.

Enabling or disabling the count operation is controlled by bits 2 and 3 (TMC12, TMC13) of 16-bit timer mode control register 1 (TMC1). If TMC12 and TMC13 are set to an operation mode other than “0” and “0”, the count operation starts, and when they are reset (TMC12 and TMC13 are set to “0” and “0”), TM1 is cleared and the count operation stops.

The count value becomes 0000H.

TM1 changes from 0000H to 0001H at the first count clock input after the count-start setting.

TM1 continues operating as is even if the same operating mode is set during operation and the timer is not cleared.

Count does not stop even during the TM1 read period.

Figure 9-5. Basic Operation Timing of TM1

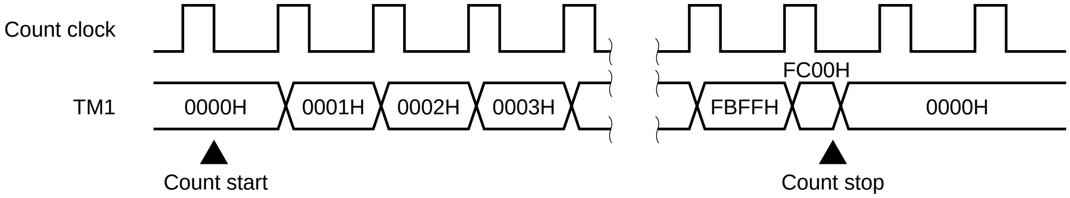
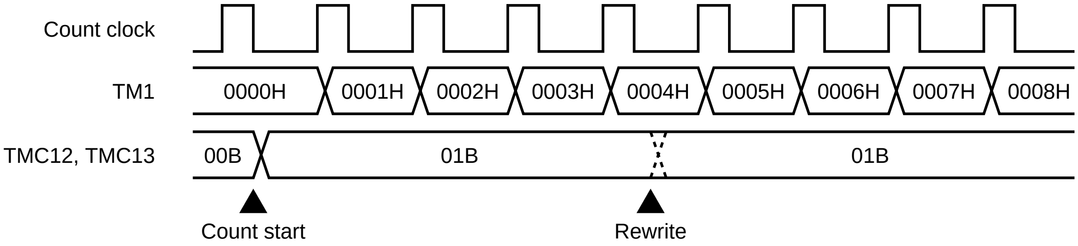


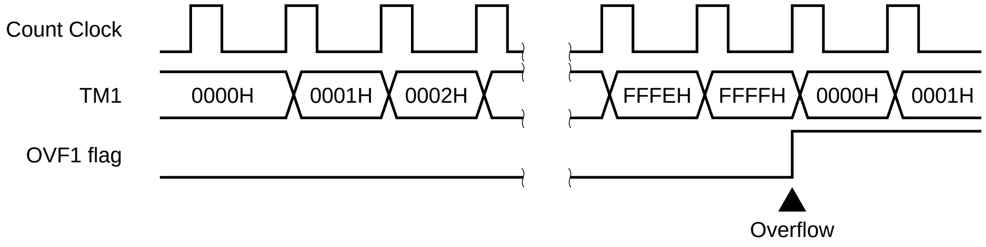
Figure 9-6. Timing for Rewriting to TMC12 and TMC13 (Free Running Mode)



9.4.2 Free running operation of TM1

If bits 2 and 3 (TMC12, TMC13) of 16-bit timer mode control register 1 (TMC1) are set to “1” and “0”, respectively, TM1 performs free-running operation. If TM1 has a full count by FFFFH, bit 0 of TMC1 (OVF1) is set to “1” at the next count clock, and TM1 is cleared. Counting continues after that, and it is possible to clear OVF1 with an instruction.

Figure 9-7. Free Running Mode Operation Timing of TM1



9.4.3 Clear and start mode operation when TM1 and CR10 match

If bits 2 and 3 (TMC12, TMC13) of 16-bit timer mode control register 1 (TMC1) are set to “1” and “1”, respectively, TM1 is set to the clear and start mode when 16-bit compare register 10 (CR10) matches. When TM1 and CR10 match, an interrupt request signal (INTTM10) is generated at the next count clock, and TM1 is cleared (0000H).

Figure 9-8. Clear and Start Mode Operation Timing When TM1 and CR10 Match (CR10 ≠ 0000H)

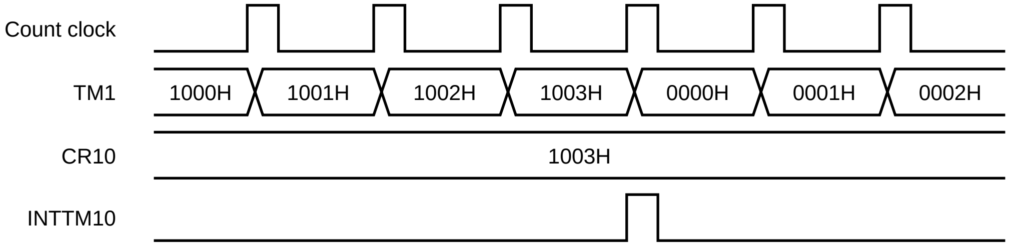
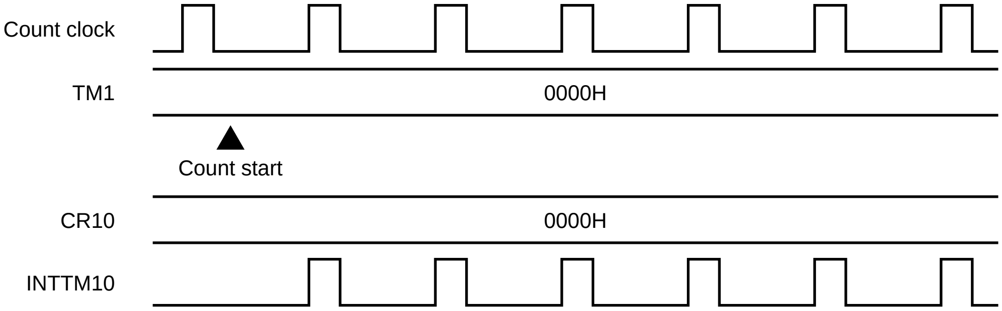


Figure 9-9. Clear and Start Mode Operation Timing When TM1 and Match CR10 (CR10 = 0000H)



Remark Interval period = (CR10 + 1) × TM1 count-clock rate

9.4.4 Operation as 16-bit PWM output

By setting bit 0 (TOE1) of timer output control register 1 (TOC1) to “1”, it operates as PWM output.

<Setting method>

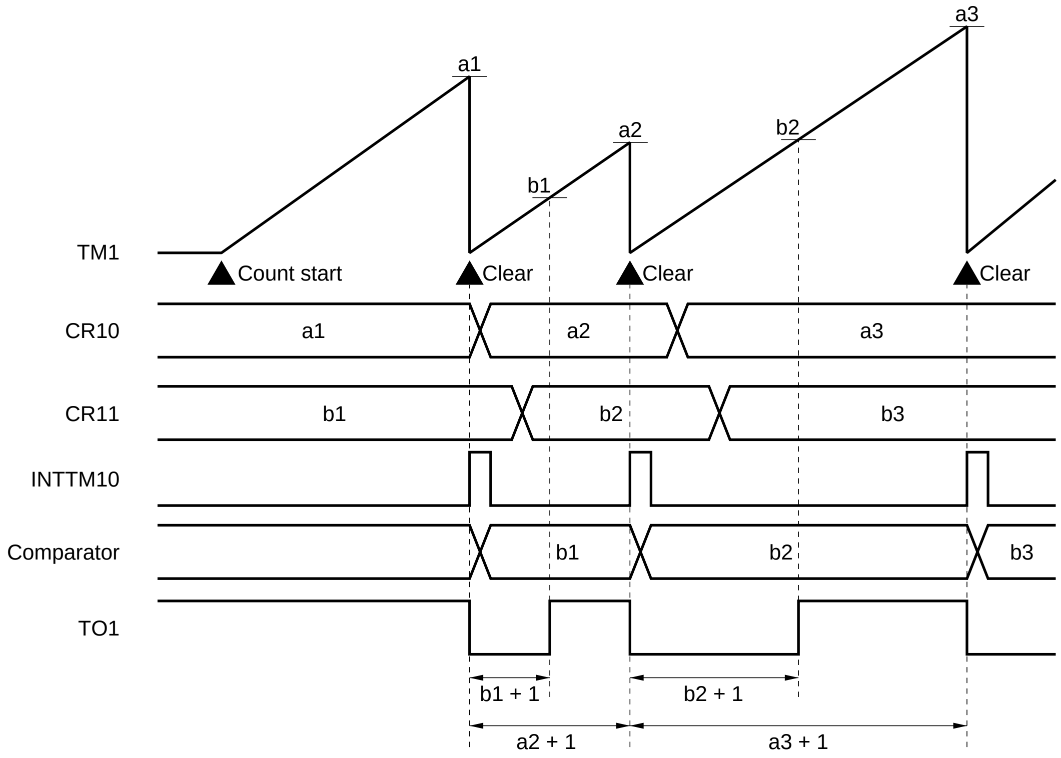
<1> Specify the active level (bit 1 (ALV1) of TOC1) of timer 1 output (TO1), and enable TO1 output (set bit 0 (TOE1) of TOC1 to “1”).

<2> Set the interval period in 16-bit compare register 10 (CR10), and set the active-level width in 16-bit compare register 11 (CR11).

<3> The count clock is selected by bits 0 and 1 (PRM10, PRM11) of prescaler mode register 1 (PRM1).

<4> By setting bits 2 and 3 (TMC12, TMC13) of 16-bit timer mode control register 1 (TMC1) to “1”, the count operation starts, and the PWM signal is output from pin TO1.

Figure 9-10. Example of PWM Output of TO1



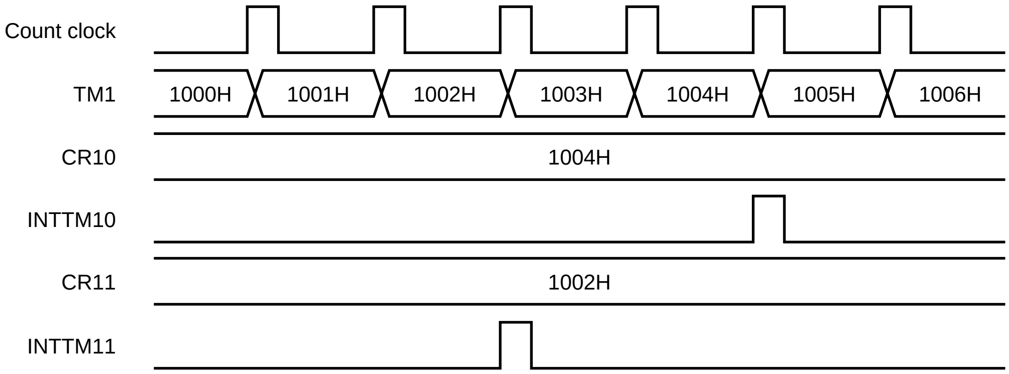
Caution If CR10 = CR11 is set, TO1 outputs inactive level ($\overline{\text{ALV1}}$).
If CR10 < CR11 is set, TO1 outputs active level (ALV1).

Remark CR10 > CR11, Active level: “0”

9.4.5 Compare operation of TM1

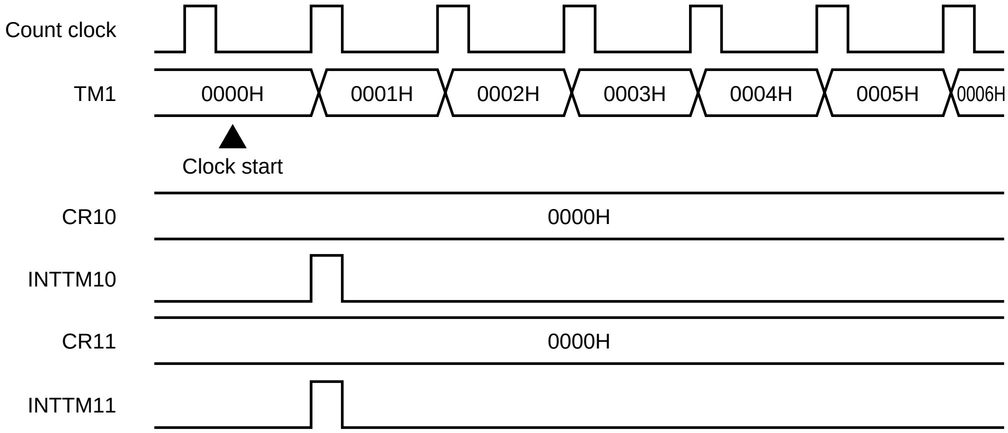
When 16-bit timer register 1 (TM1) matches 16-bit compare register 10 (CR10) or 16-bit compare register 11 (CR11), an interrupt request signal (INTTM10 or INTTM11) is generated at the next count clock.

Figure 9-11. Compare Operation Timing of TM1 (CR10, CR11 ≠ 0000H)



Caution The operating mode of TM1 shown in Figure 9-11 is a mode other than the clear and start mode when TM1 and CR10 match.
In the case of the clear and start mode when TM1 and CR10 match, TM1 is cleared at the next count clock after TM1 and CR10 match. (See Figure 9-8. Clear and Start Mode Operation Timing When TM1 and CR10 Match (CR10 ≠ 0000H).)

Figure 9-12. Compare Operation Timing of TM1 (CR10, CR11 = 0000H)



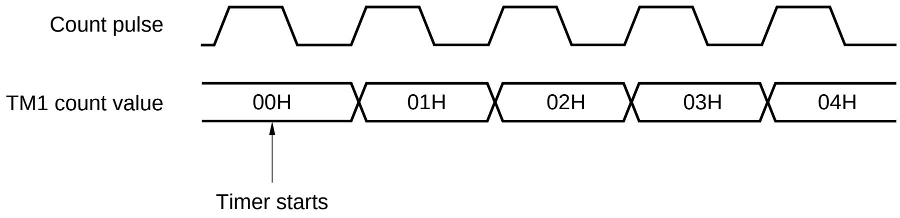
Caution The operating mode of TM1 shown in Figure 9-12 is a mode other than the clear and start mode when TM1 and CR10 match.
In the case of the clear and start mode when TM1 and CR10 match, TM1 remains 0000H. (See Figure 9-9. Clear and Start Mode Operation Timing When TM1 and CR10 Match (CR10 = 0000H).)

★
9.5 Cautions

(1) Error at timer start

After the timer starts, the time until a uniform signal is generated may have a maximum error of 1 clock. This is because the start of 16-bit timer register 1 (TM1) is asynchronous with respect to the count pulse.

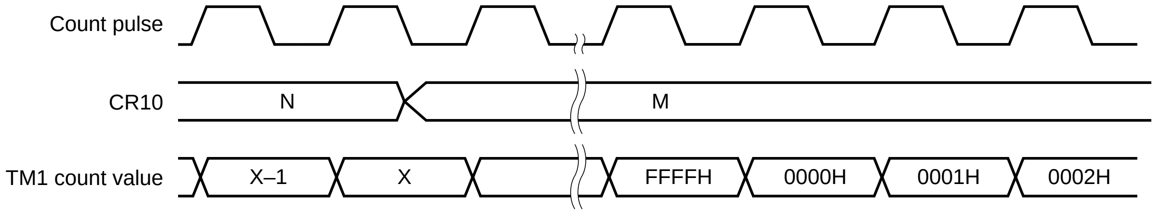
Figure 9-13. Start Timing of 16-Bit Timer Register 1



(2) Operation after changes in the compare register during timer count operation

If the value of 16-bit compare register 10 (CR10) after changing is less than the value of 16-bit timer register 1 (TM1), TM1 continues counting, and when an overflow occurs, it starts over from 0. Also, if the value of CR10 after changing (M) is less than the value before changing (N), the timer must be restarted after CR10 changes.

Figure 9-14. Timing After the Compare Register Changes During Timer Counting



Remark N > X > M

[MEMO]

CHAPTER 10 16-BIT TIMER/COUNTER 2

10.1 Functions

16-bit timer/counter 2 (TM2) has the following modes.

- Interval timer
  An interrupt request is generated at an arbitrary time interval that was set in advance.
- PWM output

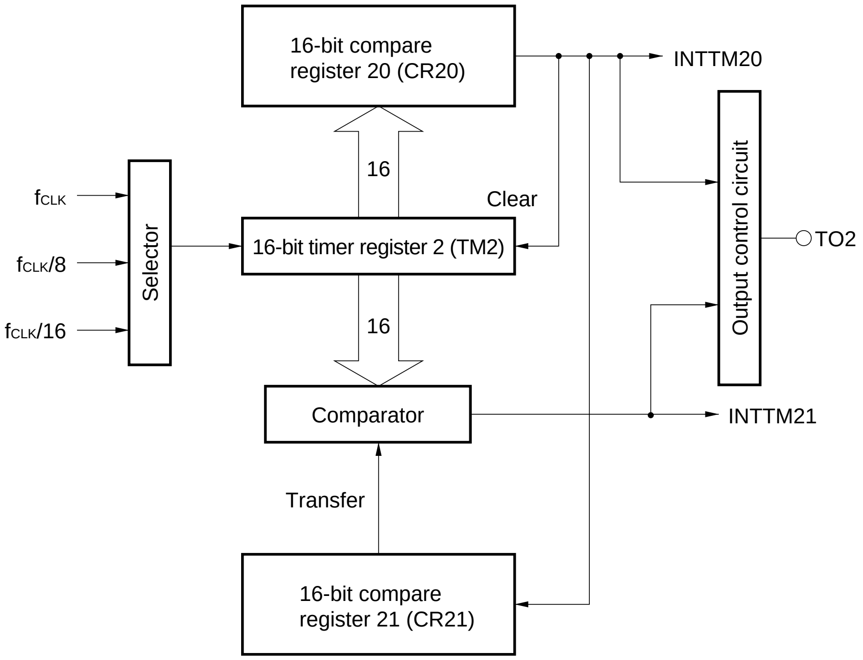
10.2 Configuration

16-bit timer/counter 2 (TM2) is comprised of the following hardware.

Table 10-1. Configuration of 16-Bit Timer/Counter 2 (TM2)

Item	Configuration
Timer register	16-bit × 1 (TM2)
Register	Compare register: 16-bit × 2 (CR20, CR21)
Timer output	1 (TO2)
Control register	16-bit timer mode control register 2 (TMC2) Timer output control register 2 (TOC2) Prescaler mode register 2 (PRM2)

Figure 10-1. Block Diagram of 16-Bit Timer/Counter 2 (TM2)



(1) 16-bit timer register 2 (TM2)

TM2 is a 16-bit free running or interval timer which counts the count pulse. The count is incremented in synchronization with the rise of the input clock.

In the following cases the count value becomes 0000H.

<1> $\overline{\text{RESET}}$ input

<2> When bits 2 and 3 (TMC22, TMC23) of 16-bit timer mode control register 2 (TMC2) are cleared

<3> When TM2 and CR20 match in the clear and start mode at the match of 16-bit compare register 20 (CR20)

(2) 16-bit compare register 20 (CR20)

This register constantly compares the value set for CR20 with the count value of 16-bit timer register 2 (TM2), and if they match, it generates an interrupt request (INTTM20). If TM2 is set as an interval timer, it can also be used as a register for holding the interval time.

CR20 is set by a 16-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input makes CR20 undefined.

(3) 16-bit compare register 21 (CR21)

This register constantly compares the value set for CR21 with the count value of 16-bit timer register 2 (TM2), and if they match, it generates an interrupt request (INTTM21). If TM2 is set as an interval timer, it can also be used as a register for holding the interval time.

CR21 is set by a 16-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input makes CR21 undefined.

10.3 Control Registers

The following three registers control 16-bit timer/counter 2 (TM2).

- 16-bit timer mode control register 2 (TMC2)
- Timer output control register 2 (TOC2)
- Prescaler mode register 2 (PRM2)

(1) 16-bit timer mode control register 2 (TMC2)

TMC2 is a register for detecting the setting and overflow of the clear mode of 16-bit timer register 2 (TM2).
TMC2 are set by a 1-bit or 8-bit memory manipulation instruction.
$\overline{\text{RESET}}$ input sets TMC2 to 00H.

Caution TM2 starts operating when bits 2 and 3 (TMC22, TMC23) of TMC2 are set to values other than “0” and “0” (operation stop mode), respectively. To stop operation, set TMC22 and TMC23 to “0” and “0”, respectively.

Figure 10-2. Format of the 16-Bit Timer Mode Control Register 2 (TMC2)

Address: 0FF6CH After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	⓪
TMC2	0	0	0	0	TMC23	TMC22	0	OVF2

TMC23	TMC22	Selection of Operating Mode and Clear Mode	Generation of Interrupt
0	0	Operation stop (TM2 is cleared to 0).	Does not generate
0	1	Free running mode.	Generates on coincidence between TM2 and CR20.
1	0	Setting prohibited	
1	1	Clears and starts on coincidence between TM2 and CR20.	

OVF2	Detection of Overflow of TM2
0	Does not overflow
1	Overflow

Caution TMC2 should be set after TM2 timer operation has stopped.

(2) Timer output control register 2 (TOC2)

This is a register for controlling the operation of the 16-bit timer/counter 2 (TM2) output control circuit. It sets or resets the R-S type flip-flop (ALV2), enables or disables output inversion, and enables or disables TM1 timer output.

TOC2 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets TOC2 to 00H.

Figure 10-3. Format of the Timer Output Control Register 2 (TOC2)

Address: 0FF7CH After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	⓪
TOC2	0	0	0	0	0	0	ALV2	TOE2

ALV2	Specify of Active Level
0	Active level “0” (low)
1	Active level “1” (high)

TOE2	Output Control of 16-Bit Timer/Counter 2 (TM2)
0	Disable output (output is set to 0 level)
1	Enable output

Cautions 1. TOC2 should be set after TM2 timer operation has been stopped.

2. If PWM is to be output from TM2, bits 2 and 3 (TMC22, TMC23) of 16-bit timer mode control register 2 (TMC2) should be set to “1” and “1”, respectively. If TMC22 and TMC23 are set to “0” and “0”, respectively, TO2 becomes inactive.

Remark Setting and resetting of the timer output is controlled by INTTM20 (set signal) and INTTM21 (reset signal).

(3) Prescaler mode register 2 (PRM2)

Prescaler mode register 2 (PRM2) is a register for specifying the count clock of 16-bit timer register 2 (TM2).
PRM2 is set by a 1-bit or 8-bit memory manipulation instruction.
$\overline{\text{RESET}}$ input sets PRM2 to 00H.

Figure 10-4. Format of the Prescaler Mode Register 2 (PRM2)

Address: 0FF86H After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
PRM2	0	0	0	0	0	0	PRM21	PRM20

PRM21	PRM20	Count Clock Selection
0	0	f_{CLK} (8 MHz)
0	1	f_{CLK}/8 (1 MHz)
1	0	f_{CLK}/16 (500 kHz)
1	1	Setting prohibited

Caution PRM2 should only be set after TM2 timer operation has stopped.

Remark Figures in parentheses apply to operation with f_{CLK} = 8 MHz.

10.4 Operation

10.4.1 Basic operation of TM2

16-bit timer/counter 2 (TM2) is a 16-bit free running or interval timer that counts the count pulse. Its count is incremented in synchronization with the rising edge of the input clock.

All of the bits of TM2 are cleared (0) by $\overline{\text{RESET}}$ input, and the count operation stops.

Enabling or disabling the count operation is controlled by bits 2 and 3 (TMC22, TMC23) of 16-bit timer mode control register 2 (TMC2). If TMC22 and TMC23 are set to an operation mode other than “0” and “0”, the count operation starts, and when they are reset (TMC22 and TMC23 are set to “0” and “0”), TM2 is cleared and the count operation stops.

Also, the count value becomes 0000H.

TM2 changes from 0000H to 0001H at the first count clock input after the count-start setting.

TM2 continues operating as is even if the same operating mode is set during operation, and the timer is not cleared.

Count does not stop even during the TM2 read period.

Figure 10-5. Basic Operation Timing of TM2

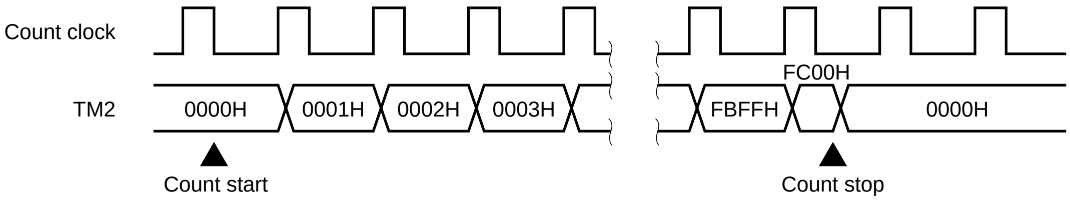
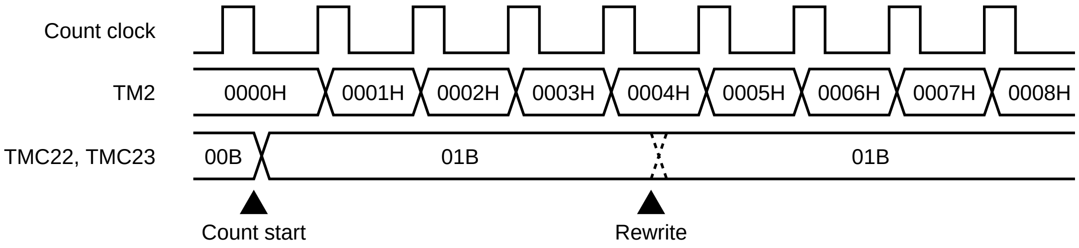


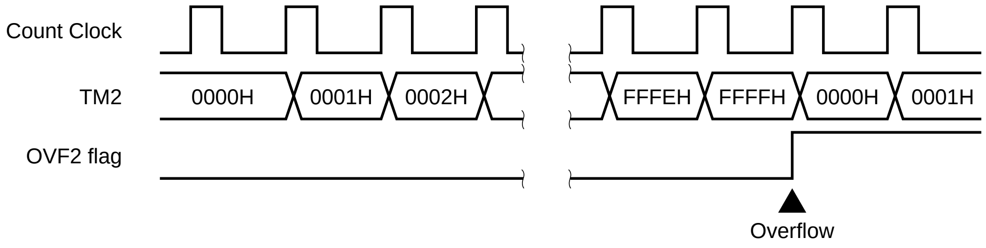
Figure 10-6. Timing for Rewriting to TMC22 and TMC23 (Free Running Mode)



10.4.2 Free running operation of TM2

If bits 2 and 3 (TMC22, TMC23) of 16-bit timer mode control register 2 (TMC2) are set to “1” and “0”, respectively, TM2 performs free running operation. If TM2 has a full count by FFFFH, bit 0 of TMC2 (OVF2) is set to “1” at the next count clock, and TM2 is cleared. Counting continues after that, and it is possible to clear OVF2 using an instruction.

Figure 10-7. Free Running Mode Operation Timing of TM2



10.4.3 Clear and start mode operation timing when TM2 and CR20 match

If bits 2 and 3 (TMC22, TMC23) of 16-bit timer mode control register 2 (TMC2) are set to “1” and “1”, respectively, TM2 is set to the clear and start mode when 16-bit compare register 20 (CR20) matches. When TM2 and CR20 match, an interrupt request signal (INTTM20) is generated at the next count clock, and TM2 is cleared (0000H). Counting continues after that.

Figure 10-8. Clear and Start Mode Operation Timing When TM2 and CR20 Match (CR20 ≠ 0000H)

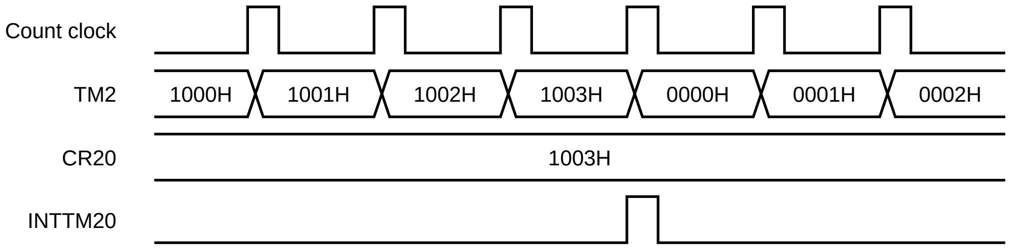
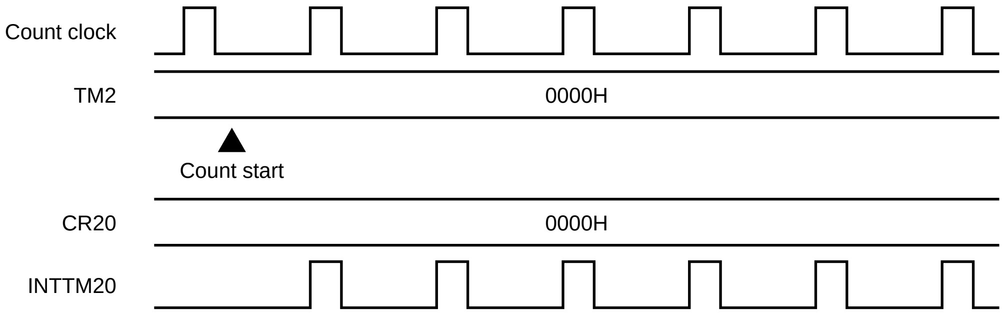


Figure 10-9. Clear and Start Mode Operation Timing When TM2 and CR20 Match (CR20 = 0000H)



Remark Interval period = (CR20 + 1) × TM2 count-clock rate

10.4.4 Operation as 16-bit PWM output

By setting bit 0 (TOE2) of timer-output-control register 2 (TOC2) to “1”, it operates as PWM output.

<Setting method>

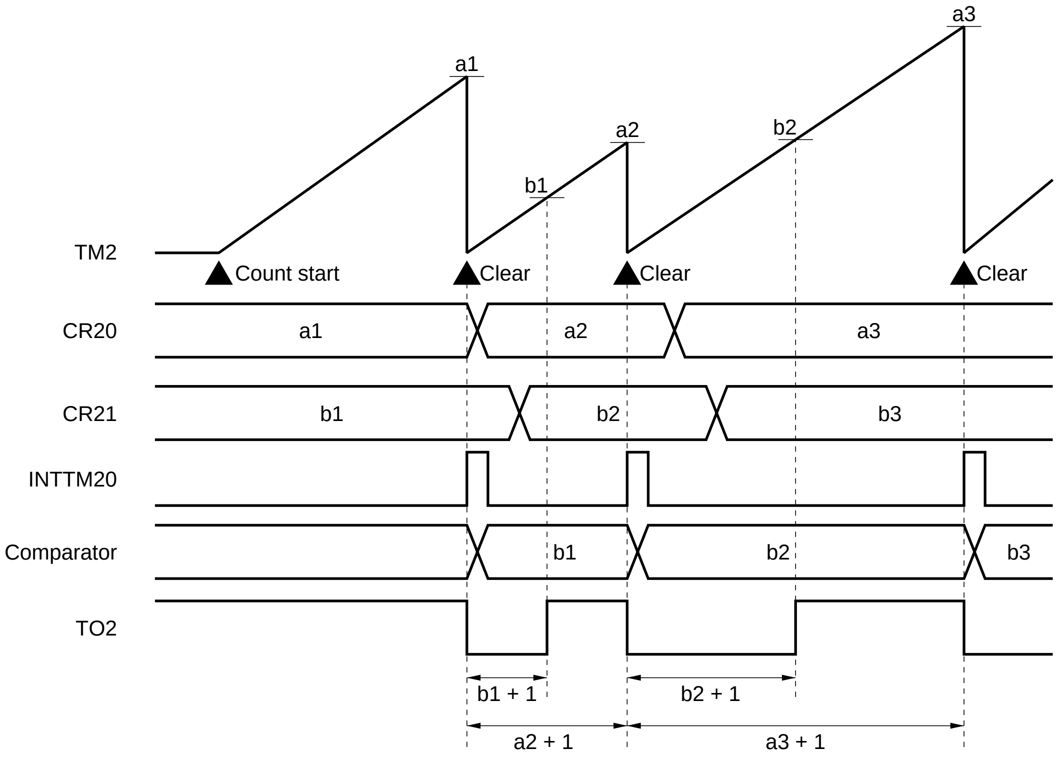
<1> Specify the active level (bit 2 (ALV2) of TOC2) of timer 2 output (TO2), and enable TO2 output (set bit 0 (TOE2) of TOC2 to “1”).

<2> Set the interval period in 16-bit compare register 20 (CR20), and set the active-level width in 16-bit compare register 21 (CR21).

<3> The count clock is selected by bits 0 and 1 (PRM20, PRM21) of prescaler mode register 2 (PRM2).

<4> By setting bits 2 and 3 (TMC22, TMC23) of 16-bit timer mode control register 2 (TMC2) to “1”, the count operation starts, and the PWM signal is output from pin TO2.

Figure 10-10. Example of PWM Output of TO2



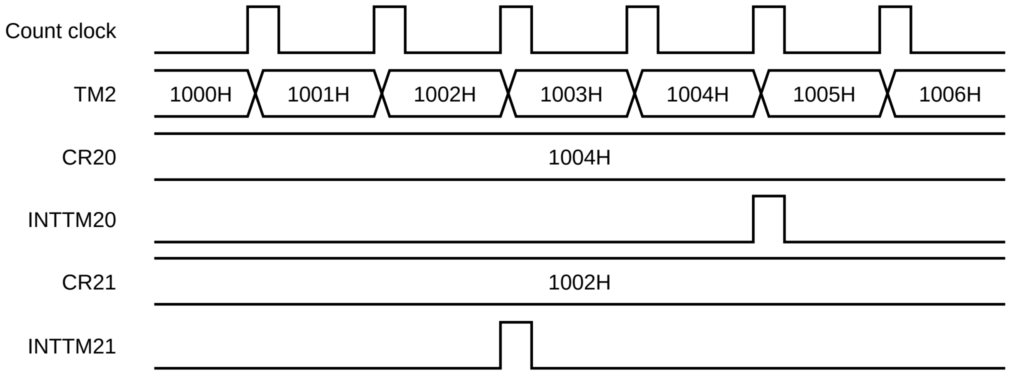
Caution If CR20 = CR21 is set, TO2 outputs inactive level ($\overline{\text{ALV2}}$).
If CR20 < CR21 is set, TO2 outputs active level (ALV2).

Remark CR20 > CR21, Active level: “0”

10.4.5 Compare operation of TM2

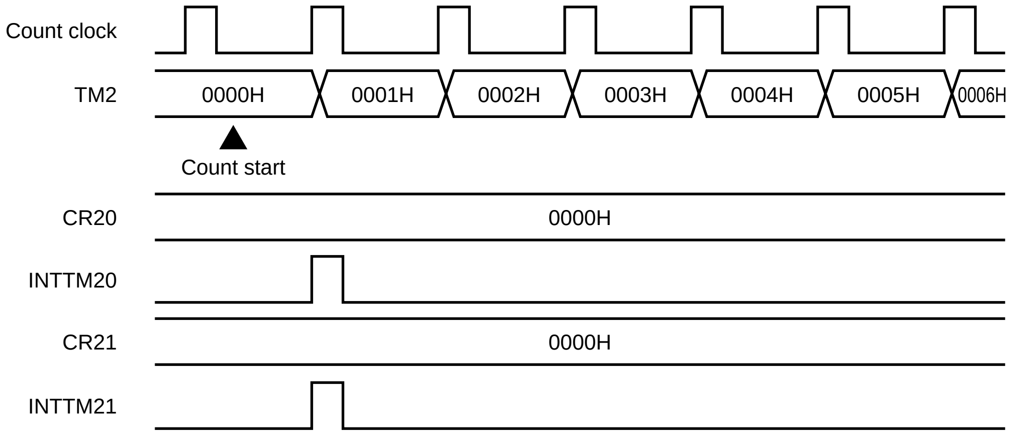
When 16-bit timer register 2 (TM2) matches 16-bit compare register 20 (CR20) or 16-bit compare register 21 (CR21), an interrupt request signal (INTTM20 or INTTM21) is generated at the next count clock.

Figure 10-11. Compare Operation Timing of TM2 (CR20, CR21 ≠ 0000H)



Caution The operating mode of TM2 shown in Figure 10-11 is a mode other than the clear and start mode when TM2 and CR20 match.
In the case of the clear and start mode when TM2 and CR20 match, TM2 is cleared at the next count clock after TM2 and CR20 match. (See Figure 10-8. Clear and Start Mode Operation Timing When TM2 and CR20 Match (CR20 ≠ 0000H).) Even if CR21 and TM2 match, TM2 is not cleared.

Figure 10-12. Compare Operation Timing of TM2 (CR20, CR21 = 0000H)



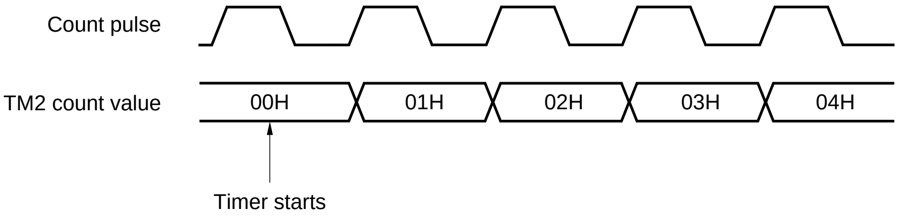
Caution The operating mode of TM2 shown in Figure 10-12 is a mode other than the clear and start mode when TM2 and CR20 match.
In the case of the clear and start mode when TM2 and CR20 match, TM2 remains 0000H. (See Figure 10-9. Clear and Start Mode Operation Timing When TM2 and CR20 Match (CR20 = 0000H).)

★ 10.5 Cautions

(1) Error at timer start

After the timer starts, the time until a uniform signal is generated has a maximum error of 1 clock. This is because the start of 16-bit register 2 (TM2) is asynchronous with respect to the count pulse.

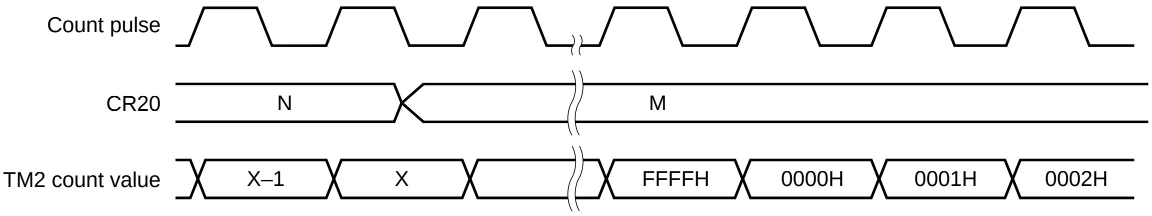
Figure 10-13. Start Timing of 16-Bit Timer Register 2



(2) Operation after changes in the compare register during timer count operation

If the value of 16-bit compare register 20 (CR20) after changing is less than the value of 16-bit timer register 2 (TM2), TM2 continues counting, and when an overflow occurs, it starts over from 0. Therefore, if the value of CR20 after changing (M) is less than the value before changing (N), the timer must be restarted after CR20 changes.

Figure 10-14. Timing After the Compare Register Changes During Timer Counting



Remark N > X > M

CHAPTER 11 16-BIT TIMER/COUNTER 3

11.1 Functions

16-bit timer/counter 3 (TM3) has the following function.

- Interval timer
  An interrupt request is generated at an arbitrary time interval that was set in advance.

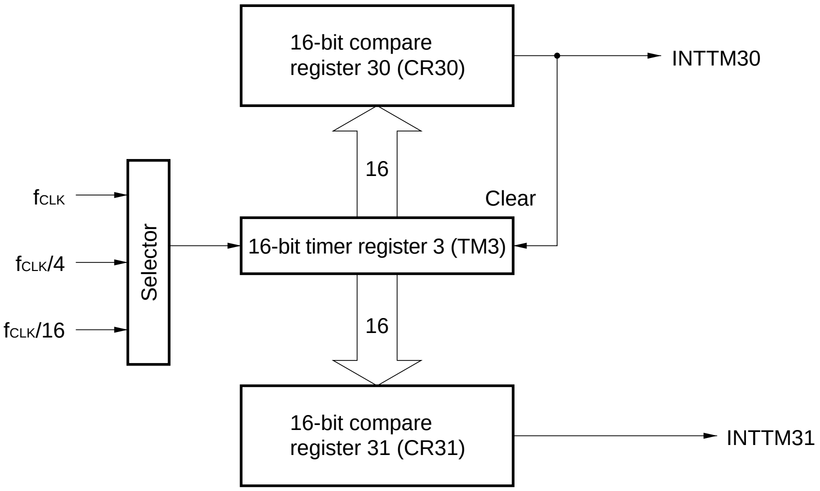
11.2 Configuration

16-bit timer/counter 3 (TM3) is comprised of the following hardware.

Table 11-1. Configuration of 16-Bit Timer/Counter 3 (TM3)

Item	Configuration
Timer register	16-bit × 1 (TM3)
Register	Compare register: 16-bit × 2 (CR30, CR31)
Control register	16-bit timer mode control register 3 (TMC3) Prescaler mode register 3 (PRM3)

Figure 11-1. Block Diagram of 16-Bit Timer/Counter 3 (TM3)


16-bit compare register 30 (CR30)
INTTM30
16
Clear
fCLK
fCLK/4
fCLK/16
Selector
16-bit timer register 3 (TM3)
16
16-bit compare register 31 (CR31)
INTTM31

(1) 16-bit timer register 3 (TM3)

TM3 is a 16-bit free running or interval timer which counts the count pulse. The count is incremented in synchronization with the rise of the input clock.

In the following cases the count value becomes 0000H.

<1> $\overline{\text{RESET}}$ input

<2> When bits 2 and 3 (TMC32, TMC33) of 16-bit timer mode control register 3 (TMC3) are cleared.

<3> When TM3 and CR30 match in the clear and start mode at the match of 16-bit compare register 30 (CR30).

(2) 16-bit compare register 30 (CR30)

This register constantly compares the value set for CR30 with the count value of 16-bit timer register 3 (TM3), and if they match, it generates an interrupt request (INTTM30). If TM3 is set as an interval timer, it can also be used as a register for holding the interval time.

CR30 is set by a 16-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input makes CR30 undefined.

(3) 16-bit compare register 31 (CR31)

This register constantly compares the value set for CR31 with the count value of TM3, and if they match, it generates an interrupt request (INTTM31). If TM3 is set as an interval timer, it can also be used as a register for holding the interval time.

CR31 is set by a 16-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input makes CR31 undefined.

11.3 Control Registers

There are two kinds of registers for controlling 16-bit timer/counter 3 (TM3).

- 16-bit timer mode control register 3 (TMC3)
- Prescaler mode register 3 (PRM3)

(1) 16-bit timer mode control register 3 (TMC3)

TMC3 is a register for detecting the setting and overflow of the clear mode of 16-bit timer register 3 (TM3).
TMC3 is set by a 1-bit or 8-bit memory manipulation instruction.
$\overline{\text{RESET}}$ input sets TMC3 to 00H.

Caution TM3 starts operating when bits 2 and 3 (TMC32, TMC33) of TMC3 are set to values other than "0" and "0" (operation stop mode), respectively. To stop operation, set TMC32 and TMC33 to "0" and "0", respectively.

Figure 11-2. Format of 16-Bit Timer Mode Control Register 3 (TMC3)

Address: 0FF6DH After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	⓪
TMC3	0	0	0	0	TMC33	TMC32	0	OVF3

TMC33	TMC32	Operating Mode and Clear Mode Selection	Generation of Interrupt
0	0	Operation stop (TM3 is cleared to 0).	Does not generate.
0	1	Free running mode.	Generated on coincidence between TM3 and CR30.
1	0	Setting prohibited.	
1	1	Clear and start on coincidence between TM3 and CR30.	

OVF3	Detection of Overflow of TM3
0	Does not overflow
1	Overflow

Caution TMC3 should be set after TM3 timer operation has stopped.

(2) Prescaler mode register 3 (PRM3)

Prescaler mode register 3 (PRM3) is a register for specifying the count clock of 16-bit timer register 3 (TM3).

PRM3 is set by a 1-bit or 8-bit memory manipulation instructions.

$\overline{\text{RESET}}$ input sets PRM3 to 00H.

Figure 11-3. Format of Prescaler Mode Register 3 (PRM3)

Address: 0FF87H After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
PRM3	0	0	0	0	0	0	PRM31	PRM30

PRM31	PRM30	Count Clock Selection
0	0	f_{CLK} (8 MHz)
0	1	f_{CLK}/4 (2 MHz)
1	0	f_{CLK}/16 (500 kHz)
1	1	Setting prohibited.

Caution PRM3 should be set after TM3 timer operation has stopped.

Remark Values inside parentheses ( ) are for f_{CLK} = 8 MHz.

11.4 Operation

11.4.1 Basic operation of TM3

16-bit timer/counter 3 (TM3) is a 16-bit free running or interval timer that counts the count pulse. Its count is incremented in synchronization with the rise of the input clock.

All of the bits of TM3 are cleared (0) by $\overline{\text{RESET}}$ input, and the count operation stops.

Enabling or disabling the count operation is controlled by bits 2 and 3 (TMC32, TMC33) of 16-bit timer mode control register 3 (TMC3). If TMC32 and TMC33 are set to an operation mode other than "0" and "0", the count operation starts, and when they are reset (TMC32 and TMC33 are set to "0" and "0"), TM3 is cleared and the count operation stops.

The count value becomes 0000H.

TM3 changes from 0000H to 0001H at the first count clock input after the count-start setting. TM3 continues operating as is even if the same operating mode is set during operation, and the timer is not cleared.

Count does not stop even during the TM3 read period.

Figure 11-4. Basic Operation Timing of TM3

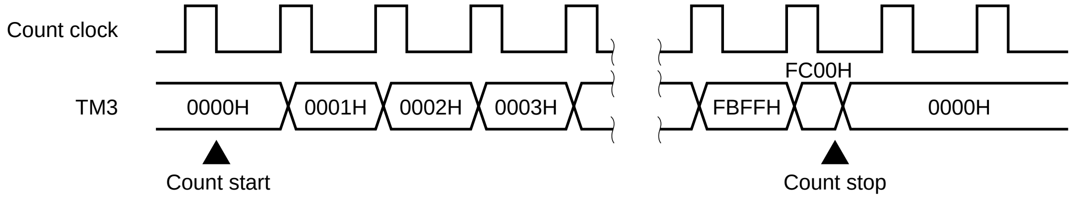
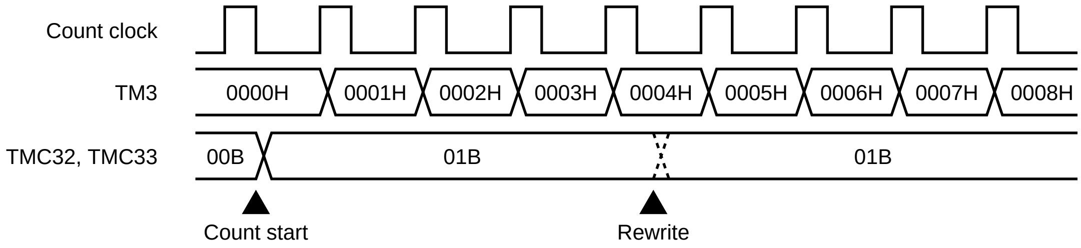


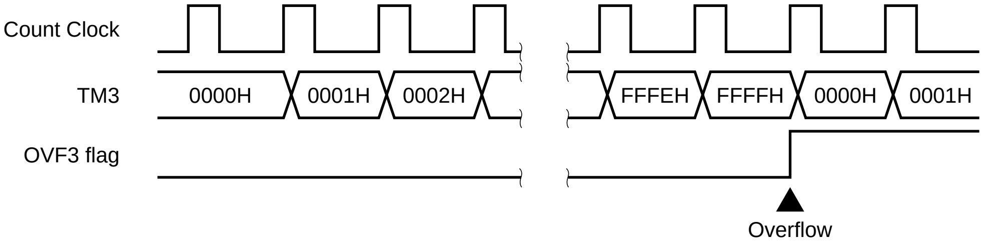
Figure 11-5. Timing for Rewriting to TMC32 and TMC33 (Free Running Mode)



11.4.2 Free running operation of TM3

If bits 2 and 3 (TMC32, TMC33) of 16-bit timer mode control register 3 (TMC3) are set to “1” and “0”, respectively, TM3 performs free running operation. If TM3 has a full count by FFFFH, bit 0 of TMC3 (OVF3) is set to “1” at the next count clock, and TM3 is cleared. Counting continues after that, and it is possible to clear OVF3 with an instruction.

Figure 11-6. Free Running Mode Operation Timing of TM3



11.4.3 Clear and start operation when TM3 and CR30 match

If bits 2 and 3 (TMC32, TMC33) of 16-bit timer mode control register 3 (TMC3) are set to “1” and “1”, respectively, TM3 is set to the clear and start mode when 16-bit compare register 30 (CR30) matches. When TM3 and CR30 match, an interrupt request signal (INTTM30) is generated at the next count clock, and TM3 is cleared (0000H). Counting continues after that.

Figure 11-7. Clear and Start Mode Operation Timing When TM3 and CR30 Match (CR30 ≠ 0000H)

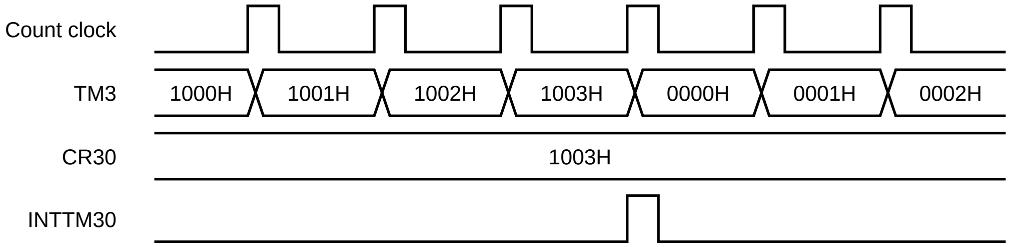
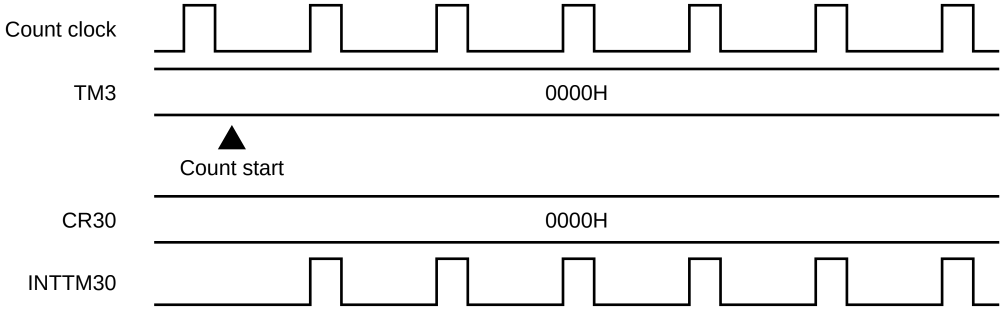


Figure 11-8. Clear and Start Mode Operation Timing When TM3 and CR30 Match (CR30 = 0000H)

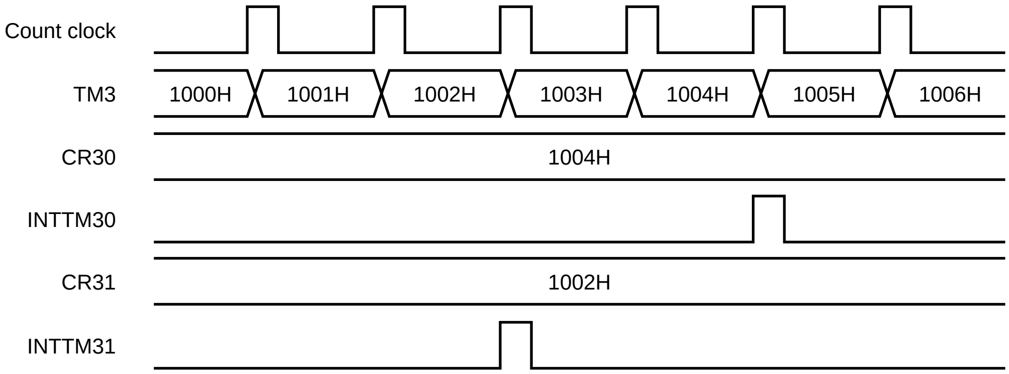


Remark Interval period = (CR30 + 1) × TM3 count-clock rate

11.4.4 Compare operation of TM3

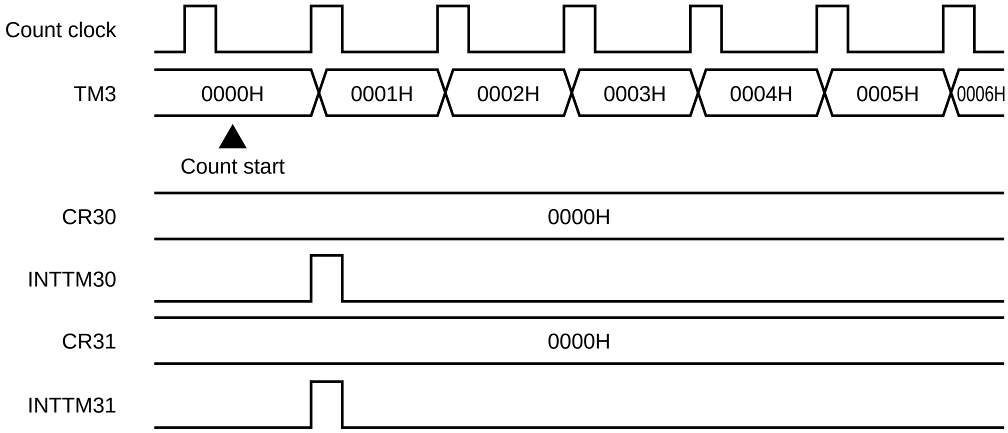
When 16-bit timer register 3 (TM3) matches 16-bit compare register 30 (CR30) or 16-bit compare register 31 (CR31), an interrupt request signal (INTTM30 or INTTM31) is generated at the next count clock.

Figure 11-9. Compare Operation Timing of TM3 (CR30, CR31 ≠ 0000H)



Caution The operating mode of TM3 shown in Figure 11-9 is a mode other than the clear and start mode when TM3 and CR30 match.
In the case of the clear and start mode when TM3 and CR30 match, TM3 is cleared at the next count clock after TM3 and CR30 match. (See Figure 11-7. Clear and Start Mode Operation Timing When TM3 and CR30 Match (CR30 ≠ 0000H).)

Figure 11-10. Compare Operation Timing of TM3 (CR30, CR31 = 0000H)



Caution The operating mode of TM3 shown in Figure 11-10 is a mode other than the clear and start mode when TM3 and CR30 match.
In the case of the clear and start mode when TM3 and CR30 match, TM3 remains 0000H. (See Figure 11-8. Clear and Start Mode Operation Timing When TM3 and CR30 Match (CR30 = 0000H).)

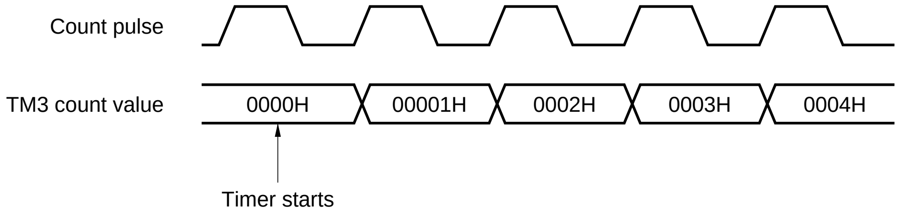
★

11.5 Cautions

(1) Error at timer start

After the timer starts, the time until a uniform signal is generated may have a maximum error of 1 clock. This is due to the fact that the start of 16-bit timer register 3 (TM3) is asynchronous with respect to the count pulse.

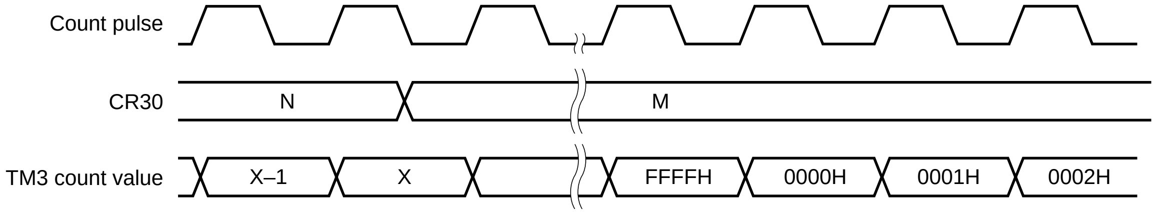
Figure 11-11. Start Timing of 16-Bit Timer Register 3



(2) Operation after changes in the compare register during timer count operation

If the value of 16-bit compare register 30 (CR30) after changing is less than the value of 16-bit timer register 3 (TM3), TM3 continues counting, and when an overflow occurs, it starts over from 0. Therefore, if the value of CR30 after changing (M) is less than the value before changing (N), the timer must be restarted after CR30 changes.

Figure 11-12. Timing After the Compare Register Changes During Timer Counting



Remark N > X > M

[MEMO]

CHAPTER 12 16-BIT TIMER/COUNTER 4

12.1 Function

16-bit timer/counter 4 (TM4) has the following function:

- Interval timer
  An interrupt request is generated at a time interval set in advance.

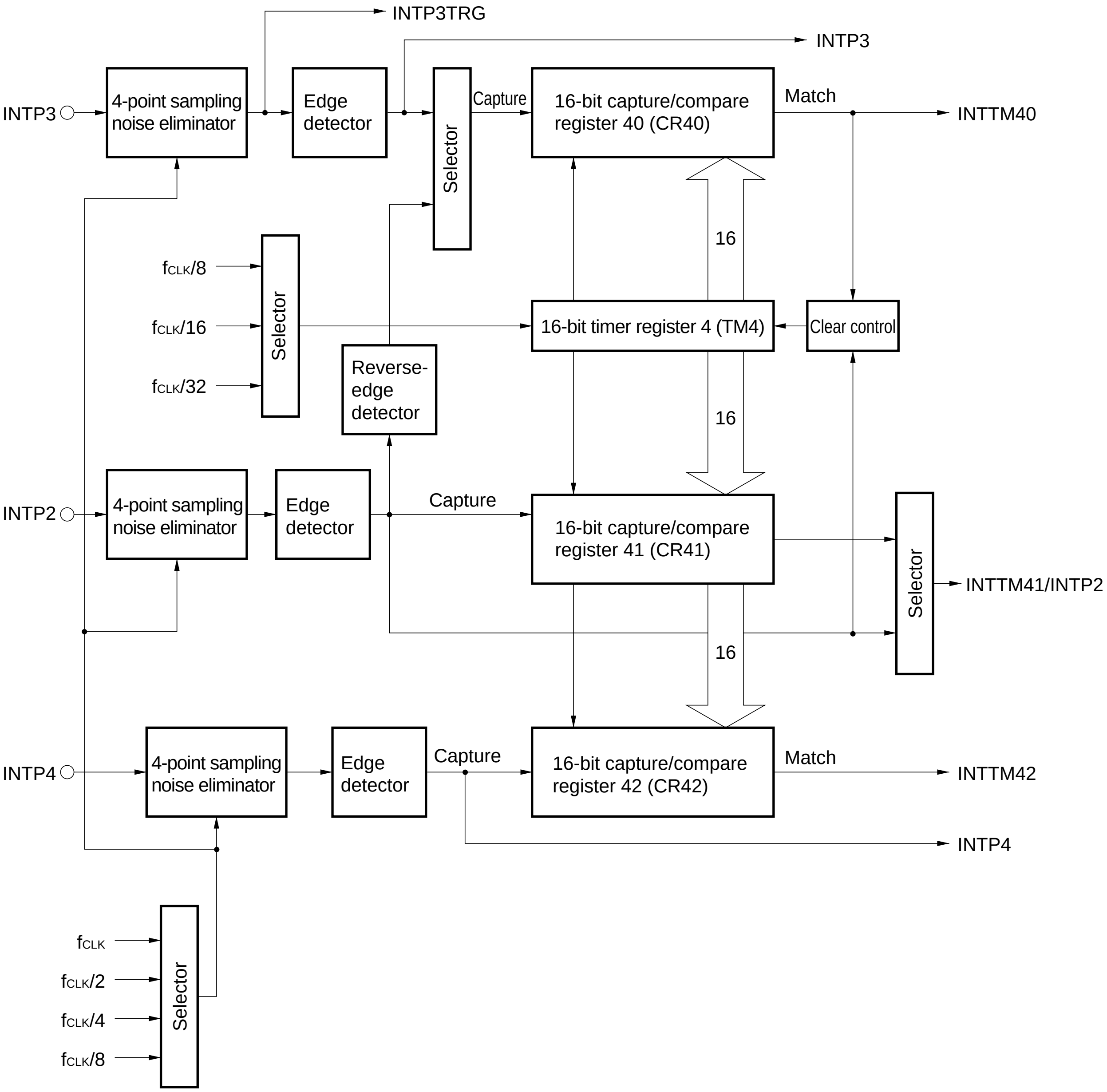
12.2 Configuration

The 16-bit timer/counter 4 (TM4) has the hardware configuration shown below.

Table 12-1. Configuration of 16-Bit Timer/Counter 4 (TM4)

Item	Configuration
Timer register	16 bits × 1 (TM4)
Register	Capture/compare register: 16 bits × 3 (CR40, CR41, CR42)
Control register	16-bit timer mode control register 4 (TMC4) Capture/compare control register 4 (CRC4) Prescaler mode register 4 (PRM4)

Figure 12-1. Block Diagram of 16-Bit Timer/Counter 4 (TM4)


INTP3TRG
INTP3
INTP3
4-point sampling noise eliminator
Edge detector
Selector
Capture
16-bit capture/compare register 40 (CR40)
Match
INTTM40
16
fCLK/8
fCLK/16
fCLK/32
Selector
16-bit timer register 4 (TM4)
Clear control
Reverse-edge detector
16
INTP2
4-point sampling noise eliminator
Edge detector
Capture
16-bit capture/compare register 41 (CR41)
Selector
INTTM41/INTP2
16
INTP4
4-point sampling noise eliminator
Edge detector
Capture
16-bit capture/compare register 42 (CR42)
Match
INTTM42
INTP4
fCLK
fCLK/2
fCLK/4
fCLK/8
Selector

(1) 16-bit timer register 4 (TM4)

The TM4 is a 16-bit free running or interval timer that counts count pulses. The counter is incremented in synchronization with the rising edge of the input clock pulse. The count value becomes 0000H during the following.

<1> $\overline{\text{RESET}}$ input
<2> Clears the 16-bit timer mode control register 4 (TMC4)'s bits 2 and 3 (TMC42 and TMC43).
<3> When the INTP2 valid edge inputs during clear and start mode by INTP2 valid edge input
<4> TM4 and CR40 are the same in the clear and start mode when the 16-bit capture/compare register 40 (CR40) is the same.

(2) 16-bit capture/compare register 40 (CR40)

CR40 is a 16-bit register that has the same functions as a capture register and compare register.
Bit 0 (CRC4) of the capture/compare control register 4 (CRC4) sets the register as to whether it will be used as a capture register or a compare register.

- **When CR40 is used as a compare register**
  It always compares the count value of the 16-bit timer register (TM4) with the values programmed in CR40 and generates an interrupt request when they are the same. When TM4 is programmed to operate as an interval timer, it can also be used as a register that saves interval time.

- **When CR40 is used as a capture register**
  Allows the selection of the valid edge of the signal at pins INTP2 or INTP3 to be used as a capture trigger. The valid edge of INTP2 and INTP3 is set by the prescaler mode register 4 (PRM4)'s bits 4 and 5 (ES20 and ES21) and bits 6 and 7 (ES30 and ES31).
  Table 12-2 shows the situation when the valid edge of the signal on pin INTP2 is specified as the capture trigger, and Table 12-3 shows the situation when the valid edge of the signal on pin INTP3 is specified as the capture trigger.

Table 12-2. Valid Edge of Pin INTP2 and CR40 Capture Trigger

ES21	ES20	Valid Edge of Pin INTP2	Capture Trigger of CR40	Capture Trigger of CR41
0	0	Falling edge	Rising edge	Falling edge
0	1	Rising edge	Falling edge	Rising edge
1	0	Setting prohibited	Setting prohibited	Setting prohibited
1	1	Both rising and falling edge	Capture does not operate	Both rising and falling edge

Table 12-3. Valid Edge of Pin INTP3 and CR40 Capture Trigger

ES31	ES30	Valid Edge of Pin INTP3	Capture Trigger of CR40
0	0	Falling edge	Falling edge
0	1	Rising edge	Rising edge
1	0	Setting prohibited	Setting prohibited
1	1	Both rising and falling edges	Both rising and falling edges

CR40 is set by a 16-bit memory manipulation instruction.
$\overline{\text{RESET}}$ input makes CR40 undefined.

Caution When switching the CR40 from the capture mode to the compare mode, the value for CR40 becomes one that will be ultimately captured. When switching from the compare mode to the capture mode, the value of CR40 becomes the value assigned to the compare register.

(3) 16-bit capture/compare register 41 (CR41)

CR41 is a 16-bit register with combined capture-register and compare-register functions.
Depending on the value of bit 2 (CRC42) of the capture/compare control register 4 (CRC4) sets whether the register is used as a capture register or compare register.

- **When CR41 is used as a compare register.**
  It always compares the count value of the 16-bit timer register (TM4) with the value programmed in CR41, and if they are the same, it generates an interrupt request (INTTM41). When TM4 is set to interval timer operation it can also be used as the register for saving the interval time.

- **When CR41 is used as a capture register.**
  The valid edge of the signal at pin INTP2 can be selected as a capture trigger. The valid edge of INTP2 is programmed by bits 4 and 5 (ES20 and ES21) of the prescaler mode register 4 (PRM4).
  If the valid edge of pin INTP2 is specified as the capture trigger, setting is as shown Table 12-4.

Table 12-4. Valid Edge of Pin INTP2 and CR41 Capture Trigger

ES21	ES20	Valid Edge of INTP2 Pin	Capture Trigger of CR41
0	0	Falling edge	Falling edge
0	1	Rising edge	Rising edge
1	0	Setting prohibited	Setting prohibited
1	1	Both rising and falling edges	Both rising and falling edges

CR41 is set by a 16-bit memory manipulation instruction.
$\overline{\text{RESET}}$ input makes CR41 undefined.

Caution When switching CR41 from the capture mode to the compare mode, the value of CR41 becomes the last captured value. Also, when switching from the compare mode to the capture mode, the value of CR41 becomes the last value that was set in the compare register.

(4) 16-bit capture/compare register 42 (CR42)

CR42 is a 16-bit register with combined capture-register and compare-register functions.

Depending on the value of bit 2 (CRC42) of the capture/compare control register 4 (CRC4) sets whether the register is used as a capture register or compare register.

- **Using CR42 as a compare register**

  Constantly compares the value set on CR42 and the count value of 16-bit timer register 4 (TM4) and if they match, generates an interrupt request (INTTM42). When TM4 is set for interval-timer operation, it can also be used as a register for containing the interval time.

- **Using CR42 as a capture register**

  The valid edge of pin INTP4 can be selected as a capture trigger, and is set by bits 2 and 3 (ES40, ES41) of prescaler mode register 4 (PRM4).

  If the valid edge of pin INTP4 is specified as the capture trigger, setting is as shown in Table 12-5.

Table 12-5. Valid Edge of Pin INTP4 and CR42 Capture Trigger

ES41	ES40	Valid Edge of INTP4 Pin	Capture Trigger of CR42
0	0	Falling edge	Falling edge
0	1	Rising edge	Rising edge
1	0	Setting prohibited	Setting prohibited
1	1	Both rising and falling edges	Both rising and falling edges

CR42 is set by a 16-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input makes CR42 undefined.

Caution When switching CR42 from the capture mode to the compare mode, the value of CR42 becomes the last captured value. Also, when switching from the compare mode to the capture mode, the value of CR42 becomes the last value that was set in the compare register.

12.3 Control Register

The following three types of registers control the 16-bit timer/counter 4 (TM4).

- 16-bit timer mode control register 4 (TMC4)
- Capture/compare control register 4 (CRC4)
- Prescaler mode register 4 (PRM4)

(1) 16-bit timer mode control register 4 (TMC4)

The TMC4 is the register that detects overflow and assigns the clear mode for the 16-bit timer register 4 (TM4).

TMC4 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets TMC4 to 000H.

Caution TM4 starts operating when a value other than "0", "0" (operation stop mode) is set in TMC4's bits 2 and 3 (TMC42 and TMC43). To stop the operation, set "0", "0" in TMC42 and TMC43.

Figure 12-2. Format of 16-Bit Timer Mode Control Register 4 (TMC4)

Address: 0FF6EH After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
TMC4	0	0	0	0	TMC43	TMC42	0	OVF4

TMC43	TMC42	Selection of Operating Mode and Clear Mode	Generation of Interrupt
0	0	Operation stop (TM4 is cleared to 0).	Does not generate.
0	1	Free running mode	Generates on coincidence between TM4 and CR40.
1	0	Clears and starts at valid edge of INTP2.	
1	1	Clears and starts on coincidence between TM4 and CR40.	

0VF0	Detection of Overflow of TM4
0	Does not overflow.
1	Overflows.

Caution Always assign TM4 after stopping TM4's timer operations.

(2) Capture/compare control register 4 (CRC4)

This is the register that controls the operations of the 16-bit capture/compare registers 40, 41 and 42 (CR40, CR41 and CR42).

CRC4 is set by a 1-bit or 8-bit memory manipulation instructions.

$\overline{\text{RESET}}$ input sets CRC4 to 00H.

Figure 12-3. Format of Capture/Compare Control Register 4 (CRC4)

Address: 0FF7DH After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
CRC4	SMPC41	SMPC40	0	0	CRC43	CRC42	CRC41	CRC40

SMPC41	SMPC40	Selection of Sampling Clock
0	0	f_{CLK}
0	1	$f_{CLK}/2$
1	0	$f_{CLK}/4$
1	1	$f_{CLK}/8$

CRC43	Selection of Operation Mode of CR42
0	Operates as compare register.
1	Operates as capture register.

CRC42	Selection of Operation Mode of CR41
0	Operates as compare register.
1	Operates as capture register.

CRC41	Selection of Capture Trigger of CR40
0	Captured at valid edge of INTP3.
1	Captured in reverse phase of valid edge of INTP2.

CRC40	Selection of Operation Mode of CR40
0	Operates as compare register.
1	Operates as capture register.

Cautions 1. Always set CRC4 after stopping timer operations

2. Do not specify CR40 to the capture register when selecting the clear and start mode when TM4 and CR40 match in 16-bit timer mode control register 4 (TMC4).

(3) Prescaler mode register 4 (PRM4)

The prescaler mode register 4 (PRM4) is a register that specifies 16-bit timer register 4's (TM4) count clock and INTP2's, INTP3's and INTP4's valid input edge.

PRM4 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets PRM4 to 00H.

Figure 12-4. Format of Prescaler Mode Register 4 (PRM4)

Address: 0FF88H After Reset : 00H R/W

Symbol	7	6	5	4	3	2	1	0
PRM4	ES31	ES30	ES21	ES20	ES41	ES40	PRM41	PRM40

ES31	ES30	Selection of Valid Edge of INTP3
0	0	Falling edge
0	1	Rising edge
1	0	Setting prohibited.
1	1	Both falling and rising edges

ES21	ES20	Selection of Valid Edge of INTP2
0	0	Falling edge
0	1	Rising edge
1	0	Setting prohibited.
1	1	Both falling and rising edges

ES41	ES40	Selection of Valid Edge of INTP4
0	0	Falling edge
0	1	Rising edge
1	0	Setting prohibited.
1	1	Both falling and rising edges

PRM41	PRM40	Selection of Count Clock
0	0	f_{CLK}/8 (1 MHz)
0	1	f_{CLK}/16 (500 MHz)
1	0	f_{CLK}/32 (250 kHz)
1	1	Setting prohibited.

Cautions 1. Do not set the valid edge of INTP2, INTP3 and INTP4 in the capture trigger and clear and start mode when assigning the valid edge of INTP2, INTP3, and INTP4 to the count cock.

2. Always set PRM4 after stopping timer operations.

Remark Figures in parentheses apply to operation with f_{CLK} = 8 MHz

12.4 Operation

12.4.1 Basic operation of TM4

The 16-bit timer counter 4 (TM4) is a 16-bit free running timer or interval timer for counting the pulse count. The counter is incremented through synchronization with the fall in the input clock pulse.

Inputting the RESET signal clears all TM4 bits (0) and stops the count operation.

The enabling and prohibiting of the count operation, is controlled by the 16-bit timer mode control register 4 (TMC4)'s bits 2 and 3 (TMC42 and TMC43). TMC42 and TMC43 start the count operations when they are set to an operation mode other than "0", "0" resetting TMC42 and TMC43 to "0", "0" clears TM4.

Resetting makes the count value 0000H.

After the count start is set, the first count click input makes TM4 0000H to 0001H.

TM4 does not clear the timer, it continues the count operations even if the same operation mode is set again.

The count does not stop during the TM4 read duration.

Figure 12-5. Basic Operation Timing of TM4

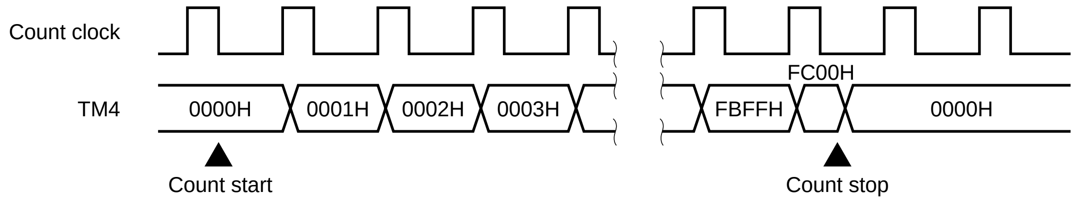
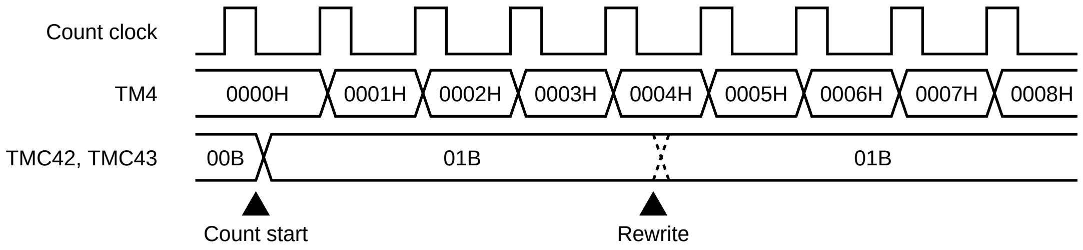


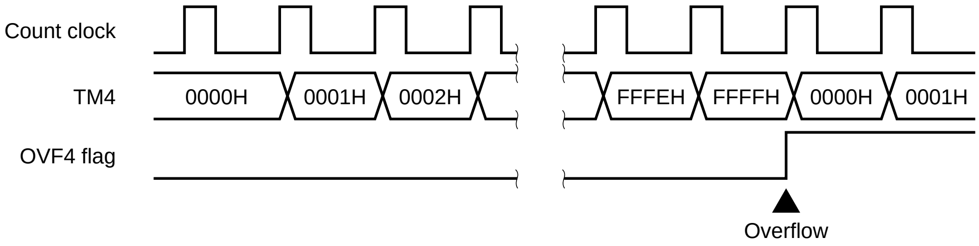
Figure 12-6. TMC42 and TMC43 Rewrite Operation Timing (Free Running Mode)



12.4.2 Free running operation of TM4

Setting the 16-bit timer mode control register 4 (TMC4)'s bits 2 and 3 (TMC42 and TMC43) to "1" and "0", respectively, allows TM4 in free running operation. When the TM4 has made a full count to FFFFH, the next count clock pulse sets TMC4's bit 0 (OVF4) to 1 and clears TM4 (00000H). The count then continues. OVF4 is cleared by command.

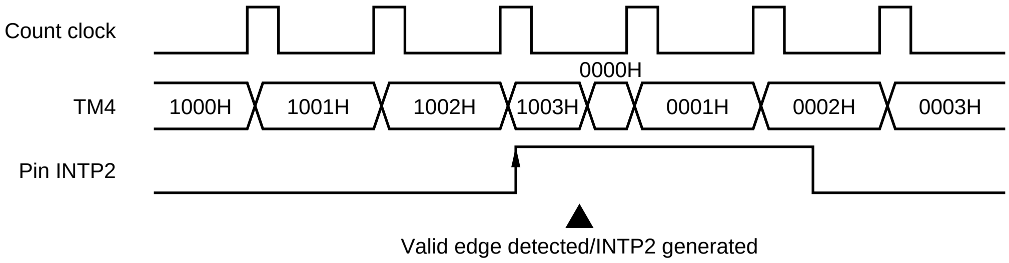
Figure 12-7. Free Running Operation Timing of TM4



12.4.3 Clear and start operation of TM4 at valid edge of INTP2

Setting the 16-bit timer mode control register 4 (TMC4)'s bits 2 and 3 (TMC42 and TMC43) to "1" and "0", respectively, allows the input of INTP2's valid edge to place TM4 in clear-and-start mode. Entering INTP2 valid edge (interrupt request signal: generates INTP2) clears TM4 0000H and the next count clock pulse resets it to 0001H. The count then continues.

Figure 12-8. Clear and Start Mode Operation Timing of TM4 at Input of the Valid Edge of INTP2



12.4.4 Clear and start operation when TM4 and CR40 Match

Setting the 16-bit timer mode control register 4 (TMC4)'s bits 2 and 3 (TMC42 and TMC43) to 1 and 1, respectively, allows a comparison of TM4 and the 16-bit capture/compare register 40 (CR40) that is the same to place TM4 in clear-and-start mode. If TM4 and CR40 are the same, the next count clock pulse generates an interrupt request signal (INTTM40) to clear TM4 (0000H). The count then continues.

Figure 12-9. Clear and Start Mode Operation Timing When TM4 and CR40 Match (CR40 ≠ 0000H)

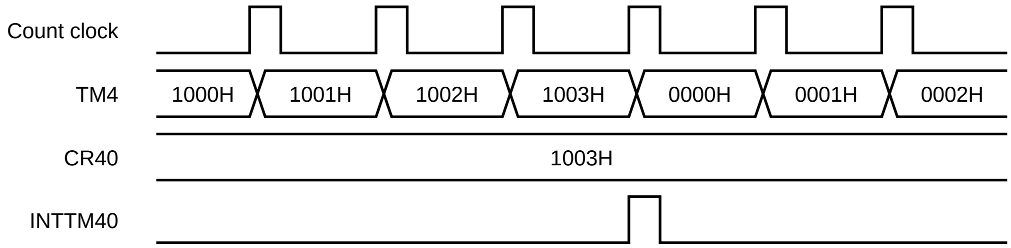
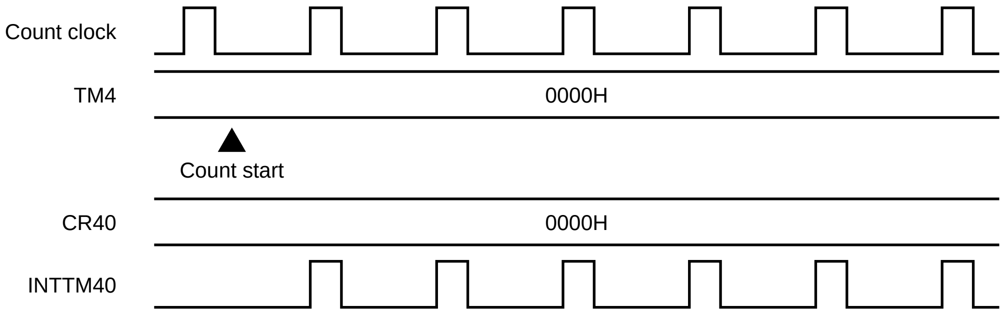


Figure 12-10. Clear and Start Mode Operation Timing When TM4 and CR40 Match (CR40 = 0000H)



Caution Always set CR40 to compare mode.

Remark Interval period = (CR40 + 1) × TM4 count-clock rate

12.4.5 Capture operation of TM4

Setting capture/compare control register 4 (CRC4)'s bits 0, 2 and 3 (CRC40, CRC42 and CRC43) to "1" allows the 16-bit capture/compare registers 40, 41 and 42 (CR40, CR41 and CR42) in capture mode. Inputting the capture trigger captures TM4's values in CR40, CR41 and CR42.

Figure 12-11. Capture Operation Timing (Free Running Mode)

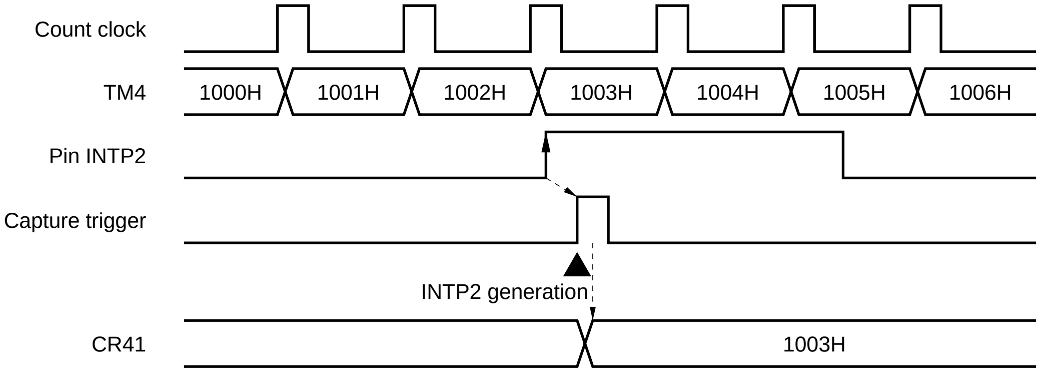
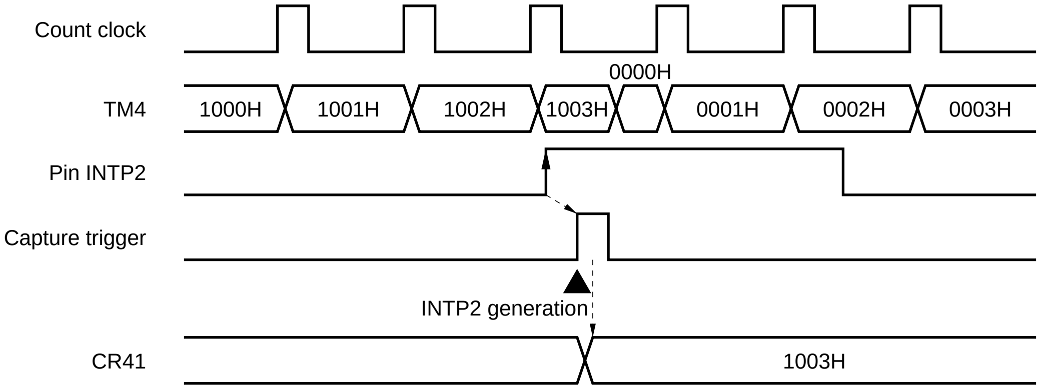


Figure 12-12. Capture Operation Timing (Clear and Start Mode at INTP2 Valid Edge Input)



12.4.6 Pulse width measurement operation

(1) Pulse width measurement (both rising and falling edges)

The 16-bit timer register 4 (TM4) can be used to measure the pulse width of signals input to pins INTP2/P03 , INTP3/P04 and INTP4/P05. The width from edge to edge is measured.

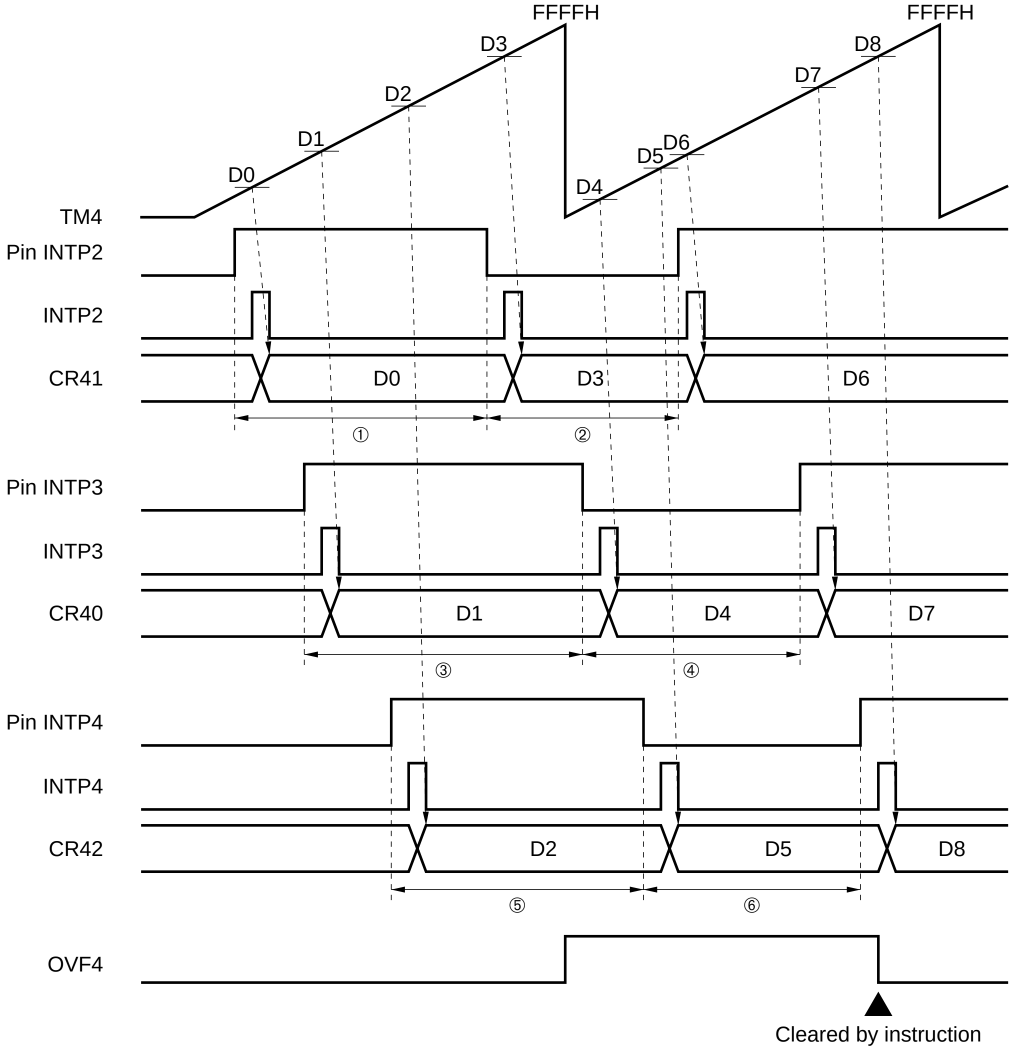
<1> Setting the 16-bit capture/compare registers 40, 41 and 42 (CR40, CR41 and CR42) enables the capture mode (sets the capture/compare control register 4 (CRC4)'s bits 0, 2 and 3 (CRC40, CRC41 and CRC42) to "1")

<2> Sets CR40's capture trigger to INTP3 (sets CRC4's bit 1 (CRC41) to "0").

<3> Sets INTP2, INTP3, INTP4's valid edges for both edges (sets the prescaler mode register 4 (PRM4)'s bits 2 to 7 (ES40, ES41, ES20, ES21, ES30 and RS31) to "1").

<4> Enables the free running mode (sets the 16-bit timer mode control register 4 (TMC4)'s bits 2 and 3 to "1" and "0", respectively (sets TMC42 to "1" and TMC43 to "0")).

Figure 12-13. Pulse Width Measurement Timing (When Both Edges Are Specified)



[Measurement Method]

- INTP2 interrupt processing reads the CR41 and OVF4 (TMC4's bit 0) flags.
  <1> is (D3 – D0) × count-clock rate.
  <2> is (10000H – D3 + D6) × count-clock rate.
- INTP3 interrupt processing reads CR40 and OVF4.
  <3> is (10000H – D1 + D4) × count-clock rate.
  <4> is (D7 – D4) × count-clock rate.
- INTP4 interrupt processing reads CR42 and OVF4.
  <5> is (10000H – D2 + D5) × count-clock rate
  <6> is (D8 – D5) × count-clock rate

Remark Dn: TM4 count value (n = 0, 1, 2, ...)

(2) Pulse width measurement (rising edge)

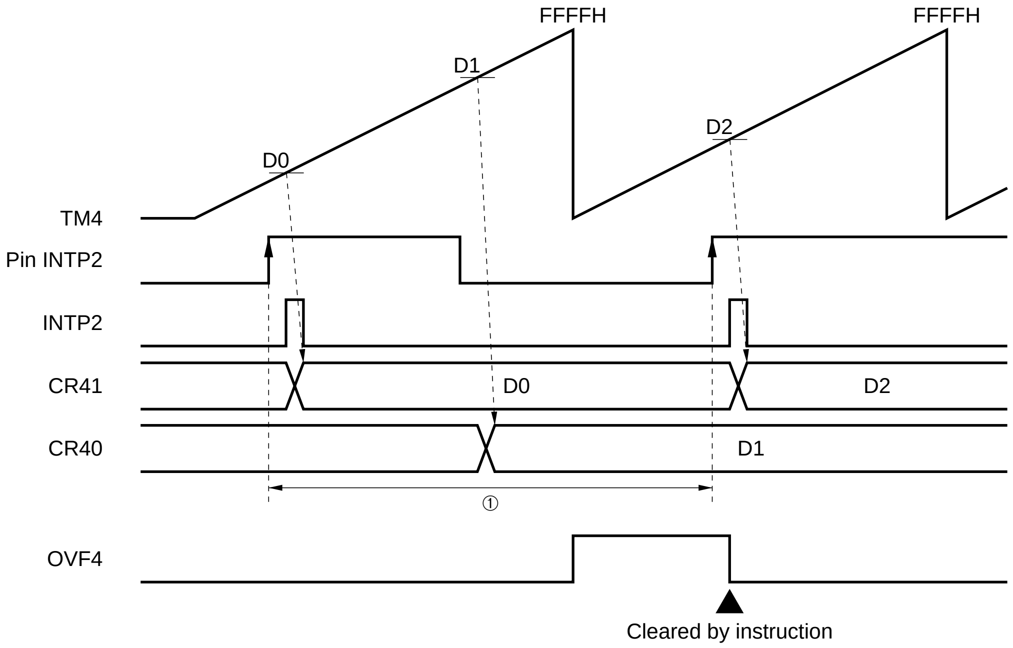
Can use the 16-bit timer register 4 (TM4) to measure the width of pulse input to pin INTP2/P03. Width is measured from edge to edge.

<1> Places the 16-bit capture/compare registers 40 and 41 in capture mode (sets capture/compare control register 4 (CRC4)'s bits 0 and 2 (CRC40 and CRC42) to "1").

<2> Sets the CR40's capture trigger to INTP2's reverse edge (sets CRC4's bit 1 (CRC41) to "1").

<3> Enables the free running mode (sets the 16-bit timer mode control register 4 (TMC4)'s bits 2 and 3 (TMC42 and TMC43) to "1" and "0" respectively.)

Figure 12-14. Pulse Width Measurement Timing (When Rising Edge Is Specified)



[Measurement Method]

- The INTP2 interrupt process reads the CR41 and OVF4 (TMC4's bit 0) flags.
  <1> is (10000H – D0 + D2) × count-clock rate.

Caution **CR40 is captured on the reverse edge signal INTP2 but the interrupt request signal (INTP2) does not generate at that time. INTP2 (interrupt request signal) is generated only when the measured valid edge is detected.**

Remark Dn: TM4 count value (n = 0, 1, 2, ...)

(3) Pulse width measurement (falling edge)

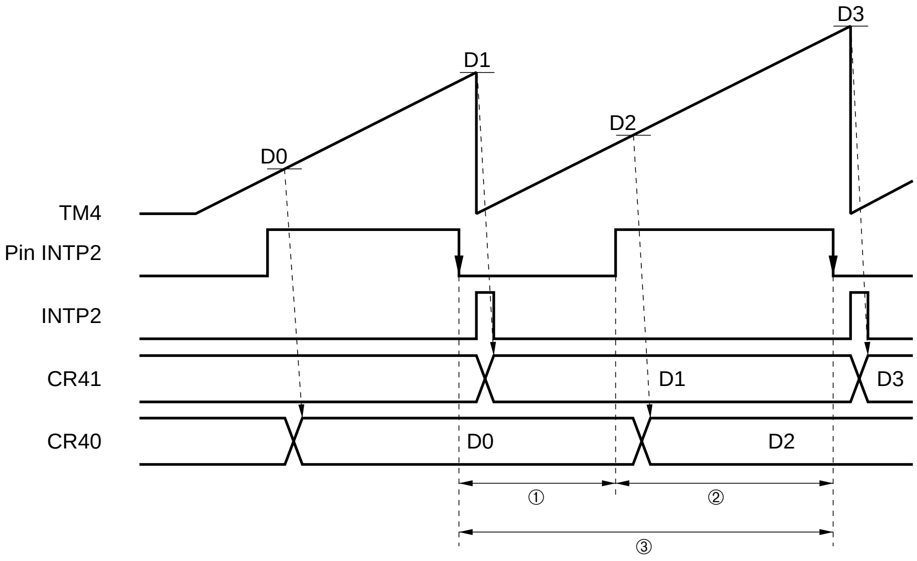
The 16-bit timer register 4 (TM4) can be used to measure the pulse width of signal INTP2/P03 input. High width and low width are measured separately.

<1> Places the 16-bit capture/compare registers 40 and 41 (CR40 and CR41) in capture mode (sets capture/compare control register 4 (CRC4)'s bits 0 and 2 (CRC40 and CRC42) to "1").

<2> Sets CR40's capture trigger on the reverse edge of INTP2 (sets CRC4's bit 1 (CRC41) to "1").

<3> Uses the input of INTP2's valid edge to enable the clear and start mode (sets 16-bit timer mode control register 4 (TMC4)'s bits 2 and 3 (TMC42 and TMC43) to "0" and "1" respectively).

Figure 12-15. Pulse Width Measurement Timing (When Falling Edge Is Specified)



[Measurement Method]

- INTP2 interrupt processing reads CR40 and CR41.
  <1> Low width is D2 × count-clock rate.
  <2> High width is (D3 – D2) × count-clock rate.
  <3> Duration 1 is D3 × count-clock rate.
  However, compensation is required if TM4 overflows.

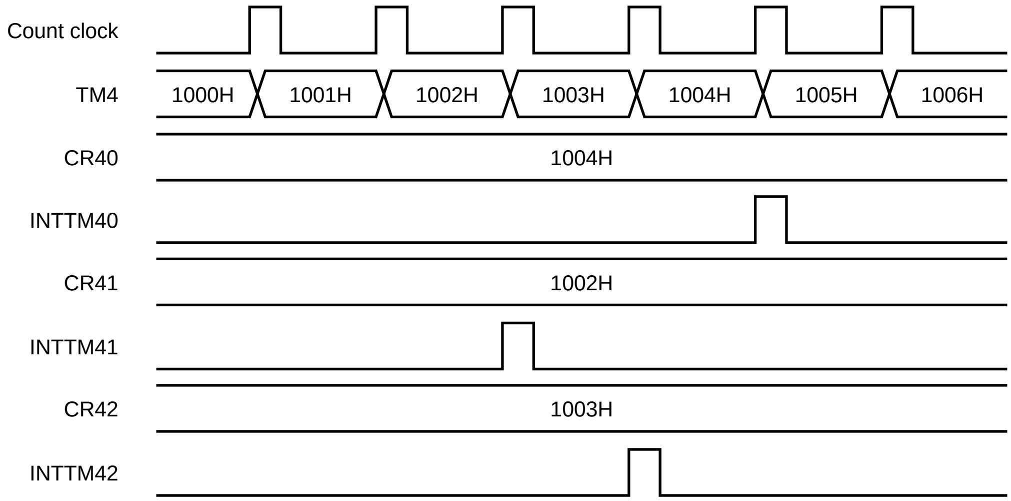
Caution CR40 is captured on the reverse edge of signal INTP2 but the interrupt request signal (INTP2) does not generate at that time. INTP2 (interrupt request signal) is generated only when the measured valid edge is detected.

Remark Dn: TM4 count value (n = 0, 1, 2, ...)

12.4.7 Compare operation of TM4

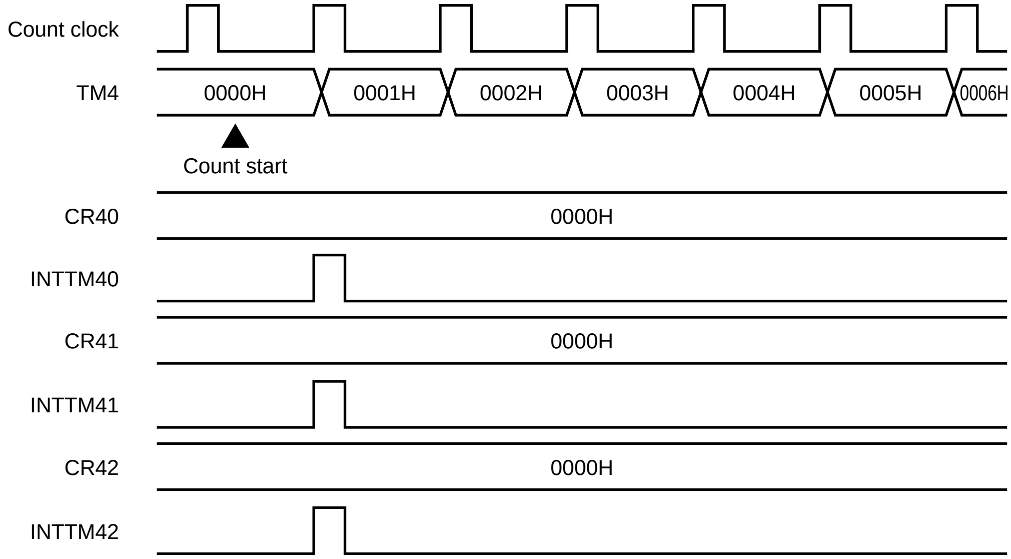
By setting the capture/compare control register 4 (CRC4)'s bits 0, 2 and 3 (CRC40, CRC42 and CRC43) to "0", the 16-bit capture/compare registers 40, 41 and 42 (CR40, CR41 and CR42) are set to compare mode. When the 16-bit timer register 4 (TM4)'s CR40, CR41 and CR42 are the same, the next count clock generates interrupt request signals (INTTM40, INTTM41 and INTTM42).

Figure 12-16. Compare Operation Timing of TM4 (CR40, CR41, and CR42 ≠ 0000H)



Caution The operating mode for TM4 in Figure 12-16 is a mode other than the clear and start mode when TM4 and CR40 match. In the case of the clear and start mode when TM4 and CR40 match, the next count clock clears TM4 (see Figure 12-9. Clear and Start Mode Operation Timing When TM4 and CR40 Match (CR40 ≠ 0000H)). TM4 is not cleared when CR41 and CR42 match.

Figure 12-17. Compare Operation Timing of TM4 (CR40, CR41, and CR42 = 0000H)

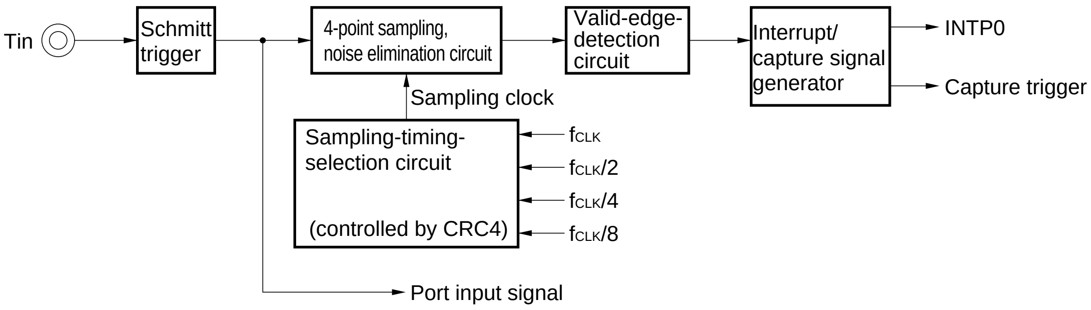


Caution The operating mode for TM4 in Figure 12-17 is a mode other than the clear and start mode when TM4 and CR40 match. In the case of the clear and start mode when TM4 and CR40 match, TM4 remains at 0000H (see Figure 12-10. Clear and Start Mode Operation Timing When TM4 and CR40 Match (CR40 = 0000H)).

12.4.8 Noise elimination circuit

The 16-bit timer/counter 4 (TM4)'s noise elimination circuit (SMPC40 and SMPC41) performs 4-point sampling in the timing specified by capture/compare control register 4 (CRC4)'s bit's 6 and 7. If in continuous sampling, the same level continues for four times that level is swept in internally.

Figure 12-18. INTP2 Block Diagram


Tin
Schmitt trigger
4-point sampling, noise elimination circuit
Valid-edge-detection circuit
Interrupt/ capture signal generator
INTP0
Capture trigger
Sampling clock
Sampling-timing-selection circuit
(controlled by CRC4)
fCLK
fCLK/2
fCLK/4
fCLK/8
Port input signal

[Sampling timing]

Tin: Width of INTP2 pin input signal, Tsmp: Sampling timing,
C1, C2: System clock, TCLK: System-clock rate (= 1/fCLK)

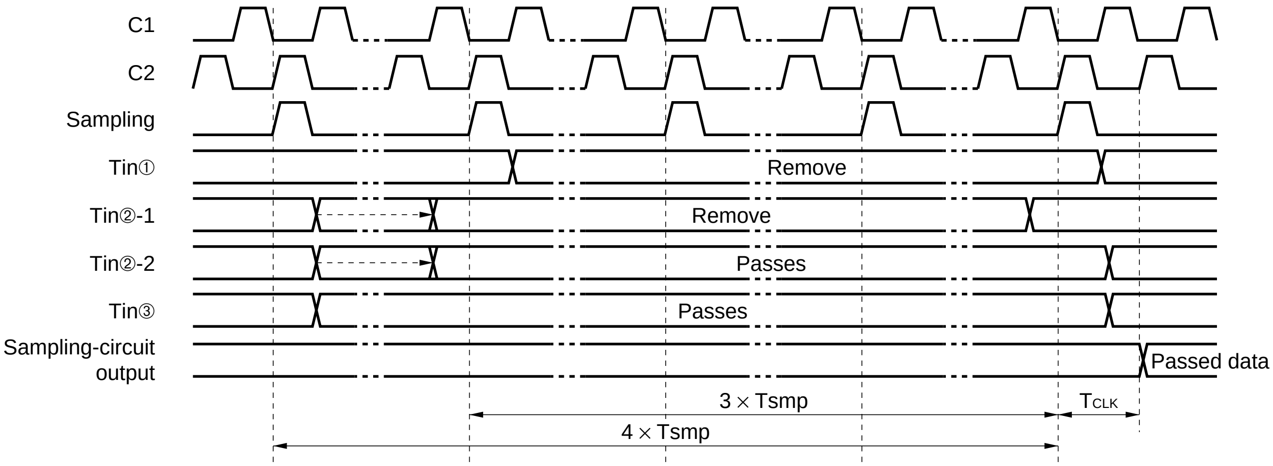
<1> Tin ≤ (3 × Tsmp) .... Removed as noise.

<2> (3 × Tsmp) < Tin < (4 × Tsmp) .... May be removed as noise, or may pass as a valid signal.

<3> Tin ≥ (4 × Tsmp) .... Passes as a valid signal.

Caution Enter a signal of 4 × Tsmp width for assured pass of valid signals.

Figure 12-19. Sampling Timing Diagram



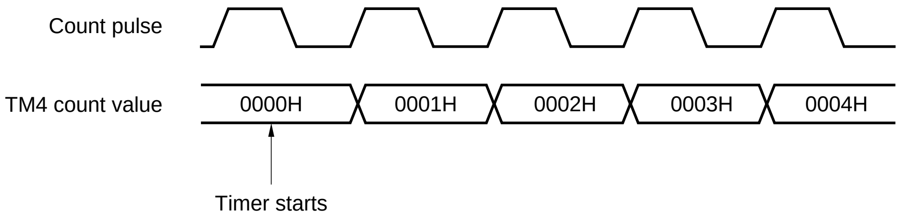
Remark There is a distortion of 1 × Tsmp in (3 × Tsmp + 1TCLK) to (4 × Tsmp + 1TCLK) in the time in which pin level (Tin) passes through the sampling circuit.

★ 12.5 Cautions

(1) Error at timer start

After the timer starts, the time until a uniform signal is generated has a maximum error of 1 clock. This is because the start of 16-bit timer register 4 (TM4) is asynchronous with respect to the count pulse.

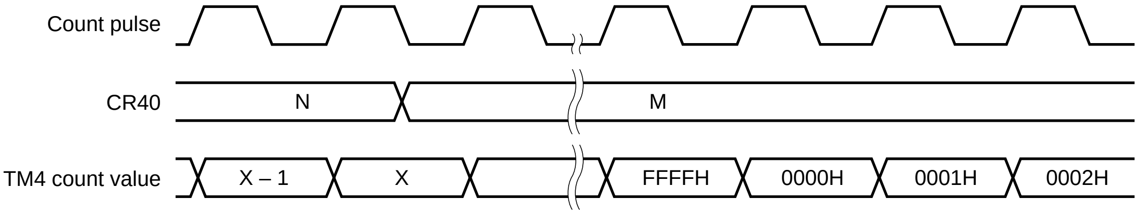
Figure 12-20. Start Timing of 16-Bit Timer Register 4



(2) Operation after changes in the compare register during timer count operation

If the value of 16-bit capture/compare register 40 (CR40) after changing is less than the value of 16-bit timer register 4 (TM4), TM4 continues counting, and when an overflow occurs, it starts over from 0. Therefore, if the value of CR40 after changing (M) is less than the value before changing (N), the timer must be restarted after CR40 changes.

Figure 12-21. Timing After Changing Compare Register During Timer Count Operation



Remark N > X > M

(3) Valid edge setting

The valid edge of pin INTP2/P03 should be set after setting bits 2 and 3 (TMC42, TMC43) of 16-bit timer mode control register 4 (TMC4) to “0” and “0”, respectively, and after timer operation stops. The valid edge is set using bits 4 and 5 (ES20, ES21) of prescaler mode register 4 (PRM4).

[MEMO]

CHAPTER 13 16-BIT TIMER/COUNTER 5

13.1 Function

16-bit timer/counter 5 (TM5) has the following functions:

- Interval timer
  An interrupt request is generated at a time interval set in advance.

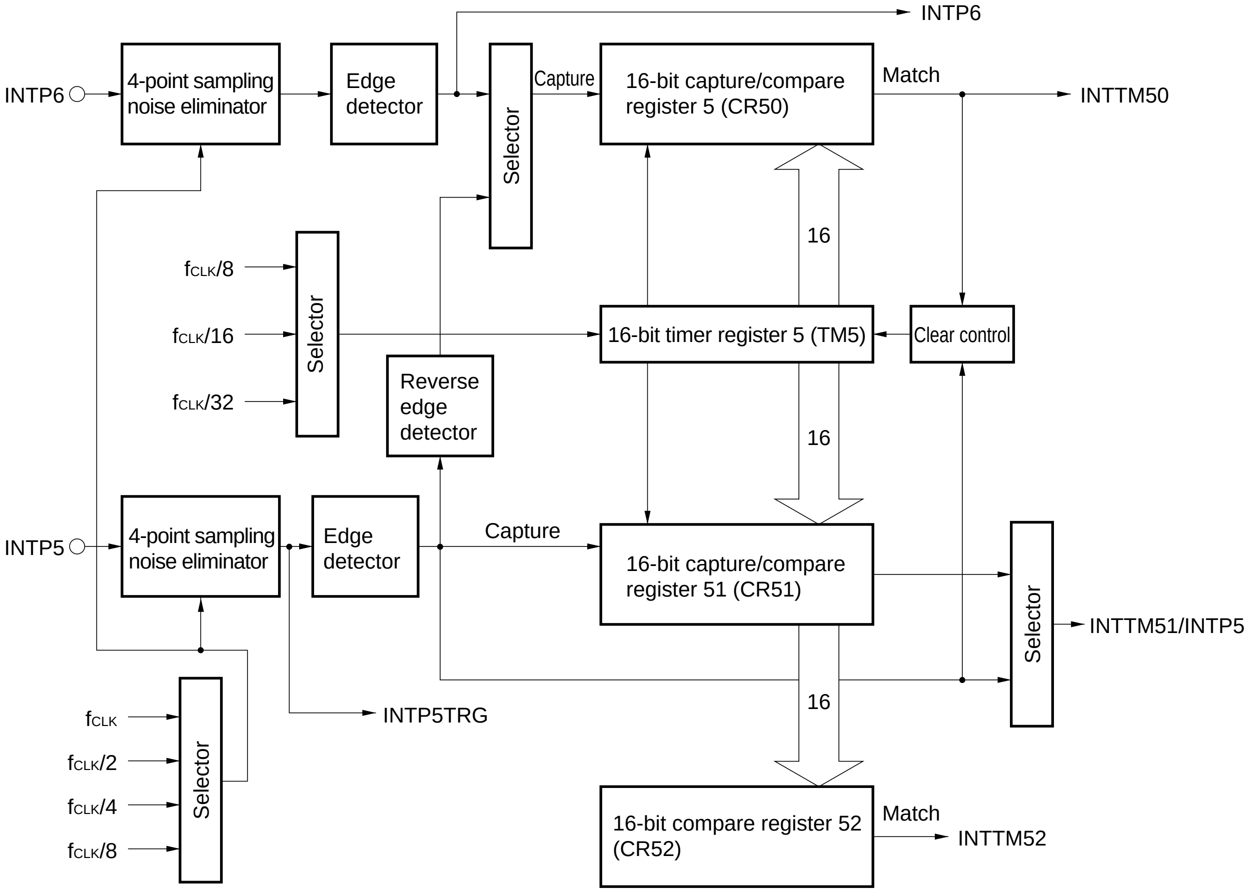
13.2 Configuration

The 16-bit timer/counter 5 (TM5) has the hardware configuration shown below.

Table 13-1. Configuration of 16-Bit Timer/Counter 5 (TM5)

Item	Configuration
Timer register	16 bits × 1 (TM5)
Register	Capture/compare register : 16-bit × 2 (CR50, CR51) compare register : 16-bit × 1 (CR52)
Control register	16-bit timer mode control register 5 (TMC5) Capture/compare control register 5 (CRC5) Prescaler mode register 5 (PRM5)

Figure 13-1. Block Diagram of 16-Bit Timer/Counter 5 (TM5)



(1) 16-bit timer register 5 (TM5)

TM5 is a 16-bit free running or interval timer that counts the count pulses.

It increments the counter in synchronization with the rise in the input clock. The count value then becomes 0000H.

<1> Input of $\overline{\text{RESET}}$

<2> Clears the 16-bit timer mode control register 5 (TMC5)'s bits 2 and 3 (TMC52 and TMC53).

<3> When the INTP5 valid edge is input during the enabling of the clear and start mode by INTP5 valid edge input.

<4> When TM5 and CR50 match in the clear and start mode when 16-bit capture/compare register 50 (CR50) match.

(2) 16-bit capture/compare register 50 (CR50)

CR50 is the 16-bit register that has the functions of both the capture and compare registers.

The capture/compare control register 5 (CRC5)'s bit 0 (CRC50) is set for the register to be used as either a capture register or a compare register.

- **When CR50 is used as a compare register**

  The value set in CR50 is always being compared with the count in 16-bit timer register 5 (TM5) and when they are the same an interrupt request (INTTM50) is generated. When the TM5 is set to operate as an interval timer, it can also be used as a register for saving interval time.

- **When CR50 is used as a capture register**

  The valid edge of the signals on pin INTP5 or INTP6 can be selected as the capture trigger. The valid edges of signals INTP5 are set by the prescaler mode register 5 (PRM5)'s bit 4 and 5 (ES50 and ES51) and the valid edges for INTP6 are set by bits 6 and 7 (ES60 and ES61).

  Table 13-2 shows the situation when the valid edge of the signal on pin INTP5 is specified as the capture trigger, and Table 13-3 shows the situation when the valid edge of the signal on pin INTP6 is specified as the capture trigger.

Table 13-2. Valid Edge of Pin INTP5 and CR50 Capture Trigger

ES51	ES50	Valid Edge of Pin INTP5	Capture Trigger of CR50	Capture Trigger of CR51
0	0	Falling edge	Rising edge	Falling edge
0	1	Rising edge	Falling edge	Rising edge
1	0	Setting prohibited	Setting prohibited	Setting prohibited
1	1	Both rising and falling edge	Capture does not operate	Both rising and falling edge

Table 13-3. Valid Edge of Pin INTP6 and CR50 Capture Trigger

ES61	ES60	Valid Edge of Pin INTP6	Capture Trigger of CR50
0	0	Falling edge	Falling edge
0	1	Rising edge	Rising edge
1	0	Setting prohibited	Setting prohibited
1	1	Both rising and falling edges	Both rising and falling edges

CR50 is set by a16-bit memory manipulation instruction.
$\overline{\text{RESET}}$ input makes CR50 undefined.

Caution When a switch is made from capture to compare mode, CR50's value is the last value captured. When a switch is made from the compare mode to the capture mode, CR50's value is the last value set in the compare register.

(3) 16-bit capture/compare register 51 (CR51)

CR51 is a 16-bit register with the functions of both capture and compare register.
The capture/compare control register 5 (CRC5)'s bit 2 (CRC52) sets the register for use as either a capture or compare register.

- **When CR51 is used as a compare register.**
  The count value of the 16-bit timer register (TM5) is always being compared with the values programmed in CR50 and an interrupt request (INTTM51) generated when they are the same. When TM5 is set to operate as an interval timer, it can also be used as a register that saves interval time.

- **When CR51 is used as a capture register.**
  Allows the selection of the valid edge of the signal at pin INTP5 to be used as a capture trigger.
  The valid edge of INTP5 is set by the prescaler mode register 5 (PRM5)'s bits 4 and 5 (ES50 and ES51).
  Table 13-4 shows the situation when the valid edge of INTP5 is specified as a capture trigger.

Table 13-4. Valid Edge of Pin INTP5 and CR51 Capture Trigger

ES51	ES50	Valid Edge of INTP5 Pin	Capture Trigger of CR51
0	0	Falling edge	Falling edge
0	1	Rising edge	Rising edge
1	0	Setting prohibited	Setting prohibited
1	1	Both rising and falling edges	Both rising and falling edges

CR51 is set by a 16-bit memory manipulation instruction.
$\overline{\text{RESET}}$ input makes CR51 undefined.

Caution When a switch is made from capture to compare mode, CR51's value is the last value captured. When a switch is made from the compare mode to the capture mode, CR51's value is the last value set in the compare register.

(4) 16-bit compare register 52 (CR52)

The count of the 16-bit timer register (TM5) is always compared with the values set in CR52 and an interrupt request generated when they match. When TM5 is set to operate as an interval timer, it can also be used as a register that saves interval time.

CR52 is set by a 16-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input makes CR52 undefined.

13.3 Control Register

The following three types of registers control the 16-bit timer/counter 5 (TM5).

- 16-bit timer mode control register 5 (TMC5)
- Capture/compare control register 5 (CRC5)
- Prescaler mode register 5 (PRM5)

(1) 16-bit timer mode control register 5 (TMC5)

The TMC5 is the register that detects overflow and assigns the clear mode for the 16-bit timer register 5 (TM5)
TMC5 is set by 1-bit or 8-bit memory manipulation instruction.
$\overline{\text{RESET}}$ input resets TMC5 to 00H.

Caution TM5 starts operating when a value other than “0”, “0” is set in TMC5’s bits 2 and 3 (TMC52 and TMC53). To stop the operation, set “0”, “0” in TMC52 and TMC53.

Figure 13-2. Format of 16-Bit Timer Mode Control Register 5 (TMC5)

Address: 0FF6FH After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	⓪
TMC5	0	0	0	0	TMC53	TMC52	0	OVF5

TMC53	TMC52	Selection of Operating Mode and Clear Mode	Generation of Interrupt
0	0	Operation stop (TM5 is cleared to 0).	Does not generate.
0	1	Free running mode	Generates on coincidence between TM5 and CR50.
1	0	Clears and starts at valid edge of INTP5.	
1	1	Clears and starts on coincidence between TM5 and CR50.	

0VF0	Detection of Overflow of TM5
0	Does not overflow.
1	Overflows.

Caution Always assign TM5 after stopping TM5’s timer operations.

(2) Capture/compare control register 5 (CRC5)

This register controls the operations of the 16-bit capture/compare registers 50, 51 and 52 (CR50, CR51 and CR52).

CRC5 is set by 1-bit or 8-bit memory manipulation instructions.

$\overline{\text{RESET}}$ input resets CRC5 to 00H.

Figure 13-3. Format of Capture/Compare Control Register 5 (CRC5)

Address: 0FF7EH After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
CRC5	SMPC51	SMPC50	0	0	0	CRC52	CRC51	CRC50

SMPC51	SMPC50	Selection of Sampling Clock
0	0	f_{CLK}
0	1	$f_{CLK}/2$
1	0	$f_{CLK}/4$
1	1	$f_{CLK}/8$

CRC52	Selection of Operation Mode of CR51
0	Operates as compare register.
1	Operates as capture register.

CRC51	Selection of Capture Trigger of CR50
0	Captured at valid edge of INTP6.
1	Captured in reverse phase of valid edge of INTP5.

CRC50	Selection of Operation Mode of CR50
0	Operates as compare register.
1	Operates as capture register.

Cautions 1. Always set CRC5 after stopping timer operations

2. Do not specify CR50 to the capture register when selecting the clear and start mode when TM4 and CR40 match in 16-bit timer mode control register 5 (TMC5).

(3) Prescaler mode register 5 (PRM5)

The prescaler mode register 5 (PRM5) is a register that specifies 16-bit timer register 5's (TM5) count clock and INTP5's and INTP6's valid input edge.

PRM5 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input resets PRM5 to 00H.

Figure 13-4. Format of Prescaler Mode Register 5 (PRM5)

Address: 0FF89H After Reset : 00H R/W

Symbol	7	6	5	4	3	2	1	0
PRM5	ES61	ES60	ES51	ES50	0	0	PRM51	PRM50

ES61	ES60	Selection of Valid Edge of INTP6
0	0	Falling edge
0	1	Rising edge
1	0	Setting prohibited.
1	1	Both falling and rising edges

ES51	ES50	Selection of Valid Edge of INTP5
0	0	Falling edge
0	1	Rising edge
1	0	Setting prohibited.
1	1	Both falling and rising edges

PRM51	PRM50	Selection of Count Clock
0	0	$f_{CLK}/8$ (1 MHz)
0	1	$f_{CLK}/16$ (500 MHz)
1	0	$f_{CLK}/32$ (250 kHz)
1	1	Setting prohibited.

Cautions 1. Do not set the valid edge of INTP5 and INTP6 in the capture trigger and clear-and-start mode when assigning the valid edge of INTP5 and INTP6 to the count clock

2. Always set PRM5 after stopping timer operations.

Remark Figures in parentheses apply to operation with f_{CLK} = 8 MHz

13.4 Operation

13.4.1 Basic operation of TM5

The 16-bit timer counter 5 (TM5) is a 16-bit free running timer or interval timer for counting pulses. The counter is incremented through synchronization with the rise in the input clock pulse.

Inputting the $\overline{\text{RESET}}$ signal clears all TM5 bits (to 0) and stops the count operation.

The enabling and prohibiting of the count operation is controlled by 16-bit timer mode control register 5 (TMC5)'s bits 2 and 3 (TMC52 and TMC53). Setting TMC52 and TMC52 to an operation mode other than "0", "0" starts the count operation. Resetting TMC52 and TMC53 to "0", "0" clears TM5 and stops the count operation.

Resetting makes the count value 0000H.

After the start count is set, the first count clock input makes TM5 0000H to 0001H.

TM5 does not clear the timer, it continues count operations even if the same operation mode is set again.

The count does not stop during TM5 read duration.

Figure 13-5. Basic Operation Timing of TM5

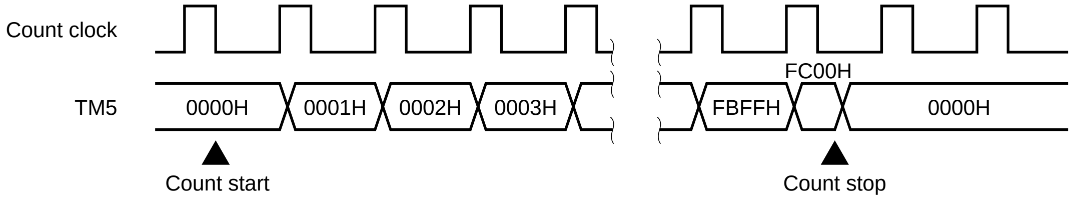
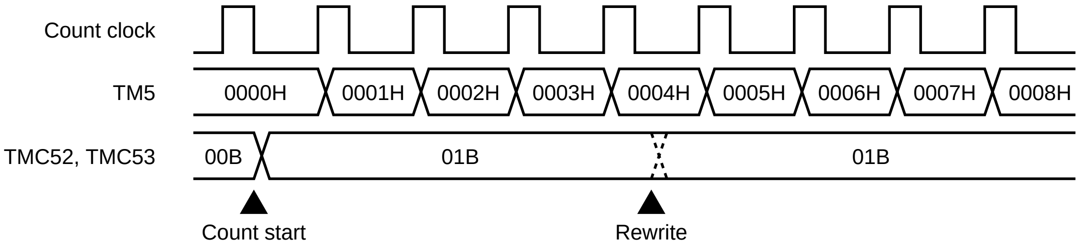


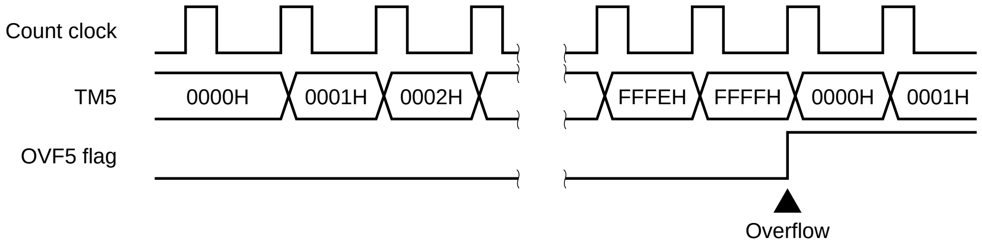
Figure 13-6. TMC52 and TMC53 Rewrite Operation Timing (Free Running Mode)



13.4.2 Free running operation of TM5

Setting the 16-bit timer mode control register 5 (TMC5)'s bits 2 and 3 (TMC52 and TMC53) to "1" and "0", respectively, allows TM5 to free running operation. When TM5 has made a full count to FFFFH, the next count clock pulse sets TMC5's bit 0 (OVF5) to 1 and clears TM5 (to 00000H). The count then continues. OVF5 can also be cleared by command.

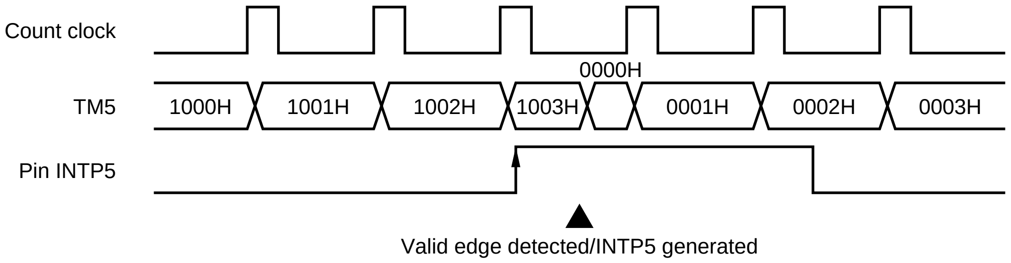
Figure 13-7. Free Running Mode Operation Timing of TM5



13.4.3 Clear and start operation of TM5 at valid edge of INTP5

Setting the 16-bit timer mode control register 5 (TMC5)'s bits 2 and 3 (TMC52 and TMC53) to "1" and "0", respectively, allows the input of INTP5's valid edge to place TM5 in clear-and-start mode. Entering INTP5 valid edge (interrupt request signal: generates INTP5) clears TM5 (to 0000H) and the next count clock pulse resets it to (0001H). The count then continues.

Figure 13-8. Clear and Start Mode Operation Timing of TM5 at Input of Valid Edge of INTP5



13.4.4 Clear and start operation when TM5 and CR50 match

Setting the 16-bit timer mode control register 5 (TMC5)'s bits 2 and 3 (TMC52 and TMC53) to "1" and "1", respectively, allows TM5 to the clear and start mode 16-bit capture/compare register 50 (CR50) match. If TM5 and CR50 are the same, the next count clock pulse generates an interrupt request signal (INTTM50) to clear TM5 (to 0000H). The count then continues.

Figure 13-9. Clear and Start Mode Operation Timing When TM5 and CR50 Match (CR50 ≠ 0000H)

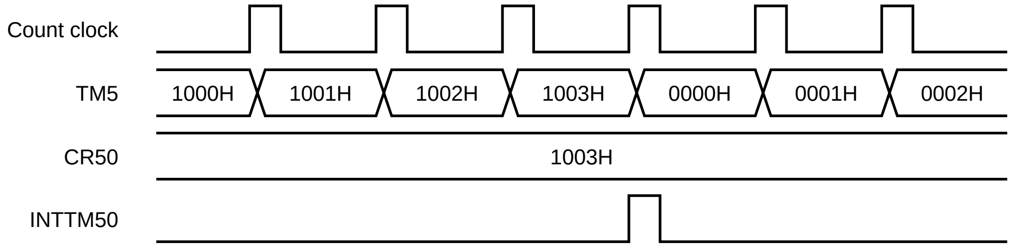
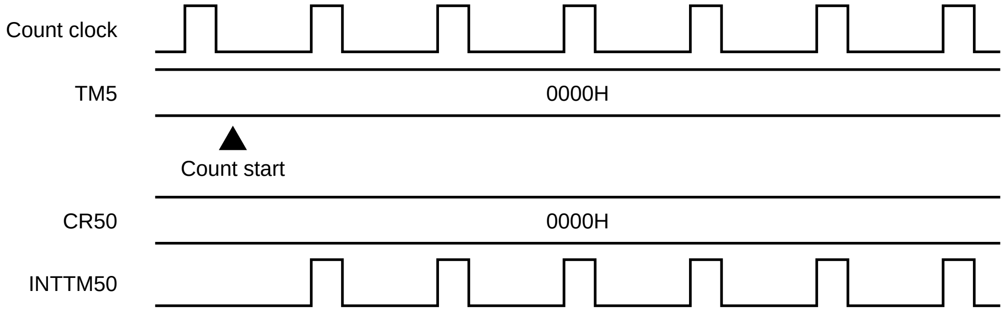


Figure 13-10. Clear and Start Mode Operation Timing When TM5 and CR50 Match (CR50 = 0000H)



Caution Always set CR50 to compare mode.

Remark Interval period = (CR50 + 1) × TM5 count-clock rate

13.4.5 Capture operation of TM5

Setting capture/compare control register 5 (CRC5)'s bits 0 and 2 (CRC50 and CRC52) to "1" places the 16-bit capture/compare registers 50 and 51 (CR50 and CR51) in capture mode. Inputting the capture trigger captures TM5's values in CR50 and CR51.

Figure 13-11. Capture Operation Timing (Free Running Mode)

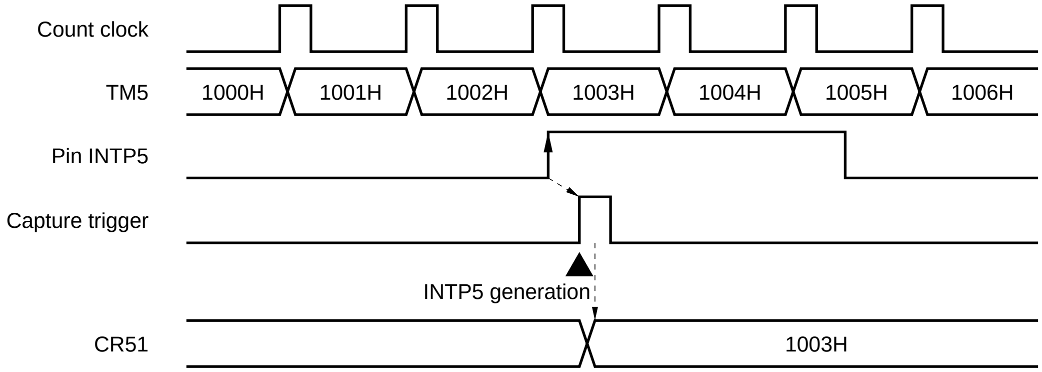
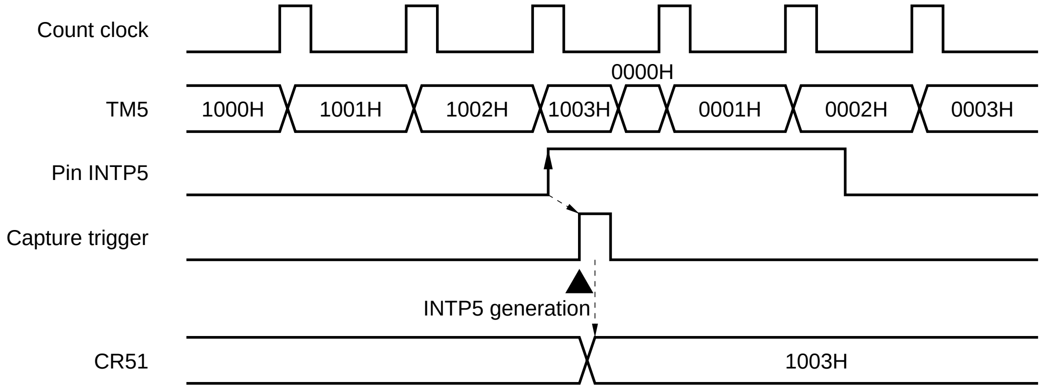


Figure 13-12. Capture Operation Timing (Clear and Start Mode at INTP5 Valid Edge Input)



13.4.6 Pulse width measurement operation

(1) Pulse width measurement (both rising and falling edge)

The 16-bit timer register 5 (TM5) can be used to measure the pulse width of signals input to pins INTP5/P06 and INTP6/P07. The width from edge to edge is measured.

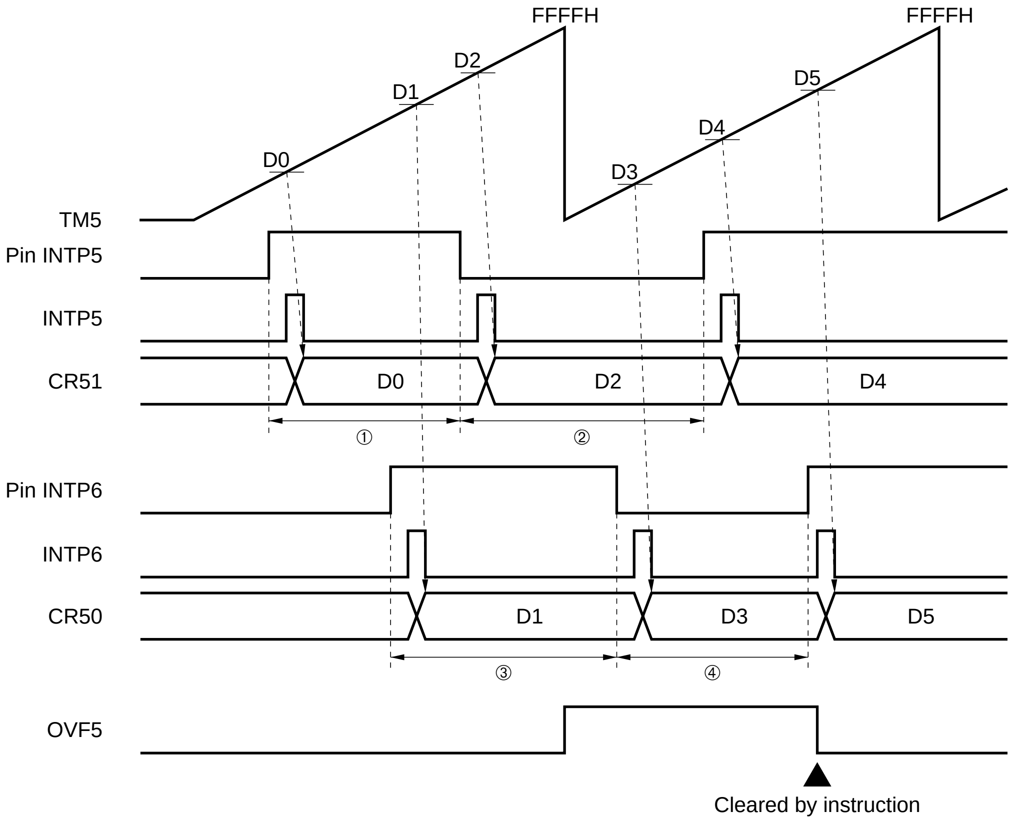
<1> Setting the 16-bit capture/compare registers 50 and 51 (CR50 and CR51) enables the capture mode (sets the capture/compare control register 5 (CRC5)'s bits 0 and 2 (CRC50 and CRC51) to "1).

<2> Sets CR50's capture trigger to INTP5 (sets CRC5's bit 1 (CRC51) to "0").

<3> Sets INTP5's and INTP6's valid edges for both edges (sets the prescaler mode register 5 (PRM5)'s bits 4 to 7 (ES50, ES51, ES60 and ES61) to "1").

<4> Enables the free running mode (Sets the 16-bit timer mode control register 5 (TMC5)'s bits 2 and 3 (sets TMC52 to "1" and TMC53 to "0")).

Figure 13-13. Pulse Width Measurement Timing (When Both Edges Are Specified)



[Measurement Method]

- INTP5 interrupt processing reads the CR50 and OVF5 (TMC5's bit 0) flags.
  <1> is (D2 – D0) × count-clock rate.
  <2> is (10000H – D2 + D4) × count-clock rate.
- INTP6 interrupt processing reads CR50 and OVF5.
  <3> is (10000H – D1 + D3) × count-clock rate.
  <4> is (D5 – D3) × count-clock rate.

Remark Dn: TM5 count value (n = 0, 1, 2, ...)

(2) Pulse width measurement (rising edge)

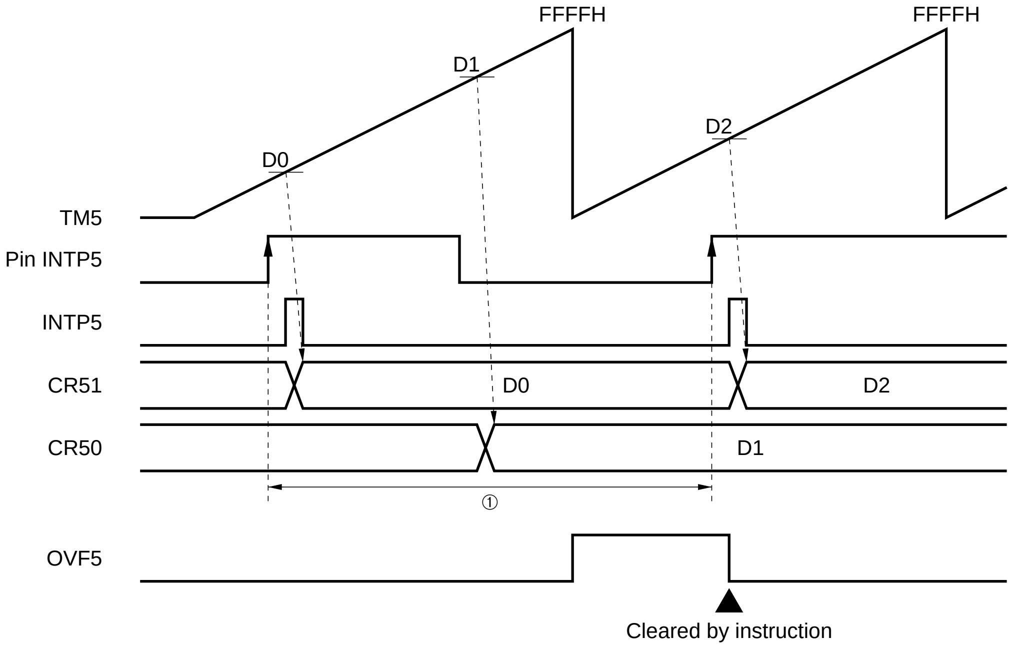
Can use the 16-bit timer register 5 (TM5) to measure the width of pulse input to pin INTP5/06. Width is measured from edge to edge.

<1> Places the 16-bit capture/compare registers 50 and 51 (CR50 and CR51) in capture mode (sets capture/compare control register 5 (CRC5)'s bits 0 and 2 (CRC50 and CRC52) to "1").

<2> Sets the CR50's capture trigger to INTP5's reverse edge (sets CRC5's bit 1 (CRC51) to "1").

<3> Enables the free running mode (sets the 16-bit timer mode control register 5 (TMC5)'s bits 2 and 3 (TMC52 and TMC53) to "1" and "0" respectively.)

Figure 13-14. Pulse Width Measurement Timing (When Rising Edge Is Specified)



[Measurement Method]

- The INTP5 interrupt process reads the CR51 and OVF5 (TMC5's bit 0) flags.
  <1> is (10000H – D0 + D2) × count-clock rate.

Caution CR50 is captured on the reverse edge of signal INTP5 but the interrupt request signal (INTP5) does not generate at that time. INTP5 (interrupt request signal) is generated only when the measured valid edge is detected.

Remark Dn: TM5 count value (n = 0, 1, 2, ...)

(3) Pulse width measurement (falling edge)

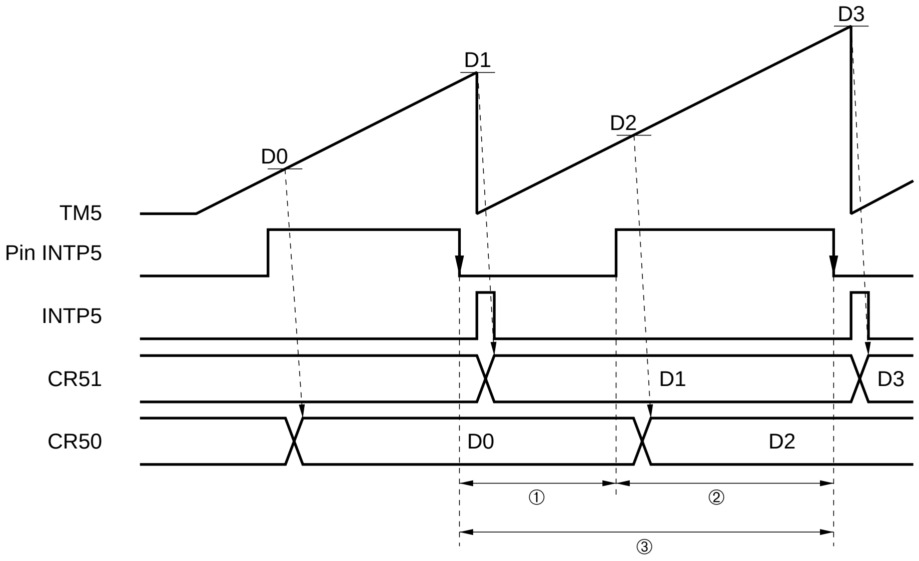
The 16-bit timer register 5 (TM5) can be used to measure the pulse width of signal INTP5/P06 input. High width and low width are measured separately.

<1> Places the 16-bit capture/compare registers 50 and 51 (CR50 and CR51) in capture mode (sets capture/compare control register 5 (CRC5)'s bits 0 and 2 (CRC50 and CRC52) to "1").

<2> Sets CR50's capture trigger on the reverse edge of INTP5 (sets CRC5's bit 1 (CRC51) to "1").

<3> Uses the input of INTP5's valid edge to enable the clear and start mode (sets 16-bit timer mode control register 5 (TMC5)'s bits 2 and 3 (TMC52 and TMC53) to "0" and "1" respectively).

Figure 13-15. Pulse Width Measurement Timing (When Falling Edge Is Specified)



[Measurement Method]

- INTP5 interrupt processing reads CR50 and CR51.
  <1> Low width is D2 × count-clock rate.
  <2> High width is (D3 – D2) × count-clock rate.
  <3> Duration 1 is D3 × count-clock rate.
  However, compensation is required if TM5 overflows.

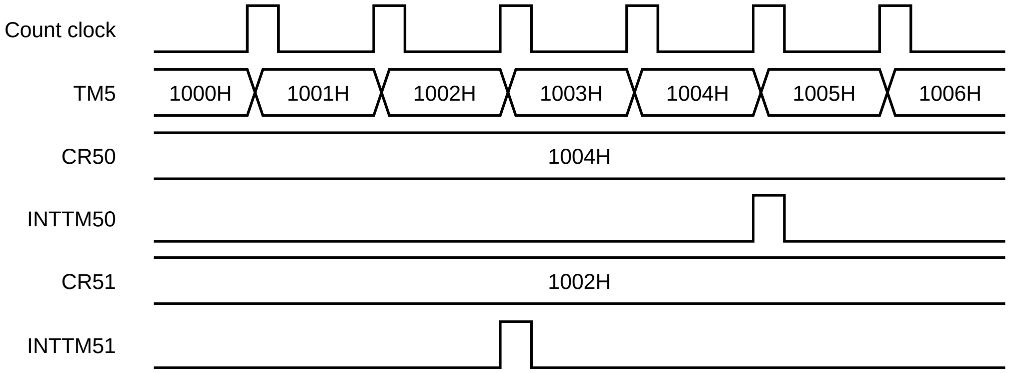
Caution CR50 is captured on the reverse edge of signal INTP5 but the interrupt request signal (INTP5) does not generate at that time. INTP5 (interrupt request signal) is generated only when the measured valid edge is detected.

Remark Dn: TM5 count value (n = 0, 1, 2, ...)

13.4.7 Compare operation of TM5

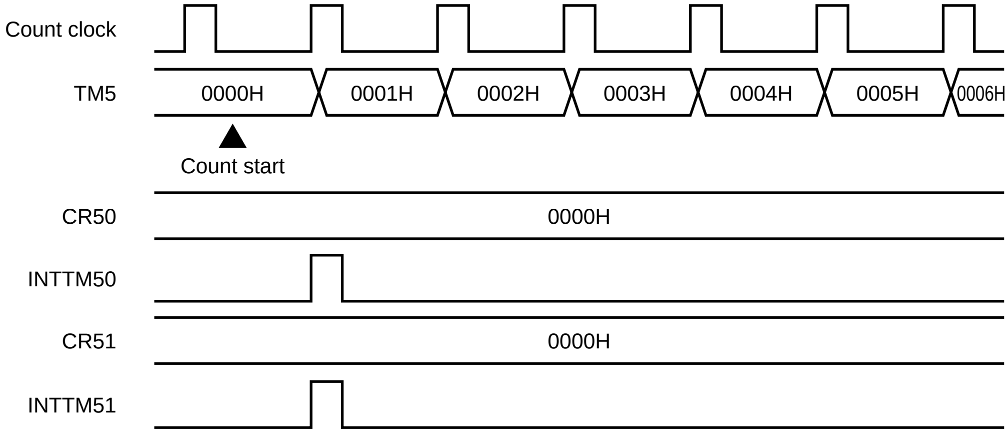
Setting the capture/compare control register 5 (CRC5)'s bits 0 and 2 (CRC50 and CRC52) to "0" places the 16-bit capture/compare registers 50 and 51 (CR50 and CR51) in compare mode. When the 16-bit timer register 5 (TM5)'s CR50 and CR51 are the same, the next count clock generates interrupt request signals (INTTM50 or INTTM51).

Figure 13-16. Compare Operation Timing of TM5 (CR50, CR51 ≠ 0000H)



Caution The operating mode for TM5 in Figure 13-16 is a mode other than the clear and start mode when TM5 and CR50 match. In the case of the clear and start mode when TM5 and CR50 match, the next count clock clears TM5 (see Figure 13-9. Clear and Start Mode Operation Timing When TM5 and CR50 Match (CR50 ≠ 0000H)). TM5 is not cleared when TM5 and CR51 match.

Figure 13-17. Compare Operation Timing of TM5 (CR50, CR51 = 0000H)

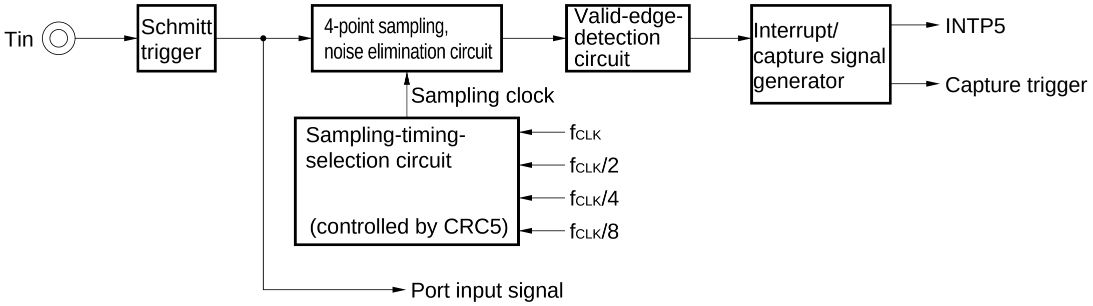


Caution The operating mode for TM5 in Figure 13-17 is a mode other than the clear-and-start mode TM5 and CR50 match, which enables. In the case of the clear and start mode when TM5 and CR50 match, TM5 remains at 0000H (see Figure 13-10. Clear and Start Mode Operation Timing When TM5 and CR50 Match (CR50 = 0000H)).

13.4.8 Noise elimination circuit

The 16-bit timer/counter 5 (TM5)'s noise elimination circuit performs 4-point sampling in the timing specified by capture/compare control register 5 (CRC5)'s bit's 6 and 7 (SMPC50 and SMPC51) . If in continuous sampling, the same level continues for four times that level is swept in internally.

Figure 13-18. INTP5 Block Diagram


Tin
Schmitt trigger
4-point sampling, noise elimination circuit
Valid-edge-detection circuit
Interrupt/ capture signal generator
INTP5
Capture trigger
Sampling clock
Sampling-timing-selection circuit
(controlled by CRC5)
fCLK
fCLK/2
fCLK/4
fCLK/8
Port input signal

[Sampling timing]

Tin: Width of INTP5 pin input signal, Tsmp: Sampling timing,
C1, C2: System clock, T_{CLK}: System-clock rate (= $1/f_{CLK}$)

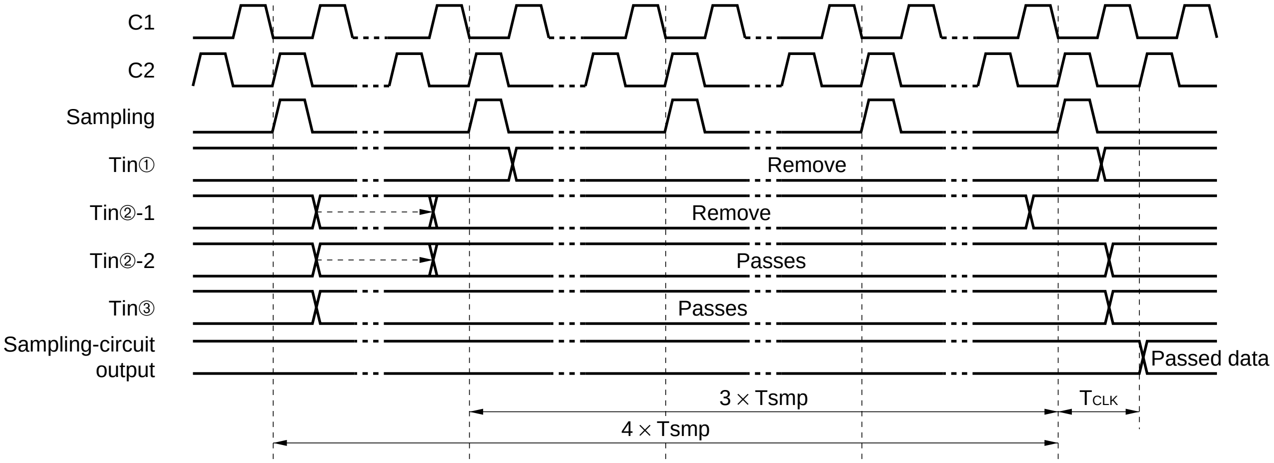
<1> Tin ≤ (3 × Tsmp) .... Removed as noise.

<2> (3 × Tsmp) < Tin < (4 × Tsmp) .... May be removed as noise, or may pass as a valid signal.

<3> Tin ≥ (4 × Tsmp) .... Passes as a valid signal.

Caution Enter a signal of 4 × Tsmp width for assured pass of valid signals.

Figure 13-19. Sampling Timing Diagram



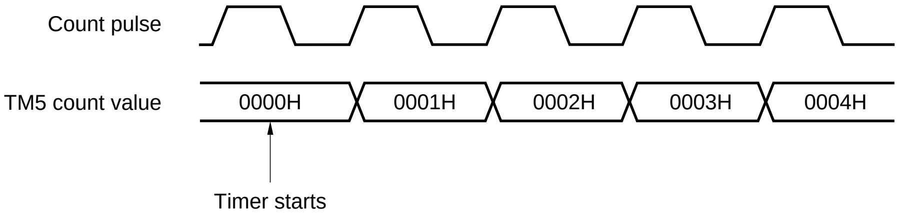
Remark There is a distortion of 1 × Tsmp in (3 × Tsmp + $1T_{CLK}$) to (4 × Tsmp + $1T_{CLK}$) in the time in which pin level (Tin) passes through the sampling circuit.

★ 13.5 Cautions

(1) Error at timer start

After the timer starts, the time until a uniform signal is generated may have a maximum error of 1 clock. This is because the start of 16-bit timer register 5 (TM5) is asynchronous with respect to the count pulse.

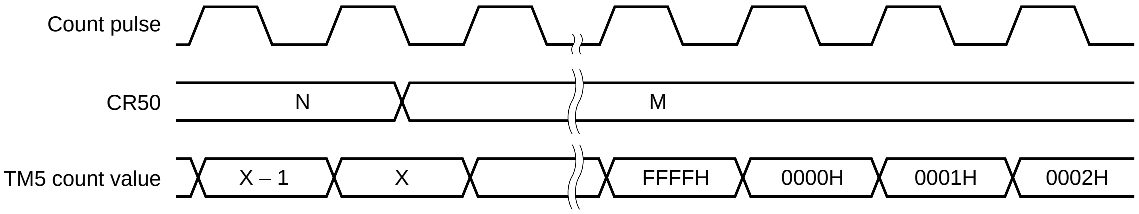
Figure 13-20. Start Timing of 16-Bit Timer Register 5



(2) Operation after changes in the compare register during timer count operation

If the value of 16-bit capture/compare register 50 (CR50) after changing is less than the value of 16-bit timer register 5 (TM5), TM5 continues counting, and when an overflow occurs, it starts over from 0. Also, if the value of CR50 after changing (M) is less than the value before changing (N), the timer must be restarted after CR50 changes.

Figure 13-21. Timing After Changing Compare Register During Timer Count Operation



Remark N > X > M

(3) Valid edge setting

The valid edge of pin INTP5/P06 should be set after setting bits 2 and 3 (TMC52, TMC53) of 16-bit timer mode control register 5 (TMC5) to “0” and “0”, respectively, and after timer operation stops. The valid edge is set using bits 4 and 5 (ES50, ES51) of prescaler mode register 5 (PRM5).

CHAPTER 14 8-BIT TIMER/COUNTER 6

14.1 Functions

The 8-bit timer/counter 6 (TM6) functions as follows.

- Interval timer
- PWM output

(1) Interval timer
An interrupt request is generated at a time interval set in advance.

(2) PWM output
Allows output of rectangular waveforms that can be set to any desired frequency and output pulse width.

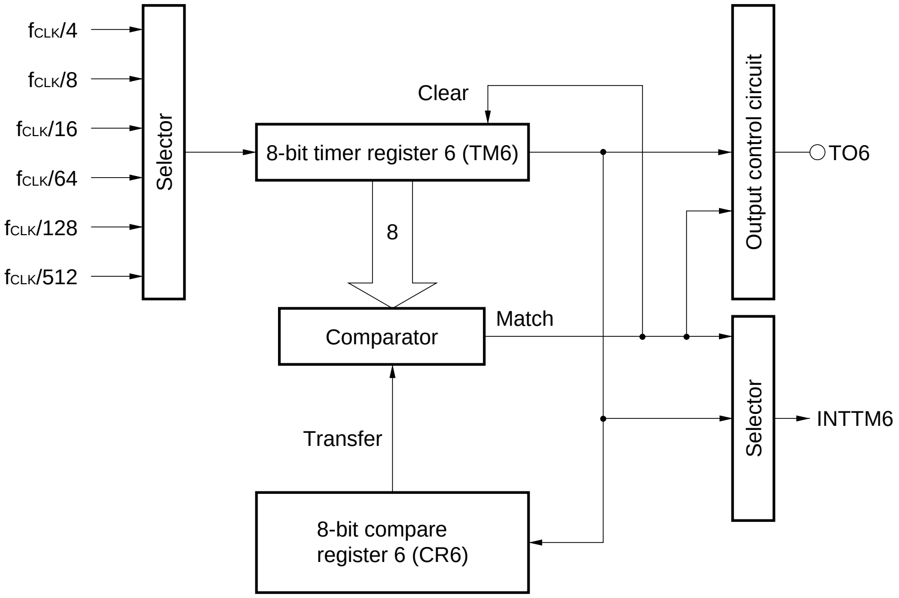
14.2 Configuration

The 8-bit timer/counter 6 (TM6) has the hardware configuration shown below.

Table 14-1. Configuration of 8-Bit Timer/Counter 6 (TM6)

Item	Configuration
Timer register	8-bit × 1 (TM6)
Register	Compare registers: 8-bit × 1 (CR6)
Timer output	1 (TO6)
Control register	Timer mode control register 6 (TMC6) Timer clock selector register 6 (TCL6)

Figure 14-1. Block Diagram of 8-Bit Timer/Counter 6



(1) 8-bit timer register 6 (TM6)

TM6 is an 8-bit free running or interval timer that counts the count pulses.
It increments the counter in synchronization with the rise in the input clock pulse.
The count then becomes 00H.

<1> Input of $\overline{\text{RESET}}$
<2> Clears the timer mode control register 6 (TMC6)'s bit 7 (TCE6).
<3> TM6 and CR6 the same in the interval mode.

(2) 8-bit compare register 6 (CR6)

CR6 has a master (CR6)/slave (comparator) configuration. Transfer times differ depending on what operating mode TM6 is in.

<1> When TM6 is in interval mode
Transfers from CR6 to the comparator are made when the write command writes into CR6. The values set in CR6 are always being compared with the TM6 count, and when they are the same, an interrupt request (INTTM6) is generated. When interval timer operations are programmed for TM6, TM6 can be used as a register for storing interval times.

<2> When TM6 is in free running mode
A TM6 overflow performs transfers from CR6 to the comparator. In the free running mode, PWM is output from pin TO6.

CR6 is set by a 8-bit memory manipulation instructions.
$\overline{\text{RESET}}$ input makes CR6 undefined.

14.3 Control Registers

The following two types of registers control the 8-bit timer/counter 6 (TM6).

- Timer mode control register 6 (TMC6)
- Timer clock selector register 6 (TCL6)

(1) Timer mode control register 6 (TMC6)

TMC6 is the register that controls the operations of 8-bit timer register 6 (TM6).
TMC6 is set by 1-bit or 8-bit memory manipulation instruction.
$\overline{\text{RESET}}$ signal input resets the register to 00H.

Figure 14-2. Format of Timer Mode Control Register 6 (TMC6)

Address: 0FF54H After Reset: 00H R/W

Symbol	⑦	6	5	4	3	2	1	⓪
TMC6	TCE6	TMC66	0	0	0	0	TMC61	TOE6

TCE6	TM6 Count Operating Control
0	Stop counting
1	Start counting

TMC66	TM6 Operating Mode Control
0	Interval mode
1	Free running mode

TMC61	Specify of Active Level of Timer Output
0	Active level “1” (high)
1	Active level “0” (low)

TOE6	TM6 Output Control
0	Disable output (output is set to 0 level)
1	Enable output

Cautions 1. Always set TM6 after stopping TM6’s timer operations.

2. When outputting PWM from TM6, set bit 6 (TOE6) to “1”. When setting TOE6 to “0”, the timer output (TO6) pin outputs “0”.

★ **3. When setting timer output enable (TOE6 = 1) in the interval mode (TMC66 = 0), the TO6 pin outputs a fixed value of inactive level. To output PWM from the TO6 pin, be sure to set the free running mode (TMC66 = 1).**

(2) Timer clock select register 6 (TCL6)

The timer clock select register 6 (TCL6) specifies the count-clock pulse for the 8-bit timer register 6 (TM6).

TCL6 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ signal input resets the register to 00H.

Figure 14-3. Format of Timer Clock Select Register 6 (TCL6)

Address: 0FF56H After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
TCL6	0	0	0	0	0	TCL62	TCL61	TCL60

TCL62	TCL61	TCL60	Count Clock Selection
0	0	0	f_{CLK}/4 (2 MHz)
0	0	1	f_{CLK}/8 (1 MHz)
0	1	0	f_{CLK}/16 (500 kHz)
0	1	1	f_{CLK}/64 (125 kHz)
1	0	0	f_{CLK}/128 (62.5 kHz)
1	0	1	f_{CLK}/512 (15.6 kHz)
1	1	0	Setting prohibited
1	1	1	

Caution Always program TCL6 after stopping timer operations

Remark Items in parentheses indicate operations at f_{CLK} = 8 MHz.

14.4 Operation

14.4.1 Basic operation of TM6

The 8-bit timer counter (TM6) is an 8-bit free running or interval timer for counting pulses. The counter is incremented through synchronization with the rise in the input clock pulse.

Inputting the $\overline{\text{RESET}}$ signal clears all TM6 bits (to 0) and stops the count operation.

The enabling and inhibiting of the count operation is controlled by the timer mode control register (TMC6)'s bit 7 (TCE6). Setting TCE6 to "1" starts the count operation. Setting TCE6 to "0" clears TM6 and stops the count operation.

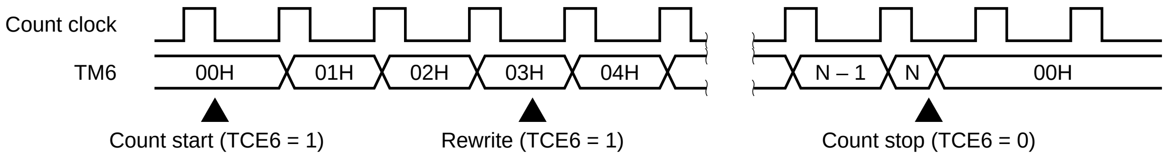
Resetting also makes the count 00H.

After the start count is set, the first count clock input makes TM 00H to 01H.

TM6 does not clear the timer, but continues count operations even if the same operation mode is set again.

The count does not stop during the TM6 read duration.

Figure 14-4. Basic Operation Timing of TM6



14.4.2 Interval operation of TM6

Setting the timer mode control register 6 (TMC6)'s bit 6 (TMC66) to "0" places the 8-bit timer/counter 6 (TM6) in interval operation. When the values in TM6 and the 8-bit compare register 6 (CR6) are the same, the next count clock pulse generates an interrupt request signal (INTTM6) that clears TM6 (to 00H).

Setting TM6's bit 0 (TOE6) to "1" enables timer output (TO6). The enabling of TO6 output outputs an inactive level of fixed value.

Caution PWM cannot be output in the interval mode.

Figure 14-5. Interval Operation Timing of TM6 (CR6 ≠ 00H)

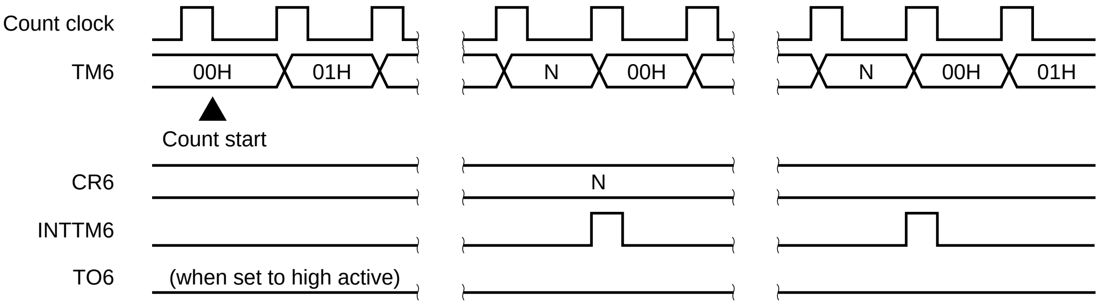
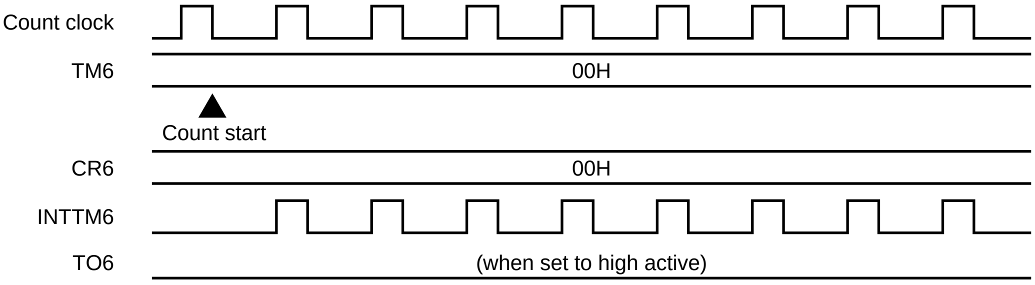


Figure 14-6. Interval Operation Timing of TM6 (CR6 = 00H)



14.4.3 Free running operation of TM6 (PWM output)

Setting the timer mode control register 6 (TMC6)'s bit 6 (TMC66) to "1" places TM6 in free running operation. TM6's overflow generates an interrupt request signal (INTTM6)

Setting TMC6's bit 0 (TOE6) to "1" enables timer output (TO6). Setting TMC6's bit 7 to "0" stops the count and places TO6 on inactive level.

PWM can be output in this mode.

(How to set PWM output)

<1> Set the timer 6 output to free running mode (TMC6's bit 6 (TMC66) to "1").

<2> Set TO6 to active level (set TMC6's bit 1 (TMC61)) and set TO6's output to enable (set TMC6's bit 0 (TOE6) to "1").

<3> Select the count clock timer using the timer clock select register 6 (TCL6)'s bits 0 to 2 (TCL60 to TCL62).

<4> Set TMC6's bit 7 (TCE6) to "1" to start the count operation and output a PWM signal from pin TO6.

★ **Cautions 1. PWM duration = count clock rate × 256 and active level width = count clock rate × CR6's value.**

2. CR6's compare equivalence interrupt is not generated in the free running mode.

Figure 14-7. Free Running Mode Operation Timing of TM6 (CR6 ≠ 00H and FFH)

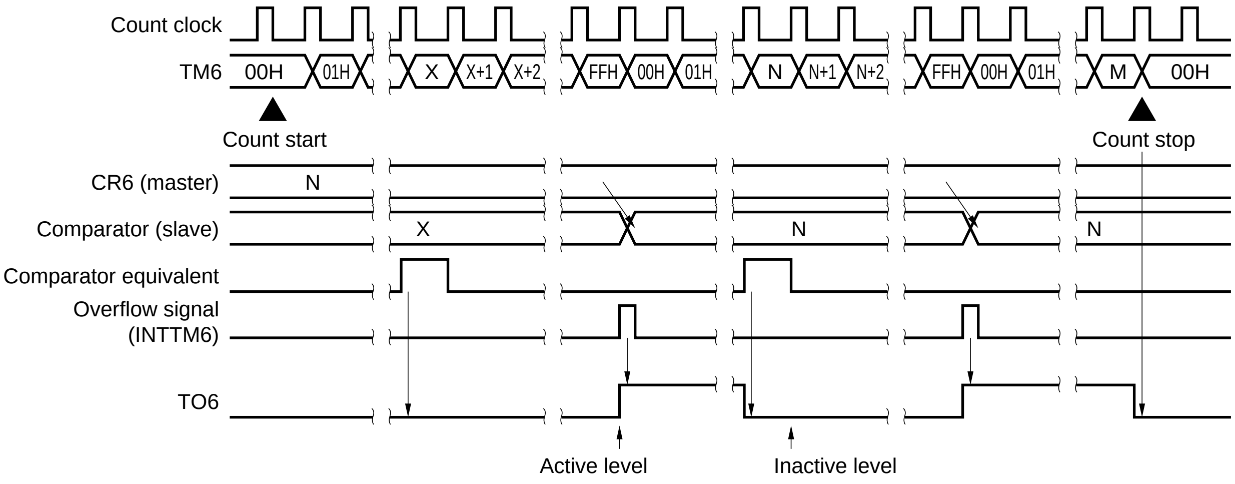


Figure 14-8. Free Running Mode Operation Timing of TM6 (CR6 = 00H)

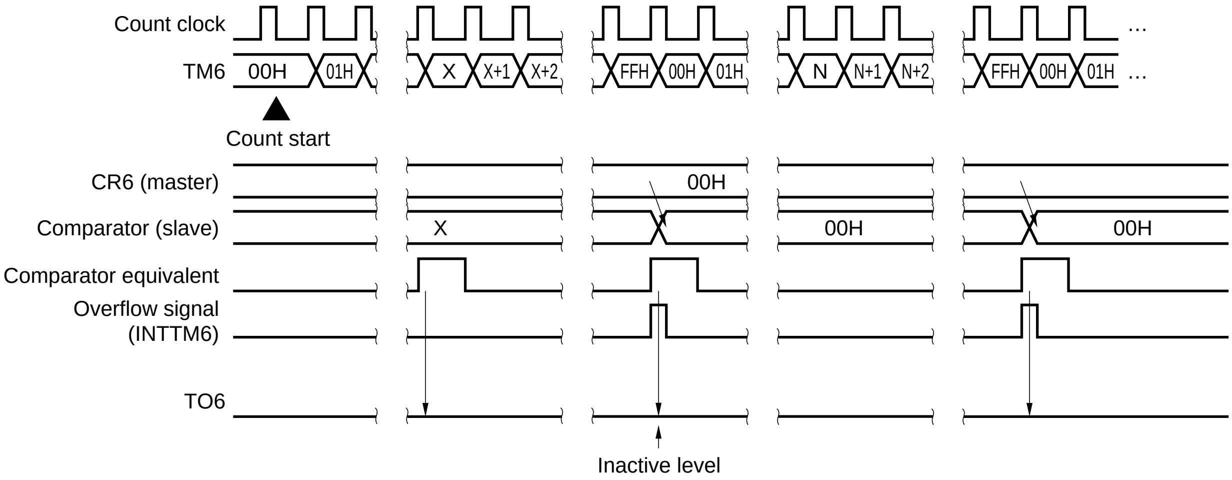
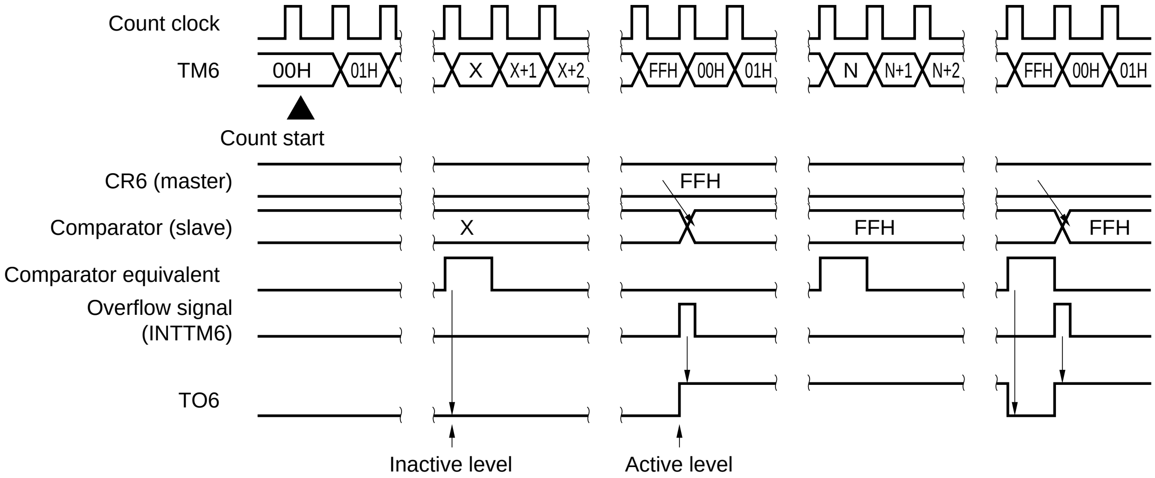


Figure 14-9. Free Running Mode Operation Timing of TM6 (CR6 = FFH)

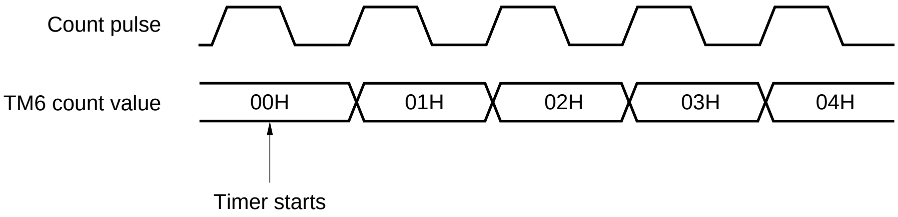


★ 14.5 Cautions

(1) Error at timer start

After the timer starts, the time until a uniform signal is generated may have a maximum error of 1 clock. This is because the start of 8-bit timer register 6 (TM6) is asynchronous with respect to the count pulse.

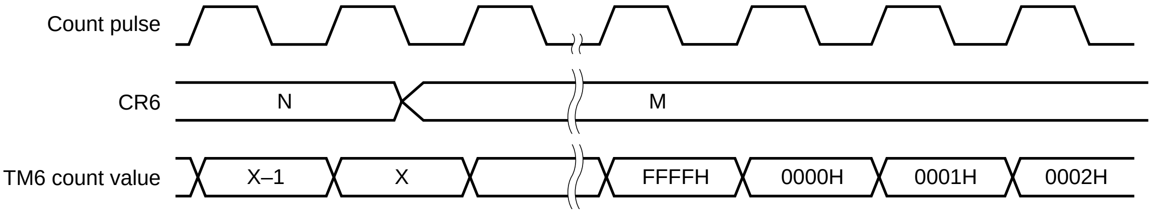
Figure 14-10. Start Timing of 8-Bit Timer Register 6



(2) Operation after changes in the compare register during timer count operation

If the value of 8-bit compare register 6 (CR6) after changing is less than the value of 8-bit timer register 6 (TM6), TM6 continues counting, and when an overflow occurs, it starts over from 0. Also, if the value of CR6 after changing (M) is less than the value before changing (N), the timer must be restarted after CR6 changes.

Figure 14-11. Timing After the Compare Register Changes During Timer Counting



Caution Always set TCE6 = 0 before setting the STOP mode.

Remark N > X > M

CHAPTER 15 8-BIT TIMER/COUNTER 7

15.1 Functions

The 8-bit timer/counter 7 (TM7) functions as follows.

- Interval timer
  An interrupt request is generated at a time interval set in advance.

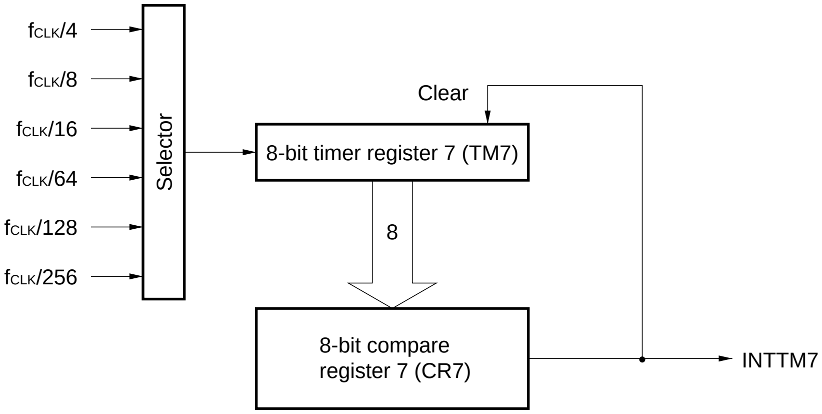
15.2 Configuration

The 8-bit timer/counter 7 (TM7) has the hardware configuration shown below.

Table 15-1. Configuration of 8-Bit Timer/Counter 7 (TM7)

Item	Configuration
Timer register	8-bit × 1 (TM7)
Register	Compare register: 8-bit × 1 (CR7)
Control registers	Timer mode control register 7 (TMC7) Timer clock selector register 7 (TCL7)

Figure 15-1. Block Diagram of 8-Bit Timer/Counter 7



(1) 8-bit timer register 7 (TM7)

TM7 is an 8-bit interval timer that counts count pulses.
It increments the counter in synchronization with the rising edge of the input clock.
The count then becomes 00H.

<1> Inputs $\overline{\text{RESET}}$ signal
<2> Clears the 8-bit timer mode control register 7 (TMC7)'s bit 7 (TCE7).
<3> TM7 and CR7 match.

(2) 8-bit compare register 7 (CR7)

The TM7 count and the values programmed in CR7 are always compared and an interrupt request (INTTM7) generated when they match. TM7 can also be used as a register for saving interval time.
CR7 is set by an 8-bit memory manipulation instruction.
$\overline{\text{RESET}}$ input makes CR7 undefined.

15.3 Control Registers

The following two types of registers control 8-bit timer/counter 7 (TM7).

- Timer mode control register 7 (TMC7)
- Timer clock select register 7 (TCL7)

(1) Timer mode control register 7 (TMC7)

TMC7 is the register that controls the operations of 8-bit timer register 7 (TM7).
TMC7 is set by 1-bit or 8-bit memory manipulation instruction.
$\overline{\text{RESET}}$ input sets TMC7 to 00H.

Figure 15-2. Format of Timer Mode Control Register 7 (TMC7)

Address: 0FF55H After Reset: 00H R/W

Symbol	⑦	6	5	4	3	2	1	0
TMC7	TCE7	0	0	0	0	0	0	0

TCE7	TM7 Count Operating Control
0	Stop counting
1	Start counting

Caution Always set bit 6 of TMC7 to 0.

(2) Timer clock select register 7 (TCL7)

The timer clock select register 7 specifies the count-clock pulse for the 8-bit timer register 7 (TM7).

TCL7 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets TCL7 to 00H.

Figure 15-3. Format of Timer Clock Select Register 7 (TCL7)

Address: 0FF57H After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
TCL7	0	0	0	0	0	TCL72	TCL71	TCL70

TCL72	TCL71	TCL70	Count Clock Selection
0	0	0	f_{CLK}/4 (2 MHz)
0	0	1	f_{CLK}/8 (1 MHz)
0	1	0	f_{CLK}/16 (500 kHz)
0	1	1	f_{CLK}/64 (125 kHz)
1	0	0	f_{CLK}/128 (62.5 kHz)
1	0	1	f_{CLK}/256 (31.25 kHz)
1	1	0	Setting prohibited
1	1	1	

Caution Always set TCL7 after stopping timer operations.

Remark Items in parentheses indicate operations at f_{CLK} = 8 MHz.

15.4 Operation

15.4.1 Basic operation of TM7

The 8-bit timer/counter (TM7) is an 8-bit interval timer for counting count pulses. The counter is incremented through synchronization with the rise in the input clock pulse.

Inputting the $\overline{\text{RESET}}$ signal clears all TM7 bits (to 0) and stops the count operation.

The enabling and inhibiting of the count operation is controlled by the timer mode control register 7 (TMC7)'s bit 7 (TCE7). Setting TCE7 to "1" starts the count operation. Setting TCE7 to "0" clears TM7 and stops the count operation.

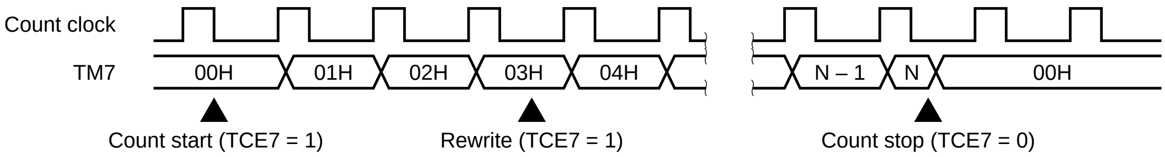
Resetting also makes the count 00H.

After the start count is set, the first count clock input makes TM7 00H to 01H.

TM7 does not clear the timer, but continues count operations even if the same operation mode is set again.

The count does not stop during the TM7 read duration.

Figure 15-4. Basic Operation Timing of TM7



15.4.2 Interval operation of TM7

When the values in TM7 and 8-bit compare register 7 (CR7) match, the next count clock pulse generates an interrupt request signal (INTTM7) that clears TM7 (to 00H). The count then continues.

Figure 15-5. Interval Operation Timing of TM7 (CR7 ≠ 00H)

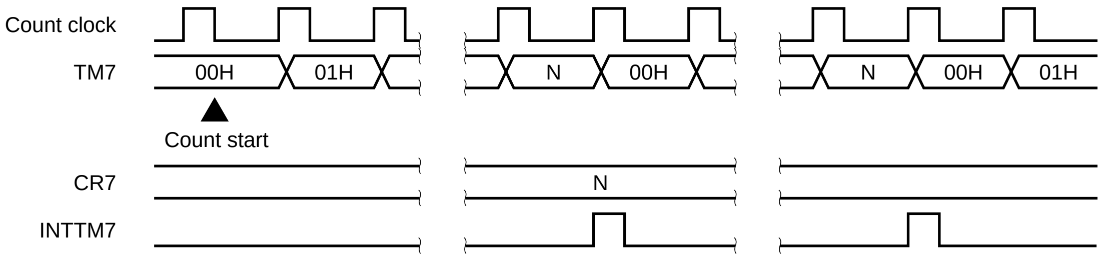
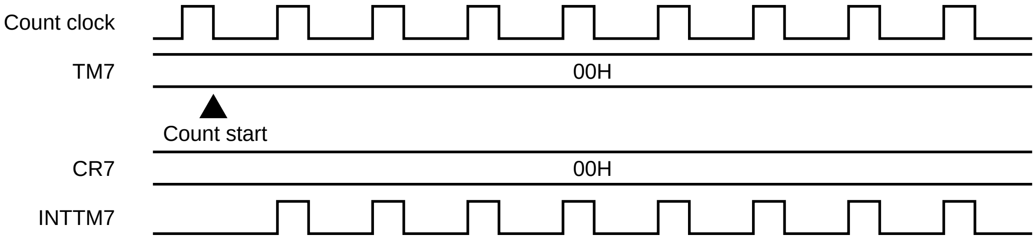


Figure 15-6. Interval Operation Timing of TM7 (CR7 = 00H)

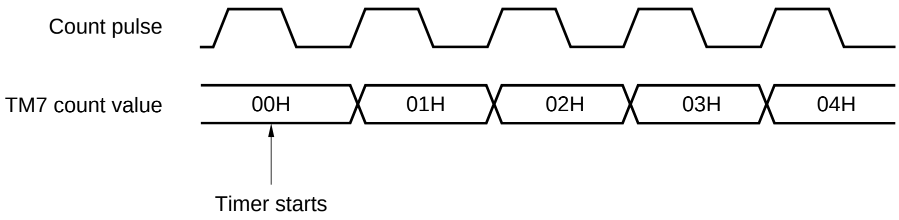


★ 15.5 Cautions

(1) Error at timer start

After the timer starts, the time until a uniform signal is generated may have a maximum error of 1 clock. This is due to the fact that the start of 8-bit timer register 7 (TM7) is asynchronous with respect to the count pulse.

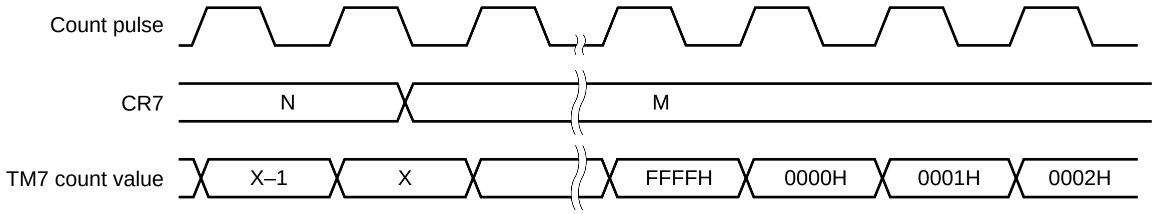
Figure 15-7. Start Timing of 8-Bit Timer Register 7



(2) Operation after changes in the compare register during timer count operation

If the value of 8-bit compare register 7 (CR7) after changing is less than the value of 8-bit timer register 7 (TM7), TM7 continues counting, and when an overflow occurs, it starts over from 0. Also, if the value of CR7 after changing (M) is less than the value before changing (N), the timer must be restarted after CR7 changes.

Figure 15-8. Timing After the Compare Register Changes During Timer Counting



Caution Always set TCE7 = 0 before setting the STOP mode.

Remark N > X > M

CHAPTER 16 WATCHDOG TIMER

The watchdog timer detects program runaway.

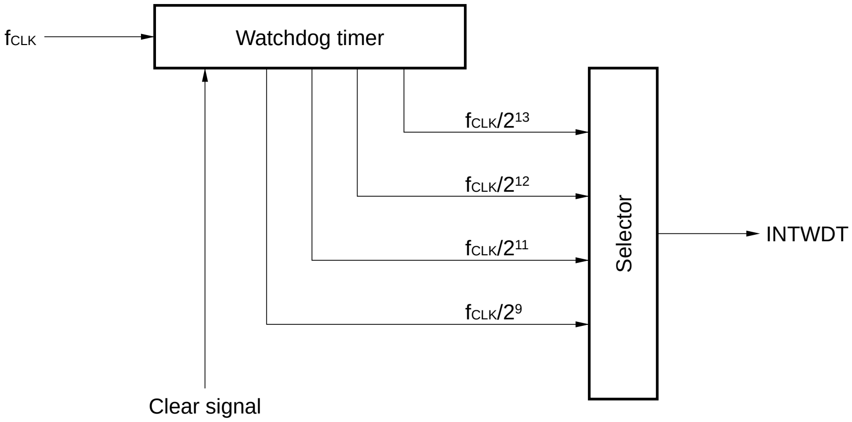
Program or system errors are detected by the generation of watchdog timer interrupts. Therefore, at each location in the program, the instruction that clears the watchdog timer (starts the count) within a constant time should be input.

If the watchdog timer overflows without executing the instruction that clears the watchdog timer within the set period, a watchdog timer interrupt (INTWDT) is generated to signal a program error.

16.1 Configuration

Figure 16-1 is a block diagram of the watchdog timer.

Figure 16-1. Watchdog Timer Block Diagram



Remark f_{CLK}: Internal system clock (8 MHz)

16.2 Control Register

- **Watchdog timer mode register (WDM)**

WDM is the 8-bit register that controls watchdog timer operation.

To prevent the watchdog timer from erroneously clearing this register due to a program runaway, this register is only written by a special instruction. This special instruction is the MOV WDM #byte instruction, which has a special code format (4 bytes). Writing takes place only when the third and fourth op codes are mutual 1's complements. If the third and fourth op codes are not mutual 1's complements and not written, the operand error interrupt is generated. In this case, the return address saved in the stack is the address of the instruction that caused the error. Therefore, the address that caused the error can be identified from the return address saved in the stack. If returning by simply using the RETB instruction from the operand error, an infinite loop results.

Since an operand error interrupt is generated only when the program is in a runaway state (the correct special instruction is only generated when "MOV WDM, #byte" is described in the RA78K4 NEC assembler), make the program initialize the system.

Other write instructions (MOV WDM, A; AND WDM, #byte; SET1 WDM.7, etc.) are ignored and no operation is executed. In other words, WDM is not written, and interrupts, such as operand error interrupts, are not generated.

After a system reset ($\overline{\text{RESET}}$ input), when the watchdog timer starts (when the RUN bit is set to one), the WDM contents cannot change. Only a reset can stop the watchdog timer. The watchdog timer can be cleared by a special instruction.

WDM can be read at any time by an 8-bit data transfer instruction.

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

Figure 16-2 shows the WDM format.

Figure 16-2. Watchdog Timer Mode Register (WDM) Format

Address: 0FFC2H After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
WDM	RUN	0	0	WDT4	0	WDT2	WDT1	0

RUN	Watchdog Timer Operation Setting
0	Stops the watchdog timer.
1	Clears the watchdog timer and starts counting.

WDT4	Watchdog Timer Interrupt Request Priority
0	Watchdog timer interrupt request <NMI pin input interrupt request
1	Watchdog timer interrupt request >NMI pin input interrupt request

WDT2	WDT1	Count Clock	Overflow Time [ms] (f_{CLK} = 8 MHz)
0	0	$f_{CLK}/2^{9}$	16.4
0	1	$f_{CLK}/2^{11}$	65.5
1	0	$f_{CLK}/2^{12}$	131.1
1	1	$f_{CLK}/2^{13}$	262.1

Cautions

1. **Only the special instruction (MOV WDM, #byte) can write to the watchdog timer mode register (WDM).**
2. **When writing to WDM to set the RUN bit to 1, write the same value every time. Even if different values are written, the contents written the first time cannot be updated.**
3. **When the RUN bit is set to 1, it cannot be reset to 0 by means of the software.**

Remark f_{CLK}: Internal system clock frequency

16.3 Operations

16.3.1 Count operation

The watchdog timer is cleared by setting the RUN bit of the watchdog timer mode register (WDM) to 1 to start counting. After the RUN bit is set to 1, when the overflow time set by bits WDT2 and WDT1 in WDM has elapsed, a non-maskable interrupt (INTWDT) is generated.

If the RUN bit is reset to 1 again before the overflow time elapses, the watchdog timer is cleared, and counting restarts.

16.3.2 Interrupt priority order

The watchdog timer interrupt (INTWDT) is a non-maskable interrupt. In addition to INTWDT, the non-maskable interrupts include the interrupt (NMI) from the NMI pin. By setting bit 4 of the watchdog timer mode register (WDM), the acknowledgement order when INTWDT and NMI are simultaneously generated can be set.

If acknowledging NMI is given priority, even if INTWDT is generated in an NMI processing program that is executing, INTWDT is not acknowledged, but is acknowledged after the NMI processing program ends.

16.4 Cautions

16.4.1 General cautions when using the watchdog timer

(1) The watchdog timer is one way to detect a runaway operation, but all runaway operations cannot be detected. Therefore, in a device that particularly demands reliability, the runaway operation must be detected early not only by the on-chip watchdog timer but by an externally attached circuit; and when returning to the normal state or while in the stable state, processing like stopping the operation must be possible.

(2) The watchdog timer cannot detect runaway operation in the following cases.

<1> When the watchdog timer is cleared in a timer interrupt servicing program
<2> When interrupt requests and macro services are temporarily held pending successively (see **22.9 When Interrupt Requests and Macro Service Are Temporarily Held Pending**)
<3> When runaway operation is caused by logical errors in the program (when each module in the program operates normally, but the entire system does not operate properly), and when the watchdog timer is periodically cleared
<4> When the watchdog timer is periodically cleared by an instruction group that is executed during a runaway operation
<5> When the STOP mode, HALT mode, or IDLE mode is the result of a runaway operation
<6> When the watchdog timer also goes into an inadvertent program loop when the CPU goes upset because of introduced noise

In cases <1>, <2>, and <3>, detection becomes possible by correcting the program.
In case <4>, the watchdog timer can be cleared only by the 4-byte special instruction. Similarly in <5>, if there is no 4-byte special instruction, the STOP mode, HALT mode, or IDLE mode cannot be set. Since the result of the runaway operation is to enter state <2>, three or more bytes of consecutive data must be a specific pattern (example, BT PSWL.bit, $$). Therefore, it is very rare that state <2> is brought about due to the results of <4>, <5>, and the runaway operation.

16.4.2 Cautions about the μPD784955 Subseries watchdog timer

(1) Only the special instruction (MOV WDM, #byte) can write to watchdog timer mode register (WDM).

(2) If the RUN bit is set to 1 by writing to the watchdog timer mode register (WDM), write the same value every time. Even when different values are written, the contents written the first time cannot be changed.

(3) If the RUN bit is set to 1, it cannot be reset to 0 by means of the software.

CHAPTER 17 A/D CONVERTER

17.1 Functions

The A/D converter converts analog inputs to digital values, and is configured by eight 8-bit resolution channels (ANI0 to ANI7).

Successive approximation is used as the conversion method, and conversion results are saved in the 8-bit A/D conversion result register 0 (ADCR0).

A/D conversion can be begun by the following two methods.

(1) Hardware start

Conversion is started by trigger input (P01) (rising edge, falling edge, or both rising and falling edges can be specified).

(2) Software start

Conversion is started by setting the A/D converter mode register 0 (ADM0).

Select one channel for analog input from ANI0 to ANI7, and perform A/D conversion. If hardware start is used, A/D conversion stops at the end of the A/D conversion operation. If software start is used, the A/D conversion operation is repeated. Each time one A/D conversion is completed, an interrupt request (INTAD) is issued.

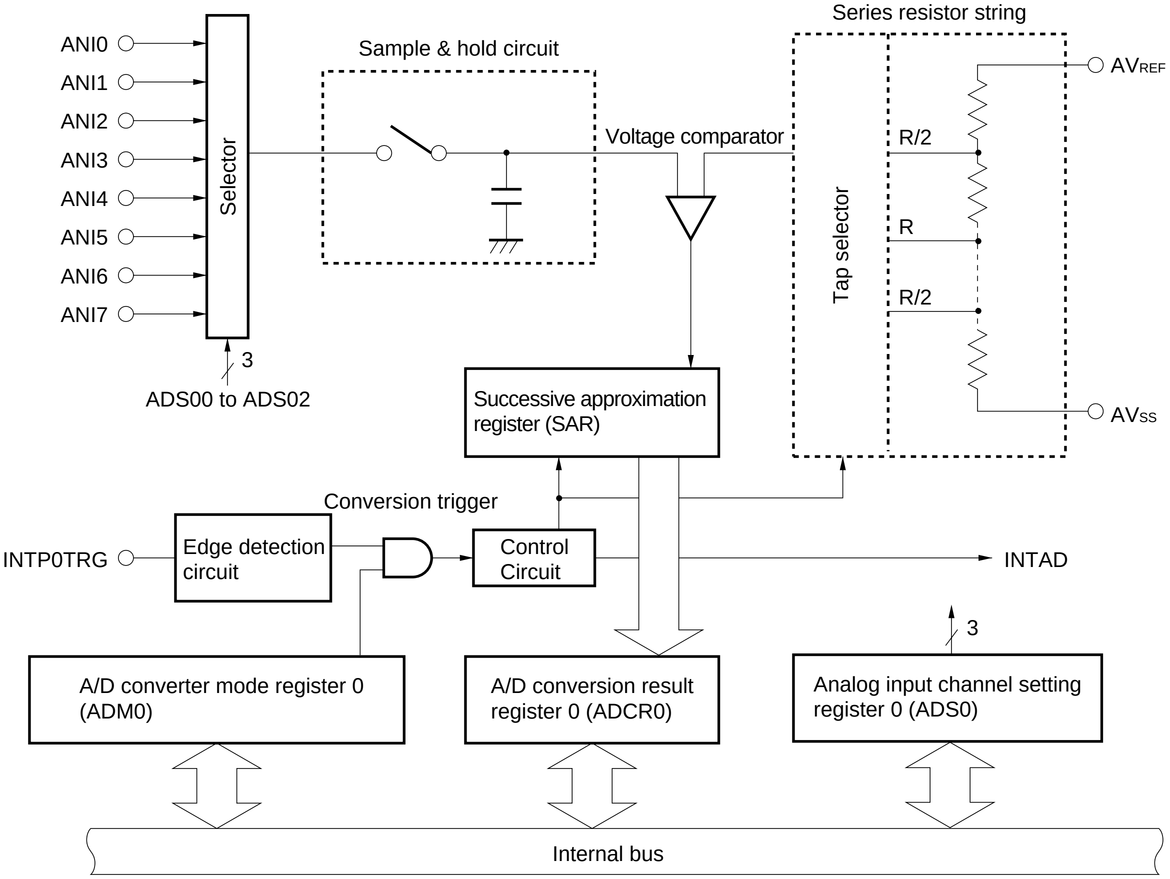
17.2 Configuration

The A/D converter has the following hardware configuration.

Table 17-1. Configuration of A/D Converter

Item	Configuration
Analog input	8 channels (ANI0 to ANI7)
Control registers	A/D converter mode register 0 (ADM0) Analog input channel setting register 0 (ADS0)
Registers	Successive approximation register (SAR) A/D conversion result register 0 (ADCR0)

★ **Figure 17-1. A/D Converter Block Diagram**



(1) Successive approximation register (SAR)

Compares the voltage of the analog input with the voltage tap (comparison voltage) from the series resistor string, and saves the result from the most significant bit (MSB).

The contents of SAR will be transmitted across to the A/D conversion result register 0 (ADCR0) everything down to the least significant bit (LSB) is retained (A/D conversion finished).

(2) A/D conversion result register 0 (ADCR0)

Holds A/D conversion results. At the end of each A/D conversion operation, the conversion result from the successive approximation register is loaded.

ADCR is read with an 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input makes its contents undefined.

(3) Sample & hold circuit

Samples analog inputs one by one as they are sent from the input circuit, and sends them to the voltage comparator. The sampled analog input voltages are held during A/D conversion.

(4) Voltage comparator

Compares the analog input voltage with the output voltage of the series resistor string.

(5) Series resistor string

Placed between AV_{REF} and AV_{SS}, generates the voltage that is compared with that of analog input signal.

(6) ANI0 to ANI7 pins

Eight analog input channels used for inputting analog data to the A/D converter for A/D conversion. Pins not selected for analog input with the analog input channel setting register 0 (ADS0) can be used as input ports.

Cautions 1. The ANI0 to ANI7 input voltages should be within the rated voltage range. Inputting a voltage equal to or greater than AV_{REF}, or equal to or smaller than AV_{SS} (even if within the absolute maximum rated range) will cause the channel's conversion values to become undefined, or may affect the conversion values of other channels.

2. Analog input (AN10 to AN17) pins alternate with input port (P70 to P77) pins. When performing an A/D conversion with the selection of any one of inputs from AN10 to AN17, do not execute input instructions to port 7 during conversion. Otherwise, the conversion resolution may decrease. When a digital pulse is applied to the pin which adjoins a pin in the A/D conversion, an expected A/D conversion value may not be acquired due to the coupling noise. Therefore do not apply a pulse to the pin which adjoins the pin in the A/D conversion.

(7) AV_{SS} pin

Ground pin of the A/D converter. Always keep this pin at the same potential as the V_{SS} pin, even when not using the A/D converter.

(8) AV_{REF} pin

Reference voltage input pin of the A/D converter. Always keep this pin at the same potential as the V_{DD} pin, even when not using the A/D converter.

17.3 Control Registers

The A/D converter is controlled by the following two registers.

- A/D converter mode register 0 (ADM0)
- Analog input channel setting register 0 (ADS0)

(1) A/D converter mode register 0 (ADM0)

Used to set the A/D conversion time of analog input to be converted, start/stop of conversion operation, and external triggers.

ADM0 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets ADM0 to 00H.

Figure 17-2. A/D Converter Mode Register 0 (ADM0) Format

Address: 0FF80H After Reset: 00H R/W

Symbol	⑦	⑥	⑤	④	③	②	①	0
ADM0	ADCS0	TRG0	FR02	FR01	FR00	EGA01	EGA00	0

ADCS0	A/D Conversion Control
0	Conversion stop
1	Conversion enable

TRG0	Software Start/Hardware Start Selection
0	Software start
1	Hardware start

FR02	FR01	FR00	A/D Conversion Time Selection	
			Number of clocks	When f_{CLK} = 8 MHz
0	0	0	144/f_{CLK}	18.0 μs
0	0	1	120/f_{CLK}	15.0 μs
0	1	0	96/f_{CLK}	12.0 μs (conversion disable)
1	0	0	288/f_{CLK}	36.0 μs
1	0	1	240/f_{CLK}	30.0 μs
1	1	0	192/f_{CLK}	24.0 μs
Other than above			—	Setting prohibited

EGA01	EGA00	External Trigger Signal Valid Edge Selection
0	0	No edge detection
0	1	Detection of falling edge
1	0	Detection of rising edge
1	1	Detection of both falling and rising edges

Cautions 1. Do not set the A/D conversion time to 14 μs or less.

2. When overwritting bits FR00 to FR02 with unidentical data, temporarily halt A/D conversion before continuing.

Remark f_{CLK} : Internal system clock frequency

(2) Analog input channel setting register 0 (ADS0)

Used to specify the input ports for analog signals to be A/D converted.

ADS0 can be set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets ADS0 to 00H.

Figure 17-3. Analog Input Channel Setting Register 0 (ADS0) Format

Address: 0FF81H After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
ADS0	0	0	0	0	0	ADS02	ADS01	ADS00

ADS02	ADS01	ADS00	Analog Input Channel Setting
0	0	0	ANI0
0	0	1	ANI1
0	1	0	ANI2
0	1	1	ANI3
1	0	0	ANI4
1	0	1	ANI5
1	1	0	ANI6
1	1	1	ANI7

17.4 Operations

★ ### 17.4.1 Basic operations of A/D converter

<1> Select one channel for A/D conversion with the analog input channel setting register 0 (ADS0).

<2> The voltage input to the selected analog input channel is sampled by the sample & hold circuit.

<3> After sampling has been performed for a certain time, the sample & hold circuit enters the hold status, and the input analog voltage is held until A/D conversion ends.

<4> Bit 7 of the successive approximation register (SAR) is set. The tap selector sets the voltage tap for the series resistor string at (1/2)AV_{REF}.

<5> The difference in voltage between the series resistor string's voltage tap and analog input is compared with the voltage comparator. If the analog input is greater than (1/2)AV_{REF}, the setting for the SAR MSB will remain the same. If it is smaller than (1/2)AV_{REF}, the MSB will be reset.

<6> Next, bit 6 of SAR is automatically set, and the next comparison is started. The series resistor string voltage tap is selected as shown below according to bit 7 to which a result has already been set.

- Bit 7 = 1 : (3/4) AV_{REF}
- Bit 7 = 0 : (1/4) AV_{REF}

The voltage tap and analog input voltage are compared, and bit 6 of SAR is manipulated according to the result, as follows.

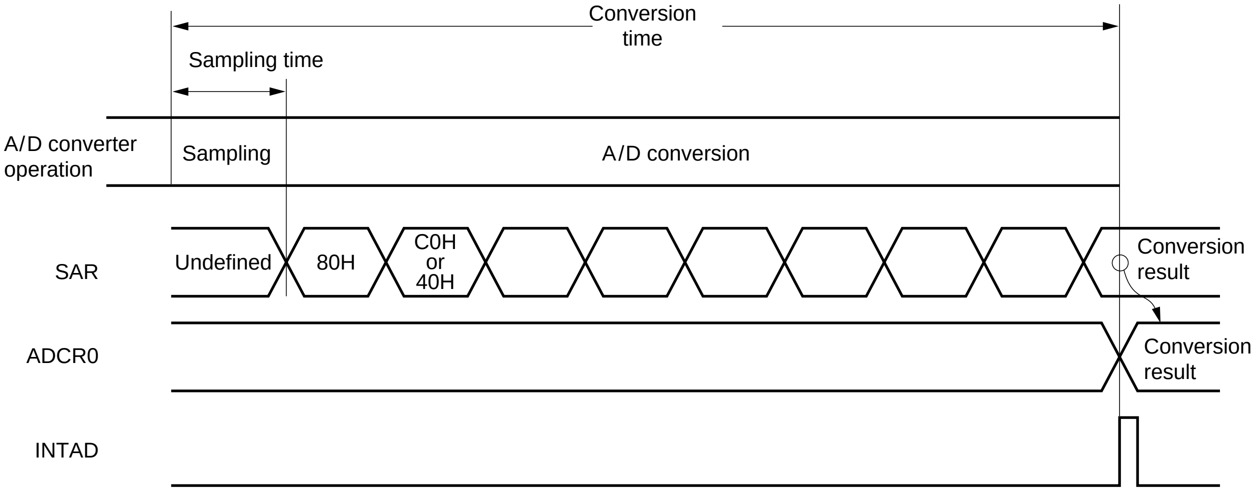
- Analog input voltage ≥ Voltage tap : Bit 6 = 1
- Analog input voltage < Voltage tap : Bit 6 = 0

<7> Comparisons of this kind are repeated until bit 0 of SAR.

<8> When comparison of all eight bits is completed, the valid digital result remains in SAR, and this value is transferred to the A/D conversion result register 0 (ADCR0) and latched.

At the same time, it is possible to have an A/D conversion end interrupt request (INTAD) issued.

Figure 17-4. Basic Operation Timing of A/D Converter



A/D conversion is performed continuously until the bit 7 (ADCS0) of the A/D converter mode register 0 (ADM0) is reset to 0 by means of software.

If a write operation to ADM0 or analog input channel setting register 0 (ADS0) is performed during A/D conversion, the conversion operation is initialized and conversion starts from the beginning if the ADCS 0 bit is set 1.

$\overline{\text{RESET}}$ input makes the A/D conversion result register 0 (ADCR0) undefined.

17.4.2 Input voltage and conversion result

The relationship between the analog input voltage input to the analog input pins (ANI0 to ANI7) and the A/D conversion result (value saved in A/D conversion result register 0 (ADCR0)) is expressed by the following equation.

$$\mathrm{ADCR} = \mathrm{INT}\,(\frac{V_{\mathrm{IN}}}{AV_{\mathrm{REF}}} \times 256 + 0.5)$$

or

$$(\mathrm{ADCR} - 0.5) \times \frac{AV_{\mathrm{REF}}}{256} \leq V_{\mathrm{IN}} < (\mathrm{ADCR} + 0.5) \times \frac{AV_{\mathrm{REF}}}{256}$$

Remark INT( ) : Function returning the integer portion of the value in parentheses

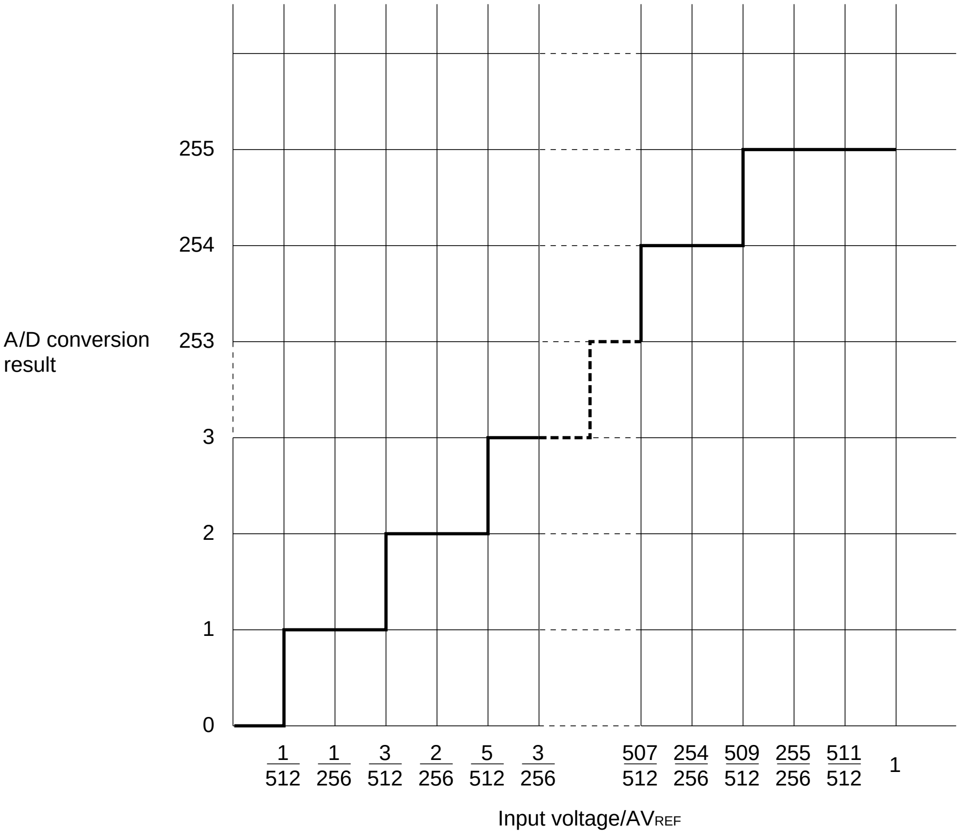
V_{IN} : Analog input voltage

AV_{REF} : AV_{REF} pin voltage

ADCR : A/D conversion result register 0 (ADCR0) value

Figure 17-5 shows the relationship between analog input voltage and the A/D conversion result.

Figure 17-5. Relationship between Analog Input Voltage and A/D Conversion Result



17.4.3 Operations mode of A/D converter

Select one channel for analog input from among ANI0 to ANI7 with the analog input channel setting register 0 (ADS0) and commence A/D conversion.

A/D conversion can be started in the following two ways.

- Hardware start : Conversion start by trigger input (P01)
- Software start : Conversion start by setting A/D converter mode register 0 (ADM0)

The result of A/D conversion will be stored in the A/D conversion result register 0 (ADCR0), and at the same time, an interrupt request signal (INTAD) will be issued.

(1) A/D conversion operation by hardware start

The A/D conversion operation can be made to enter the standby status by setting “1” to bit 6 (TRG) and bit 7 (ADCS0) of the A/D converter mode register 0 (ADM0). When an external trigger signal (P01) is input, conversion of the voltage applied to the analog input pin set with ADS0 begins.

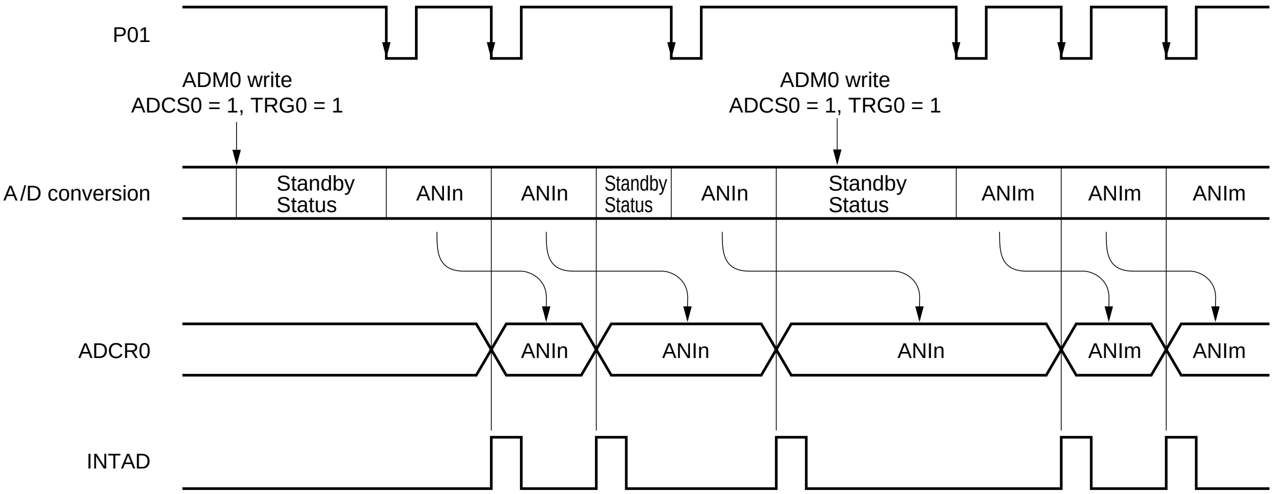
The result of conversion will be stored in the A/D conversion result register 0 (ADCR0) when A/D conversion operation have finished, and an interrupt request signal (INTAD) will be issued. When the A/D conversion operation that was started completes the first A/D conversion, no other A/D conversion operation is started unless an external trigger signal is input.

Rewriting ADM0’s bit 7 (ADCS0) during an A/D conversion operation cancels the A/D conversion operating at that time and waits until a new external trigger signal is input. The A/D conversion process will be restarted from the beginning when an external trigger signal is input once again. The A/D conversion process will be started when the next external trigger signal is received when ADCS is overwritten with A/D conversion in the stand-by mode.

If, during A/D conversion, data whose ADCS0 is 0 is written to ADM0, A/D conversion is immediately stopped.

Caution When P01/INTP0 is used as the external trigger input (P01), specify a valid edge with bits 1 and 2 (EGA00 and EGA01) of the A/D converter mode register 0 (ADM0) and set 1 to the interrupt mask flag (PMK1).

Figure 17-6. A/D Conversion Operation by Hardware Start (When Falling Edge Is Specified)



Remark n = 0, 1, ... , 7
m = 0, 1, ... , 7

(2) A/D conversion operation by software start

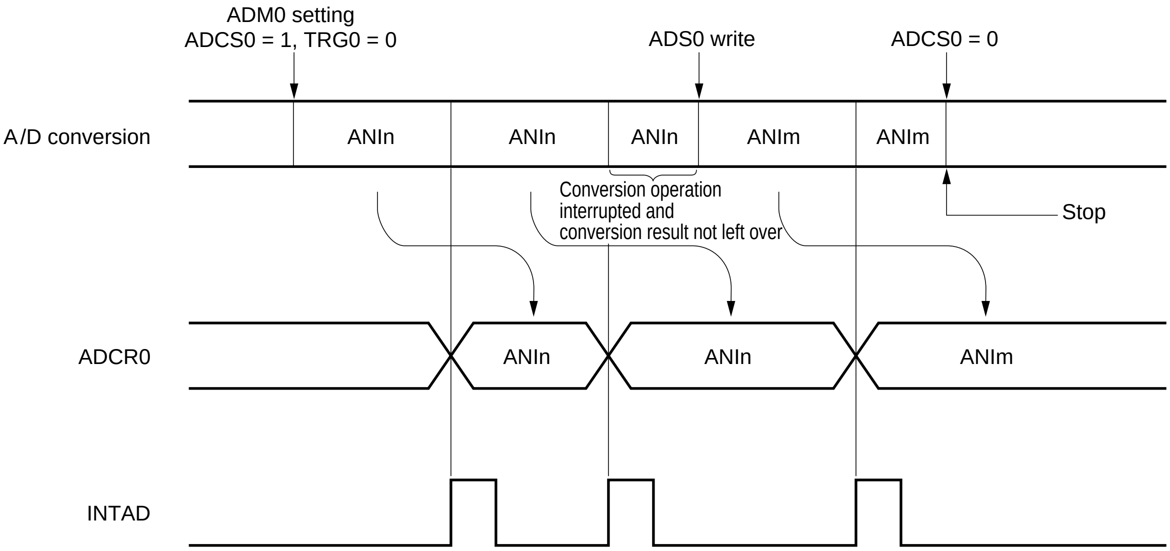
A/D conversion of the voltage applied to the analog input pin specified with analog input channel setting register 0 (ADS0) is started by setting “0” to bit 6 (TRG0) and “1” to bit 7 (ADCS0) of the A/D converter mode register 0 (ADM0).

When A/D conversion ends, the conversion result is saved in the A/D conversion result register 0 (ADCR0), and an interrupt request signal (INTAD) is issued. When an A/D conversion operation that was started completes the first A/D conversion, the next A/D conversion starts immediately. A/D conversion operations are performed continuously until new data is written to ADM0.

The A/D conversion process will be suspended if ADCS0 is overwritten during A/D conversion operations, and A/D conversion operations for the newly selected analog input channel will be started.

If, during A/D conversion, data where ADCS0 is 0 is written to ADM0, the A/D conversion operation is immediately stopped.

Figure 17-7. A/D Conversion Operation by Software Start



Remark n = 0, 1, ... , 7
m = 0, 1, ... , 7

17.5 Cautions

(1) ANI0 to ANI7 input range

ANI0 to ANI7 input voltages should be within the rated voltage range. Inputting a voltage equal to or greater than AV_{REF}, or equal to or smaller than AV_{SS} (even if within the absolute maximum rated range) will cause the channel's conversion values to become undefined, or may affect the conversion values of other channels.

(2) Contention operation

<1> Contention between A/D conversion result register 0 (ADCR0) write at conversion end and ADCR0 read clue to instruction

The read operation to ADCR0 is prioritized. After the read operation, a new conversion result is written to ADCR0.

<2> Contention between ADCR0 write at conversion end and external trigger signal input

External trigger signals cannot be received during A/D conversion. Therefore, external trigger signals during ADCR0 write operation are not accepted.

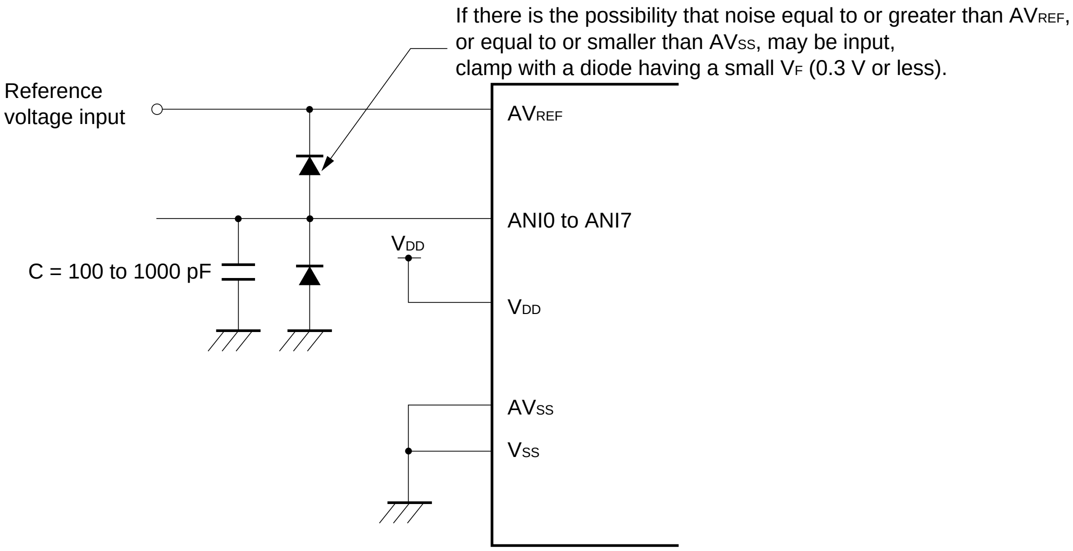
<3> Contention between ADCR0 write at conversion end and A/D converter mode register 0 (ADM0) write, or between ADCR0 write at conversion end and analog input channel setting register 0 (ADS0) write.

The write operation to ADM0 or ADS0 is prioritized. Write to ADCR0 is not performed. Moreover, no interrupt signal (INTAD) is issued at conversion end.

(3) Anti-noise measures

Attention must be paid to noise fed to AV_{REF} and ANI0 to ANI7 to preserve the 8-bit resolution. The influence of noise grows proportionally to the output impedance of the analog input source. Therefore, it is recommended to connect C externally, as shown in Figure 17-8.

Figure 17-8. Connection of Analog Input Pin



If there is the possibility that noise equal to or greater than AV_{REF}, or equal to or smaller than AV_{SS}, may be input, clamp with a diode having a small V_F (0.3 V or less).

(4) ANI0/P70 to ANI7/P77

The analog input pins (ANI0 to ANI7) can also be used as an input port pin (P70 to P77).

Do not execute the input instruction that corresponds with port 7 during conversion if any pin from among ANI0 to ANI7 has been selected and A/D conversion run. This may degrade the resolution.

Moreover, if a digital pulse is applied to pins adjacent to the pin for which A/D conversion is being performed, the A/D conversion value will not be obtained as expected because of coupling noise. Therefore, do not apply a pulse to pins adjacent to the pin for which A/D conversion is being performed.

(5) Input impedance of AV_{REF} pin

Connects a series resistor string across the AV_{REF} pin and AV_{SS} pin.

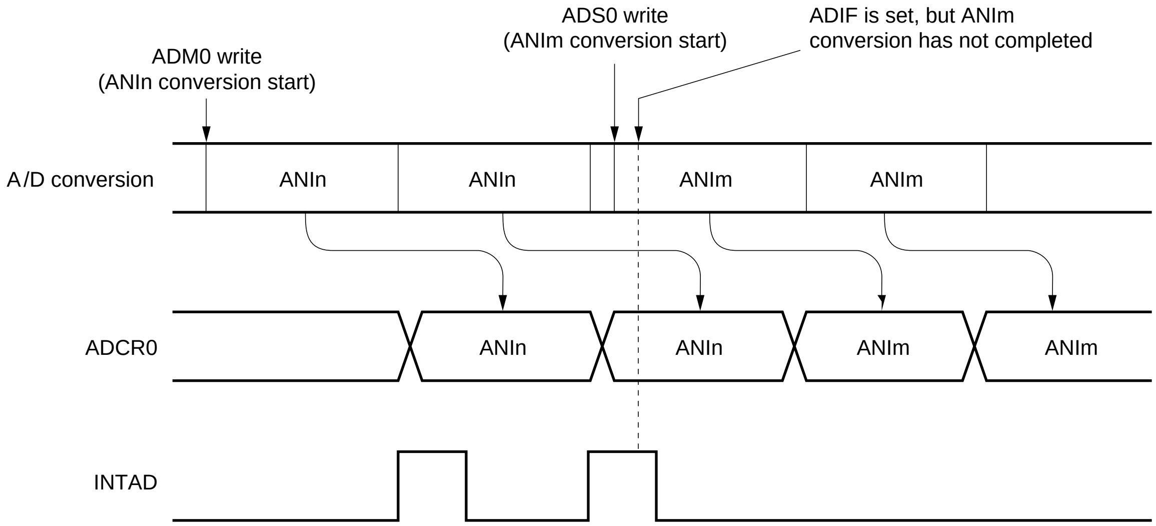
Therefore, if the output impedance of the reference voltage source is high, connecting in parallel a series resistor string between the AV_{REF} and AV_{SS} pins will result in a large reference voltage error.

(6) Interrupt request flag (ADIF)

The interrupt request flag (ADIF) is not cleared even if the analog input channel register 0 (ADS0) is changed. Owing to this, there will be cases when the A/D conversion result and ADIF that correspond with the pre-changed analog input immediately prior to ADS0 overwriting will be set if the analog input pin is changed during A/D conversion. It must therefore be noted that ADIF will be set regardless of whether the A/D conversion for the changed analog input has finished or not when ADIF is read immediately after ADS0 has been overwritten. These facts should be kept in mind.

Moreover, if A/D conversion is stopped once and then resumed, clear ADIF before resuming conversion.

Figure 17-9. A/D Conversion End Interrupt Request Generation Timing



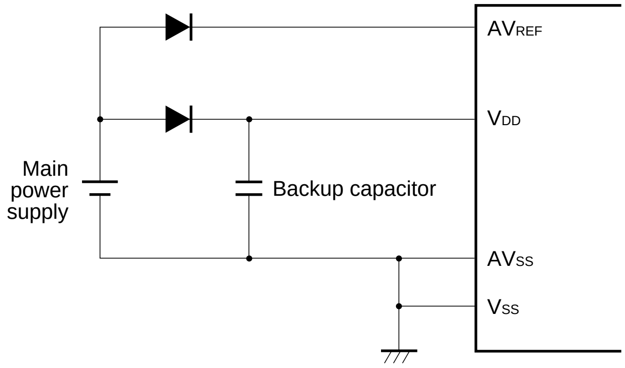
Remark n = 0, 1, ···, 7
m = 0, 1, ···, 7

(7) AV_{REF} pin

The AV_{REF} pin is the analog circuit's power pin and A/D converter's reference voltage input pin and also supplies power to the ANI0/P70 to ANI7/P77 input circuits.

Therefore, be sure to apply the same potential level as V_{DD} as shown in Figure 17-10, even for applications in which a backup power supply can be switched.

Figure 17-10. Connection of AV_{REF} Pin



[MEMO]

CHAPTER 18 SERIAL INTERFACE OVERVIEW

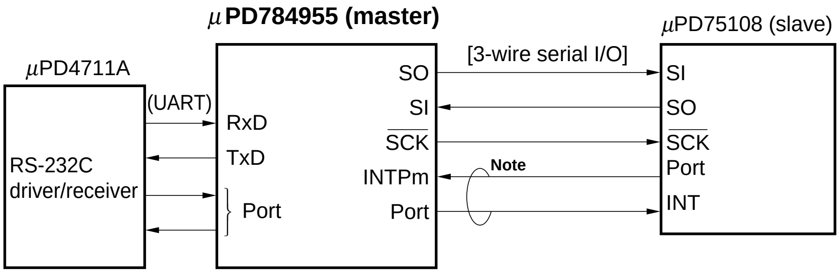
The μPD784955 Subseries has a serial interface with two independent channels. Therefore, communication outside and within the system can be simultaneous on the two channels.

- Asynchronous serial interface (UART) × 1 channel
  → See **CHAPTER 19**.

- Clocked serial interface (CSI) × 1 channel
  - 3-wire serial I/O mode (MSB or LSB first)
    → See **CHAPTER 20**.

Figure 18-1. Serial Interface Example

UART + 3-wire serial I/O



Note Handshake lines

[MEMO]

CHAPTER 19 ASYNCHRONOUS SERIAL INTERFACE

The μPD784955 Series has one channel of serial interface in asynchronous serial interface mode.

19.1 Asynchronous Serial Interface Mode

The asynchronous serial interface (UART: Universal Asynchronous Receiver Transmitter) offers the following two modes.

(1) Operation stop mode

This mode is used when serial transfer is not performed to reduce the power consumption.

(2) Asynchronous serial interface (UART) mode

This mode is used to send and receive 1-byte data that follows the start bit, and supports full-duplex transmission. A UART-dedicated baud rate generator is provided on-chip, enabling transmission at any baud rate within a broad range. The MIDI standard's baud rate (31.25 kbps) can be used by utilizing the UART-dedicated baud rate generator.

19.1.1 Configuration

The asynchronous serial interface has the following hardware configuration.

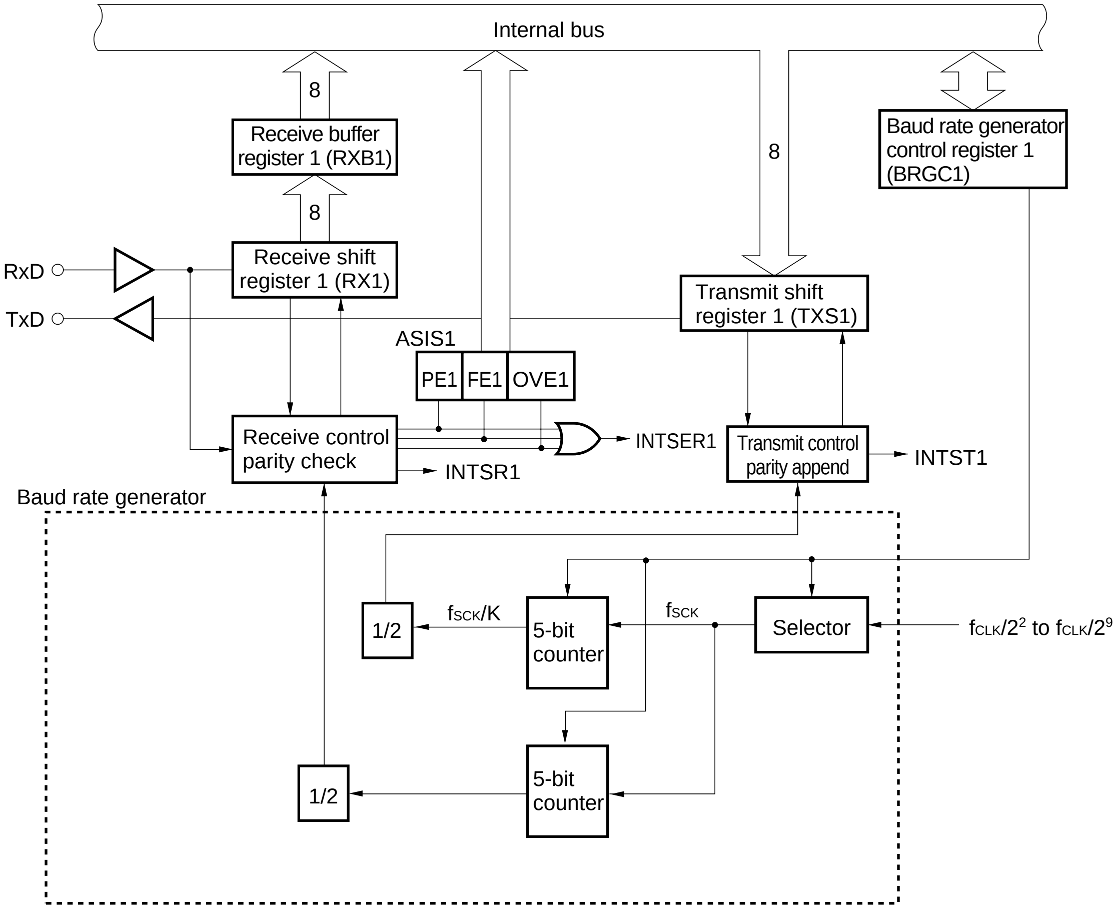
Figure 19-1 gives the block diagram of the asynchronous serial interface.

Table 19-1. Configuration of Asynchronous Serial Interface

Item	Configuration
Registers	Transmit shift register (TXS1) Receive shift register (RX1) Receive buffer register (RXB1)
Control registers	Asynchronous serial interface mode register 1 (ASIM1) Asynchronous serial interface status register 1 (ASIS1) Baud rate generator control register 1 (BRGC1)

★

Figure 19-1. Block Diagram in Asynchronous Serial Interface Mode


Internal bus
8
Receive buffer register 1 (RXB1)
8
Receive shift register 1 (RX1)
8
Baud rate generator control register 1 (BRGC1)
RxD
TxD
Transmit shift register 1 (TXS1)
ASIS1
PE1
FE1
OVE1
Receive control parity check
INTSER1
Transmit control parity append
INTST1
INTSR1
Baud rate generator
fSCK/K
fSCK
1/2
5-bit counter
Selector
fCLK/2^2 to fCLK/2^9
5-bit counter
1/2

(1) Transmit shift register 1 (TXS1)

This register is used to set transmit data. Data written to TXS1 is sent as serial data.

If a data length of 7 bits is specified, bits 0 to 6 of the data written to TXS1 are transferred as transmit data. Transmission is started by writing data to TXS1.

TX1 can be written with an 8-bit memory manipulation instruction, but cannot be read.

$\overline{\text{RESET}}$ input sets TXS1 to FFH.

Caution Do not write to TXS1 during transmission.
TXS1 and receive buffer register 1 (RXB1) are allocated to the same address. Therefore, attempting to read TXS1 will result in reading the values of RXB1.

(2) Receive shift register 1 (RX1)

This register is used to convert serial data input to the RxD1 pin to parallel data. Receive data is transferred to the receive buffer register (RXB1) when 1-byte data is received.

RX1 cannot be directly manipulated by program.

(3) Receive buffer register 1 (RXB1)

This register is used to hold receive data. Each time one byte of data is received, new receive data is transferred from the receive shift register 1 (RX1)

If a data length of 7 bits is specified, receive data is transferred to bits 0 to 6 of RXB1, and the MSB of RXB1 always becomes 0.

RXB1 can be read by an 8-bit memory manipulation instruction, but cannot be written.

$\overline{\text{RESET}}$ input sets RXB1 to FFH.

Caution RXB1 and transmit shift register 1 (TXB1) are allocated to the same address. Therefore, attempting to read RXB1 will result in reading the values of TXB1.

(4) Transmission control circuit

This circuit controls transmit operations such as the addition of a start bit, parity bit, and stop bit(s) to data written to transmit shift register 1 (TXS1), according to the contents set to the asynchronous serial interface mode register 1 (ASIM1).

(5) Reception control circuit

This circuit controls reception according to the contents set to the asynchronous serial interface mode register 1 (ASIM1). It also performs error check for parity errors, etc., during reception and transmission. If it detects an error, it sets a value corresponding to the nature of the error in the asynchronous serial interface status register 1 (ASIS1).

19.1.2 Control registers

The asynchronous serial interface controls the following three types of registers.

- Asynchronous serial interface mode register 1 (ASIM1)
- Asynchronous serial interface status register 1 (ASIS1)
- Baud rate generator control register 1 (BRGC1)

(1) Asynchronous serial interface mode register 1 (ASIM1)

ASIM1 is an 8-bit register that controls serial transfer operation of the asynchronous serial interface.

ASIM1 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets ASIM1 to 00H.

Figure 19-2. Asynchronous Serial Interface Mode Register 1 (ASIM1) Format

Address: 0FF70H After Reset: 00H R/W

Symbol	⑦	⑥	5	4	3	2	1	0
ASIM1	TXE1	RXE1	PS11	PS10	CL1	SL1	ISRMn	0 Note

TXE1	RXE1	Operation Mode	RxD/P20 Pin Function	TxD/P21 Pin Function
0	0	Operation stop	Port function	Port function
0	1	UART mode (Receive only)	Serial function	Port function
1	0	UART mode (Transmit only)	Port function	Serial function
1	1	UART mode (Transmit/Receive)	Serial function	Serial function

PS11	PS10	Parity Bit Specification
0	0	No parity
0	1	Always add 0 parity during transmission Do not perform parity check during reception (parity error not generated)
1	0	Odd parity
1	1	Even parity

CL1	Character Length Specification
0	7 bits
1	8 bits

SL1	Transmit Data Stop Bit Length Specification
0	1 bit
1	2 bits

ISRM1	Receive Completion Interrupt Control at Error Occurrence
0	Generate receive completion interrupt request when error occurs
1	Do not generate receive completion interrupt request when error occurs

Note Ensure that “0” is written in bit 0.

Caution Switch across to the operational mode after stopping serial sending and receiving operations.

(2) Asynchronous serial interface status register 1 (ASIS1)

ASIS1 is a register used to display the type of error when a receive error occurs.

ASIS1 can be read with a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets ASIS1 to 00H.

Figure 19-3. Asynchronous Serial Interface Status Register 1 (ASIS1) Format

Address: 0FF72H After Reset: 00H R

Symbol	7	6	5	4	3	②	①	⓪
ASIS1	0	0	0	0	0	PE1	FE1	OVE1

PE1	Parity Error Flag
0	Parity error not generated
1	Parity error generated (when parity of transmit data does not match)

FE1	Framing Error Flag
0	Framing error not generated
1	Framing error generated[Note 1] (when stop bit(s) is not detected)

OVE1	Overrun Error Flag
0	Overrun error not generated
1	Overrun error generated[Note 2] (When next receive operation is completed before data from receive buffer register is read)

Notes 1. Even if the stop bit length has been set to 2 bits with bit 2 (SL1) of the asynchronous serial interface mode register 1 (ASIM1), stop bit detection during reception is only 1 bit.

2. Be sure to read the receive buffer register 1 (RXB1) when an overrun error occurs.
An overrun error is generated each time data is received until RXB1 is read.

(3) Baud rate generator control register 1 (BRGC1)

BRGC1 is a register used to set the serial clock of the asynchronous serial interface.

BRGC1 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets BRGC1 to 00H.

Figure 19-4. Baud Rate Generator Control Register 1 (BRGC1) Format

Address: 0FF76H After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
BRGC1	0	TPS12	TPS11	TPS10	MDL13	MDL12	MDL11	MDL10

TPS12	TPS11	TPS10	5-Bit Counter Source Clock Selection
0	0	0	f_{CLK}/4 (2 MHz)
0	0	1	f_{CLK}/8 (1 MHz)
0	1	0	f_{CLK}/16 (500 kHz)
0	1	1	f_{CLK}/32 (250 kHz)
1	0	0	f_{CLK}/64 (125 kHz)
1	0	1	f_{CLK}/128 (62.5 kHz)
1	1	0	f_{CLK}/256 (31.3 kHz)
1	1	1	f_{CLK}/512 (15.6 kHz)

MDL13	MDL12	MDL11	MDL10	Baud Rate Generator Input Clock Selection	k
0	0	0	0	f_{SCK}/16	0
0	0	0	1	f_{SCK}/17	1
0	0	1	0	f_{SCK}/18	2
0	0	1	1	f_{SCK}/19	3
0	1	0	0	f_{SCK}/20	4
0	1	0	1	f_{SCK}/21	5
0	1	1	0	f_{SCK}/22	6
0	1	1	1	f_{SCK}/23	7
1	0	0	0	f_{SCK}/24	8
1	0	0	1	f_{SCK}/25	9
1	0	1	0	f_{SCK}/26	10
1	0	1	1	f_{SCK}/27	11
1	1	0	0	f_{SCK}/28	12
1	1	0	1	f_{SCK}/29	13
1	1	1	0	f_{SCK}/30	14
1	1	1	1	Setting prohibited	–

Caution If a write operation to BRGC1 is performed during communication, the baud rate generator output will become garbled and normal communication will not be achieved. Consequently, do not write in BRGC1 during communications.

Remarks 1. Figures in parentheses apply to operation with f_{CLK} = 8 MHz.

2. f_{SCK} : Source clock of 5-bit counter

3. k : Value set in MDL10 to MDL13 ($0 \le k \le 14$)

19.2 Operation

The asynchronous serial interface has the following two types of operation modes.

- Operation stop mode
- Asynchronous serial interface (UART) mode

19.2.1 Operation stop mode

Serial transfer cannot be performed in the operation stop mode, resulting in reduced power consumption. Moreover, in the operation stop mode, pins can be used as regular ports.

(1) Register setting

Setting of the operation stop mode is done with asynchronous serial interface mode register 1 (ASIM1). ASIM1 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets ASIM1 to 00H.

Address: 0FF70H After Reset: 00H R/W

Symbol	⑦	⑥	5	4	3	2	1	0
ASIM1	TXE1	RXE1	PS11	PS10	CL1	SL1	ISRM1	0 [Note]

TXE1	RXE1	Operation Mode	RxD/P20 Pin Function	TxD/P21 Pin Function
0	0	Operation stop	Port function	Port function
0	1	UART mode (Receive only)	Serial function	Port function
1	0	UART mode (Transmit only)	Port function	Serial function
1	1	UART mode (Transmit/Receive)	Serial function	Serial function

Note Ensure that “0” is written in bit 0.

Caution Switch across to the operational mode after stopping serial sending and receiving operations.

19.2.2 Asynchronous serial interface (UART) mode

This mode is used to transmit and receive the 1-byte data following the start bit. It supports full-duplex operation.

A UART-dedicated baud rate generator is incorporated enabling communication using any baud rate within a large range.

The MIDI standard's baud rate (31.25 kbps) can be used utilizing the UART-dedicated baud rate generator.

(1) Register setting

The UART mode is set with asynchronous serial interface mode register 1 (ASIM1), asynchronous serial interface status register 1 (ASIS1), and baud rate generator control register 1 (BRGC1).

(a) Asynchronous serial interface mode register 1 (ASIM1)

ASIM1 can be set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets ASIM1 to 00H.

Address: 0FF70H After Reset: 00H R/W

Symbol	⑦	⑥	5	4	3	2	1	0
ASIM1	TXE1	RXE1	PS11	PS10	CL1	SL1	ISRM1	0 Note

TXE1	RXE1	Operation Mode	RxD/P20 Pin Function	TxD/P21 Pin Function
0	0	Operation stop	Port function	Port function
0	1	UART mode (Receive only)	Serial function	Port function
1	0	UART mode (Transmit only)	Port function	Serial function
1	1	UART mode (Transmit/Receive)	Serial function	Serial function

PS11	PS10	Parity Bit Specification
0	0	No parity
0	1	Always add 0 parity during transmission Do not perform parity check during reception (parity error not generated)
1	0	Odd parity
1	1	Even parity

CL1	Character Length Specification
0	7 bits
1	8 bits

SL1	Transmit Data Stop Bit Length Specification
0	1 bit
1	2 bits

ISRM1	Receive Completion Interrupt Control at Error Occurrence
0	Generate receive completion interrupt when error occurs
1	Do not generate receive completion interrupt when error occurs

Note Ensure that “0” is written in bit 0.

Caution Switch the operation mode after stopping serial sending and receiving operations.

(b) Asynchronous serial interface status register 1 (ASIS1)

ASIS1 can be read by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets ASIS1 to 00H.

Address: 0FF72H After Reset: 00H R

Symbol	7	6	5	4	3	②	①	⓪
ASIS1	0	0	0	0	0	PE1	FE1	OVE1

PE1	Parity Error Flag
0	Parity error not generated
1	Parity error generated (when parity of transmit data does not match)

FE1	Framing Error Flag
0	Framing error not generated
1	Framing error generated[Note 1] (when stop bit(s) is not detected)

OVE1	Overrun Error Flag
0	Overrun error not generated
1	Overrun error generated[Note 2] (When next receive operation is completed before data from receive buffer register is read)

Notes 1. Even if the stop bit length has been set to 2 bits with bit 2 (SLn) of the asynchronous serial interface mode register 1 (ASIM1), stop bit detection during reception is only for 1 bit.

2. Be sure to read the receive buffer register 1 (RXB1) when an overrun error occurs.
An overrun error is generated each time data is received until RXB1 is read.

(c) Baud rate generator control register 1 (BRGC1)

BRGC1 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets BRGC1 to 00H.

Address: 0FF76H After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
BRGC1	0	TPS12	TPS11	TPS10	MDL13	MDL12	MDL11	MDL10

TPS12	TPS11	TPS10	5-Bit Counter Source Clock Selection
0	0	0	f_{CLK}/4 (2 MHz)
0	0	1	f_{CLK}/8 (1 MHz)
0	1	0	f_{CLK}/16 (500 kHz)
0	1	1	f_{CLK}/32 (250 kHz)
1	0	0	f_{CLK}/64 (125 kHz)
1	0	1	f_{CLK}/128 (62.5 kHz)
1	1	0	f_{CLK}/256 (31.3 kHz)
1	1	1	f_{CLK}/512 (15.6 kHz)

MDL13	MDL12	MDL11	MDL10	Baud Rate Generator Input Clock Selection	k
0	0	0	0	f_{SCK}/16	0
0	0	0	1	f_{SCK}/17	1
0	0	1	0	f_{SCK}/18	2
0	0	1	1	f_{SCK}/19	3
0	1	0	0	f_{SCK}/20	4
0	1	0	1	f_{SCK}/21	5
0	1	1	0	f_{SCK}/22	6
0	1	1	1	f_{SCK}/23	7
1	0	0	0	f_{SCK}/24	8
1	0	0	1	f_{SCK}/25	9
1	0	1	0	f_{SCK}/26	10
1	0	1	1	f_{SCK}/27	11
1	1	0	0	f_{SCK}/28	12
1	1	0	1	f_{SCK}/29	13
1	1	1	0	f_{SCK}/30	14
1	1	1	1	Setting prohibited	–

Caution If a write operation to BRGC1 is performed during communication, the baud rate generator output will become garbled and normal communication will not be achieved. Consequently, do not write in BRGC1 during communications.

Remarks 1. Figures in parentheses apply to operation with f_{CLK} = 8 MHz.

2. f_{SCK} : Source clock of 5-bit counter

3. k : Value set in MDL10 to MDL13 ($0 \leq k \leq 14$)

The send/receive clock pulse used for generating baud rate is a signal that is a frequency-division of the system clock pulse.

- **The system clock generates a clock pulse that is used for baud rate.**
  The system clock pulses is frequency-divided to generate a send/receive clock pulse. The baud rate generated from the system clock is determined by the equation shown below.

★

$$[\text{Baud rate}] = \frac{f_{CLK}}{2^{m+1}\,(k + 16)}\ [\text{Hz}]$$

f_{CLK} : Internal system clock frequency
m : Value set in TPS10 to TPS12 ($2 \le m \le 9$)
k : Value set in MDL10 to MDL13 ($0 \le k \le 14$)

The relationship between the source clock of the 5-bit counter and the m value is shown in Table 19-2.

★ **Table 19-2. Relationship between 5-Bit Counter Source Clock and m Value**

TPS12	TPS11	TPS10	5-Bit Counter Source Clock Selection	m
0	0	0	f_{CLK}/4 (2 MHz)	2
0	0	1	f_{CLK}/8 (1 MHz)	3
0	1	0	f_{CLK}/16 (500 kHz)	4
0	1	1	f_{CLK}/32 (250 kHz)	5
1	0	0	f_{CLK}/64 (125 kHz)	6
1	0	1	f_{CLK}/128 (62.5 kHz)	7
1	1	0	f_{CLK}/256 (31.3 kHz)	8
1	1	1	f_{CLK}/512 (15.6 kHz)	9

Remark Figures in parentheses apply to operation with f_{CLK} = 8 MHz.

- **Permissible baud rate range**

  The permissible baud rate range is dependent on the number of bits in one frame and the ratio of frequency division of the counter. [1/(16+k)]

  Table 19-3 shows the relationship between system clock and baud rate.

Table 19-3. Relationship between System Clock and Baud Rate

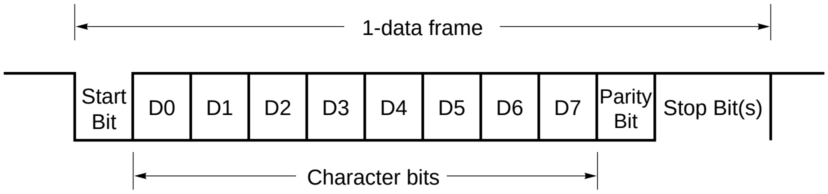
Baud Rate (bps)	f_{CLK} = 12.5 MHz		f_{CLK} = 8.0 MHz		f_{CLK} = 6.0 MHz	
	BRGC1 value	Error (%)	BRGC1 value	Error (%)	BRGC1 value	Error (%)
195	—	—	7EH	33.55	7EH	0.16
300	7EH	35.63	7AH	0.16	74H	–2.34
600	74H	1.73	6AH	0.16	64H	–2.34
976	69H	1.00	60H	0.06	58H	0.06
1200	64H	1.73	5AH	0.16	54H	–2.34
2400	54H	1.73	4AH	0.16	44H	–2.34
4800	44H	1.73	3AH	0.16	34H	–2.34
9615	34H	1.56	2AH	0.00	24H	–2.50
19200	24H	1.73	1AH	0.16	14H	–2.34
31250	19H	0.00	10H	0.00	08H	0.00
38400	14H	1.73	0AH	0.16	04H	–2.34
76800	04H	1.73	00H	–18.62	00H	–38.96
97656	00H	1.00	—	—	—	—

(2) Communication operation

(a) Data format

The format for sending and receiving data is shown in Figure 19-5.

Figure 19-5. Asynchronous Serial Interface Transmit/Receive Data Format



Each data frame is composed for the bits outlined below.

- Start bit......................... 1 bit
- Character bits .... 7 bits/8 bits
- Parity bit .... Even parity/Odd parity/0 parity/No parity
- Stop bit(s) .... 1 bit/2 bits

Specification of the character bit length inside one data frame, selection of the parity, and selection of the stop bit length, are performed with the asynchronous serial interface mode register 1 (ASIM1).
If 7 bits has been selected as the number of character bits, only the low-order 7 bits (bits 0 to 6) are valid. In the case of transmission, the highest-order bit (bit 7) is ignored. In the case of reception, the highest-order bit (bit 7) always becomes “0”.
The setting of the serial transfer rate is performed with the ASIM1 and the baud rate generator control register 1 (BRGC1).
If a serial data reception error occurs, it is possible to determine the contents of the reception error by reading the status of the asynchronous serial interface status register 1 (ASIS1).

(b) Parity types and operations

Parity bits serve to detect bit errors in transmit data. Normally, the parity bits used on the transmit side and the receive side are of the same type. In the case of even parity and odd parity, it is possible to detect “1” bit (odd number) errors. In the case of 0 parity and no parity, errors cannot be detected.

(i) Even parity

- **During transmission**

  Makes the number of “1”s in transmit data that includes the parity bit even. The value of the parity bit changes as follows.

  If the number of “1” bits in transmit data is odd : 1
  If the number of “1” bits in transmit data is even : 0

- **During reception**

  The number of “1” bits in receive data that includes the parity bit is counted, and if it is odd, a parity error occurs.

(ii) Odd parity

- **During transmission**

  Odd parity is the reverse of even parity. It makes the number of “1”s in transmit data that includes the parity bit even. The value of the parity bit changes as follows.

  If the number of “1” bits in transmit data is odd : 0
  If the number of “1” bits in transmit data is even : 1

- **During reception**

  The number of “1” bits in receive data is counted, and if it is even, a parity error occurs.

(iii) 0 Parity

During transmission, makes the parity bit “0”, regardless of the transmit data.

Parity bit check is not performed during reception. Therefore, no parity error occurs, regardless of whether the parity bit value is “0” or “1”.

(iv) No parity

No parity is appended to transmit data.

Transmit data is received assuming that it has no parity bit. No parity error can occur because there is no parity bit.

(c) Transmission

Transmission is started by writing transmit data to the transmit shift register 1 (TXS1). The start bit, parity bit, and stop bit(s) are automatically added.

The contents of the transmit shift register 1 (TXS1) are shifted out upon transmission start, and when the transmit shift register 1 (TXS1) becomes empty, a transmit completion interrupt request (INTST1) is generated.

★ **Caution In the case of UART transmission, follow the procedure below when performing transmission for the first time.**

<1> Set the port to the input mode (PM21 = 1), and write 0 to the port latch.

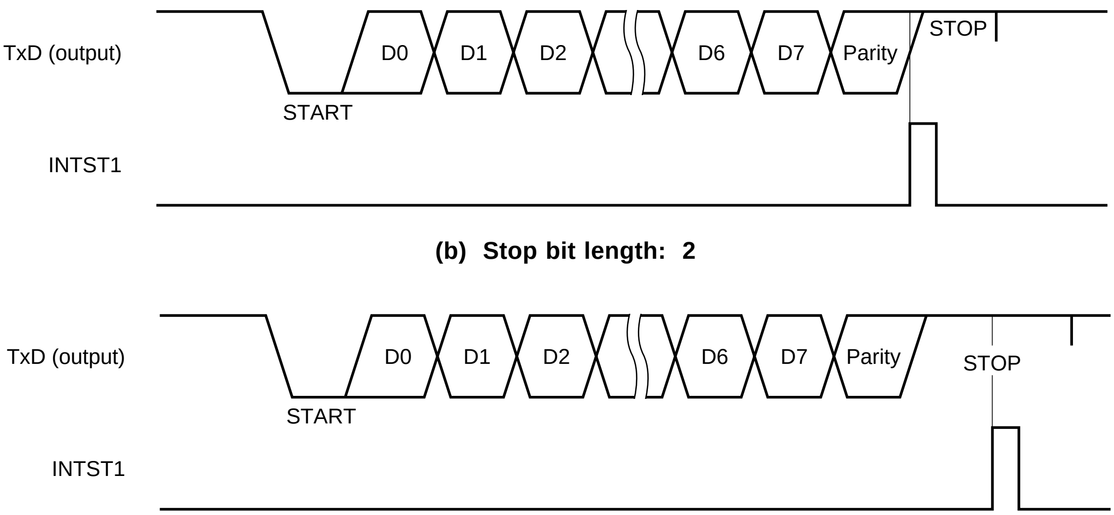
<2> Write 1 to bit 7 (TXE1) of asynchronous serial interface mode register 1 (ASIM1) to enable transmission.

<3> Set the port to the output mode (PM21 = 0).

<4> Write transmit data to TXS to start transmission.

If the port is set to the output mode first, 0 will be output from the pins, which may cause malfunction.

Figure 19-6. Asynchronous Serial Interface Transmit Completion Interrupt Request Timing



(a) Stop bit length: 1

(b) Stop bit length: 2

Caution Do not write to the asynchronous serial interface mode register 1 (ASIM1) during transmission. If you write to the ASIM1 register during transmission, further transmission operations may become impossible (in this case, input $\overline{\text{RESET}}$ to return to normal). Whether transmission is in progress or not can be judged by software, using the transmit completion interrupt (INTST1) or the interrupt request flag (STIF1) set by INTST1.

(d) Reception

When the RXE bit of the asynchronous serial interface mode register 1 (ASIM1) is set to 1, reception is enabled and sampling of the RxD pin input is performed.

Sampling of the RxD pin input is performed by the serial clock set in the baud rate generator control register 1 (BRGC1).

The 5-bit counter for the baud rate generator will begin counting when the RxD pin input reaches low level, and the 'start timing' signal for data sampling will be output when half of the time set for the baud rate has passed. If the result of re-sampling the RxD pin input with this start timing signal is low level, the RxDn pin input is perceived as the start bit, the 5-bit counter is initialized and begins counting, and data sampling is performed. Following the start bit, when the character data, parity bit, and one stop bit are detected, reception of one frame of data is completed.

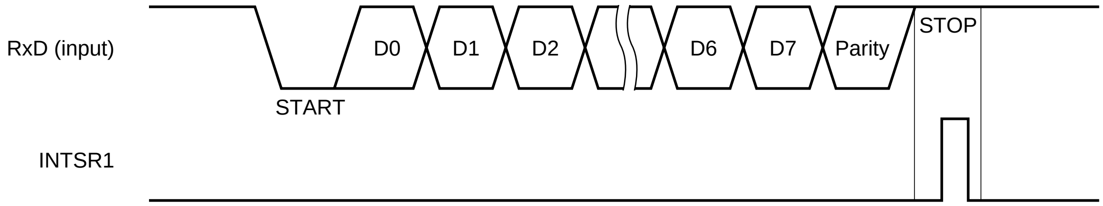
When reception of one frame of data is completed, the receive data in the shift register is transferred to the receive buffer 1 register (RXB1),and a receive completion interrupt request (INTSR1) is generated.

Also, even if an error occurs, the receiving data for which the error occurred is transferred to RXB1. If an error occurs, when bit 1 (ISRM1) of ASIM1 is cleared (0), INTSR1 is generated. (refer to **Figure 19-7**).

When bit ISRM1 is set (1), INTSR1 is not generated.

When bit RXE1 is reset to 0 during a receive operation, the receive operation is immediately stopped. At this time, the contents of RXB1 and ASIS1 remain unchanged, and INTSR1 and INTSER1 are not generated.

Figure 19-7. Asynchronous Serial Interface Receive Completion Interrupt Request Timing



Caution Even when a receive error occurs, be sure to read the receive buffer register 1 (RXB1). If RXB1 is not read, an overrun error will occur during reception of the next data, and the reception error status will continue indefinitely.

(e) Receive error

Errors that occur during reception are of three types: parity errors, framing errors, and overrun errors. As the data reception result error flag is set inside the asynchronous serial interface status register 1 (ASIS1), the receive error interrupt request (INTSER1) is generated. A receive error interrupt is generated before a receive completion interrupt request (INTSR1). Receive error causes are shown in Table 19-4.

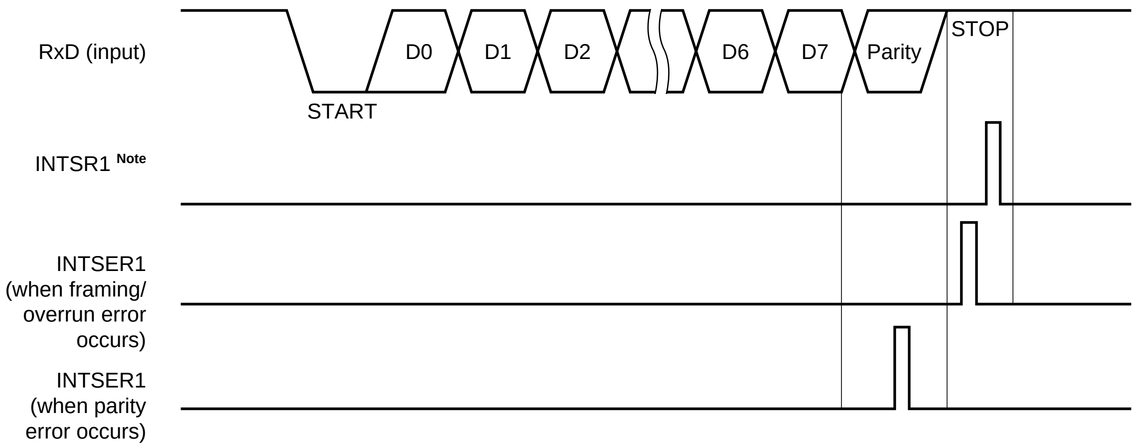
What type of error has occurred during reception can be detected by reading the contents of the asynchronous serial interface status register 1 (ASIS1) during processing of the receive error interrupt (INTSER1) (refer to **Table 19-4** and **Figure 19-8**).

The contents of ASIS1 are reset to 0 either when the receive buffer register 1 (RXB1) is read or when the next data is received (If the next data has an error, this error flag is set).

Table 19-4. Receive Error Causes

Receive Error	Cause	ASIS1
Parity error	Parity specified for transmission and parity of receive data don't match	04H
Framing error	Stop bit was not detected	02H
Overrun error	Next data reception was completed before data was read from the receive buffer register	01H

Figure 19-8. Receive Error Timing


RxD (input)
D0
D1
D2
D6
D7
Parity
STOP
START
INTSR1 Note
INTSER1 (when framing/ overrun error occurs)
INTSER1 (when parity error occurs)

Note INTSR1 will not be triggered if an error occurs when the ISRM1 bit has been set (1).

Cautions 1. The contents of the ASIS1 register are reset to 0 either when the receive buffer register 1 (RXB1) is read or when the next data is received. To find out the contents of the error, be sure to read ASIS1 before reading RXB1.

2. Be sure to read the receive buffer register 1 (RXB1) even when a receive error occurs. If RXB1 is not read, an overrun error will occur at reception of the next data, and the receive error status will continue indefinitely.

19.2.3 Standby mode operation

(1) HALT mode operation

Serial transfer operation is normally performed.

★ **(2) STOP mode or IDLE mode operation**

The asynchronous serial interface mode register 1 (ASIM1), transmit shift register 1 (TXS1), receive shift register 1 (RX1), and receive buffer register 1 (RXB1) stop operation holding the value immediately before the clock stops. If the clock stops (STOP mode) during transmission, the TxD pin output data immediately before the clock stopped is held. If the clock stops during reception, receive data up to immediately before the clock stopped is stored, and subsequent operation is stopped. When the clock is restarted, reception is resumed.

CHAPTER 20 3-WIRE SERIAL I/O MODE

20.1 Function

This mode transfers 8-bit data by using the three lines of the serial clock ($\overline{\text{SCK}}$), the serial output (SO), and the serial input (SI).

Since the 3-wire serial I/O mode can perform simultaneous transmission and reception, the data transfer processing time becomes shorter.

The first bit in the serially transferred 8-bit data can be selected from either MSB or LSB.

The 3-wire serial I/O mode is valid when the peripheral I/O or display controller equipped with a clocked serial interface is connected.

20.2 Configuration

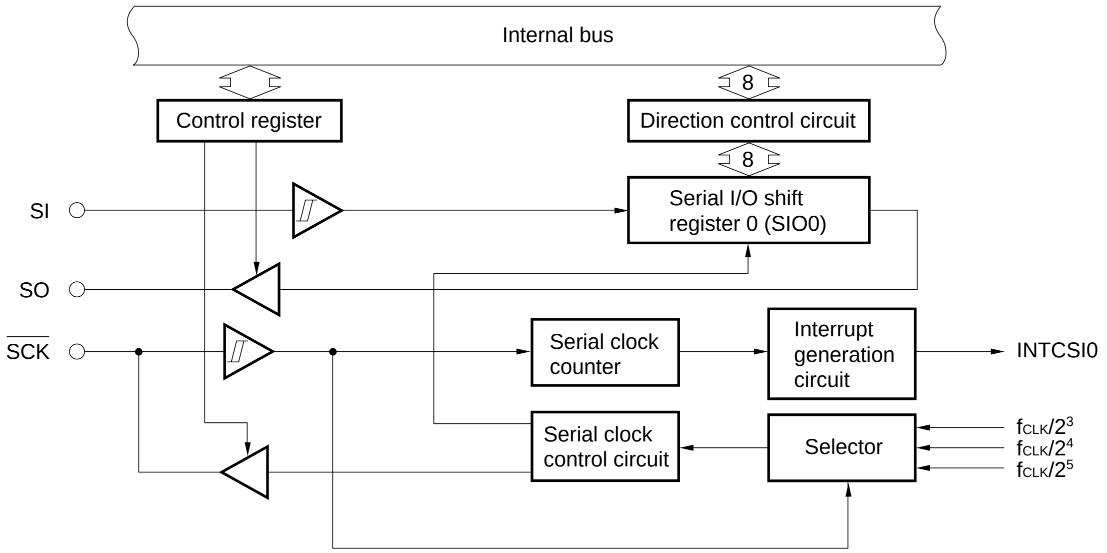
The 3-wire serial I/O mode is installed in the following hardware.

Figure 20-1 is a block diagram of the clocked serial interface (CSI) in the 3-wire serial I/O mode.

Table 20-1. 3-Wire Serial I/O Configuration

Item	Configuration
Register	Serial I/O shift register 0 (SIO0)
Control register	Serial operating mode register 0 (CSIM0)

★ **Figure 20-1. Block Diagram of Clocked Serial Interface (3-Wire Serial I/O Mode)**


Internal bus
8
Control register
Direction control circuit
8
SI
Serial I/O shift register 0 (SIO0)
SO
SCK
Serial clock counter
Interrupt generation circuit
INTCSI0
Serial clock control circuit
Selector
fCLK/2^3
fCLK/2^4
fCLK/2^5

- **Serial I/O shift register 0 (SIO0)**

This 8-bit shift register performs parallel to serial conversion and serial communication (shift operation) synchronized with the serial clock.

SIO0 is set by an 8-bit memory manipulation instruction.

When bit 7 (CSIE0) in serial operating mode register 0 (CSIM0) is one, serial operation starts by writing data to or reading it from SIO0.

When transmitting, the data written to SIO0 is output to the serial output (SO).

When receiving, data is read from the serial input (SI) to SIO0.

$\overline{\text{RESET}}$ input sets SIO0 to 00H.

Caution Do not access SIO0 during a transfer except for an access that becomes a transfer start trigger. (When MODE0 = 0, reading is disabled; and when MODE0 = 1, writing is disabled.)

20.3 Control Registers

- **Serial operating mode register 0 (CSIM0)**

  The CSIM0 register sets the serial clock and operating mode to the 3-wire serial I/O mode, and enables or stops operation.

  CSIM0 is set by a 1-bit or 8-bit memory manipulation instruction.

  $\overline{\text{RESET}}$ input sets CSIM0 to 00H.

Figure 20-2. Serial Operating Mode Register 0 (CSIM0) Format

Address: 0FF90H After Reset: 00H R/W

Symbol	⑦	6	5	4	3	2	1	0
CSIM0	CSIE0	0	0	0	MODE1	MODE0	SCL01	SCL00

CSIE0	SIO0 Operation Enable/Disable Setting		
	Shift Register Operation	Serial Counter	Port
0	Disable operation	Clear	Port function[Note]
1	Enable operation	Enable operation count	Serial function + Port function

MODE1	Specification of First Transfer Data Bit
0	MSB
1	LSB

MODE0	Transfer Operation Mode Flag		
	Operation Mode	Transfer Start Trigger	SO Output
0	Transmit/receive communication mode	SIO0 write	Normal output
1	Receive only mode	SIO0 read	“0” fixed

SCL01	SCL00	Clock Selection
0	0	External clock to $\overline{\text{SCK}}$
0	1	$f_{CLK}/8$
1	0	$f_{CLK}/16$
1	1	$f_{CLK}/32$

Note If CSIE0 = 0 (SIO0 operation stopped state), the pins connected to SI, SO, or $\overline{\text{SCK}}$ can function as ports.

Remark f_{CLK}: Internal system clock

20.4 Operation

3-wire serial I/O has the following two operating modes.

- Operation stopped mode
- 3-wire serial I/O mode

(1) Operation stopped mode

Since serial transfers are not performed in the operation stopped mode, power consumption can be decreased. In the operation stopped mode, the pin can be used as an ordinary I/O port.

(a) Register settings

The operation stopped mode is set in serial operating mode register 0 (CSIM0).
CSIM0 is set by a 1-bit or 8-bit memory manipulation instruction.
$\overline{\text{RESET}}$ input sets CSIM0 to 00H.

Figure 20-3. Serial Operating Mode Register 0 (CSIM0) Format (Operation Stopped Mode)

Address: 0FF90H After Reset: 00H R/W

Symbol	⑦	6	5	4	3	2	1	0
CSIM0	CSIE0	0	0	0	MODE1	MODE0	SCL01	SCL00

CSIE0	SIO0 Operating Enable/Disable Setting		
	Shift Register Operation	Serial Counter	Port
0	Disable operation	Clear	Port function[Note]
1	Enable operation	Enable count operation	Serial function + port function

Note If CSIE0 = 0 (SIO0 operation stopped state), the pins connected to SI, SO, or $\overline{\text{SCK}}$ can function as ports.

(2) 3-wire serial I/O mode

The 3-wire serial I/O mode is valid when connecting a peripheral I/O or a display controller equipped with the clocked serial interface.

Communication is over three lines, the serial clock ($\overline{SCK}$), serial output (SO), and serial input (SI).

(a) Register setting

The 3-wire serial I/O mode is set in serial operating mode register 0 (CSIM0).

CSIM0 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{RESET}$ input sets CSIM0 to 00H.

Figure 20-4. Serial Operating Mode Register 0 (CSIM0) Format (3-Wire Serial I/O Mode)

Address: 0FF90H After Reset: 00H R/W

Symbol	⑦	6	5	4	3	2	1	0
CSIM0	CSIE0	0	0	0	MODE1	MODE0	SCL01	SCL00

CSIE0	SIO0 Operation Enable/Disable Setting		
	Shift Register Operation	Serial Counter	Port
0	Disable operation	Clear	Port function[Note]
1	Enable operation	Enable count operation	Serial function + port function

MODE1	Specification of First Transfer Data Bit
0	MSB
1	LSB

MODE0	Transfer Operation Mode Flag		
	Operation Mode	Transfer Start Trigger	SO Output
0	Transmit/receive communication mode	SIO0 write	Normal output
1	Receive only mode	SIO0 read	“0” fixed

SCL01	SCL00	Clock Selection
0	0	External clock to $\overline{SCK}$
0	1	$f_{CLK}/8$
1	0	$f_{CLK}/16$
1	1	$f_{CLK}/32$

Note If CSIE0 = 0 (SIO0 operation stopped state), the pins connected to SI, SO, or $\overline{SCK}$ can function as ports.

Remark f_{CLK}: Internal system clock

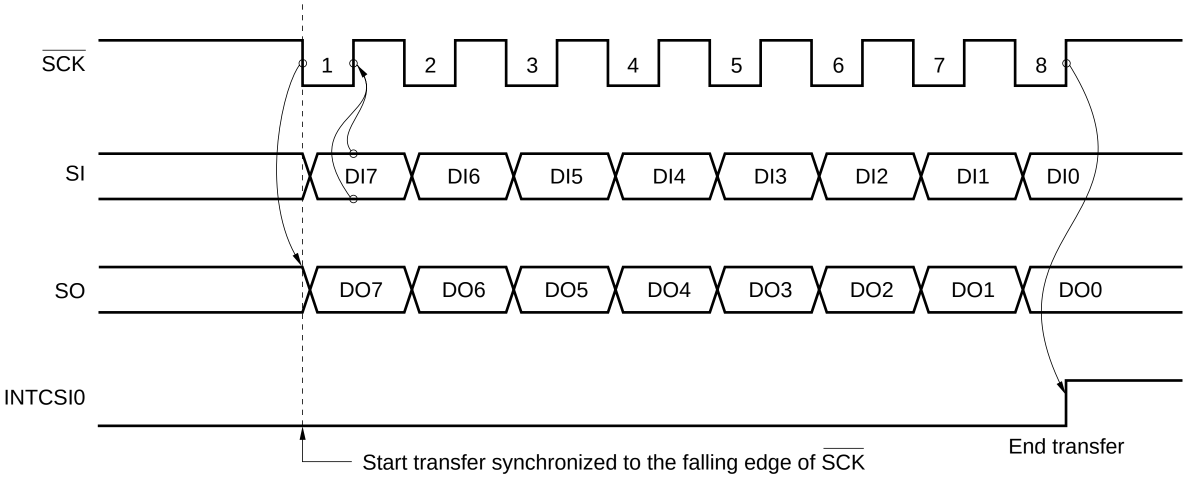
(b) Communication

The 3-wire serial I/O mode transmits and receives data in 8-bit units. Data is transmitted and received with each bit synchronized to the serial clock.

The shifting of the serial I/O shift register 0 (SIO0) is synchronized to the falling edge of the serial clock ($\overline{\text{SCK}}$). The transmitted data are held in the SO latch and output from the SO pin. At the rising edge of $\overline{\text{SCK}}$, the received data that was input at the SI pin is latched to SIO0.

The end of the 8-bit transfer automatically stops SIO0 operation and sets the interrupt request flag (INTCSI0).

Figure 20-5. 3-Wire Serial I/O Mode Timing



(c) Start transfer

If the following two conditions are satisfied, the serial transfer starts when the transfer data is set in the serial I/O shift register 0 (SIO0).

- Control bit (CSIE0) = 1 during SIO0 operation
- After an 8-bit serial transfer, the internal serial clock enters the stopped state or $\overline{\text{SCK}}$ is high.
- Transmit and transmit/receive communication mode
  When CSIE0 = 1 and MODE0 = 0, the transfer starts with an SIO0 write.
- Receive only mode
  When CSIE0 = 1 and MODE0 = 1, the transfer starts with an SIO0 read.

Caution Even if CSIE0 becomes “1” after the data is written to SIO0, transfer does not start.

Serial transfer is automatically stopped by the end of the 8-bit transfer, and the interrupt request flag (INTCSI0) is set.

CHAPTER 21 EDGE DETECTION FUNCTION

★ The P00 pin has an edge detection function that can be programmed to detect the rising edge or falling edge. The edge detection function is always functioning, even in the STOP mode and IDLE mode.

21.1 Control Registers

- **External Interrupt Rising Edge Enable Register (EGP0), External Interrupt Falling Edge Enable Register (EGN0)**
  The EGP0 and EGN0 registers specify the effective edge to be detected by the NMI pin.
  They can read/write with an 8-bit manipulation instruction or a bit manipulation instruction.
  $\overline{\text{RESET}}$ input sets the EGP0 and EGN0 to 00H.

Figure 21-1. Format of External Interrupt Rising Edge Enable Register (EGP0) and External Interrupt Falling Edge Enable Register (EGN0)

Address: 0FFA0H After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
EGP0	0	0	0	0	0	0	0	EGP00

Address: 0FFA2H After Reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
EGN0	0	0	0	0	0	0	0	EGN00

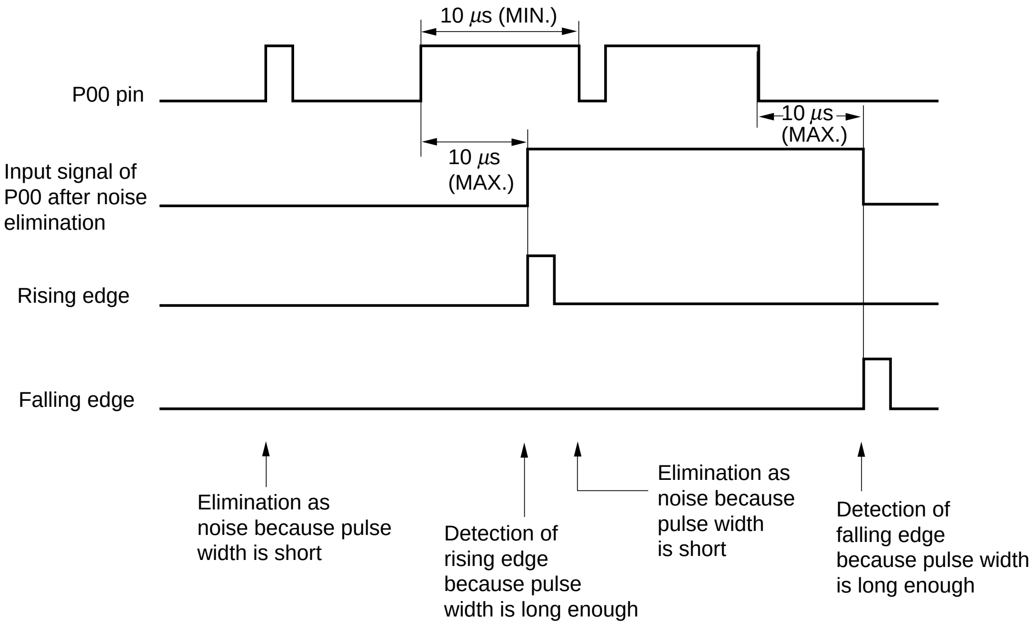
EGP00	EGN00	NMI Pin Effective Edge
0	0	Interrupt disable
0	1	Falling edge
1	0	Rising edge
1	1	Both rising and falling edges

Valid edges of edges detected at pins INTP0 to INTP6 are specified by prescaler mode registers 0, 4, and 5 (PRM0, PRM4, and PRM5) (see **Figure 8-5 Format of Prescaler Mode Register 0 (PRM0)**; **Figure 12-4 Format of Prescaler Mode Register 4 (PRM4)**; and **Figure 13-4 Format of Prescaler Mode Register 5 (PRM5)**).

★ 21.2 Edge Detection of P00 Pin

The P00 pin detects the edges after noise elimination by analog delay. Therefore, edge detection requires that a pulse width be maintained for at least 10 μs.

Figure 21-2. Edge Detection of P00 Pin


10 µs (MIN.)
P00 pin
10 µs (MAX.)
Input signal of P00 after noise elimination
10 µs (MAX.)
Rising edge
Falling edge
Elimination as noise because pulse width is short
Detection of rising edge because pulse width is long enough
Elimination as noise because pulse width is short
Detection of falling edge because pulse width is long enough

Caution **Since noise at the P00 pin is eliminated by analog delay, the edge is detected at most 10 μs later after an actual edge is input. The delay time until edge detection is not a constant value because of differences in device characteristics.**

CHAPTER 22 INTERRUPT FUNCTIONS

The μPD784955 is provided with three interrupt request service modes – vectored interrupt, context switching, and macro service (refer to **Table 22-1**). These three service modes can be set as required in the program. However interrupt service by macro service can only be selected for interrupt request sources provided with the macro service processing mode shown in Table 22-2. Context switching cannot be selected for non-maskable interrupts or operand error interrupts.

Multiple-interrupt control using 4 priority levels can easily be performed for maskable vectored interrupts.

Table 22-1. Interrupt Request Service Modes

Interrupt Request Service Mode	Servicing Performed	PC & PSW Contents	Service
Vectored interrupts	Software	Saving to & restoration from stack	Executed by branching to service program at address[Note] specified by vector table
Context switching		Saving to & restoration from fixed area in register bank	Executed by automatic switching to register bank specified by vector table and branching to service program at address[Note] specified by fixed area in register bank
Macro service	Hardware (firmware)	Retained	Execution of pre-set service such as data transfers between memory and I/O

Note The start addresses of all interrupt service programs must be in the base area. If the body of a service program cannot be located in the base area, a branch instruction to the service program should be written in the base area.

22.1 Interrupt Request Sources

The μPD784955 has the 31 interrupt request sources shown in Table 22-2, with a vector table allocated to each.

Table 22-2. Interrupt Request Sources (1/2)

Type of Interrupt Request	Default Priority	Interrupt Request Generating Source	Generating Unit	Interrupt Control Register Name	Context Switching	Macro Service	Macro Service Control Word Address	Vector Table Address
Software	None	BRK instruction execution	—	—	Not possible	Not possible	—	003EH
		BRKCS instruction execution	—	—	Possible	Not possible	—	—
Operand error	None	Invalid operand in “MOV STBC, #byte” instruction or “MOV WDM, #byte instruction”, and LOCATION instruction	—	—	Not possible	Not possible	—	003CH
Non-maskable	None	NMI (pin input edge detection)	Edge detection	—	Not possible	Not possible	—	0002H
		INTWDT (watchdog timer overflow)	Watchdog timer	—	Not possible	Not possible	—	0004H

Table 22-2. Interrupt Request Sources (2/2)

Type of Interrupt Request	Default Priority	Interrupt Request Generating Source	Generating Unit	Interrupt Control Register Name	Context Switching	Macro Service	Macro Service Control Word Address	Vector Table Address
Maskable	0	INTP0 (Pin input edge detection)	Edge detection	PIC0	Possible	Possible	0FE06H	0006H
	1	INTP1 (Pin input edge detection)		PIC1			0FE08H	0008H
	2	INTP2/INTTM41 (pin input edge detection/TM4-CR41 match signal)	Edge detection/ TM4	PIC2			0FE0AH	000AH
	3	INTP3 (Pin input edge detection)	Edge detection	PIC3			0FE0CH	000CH
	4	INTP4 (Pin input edge detection)		PIC4			0FE0EH	000EH
	5	INTP5/INTTM51 (pin input edge detection/TM5-CR51 match signal)	Edge detection/ TM5	PIC5			0FE10H	0010H
	6	INTP6 (Pin input edge detection)	Edge detection	PIC6			0FE12H	0012H
	7	INTTM00 (TM0-CR00 match signal)	TM0	TMIC00			0FE14H	0014H
	8	INTTM01 (TM0-CR01 match signal)		TMIC01			0FE16H	0016H
	9	INTTM10 (TM1-CR10 match signal)	TM1	TMIC10			0FE18H	0018H
	10	INTTM11 (TM1-CR11 match signal)		TMIC11			0FE1AH	001AH
	11	INTTM20 (TM2-CR20 match signal)	TM2	TMIC20			0FE1CH	001CH
	12	INTTM21 (TM2-CR21 match signal)		TMIC21			0FE1EH	001EH
	13	INTTM30 (TM3-CR30 match signal)	TM3	TMIC30			0FE20H	0020H
	14	INTTM31 (TM3-CR31 match signal)		TMIC31			0FE22H	0022H
	15	INTTM40 (TM4-CR40 match signal)	TM4	TMIC40			0FE24H	0024H
	16	INTTM42 (TM4-CR42 match signal)		TMIC42			0FE26H	0026H
	17	INTTM50 (TM5-CR50 match signal)	TM5	TMIC50			0FE28H	0028H
	18	INTTM52 (TM5-CR52 match signal)		TMIC52			0FE2AH	002AH
	19	INTTM6 (TM6-CR6 match signal)	TM6	TMIC6			0FE2CH	002CH
	20	INTTM7 (TM7-CR7 match signal)	TM7	TMIC7			0FE2EH	002EH
	21	INTSER1 (UART receive error)	UART	SERIC1			0FE30H	0030H
	22	INTSR1 (UART receive completion)		SRIC1			0FE32H	0032H
	23	INTST1 (UART transmit completion)		STIC1			0FE34H	0034H
	24	INTCSI0 (transmit/receive completion clocked-serial interface)	CSI	CSIIC0			0FE36H	0036H
	25	INTAD (A/D conversion completion)	A/D converter	ADIC			0FE38H	0038H

Remarks 1. The default priority is a fixed number. It indicates the priority when multiple interrupt requests with the same specified priority are generated simultaneously.

2.

TM	: timer/counter
CR (00, 01, 40, 41, 42, 50, 51)	: capture/compare registers
CR (10, 11, 20, 21, 30, 31, 52, 6, 7)	: compare registers
UART	: asynchronous serial interface
CSI	: clocked serial interface

22.1.1 Software interrupts

Interrupts by software consist of the BRK instruction that generates a vectored interrupt and the BRKCS instruction that performs context switching.

Software interrupts are acknowledged even in the interrupt disabled state, and are not subject to priority control.

22.1.2 Operand error interrupts

These interrupts are generated if there is an illegal operand in an "MOV STBC, #byte" instruction or "MOV WDMC, #byte" instruction, and LOCATION instruction.

Operand error interrupts are acknowledged even in the interrupt disabled state, and are not subject to priority control.

22.1.3 Non-maskable interrupts

A non-maskable interrupt is generated by NMI pin input or the watchdog timer.

Non-maskable interrupts are acknowledged unconditionally[Note], even in the interrupt disabled state. They are not subject to interrupt priority control, and are of higher priority that any other interrupt.

Note Except during execution of the service program for the same non-maskable interrupt, and during execution of the service program for a higher-priority non-maskable interrupt

22.1.4 Maskable interrupts

A maskable interrupt is one subject to masking control according to the setting of an interrupt mask flag. In addition, acknowledgment enabling/disabling can be specified for all maskable interrupts by means of the IE flag in the program status word (PSW).

In addition to normal vectored interrupts, maskable interrupts can be acknowledged by context switching and macro service (though some interrupts cannot use macro service: refer to **Table 22-2**).

The priority order for maskable interrupt requests when interrupt requests of the same priority are generated simultaneously is predetermined (default priority) as shown in Table 22-2. Also, multiprocessing control can be performed with interrupt priorities divided into 4 levels. However, macro service requests are acknowledged without regard to priority control or the IE flag.

22.2 Interrupt Service Modes

There are three μPD784955 interrupt service modes, as follows:

- Vectored interrupt service
- Macro service
- Context switching

22.2.1 Vectored interrupt service

When an interrupt is acknowledged, the program counter (PC) and program status word (PSW) are automatically saved to the stack, a branch is made to the address indicated by the data stored in the vector table, and the interrupt service routine is executed.

22.2.2 Macro service

When an interrupt is acknowledged, CPU execution is temporarily suspended and a data transfer is performed by hardware. Since macro service is performed without the intermediation of the CPU, it is not necessary to save or restore CPU statuses such as the program counter (PC) and program status word (PSW) contents. This is therefore very effective in improving the CPU service time (refer to **22.8 Macro Service Function**).

22.2.3 Context switching

When an interrupt is acknowledged, the prescribed register bank is selected by hardware, a branch is made to a pre-set vector address in the register bank, and at the same time the current program counter (PC) and program status word (PSW) are saved in the register bank (refer to **22.4.2 BRKCS instruction software interrupt (software context switching) acknowledgment operation** and **22.7.2 Context switching**).

Remark “Context” refers to the CPU registers that can be accessed by a program while that program is being executed. These registers include general registers, the program counter (PC), program status word (PSW), and stack pointer (SP).

22.3 Interrupt Processing Control Registers

μPD784955 interrupt service is controlled for each interrupt request by various control registers that perform interrupt service specification. The interrupt control registers are listed in Table 22-3.

Table 22-3. Control Registers

Register Name	Symbol	Function
Interrupt control registers	PIC0, PIC1, PIC2, PIC3, PIC4, PIC5, PIC6, TMIC00, TMIC01, TMIC10, TMIC11, TMIC20, TMIC21, TMIC30, TMIC31, TMIC40, TMIC42, TMIC50, TMIC52, TMIC6, TMIC7, SERIC1, SRIC1, STIC1,CSIIC0, ADIC	Registers to record generation of interrupt request, control masking, specify vectored interrupt service or macro service processing, enable or disable context switching function, and specify priority.
Interrupt mask registers	MK0 (MK0L, MK0H) MK1 (MK1L, MK1H)	Control masking of maskable interrupt request. Associated with mask control flag in interrupt control register. Can be accessed in word or byte units.
In-service priority register	ISPR	Records priority of interrupt request currently acknowledged.
Interrupt mode control register	IMC	Controls nesting of maskable interrupt with priority specified to lowest level (level 3).
Watchdog timer mode register	WDM	Specifies priorities of interrupt by NMI pin input and overflow of watchdog timer.
Program status word	PSW	Enables or disables acknowledging maskable interrupt.

An interrupt control register is allocated to each interrupt source. The flags of each register perform control of the contents corresponding to the relevant bit position in the register. The interrupt control register flag names corresponding to each interrupt request signal are shown in Table 22-4.

Table 22-4. Flag List of Interrupt Control Registers for Interrupt Requests

Default Priority	Interrupt Request Signal	Interrupt Control Register						
			Interrupt Request Flag	Interrupt Mask Flag	Macro Service Enable Flag	Context Switching Enable Flag	Priority Specification Flag	
0	INTP0	PIC0	PIF0	PMK0	PISM0	PCSE0	PPR01	PPR00
1	INTP1	PIC1	PIF1	PMK1	PISM1	PCSE1	PPR11	PPR10
2	INTP2/INTTM41	PIC2	PIF2	PMK2	PISM2	PCSE2	PPR21	PPR20
3	INTP3	PIC3	PIF3	PMK3	PISM3	PCSE3	PPR31	PPR30
4	INTP4	PIC4	PIF4	PMK4	PISM4	PCSE4	PPR41	PPR40
5	INTP5/INTTM51	PIC5	PIF5	PMK5	PISM5	PCSE5	PPR51	PPR50
6	INTP6	PIC6	PIF6	PMK6	PISM6	PCSE6	PPR61	PPR60
7	INTTM00	TMIC00	TMIF00	TMMK00	TMISM00	TMCSE00	TMPR001	TMPR000
8	INTTM01	TMIC01	TMIF01	TMMK01	TMISM01	TMCSE01	TMPR011	TMPR010
9	INTTM10	TMIC10	TMIF10	TMMK10	TMISM10	TMCSE10	TMPR101	TMPR100
10	INTTM11	TMIC11	TMIF11	TMMK11	TMISM11	TMCSE11	TMPR111	TMPR110
11	INTTM20	TMIC20	TMIF20	TMMK20	TMISM20	TMCSE20	TMPR201	TMPR200
12	INTTM21	TMIC21	TMIF21	TMMK21	TMISM21	TMCSE21	TMIC211	TMIC210
13	INTTM30	TMIC30	TMIF30	TMMK30	TMISM30	TMCSE30	TMPR301	TMPR300
14	INTTM31	TMIC31	TMIF31	TMMK31	TMISM31	TMCSE31	TMPR311	TMPR310
15	INTTM40	TMIC40	TMIF40	TMMK40	TMISM40	TMCSE40	TMPR401	TMPR400
16	INTTM42	TMIC42	TMIF42	TMMK42	TMISM42	TMCSE42	TMPR421	TMPR420
17	INTTM50	TMIC50	TMIF50	TMMK50	TMISM50	TMCSE50	TMPR501	TMPR500
18	INTTM52	TMIC52	TMIF52	TMMK52	TMISM52	TMCSE52	TMPR521	TMPR520
19	INTTM6	TMIC6	TMIF6	TMMK6	TMISM6	TMCSE6	TMPR61	TMPR60
20	INTTM7	TMIC7	TMIF7	TMMK7	TMISM7	TMCSE7	TMPR71	TMPR70
21	INTSER1	SERIC1	SERIF1	SERMK1	SERISM1	SERCSE1	SERPR11	SERPR10
22	INTSR1	SRIC1	SRIF1	SRMK1	SRISM1	SRCSE1	SRPR11	SRPR10
23	INTST1	STIC1	STIF1	STMK1	STISM1	STCSE1	STPR11	STPR10
24	INTCSI0	CSIIC0	CSIIF0	CSIMK0	CSIISM0	CSICSE0	CSIPR01	CSIPR00
25	INTAD	ADIC	ADIF	ADMK	ADISM	ADICSE	ADPR1	ADPR0

22.3.1 Interrupt control registers

An interrupt control register is allocated to each interrupt source, and performs priority control, mask control, etc., for the corresponding interrupt request. The interrupt control register format is shown in Figure 22-1.

(1) Priority specification flags (××PR1/××PR0)

The priority specification flags specify the priority on an individual interrupt source basis for the 26 maskable interrupts.

Up to 4 priority levels can be specified, and multiple interrupt sources can be specified at the same level. Among maskable interrupt sources, level 0 is the highest priority.

If multiple interrupt requests are generated simultaneously among interrupt source of the same priority level, they are acknowledged in default priority order.

These flags can be manipulated bit-wise by software.

$\overline{\text{RESET}}$ input sets all bits to 1.

(2) Context switching enable flag (××CSE)

The context switching enable flag specifies that a maskable interrupt request is to be serviced by context switching.

In context switching, the register bank specified beforehand is selected by hardware, a branch is made to a vector address stored beforehand in the register bank, and at the same time the current contents of the program counter (PC) and program status word (PSW) are saved in the register bank.

Context switching is suitable for real-time processing, since execution of interrupt servicing can be started faster than with normal vectored interrupt servicing.

This flag can be manipulated bit-wise by means of software.

$\overline{\text{RESET}}$ input sets all bits to 0.

(3) Macro service enable flag (××ISM)

The macro service enable flag specifies whether an interrupt request corresponding to each flag is to be processed by vectored interrupt or context switching, or by macro service.

When macro service processing is selected, at the end of the macro service (when the macro service counter reaches 0) the macro service enable flag is automatically cleared (0) by hardware (vectored interrupt service/context switching service).

This flag can be manipulated bit-wise by means of software.

$\overline{\text{RESET}}$ input sets all bits to 0.

(4) Interrupt mask flag (××MK)

An interrupt mask flag specifies enabling/disabling of vectored interrupt servicing and macro service processing for the interrupt request corresponding to each flag.

The interrupt mask contents are not changed by the start of interrupt service, etc., and are the same as the interrupt mask register contents (refer to **22.3.2 Interrupt mask registers (MK0/MK1)**).

Macro service processing requests are also subject to mask control, and macro service requests can also be masked with this flag.

This flag can be manipulated by means of software.

$\overline{\text{RESET}}$ input sets all bits to 1.

(5) Interrupt request flag (××IF)

An interrupt request flag is set (1) by generation of the interrupt request that corresponds to that flag. When the interrupt is acknowledged, the flag is automatically cleared (0) by hardware.

This flag can be manipulated by means of software.

$\overline{\text{RESET}}$ input sets all bits to 0.

Figure 22-1. Interrupt Control Register (xxICn) (1/3)

Address : 0FFE0H to 0FFE8H After Reset : 43H R/W

Symbol	⑦	⑥	⑤	④	3	2	①	⓪
PIC0	PIF0	PMK0	PISM0	PCSE0	0	0	PPR01	PPR00
PIC1	PIF1	PMK1	PISM1	PCSE1	0	0	PPR11	PPR10
PIC2	PIF2	PMK2	PISM2	PCSE2	0	0	PPR21	PPR20
PIC3	PIF3	PMK3	PISM3	PCSE3	0	0	PPR31	PPR30
PIC4	PIF4	PMK4	PISM4	PCSE4	0	0	PPR41	PPR40
PIC5	PIF5	PMK5	PISM5	PCSE5	0	0	PPR51	PPR50
PIC6	PIF6	PMK6	PISM6	PCSE6	0	0	PPR61	PPR60
TMIC00	TMIF00	TMMK00	TMISM00	TMCSE00	0	0	TMPR001	TMPR000
TMIC01	TMIF01	TMMK01	TMISM01	TMCSE01	0	0	TMPR011	TMPR010

xxIFn	Interrupt Request Generation
0	No interrupt request (Interrupt signal is not generated.)
1	Interrupt request (Interrupt signal is generated.)

xxMKn	Interrupt Processing Enable/Disable
0	Interrupt processing enable
1	Interrupt processing disable

xxISMn	Interrupt Processing Mode Specification
0	Vectored interrupt processing/Context switching processing
1	Macro service processing

xxCSEn	Context Switching Processing Specification
0	Processing with vectored interrupt
1	Processing with context switching

xxPRn1	xxPRn0	Interrupt Request Priority Specification
0	0	Priority 0 (Highest priority)
0	1	Priority 1
1	0	Priority 2
1	1	Priority 3

Figure 22-1. Interrupt Control Register (xxICn) (2/3)

Address : 0FFE9H to 0FFF1H   After Reset : 43H   R/W

Symbol	7	6	5	4	3	2	1	0
TMIC10	TMIF10	TMMK10	TMISM10	TMCSE10	0	0	TMPR101	TMPR100
TMIC11	TMIF11	TMMK11	TMISM11	TMCSE11	0	0	TMPR111	TMPR110
TMIC20	TMIF20	TMMK20	TMISM20	TMCSE20	0	0	TMPR201	TMPR200
TMIC21	TMIF21	TMMK21	TMISM21	TMCSE21	0	0	TMPR211	TMPR210
TMIC30	TMIF30	TMMK30	TMISM30	TMCSE30	0	0	TMPR301	TMPR300
TMIC31	TMIF31	TMMK31	TMISM31	TMCSE31	0	0	TMPR311	TMPR310
TMIC40	TMIF40	TMMK40	TMISM40	TMCSE40	0	0	TMPR401	TMPR400
TMIC42	TMIF42	TMMK42	TMISM42	TMCSE42	0	0	TMPR421	TMPR420
TMIC50	TMIF50	TMMK50	TMISM50	TMCSE50	0	0	TMPR501	TMPR500

xxIFn	Interrupt Request Generation
0	No interrupt request (Interrupt signal is not generated.)
1	Interrupt request (Interrupt signal is generated.)

xxMKn	Interrupt Processing Enable/Disable
0	Interrupt processing enable
1	Interrupt processing disable

xxISMn	Interrupt Processing Mode Specification
0	Vectored interrupt processing/Context switching processing
1	Macro service processing

xxCSEn	Context Switching Processing Specification
0	Processing with vectored interrupt
1	Processing with context switching

xxPRn1	xxPRn0	Interrupt Request Priority Specification
0	0	Priority 0 (Highest priority)
0	1	Priority 1
1	0	Priority 2
1	1	Priority 3

Figure 22-1. Interrupt Control Register (xxICn) (3/3)

Address : 0FFF2H to 0FFF9H After Reset : 43H R/W

Symbol	⑦	⑥	⑤	④	3	2	①	⓪
TMIC52	TMIF52	TMMK52	TMISM52	TMCSE52	0	0	TMPR521	TMPR520
TMIC6	TMIF6	TMMK6	TMISM6	TMCSE6	0	0	TMPR61	TMPR60
TMIC7	TMIF7	TMMK7	TMISM7	TMCSE7	0	0	TMPR71	TMPR70
SERIC1	SERIF1	SERMK1	SERISM1	SERCSE1	0	0	SERPR11	SERPR10
SRIC1	SRIF1	SRMK1	SRISM1	SRCSE1	0	0	SRPR11	SRPR10
STIC1	STIF1	STMK1	STISM1	STCSE1	0	0	STPR11	STPR10
CSIIC0	CSIIF0	CSIMK0	CSIISM0	CSICSE0	0	0	CSIPR01	CSIPR00
ADIC	ADIF	ADMK	ADISM	ADCSE	0	0	ADPR01	ADPR00

xxIFn	Interrupt Request Generation
0	No interrupt request (Interrupt signal is not generated.)
1	Interrupt request (Interrupt signal is generated.)

xxMKn	Interrupt Processing Enable/Disable
0	Interrupt processing enable
1	Interrupt processing disable

xxISMn	Interrupt Processing Mode Specification
0	Vectored interrupt processing/Context switching processing
1	Macro service processing

xxCSEn	Context Switching Processing Specification
0	Processing with vectored interrupt
1	Processing with context switching

xxPRn1	xxPRn0	Interrupt Request Priority Specification
0	0	Priority 0 (Highest priority)
0	1	Priority 1
1	0	Priority 2
1	1	Priority 3

22.3.2 Interrupt mask registers (MK0, MK1)

The MK0 and MK1 are composed of interrupt mask flags. MK0 and MK1 are 16-bit registers that can be manipulated as a 16-bit unit. MK0 can be manipulated in 8 bit units as MK0L and MK0H, and similarly MK1 can be manipulated as MK1L and MK1H.

In addition, each bit of the MK0 and MK1 can be manipulated individually with a bit manipulation instruction. Each interrupt mask flag controls enabling/disabling of the corresponding interrupt request.

When an interrupt mask flag is set (1), acknowledgment of the corresponding interrupt request is disabled.

When an interrupt mask flag is cleared (0), the corresponding interrupt request can be acknowledged as a vectored interrupt or macro service request.

Each interrupt mask flag in the MK0 and MK1 is the same flag as the interrupt mask flag in the interrupt control register. The MK0 and MK1 are provided for en bloc control of interrupt masking.

After $\overline{\text{RESET}}$ input, the MK0 and MK1 are set to FFFFH, and all maskable interrupts are disabled.

Figure 22-2. Format of Interrupt Mask Registers (MK0, MK1)

<Byte access>

Address : 0FFACH to 0FFAFH After Reset : FFH R/W

Symbol	7	6	5	4	3	2	1	0
MK0L	TMMK00	PMK6	PMK5	PMK4	PMK3	PMK2	PMK1	PMK0
MK0H	TMMK40	TMMK31	TMMK30	TMMK21	TMMK20	TMMK11	TMMK10	TMMK01
MK1L	STMK1	SRMK1	SERMK1	TMMK7	TMMK6	TMMK52	TMMK50	TMMK42
MK1H	1	1	1	1	1	1	ADMK	CSIMK0

xxMKn	Interrupt Request Enable/Disable
0	Interrupt processing enable
1	Interrupt processing disable

<Word access>

Address : 0FFACH, 0FFAEH After Reset : FFFFH R/W

Symbol	15	14	13	12	11	10	9	8
MK0	TMMK40	TMMK31	TMMK30	TMMK21	TMMK20	TMMK11	TMMK10	TMMK01
	7	6	5	4	3	2	1	0
	TMMK00	PMK6	PMK5	PMK4	PMK3	PMK2	PMK1	PMK0
	15	14	13	12	11	10	9	8
MK1	1	1	1	1	1	1	ADMK	CSIMK0
	7	6	5	4	3	2	1	0
	STMK1	SRMK1	SERMK1	TMMK7	TMMK6	TMMK52	TMMK50	TMMK42

xxMKn	Interrupt Request Enable/Disable
0	Interrupt processing enable
1	Interrupt processing disable

22.3.3 In-service priority register (ISPR)

ISPR shows the priority level of the maskable interrupt currently being serviced and the non-maskable interrupt being serviced. When a maskable interrupt request is acknowledged, the bit corresponding to the priority of that interrupt request is set to 1, and remains set until the service program ends. When a non-maskable interrupt is acknowledged, the bit corresponding to the priority of that non-maskable interrupt is set to 1, and remains set until the service program ends.

When a RETI instruction or RETCS instruction is executed, the bit, among those set to 1 in ISPR, that corresponds to the highest-priority interrupt request is automatically cleared to 0 by hardware.

The contents of ISPR are not changed by execution of a RETB or RETCSB instruction.

$\overline{\text{RESET}}$ input sets the ISPR register to 00H.

Figure 22-3. Format of In-Service Priority Register (ISPR)

Address : 0FFA8H   After Reset : 00H   R

Symbol	7	6	5	4	3	2	1	0
ISPR	NMIS	WDTS	0	0	ISPR3	ISPR2	ISPR1	ISPR0

NMIS	NMI Processing Status
0	NMI interrupt is not acknowledged.
1	NMI interrupt is acknowledged.

WDTS	Watchdog Timer Interrupt Processing Status
0	Watchdog timer interrupt is not acknowledged.
1	Watchdog timer interrupt is acknowledged.

ISPRn	Priority Level (n = 0 to 3)
0	Interrupt of priority level n is not acknowledged.
1	Interrupt of priority level n is acknowledged.

Caution The in-service priority register (ISPR) is a read-only register. The microcontroller may malfunction if this register is written.

22.3.4 Interrupt mode control register (IMC)

IMC contains the PRSL flag. The PRSL flag specifies enabling/disabling of nesting of maskable interrupts for which the lowest priority level (level 3) is specified.

When IMC is manipulated, the interrupt disabled state (DI state) should be set first to prevent malfunction.

IMC can be read or written with an 8-bit manipulation instruction or bit manipulation instruction.

$\overline{\text{RESET}}$ input sets the IMC register to 80H.

Figure 22-4. Format of Interrupt Mode Control Register (IMC)

Address : 0FFAAH After Reset : 80H R/W

Symbol	7	6	5	4	3	2	1	0
IMC	PRSL	0	0	0	0	0	0	0

PRSL	Nesting Control of Maskable Interrupt (lowest level)
0	Nesting of interrupts with level 3 (lowest level) is enabled.
1	Nesting of interrupts with level 3 (lowest level) is disabled.

22.3.5 Watchdog timer mode register (WDM)

The WDT4 bit of WDM specifies the priority of NMI pin input non-maskable interrupts and watchdog timer overflow non-maskable interrupts.

WDM can be written to only by a dedicated instruction. This dedicated instruction, “MOV WDM, #byte”, has a special code configuration (4 bytes), and a write is not performed unless the 3rd and 4th bytes of the operation code are mutual complements.

If the 3rd and 4th bytes of the operation code are not mutual 1’s complements, a write is not performed and an operand error interrupt is generated. In this case, the return address saved in the stack area is the address of the instruction that was the source of the error, and thus the address that was the source of the error can be identified from the return address saved in the stack area.

If recovery from an operand error is simply performed by means of a RETB instruction, an endless loop will result.

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

Other write instructions (MOV WDM, A; AND WDM, #byte; SET1 WDM.7, etc.) are ignored and do not perform any operation. That is, a write is not performed to the WDM, and an interrupt such as an operand error interrupt is not generated.

WDM can be read at any time by a data transfer instruction.

$\overline{\text{RESET}}$ input sets the WDM register to 00H.

Figure 22-5. Format of Watchdog Timer Mode Register (WDM)

Address : 0FFC2H After Reset : 00H R/W

Symbol	7	6	5	4	3	2	1	0
WDM	RUN	0	0	WDT4	0	WDT2	WDT1	0

RUN	Specifies Operation of Watchdog Timer (refer to **Figure 12-2**).

WDT4	Priority of Watchdog Timer Interrupt Request
0	Watchdog timer interrupt request < NMI pin input interrupt request
1	Watchdog timer interrupt request > NMI pin input interrupt request

WDT2	WDT1	Specifies Count Clock of Watchdog Timer (refer to **Figure 12-2**).

Caution The watchdog timer mode register (WDM) can be written only by using a dedicated instruction (MOV WDM, #byte).

22.3.6 Program status word (PSW)

PSW is a register that holds the current status regarding instruction execution results and interrupt requests. The IE flag that sets enabling/disabling of maskable interrupts is mapped in the low-order 8 bits of the PSW (PSWL).

PSWL can be read or written to with an 8-bit manipulation instruction, and can also be manipulated with a bit manipulation instruction or dedicated instruction (EI/DI).

When a vectored interrupt is acknowledged or a BRK instruction is executed, PSWL is saved to the stack and the IE flag is cleared (0). PSWL is also saved to the stack by the PUSH PSW instruction, and is restored from the stack by the RETI, RETB, or POP PSW instruction.

When context switching or a BRKCS instruction is executed, PSWL is saved to a fixed area in the register bank, and the IE flag is cleared to 0. PSWL is restored from the fixed area in the register bank by a RETCSI or RETCSB instruction.

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

Figure 22-6. Format of Program Status Word (PSWL)

After Reset : 00H

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

S	Used for normal instruction execution
Z	
RSS	
AC	

IE	Enable or Disable Accepting Interrupt
0	Disable
1	Enable

P/V	Used for normal instruction execution
CY	

22.4 Software Interrupt Acknowledgment Operations

A software interrupt is acknowledged in response to execution of a BRK or BRKCS instruction. Software interrupts cannot be disabled.

22.4.1 BRK instruction software interrupt acknowledgment operation

When a BRK instruction is executed, the program status word (PSW) and program counter (PC) are saved in that order to the stack, the IE flag is cleared (0), the vector table (003EH/003FH) contents are loaded into the low-order 16 bits of the PC, and 0000B into the high-order 4 bits, and a branch is performed (the start of the service program must be in the base area).

The RETB instruction must be used to return from a BRK instruction software interrupt.

Caution The RETI instruction must not be used to return from a BRK instruction software interrupt.

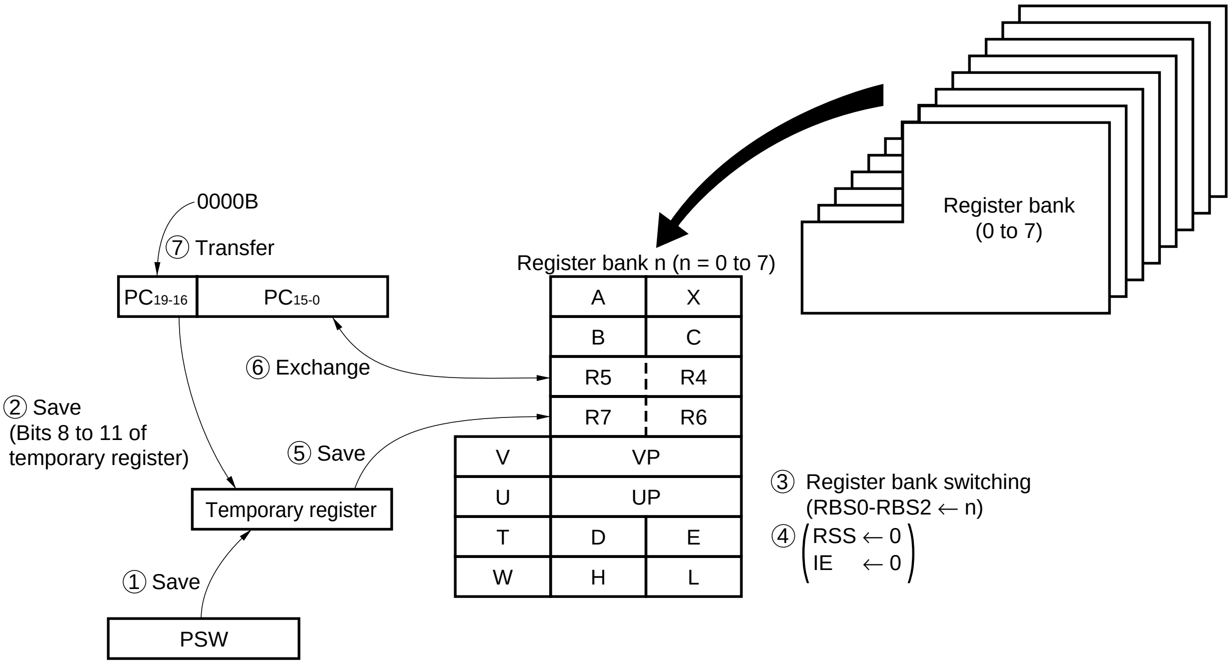
22.4.2 BRKCS instruction software interrupt (software context switching) acknowledgment operation

The context switching function can be initiated by executing a BRKCS instruction.

The register bank to be used after context switching is specified by the BRKCS instruction operand.

When a BRKCS instruction is executed, the program branches to the start address of the interrupt service program (which must be in the base area) stored beforehand in the specified register bank, and the contents of the program status word (PSW) and program counter (PC) are saved in the register bank.

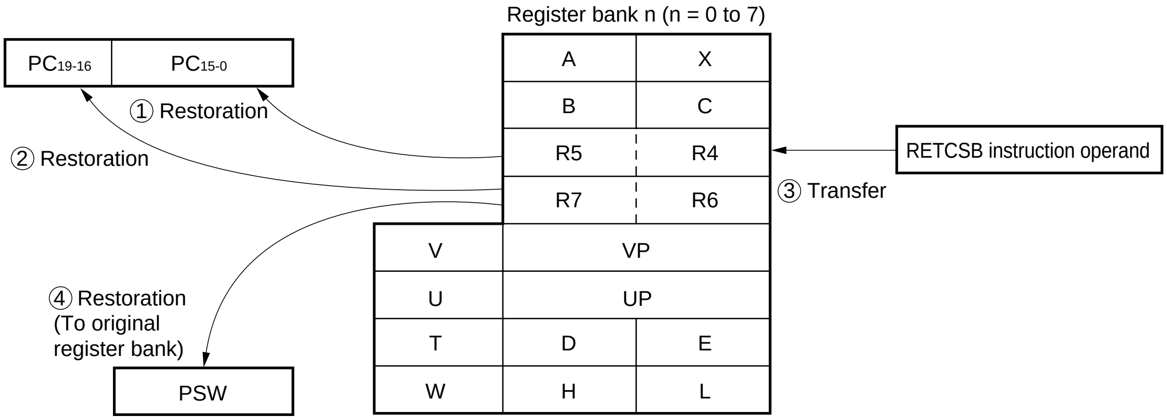
Figure 22-7. Context Switching Operation by Execution of a BRKCS Instruction



The RETCSB instruction is used to return from a software interrupt due to a BRKCS instruction. The RETCSB instruction must specify the start address of the interrupt service program for the next time context switching is performed by a BRKCS instruction. This interrupt service program start address must be in the base area.

Caution The RETCS instruction must not be used to return from a BRKCS instruction software interrupt.

Figure 22-8. Return from BRKCS Instruction Software Interrupt (RETCSB Instruction Operation)


Register bank n (n = 0 to 7)
PC19-16
PC15-0
① Restoration
② Restoration
A
X
B
C
R5
R4
R7
R6
RETCSB instruction operand
③ Transfer
V
VP
U
UP
T
D
E
W
H
L
④ Restoration (To original register bank)
PSW

22.5 Operand Error Interrupt Acknowledgment Operation

An operand error interrupt is generated when the data obtained by inverting all the bits of the 3rd byte of the operand of a “MOV STBC, #byte” instruction or LOCATION instruction or a “MOV WDM, #byte” instruction does not match the 4th byte of the operand. Operand error interrupts cannot be disabled.

When an operand error interrupt is generated, the program status word (PSW) and the start address of the instruction that caused the error are saved to the stack, the IE flag is cleared to 0, the vector table value is loaded into the program counter (PC), and a branch is performed (within the base area only).

As the address saved to the stack is the start address of the instruction in which the error occurred, simply writing a RETB instruction at the end of the operand error interrupt service program will result in generation of another operand error interrupt. You should therefore either process the address in the stack or initialize the program by referring to **22.12 Restoring Interrupt Function to Initial State**.

22.6 Non-maskable Interrupt Acknowledgment Operation

Non-maskable interrupts are acknowledged even in the interrupt disabled state. Non-maskable interrupts can be acknowledged at all times except during execution of the service program for an identical non-maskable interrupt or a non-maskable interrupt of higher priority.

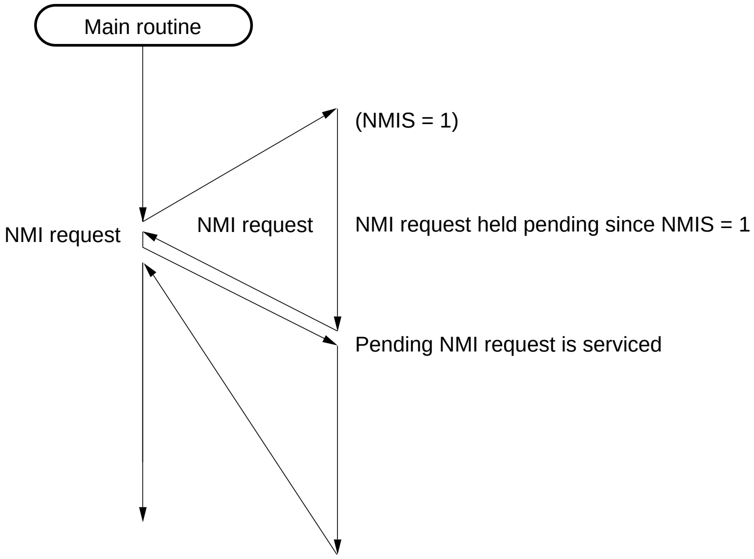
The relative priorities of non-maskable interrupts are set by the WDT4 bit of the watchdog timer mode register (WDM) (see **22.3.5 Watchdog timer mode register (WDM)**).

Except in the cases described in **22.9 When Interrupt Requests and Macro Service are Temporarily Held Pending**, a non-maskable interrupt request is acknowledged immediately. When a non-maskable interrupt request is acknowledged, the program status word (PSW) and program counter (PC) are saved in that order to the stack, the IE flag is cleared to 0, the in-service priority register (ISPR) bit corresponding to the acknowledged non-maskable interrupt is set to 1, the vector table contents are loaded into the PC, and a branch is performed. The ISPR bit that is set to 1 is the NMIS bit in the case of a non-maskable interrupt due to edge input to the NMI pin, and the WDTS bit in the case of watchdog timer overflow.

When the non-maskable interrupt service program is executed, non-maskable interrupt requests of the same priority as the non-maskable interrupt currently being executed and non-maskable interrupts of lower priority than the non-maskable interrupt currently being executed are held pending. A pending non-maskable interrupt is acknowledge after completion of the non-maskable interrupt service program currently being executed (after execution of the RETI instruction). However, even if the same non-maskable interrupt request is generated more than once during execution of the non-maskable interrupt service program, only one non-maskable interrupt is acknowledged after completion of the non-maskable interrupt service program.

Figure 22-9. Non-Maskable Interrupt Request Acknowledgment Operations (1/2)

(a) When a new NMI request is generated during NMI service program execution


Main routine
(NMIS = 1)
NMI request
NMI request
NMI request held pending since NMIS = 1
Pending NMI request is serviced

(b) When a watchdog timer interrupt request is generated during NMI service program execution (when the watchdog timer interrupt priority is higher (when WDT4 in the WDM = 1))

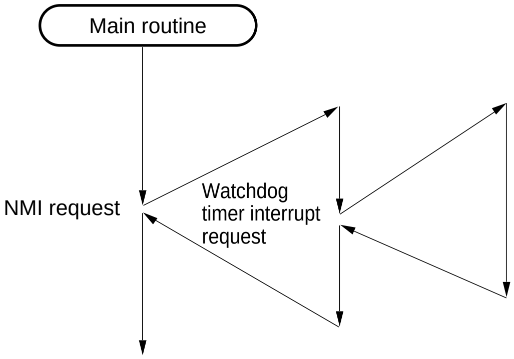
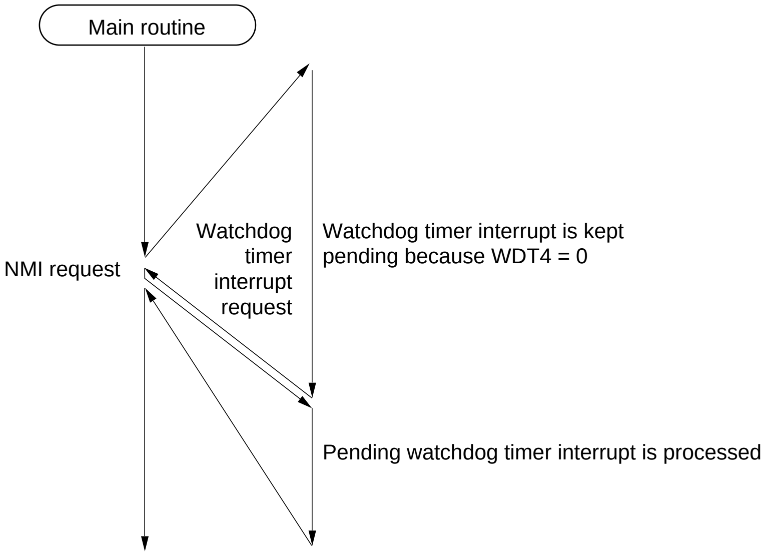
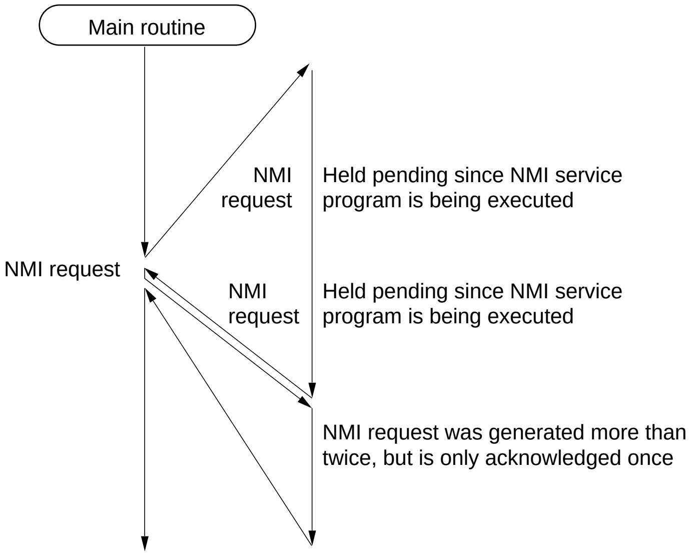


Figure 22-9. Non-Maskable Interrupt Request Acknowledgment Operations (2/2)

(c) When a watchdog timer interrupt request is generated during NMI service program execution (when the NMI interrupt priority is higher (when WDT4 in the WDM = 0))


Main routine
NMI request
Watchdog timer interrupt request
Watchdog timer interrupt is kept pending because WDT4 = 0
Pending watchdog timer interrupt is processed

(d) When an NMI request is generated twice during NMI service program execution


Main routine
NMI request
Held pending since NMI service program is being executed
NMI request
NMI request
Held pending since NMI service program is being executed
NMI request was generated more than twice, but is only acknowledged once

Cautions 1. Macro service requests are acknowledged and serviced even during execution of a non-maskable interrupt service program. If you do not want macro service processing to be performed during a non-maskable interrupt service program, you should manipulate the interrupt mask register in the non-maskable interrupt service program to prevent macro service generation.

2. The RETI instruction must be used to return from a non-maskable interrupt. Subsequent interrupt acknowledgment will not be performed normally if a different instruction is used. Refer to Section 22.12 Restoring Interrupt Function to Initial State when a program is to be restarted from the initial status after a non-maskable interrupt acknowledgement.

3. Non-maskable interrupts are always acknowledged, except during non-maskable interrupt service program execution (except when a high non-maskable interrupt request is generated during execution of a low-priority non-maskable interrupt service program) and for a certain period after execution of the special instructions shown in 22.9. Therefore, a non-maskable interrupt will be acknowledged even when the stack pointer (SP) value is undefined, in particular after reset release, etc. In this case, depending on the value of the SP, it may happen that the program counter (PC) and program status word (PSW) are written to the address of a write-inhibited special function register (SFRs) (see Table 3-6 in 3.8 Special Function Registers (SFRs)), and the CPU becomes deadlocked, or an unexpected signal is output from a pin, or the PC and PSW are written to an address in which RAM is not mounted, with the result that the return from the non-maskable interrupt service program is not performed normally and a software upset occurs.

Therefore, the program following $\overline{\text{RESET}}$ release must be as shown below.

```
            CSEG  AT 0
            DW    STRT
            CSEG  BASE
    STRT:
            LOCATION 0FH; or LOCATION 0
            MOVG SP, #imm24
```

22.7 Maskable Interrupt Acknowledgment Operation

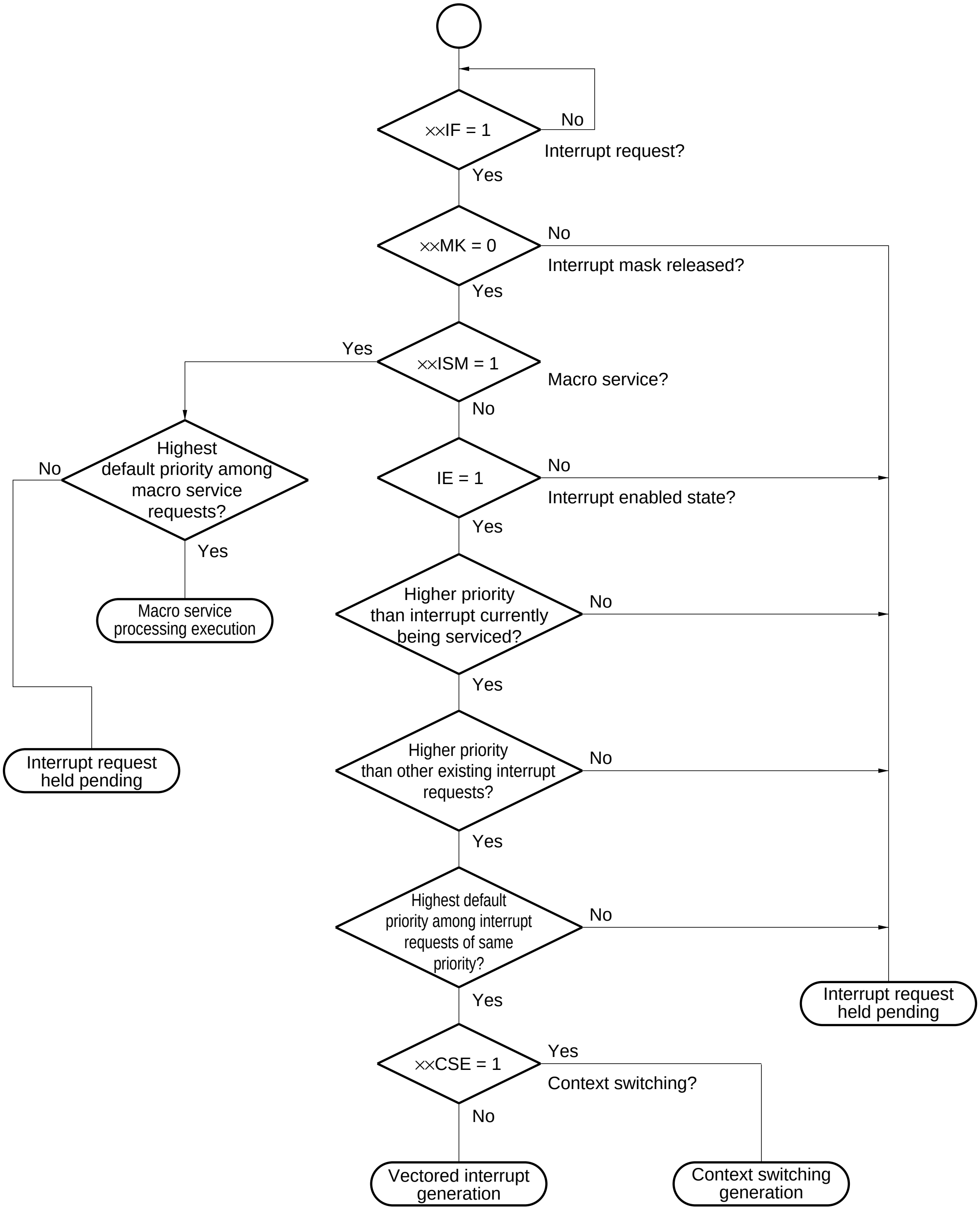
A maskable interrupt can be acknowledged when the interrupt request flag is set to 1 and the mask flag for that interrupt is cleared to 0. When servicing is performed by macro service, the interrupt is acknowledged and serviced by macro service immediately. In the case of vectored interrupt and context switching, an interrupt is acknowledged if the interrupt enabled state (when the IE flag is set to 1) and the priority of that interrupt is one for which acknowledgment is permitted.

If maskable interrupt requests are generated simultaneously, the interrupt for which the highest priority is specified by the priority specification flag is acknowledged. If the interrupts have the same priority specified, they are acknowledged in accordance with their default priorities.

A pending interrupt is acknowledged when a state in which it can be acknowledged is established.

The interrupt acknowledgment algorithm is shown in Figure 22-10.

Figure 22-10. Interrupt Request Acknowledgment Processing Algorithm


××IF = 1
No
Interrupt request?
Yes
××MK = 0
No
Interrupt mask released?
Yes
Yes
××ISM = 1
Macro service?
No
No
Highest default priority among macro service requests?
Yes
Macro service processing execution
Interrupt request held pending
IE = 1
No
Interrupt enabled state?
Yes
Higher priority than interrupt currently being serviced?
No
Yes
Higher priority than other existing interrupt requests?
No
Yes
Highest default priority among interrupt requests of same priority?
No
Yes
Interrupt request held pending
××CSE = 1
Yes
Context switching?
No
Vectored interrupt generation
Context switching generation

22.7.1 Vectored interrupt

When a vectored interrupt maskable interrupt request is acknowledged, the program status word (PSW) and program counter (PC) are saved in that order to the stack, the IE flag is cleared to 0 (the interrupt disabled state is set), and the in-service priority register (ISPR) bit corresponding to the priority of the acknowledged interrupt is set to 1. Also, data in the vector table predetermined for each interrupt request is loaded into PC, and a branch is performed. The return from a vectored interrupt is performed by means of the RETI instruction.

Caution When a maskable interrupt is acknowledged by vectored interrupt, the RETI instruction must be used to return from the interrupt. Subsequent interrupt acknowledgment will not be performed normally if a different instruction is used.

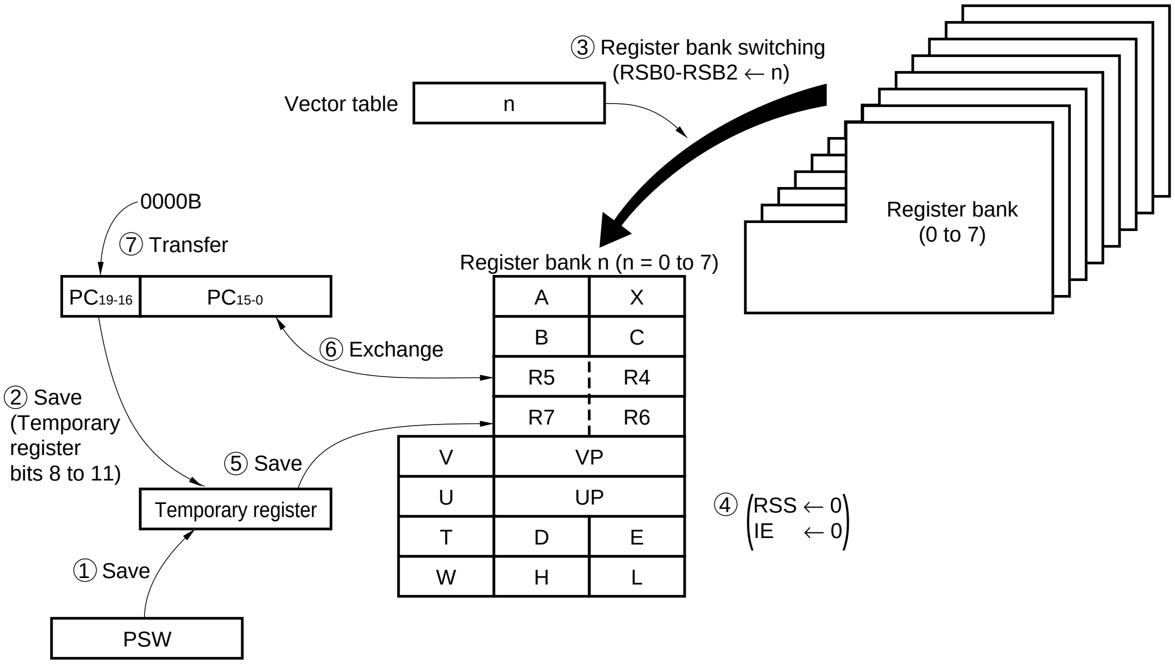
22.7.2 Context switching

Initiation of the context switching function is enabled by setting the context switching enable flag of the interrupt control register to 1.

When an interrupt request for which the context switching function is enabled is acknowledged, the register bank specified by 3 bits of the lower address (even address) of the corresponding vector table address is selected.

The vector address stored beforehand in the selected register bank is transferred to the program counter (PC), and at the same time the contents of PC and the program status word (PSW) up to that time are saved in the register bank and branching is performed to the interrupt service program.

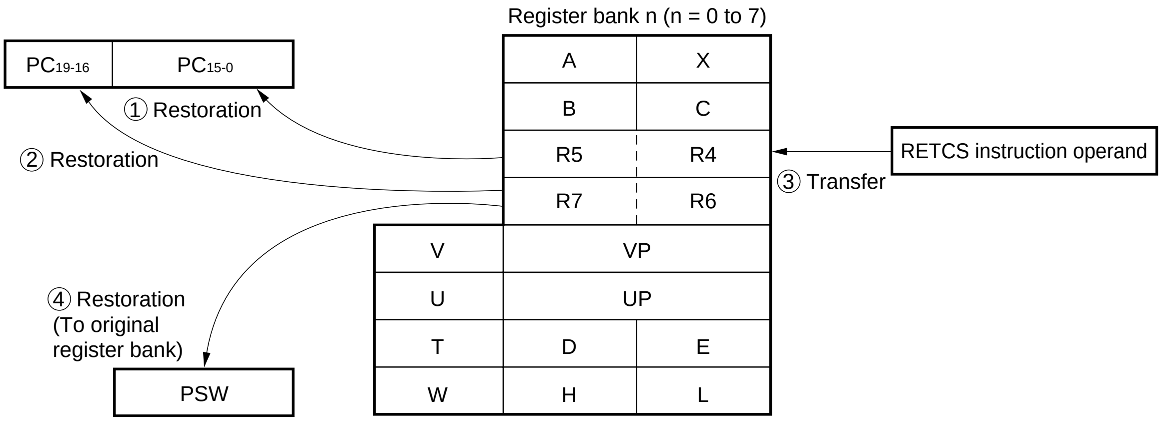
Figure 22-11. Context Switching Operation by Generation of an Interrupt Request



The RETCS instruction is used to return from an interrupt that uses the context switching function. The RETCS instruction must specify the start address of the interrupt service program to be executed when that interrupt is acknowledged next. This interrupt service program start address must be in the base area.

Caution The RETCS instruction must be used to return from an interrupt serviced by context switching. Subsequent interrupt acknowledgment will not be performed normally if a different instruction is used.

Figure 22-12. Return from Interrupt that Uses Context Switching by Means of RETCS Instruction


Register bank n (n = 0 to 7)
PC19-16
PC15-0
① Restoration
② Restoration
③ Transfer
RETCS instruction operand
④ Restoration (To original register bank)
PSW
A
X
B
C
R5
R4
R7
R6
V
VP
U
UP
T
D
E
W
H
L

22.7.3 Maskable interrupt priority levels

The μPD784955 performs multiple interrupt servicing in which an interrupt is acknowledged during servicing of another interrupt. Multiple interrupts can be controlled by priority levels.

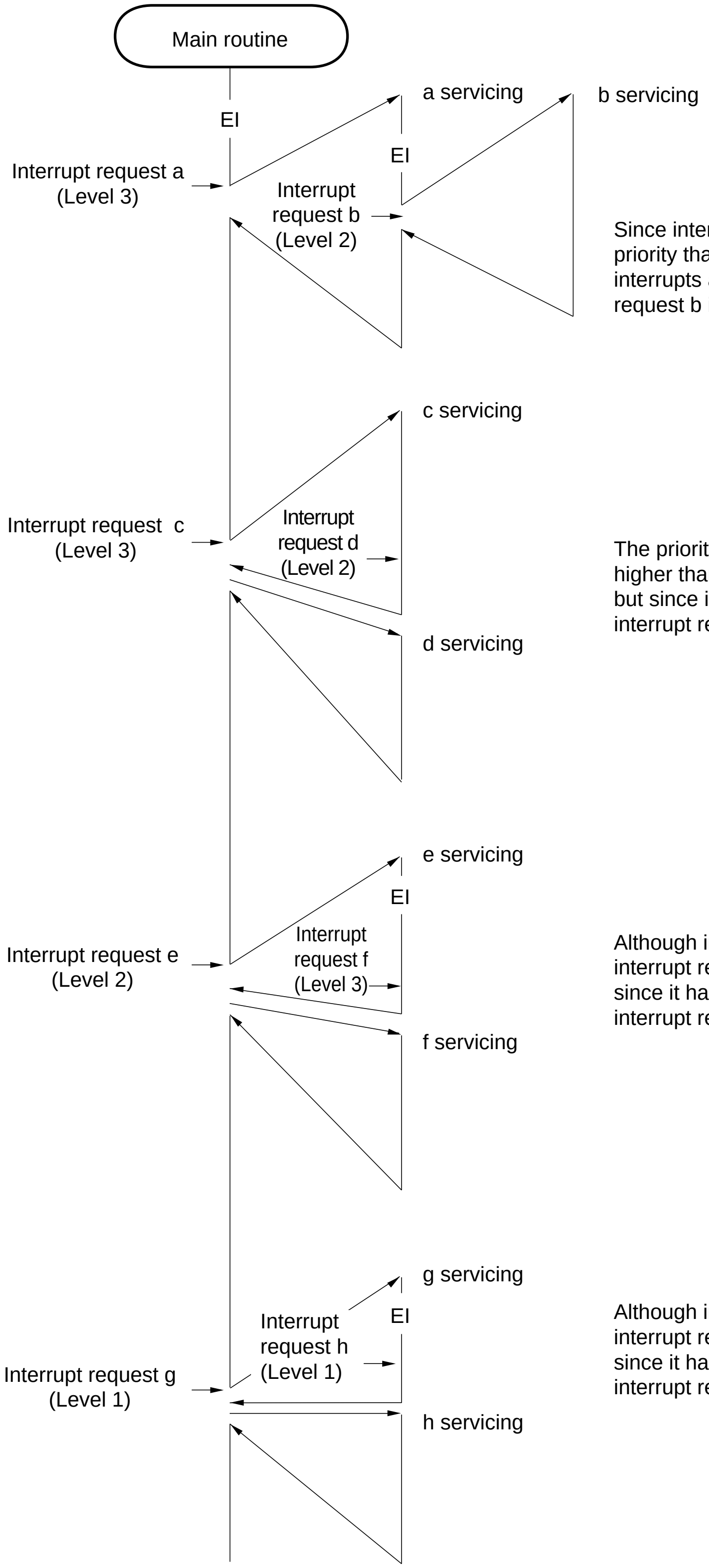
There are two kinds of priority control, control by default priority and programmable priority control in accordance with the setting of the priority specification flag. In priority control by means of default priority, interrupt service is performed in accordance with the priority preassigned to each interrupt request (default priority) (refer to **Table 22-2**). In programmable priority control, interrupt requests are divided into four levels according to the setting of the priority specification flag. Interrupt requests for which multiple interrupt is permitted are shown in Table 22-5.

Since the IE flag is cleared to 0 automatically when an interrupt is acknowledged, when multiple interrupt is used, the IE flag should be set to 1 to enable interrupts by executing an IE instruction in the interrupt service program, etc.

Table 22-5. Multiple Interrupt Servicing

Priority of Interrupt Currently Being Acknowledged	ISPR Value	IE Flag in PSW	PRSL in IMC Register	Acknowledgeable Maskable Interrupts
No interrupt being acknowledged	00000000	0	×	• All macro service only
		1	×	• All maskable interrupts
3	00001000	0	×	• All macro service only
		1	0	• All maskable interrupts
		1	1	• All macro service • Maskable interrupts specified as priority 0/1/2
2	0000×100	0	×	• All macro service only
		1	×	• All macro service • Maskable interrupts specified as priority 0/1
1	0000××10	0	×	• All macro service only
		1	×	• All macro service • Maskable interrupts specified as priority 0
0	0000×××1	×	×	• All macro service only
Non-maskable interrupts	1000×××× 0100×××× 1100××××	×	×	• All macro service only

Figure 22-13. Examples of Servicing When Another Interrupt Request Is Generated during Interrupt Service (1/3)



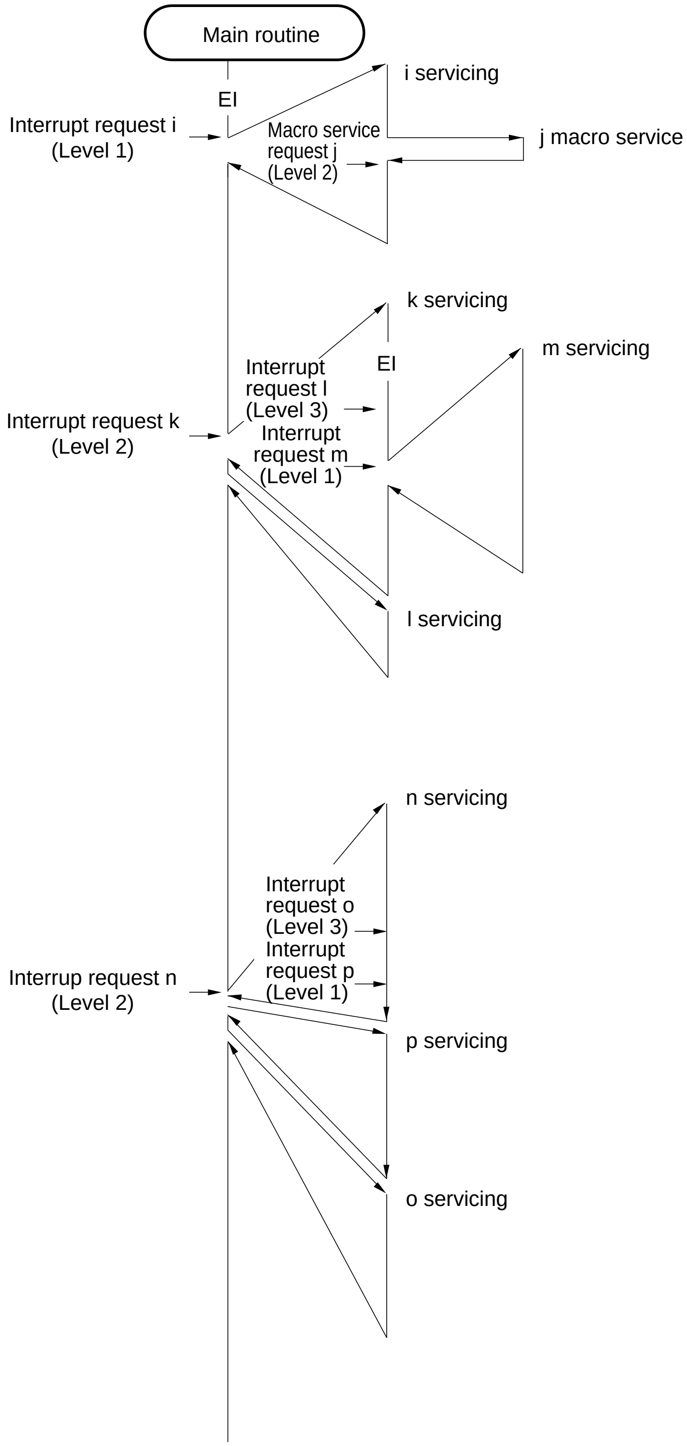
Since interrupt request b has a higher priority than interrupt request a, and interrupts are enabled, interrupt request b is acknowledged.

The priority of interrupt request d is higher than that of interrupt request c, but since interrupts are disabled, interrupt request d is held pending.

Although interrupts are enabled, interrupt request f is held pending since it has a lower priority than interrupt request e.

Although interrupts are enabled, interrupt request h is held pending since it has the same priority as interrupt request g.

Figure 22-13. Examples of Servicing When Another Interrupt Request Is Generated during Interrupt Service (2/3)

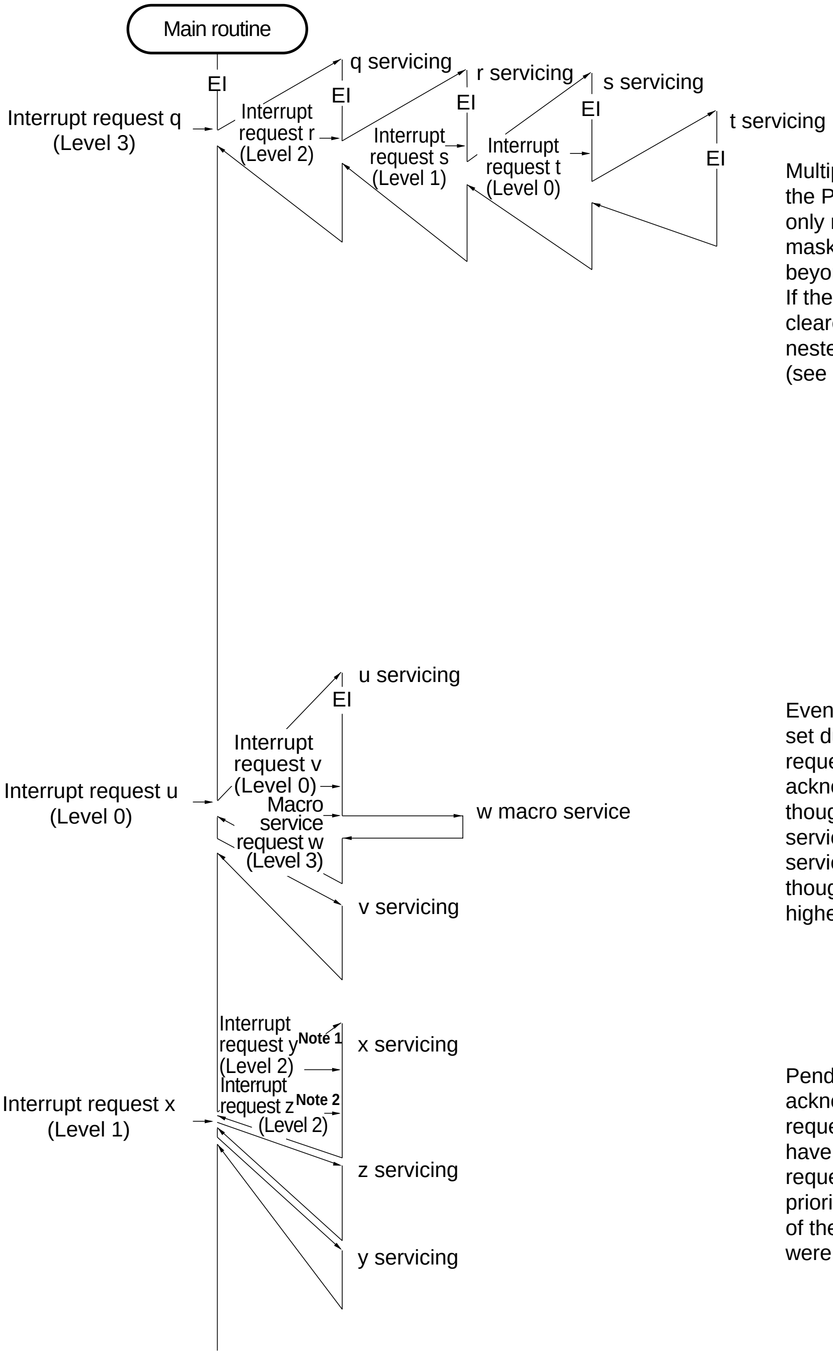


The macro service request is serviced irrespective of interrupt enabling/disabling and priority.

The interrupt request is held peding since it has a lower priority than interrupt request k. Interrupt request m generated after interrupt request l has a higher priority, and is therefore acknowledged first.

Since servicing of interrupt request n performed in the interrupt disabled state, interrupt requests o and p are held pending. After interrupt request n servicing, the pending interrupt requests are acknowledged. Although interrupt request o was generated first, interrupt request p has a higher priority and is therefore acknowledged first.

Figure 22-13. Examples of Servicing When Another Interrupt Request Is Generated during Interrupt Service (3/3)



Multiple acknowledgment of levels 3 to 0. If the PRSL bit of the IMC register is set (1), only macro service requests and non-maskable interrupts generate nesting beyond this.
If the PRSL bit of the IMC register is cleared (0), level 3 interrupts can also be nested during level 3 interrupt servicing (see Figure 22-15).

Even though the interrupt enabled state is set during servicing of level 0 interrupt request u, the interrupt request is not acknowledged but held pending even though its priority is 0. However, the macro service request is acknowledged and serviced irrespective of its level and even though there is a peding interrupt with a higher priority level.

Pending interrupt requests y and z are acknowledged after servicing of interrupt request x. As interrupt requests y and z have the same priority level, interrupt request z which has the higher default priority is acknowledged first, irrespective of the order in which the interrupt requests were generated.

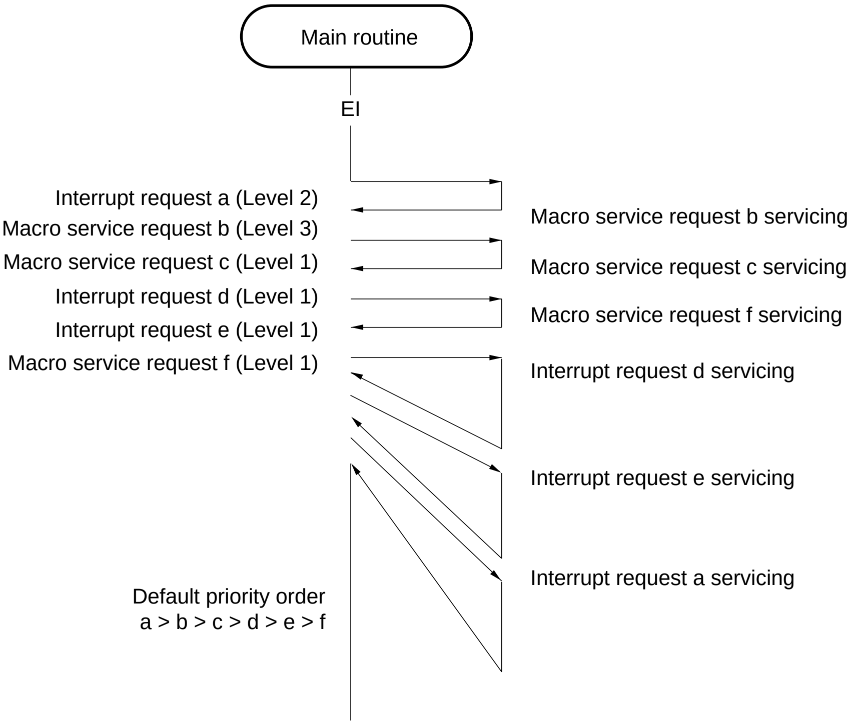
Notes 1. Low default priority

2. High default priority

Remarks 1. “a” to “z” in the figure above are arbitrary names used to differentiate between the interrupt requests and macro service requests.

2. High/low default priorities in the figure indicate the relative priority levels of the two interrupt requests.

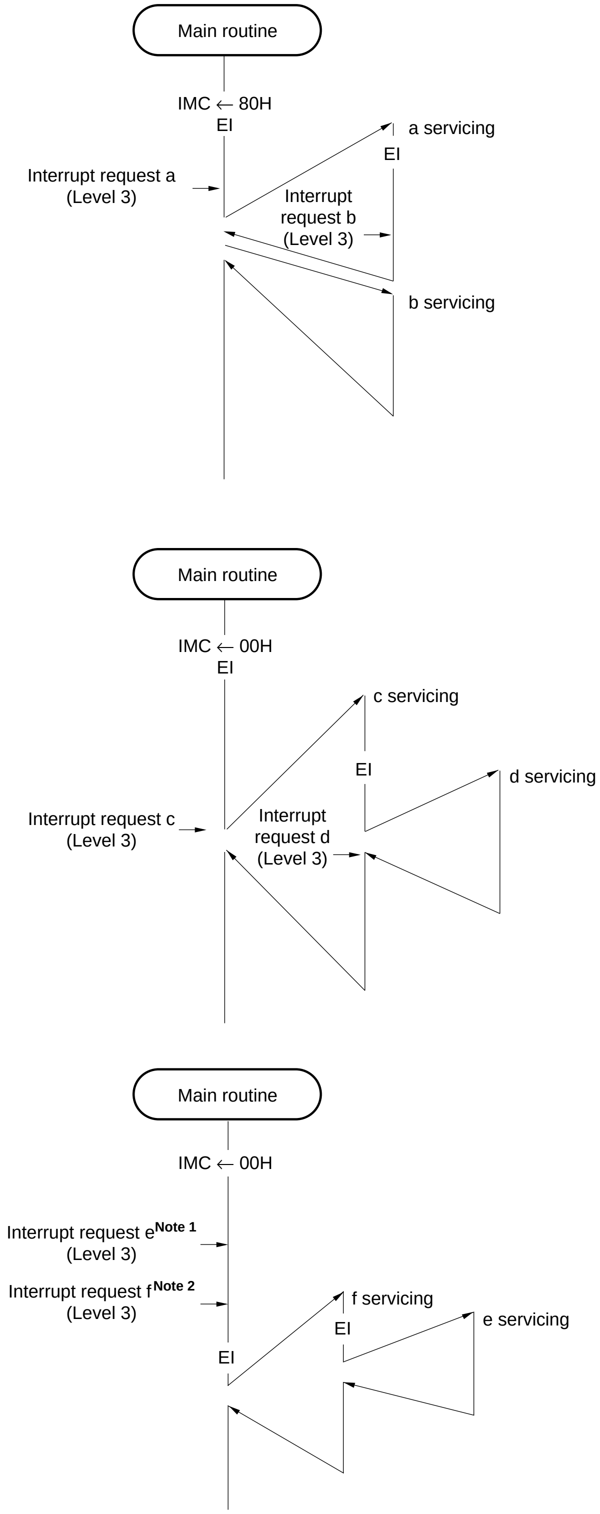
Figure 22-14. Examples of Servicing of Simultaneously Generated Interrupt Requests


Main routine
EI
Interrupt request a (Level 2)
Macro service request b (Level 3)
Macro service request c (Level 1)
Interrupt request d (Level 1)
Interrupt request e (Level 1)
Macro service request f (Level 1)
Macro service request b servicing
Macro service request c servicing
Macro service request f servicing
Interrupt request d servicing
Interrupt request e servicing
Interrupt request a servicing
Default priority order
a > b > c > d > e > f

- When requests are generated simultaneously, they are acknowledged in order starting with macro service.
- Macro service requests are acknowledged in default priority order (b/c/f) (not dependent upon the programmable priority order).
- As interrupt requests are acknowledged in high-to-low priority level order, d and e are acknowledged first.
- As d and e have the same prority level, the interrupt request with the higher default priority, d, is acknowledged first.

Remark "a" to "f" in the figure above are arbitrary names used to differentiate between the interrupt requests and macro service requests.

Figure 22-15. Differences in Level 3 Interrupt Acknowledgment According to IMC Register Setting



The PRSL bit of the IMC is set to 1, and nesting between level 3 interrupts is disabled.

Even though interrupts are enabled, interrupt request b is held pending since it has the same priority as interrupt request a.

The PRSL bit of the IMC is set to 0, so that a level 3 interrupt is acknowledged even during level 3 interrupt servicing (nesting is possible).

Since level 3 interrupt request c is being serviced in the interrupt enabled state and PRSL = 0, interrupt request d, which is also level 3, is acknowledged.

As interrupt request 3 and f are both of the same level, the one with the higher default priority, f, is acknowledged first.
When the interrupt enabled state is set during servicing of interrupt request f, pending interrupt request e is acknowledged since PRSL = 0.

Notes 1. Low default priority

2. High default priority

Remarks 1. "a" to "f" in the figure above are arbitrary names used to differentiate between the interrupt requests and macro service requests.

2. High/low default priorities in the figure indicate the relative priority levels of the two interrupt requests.

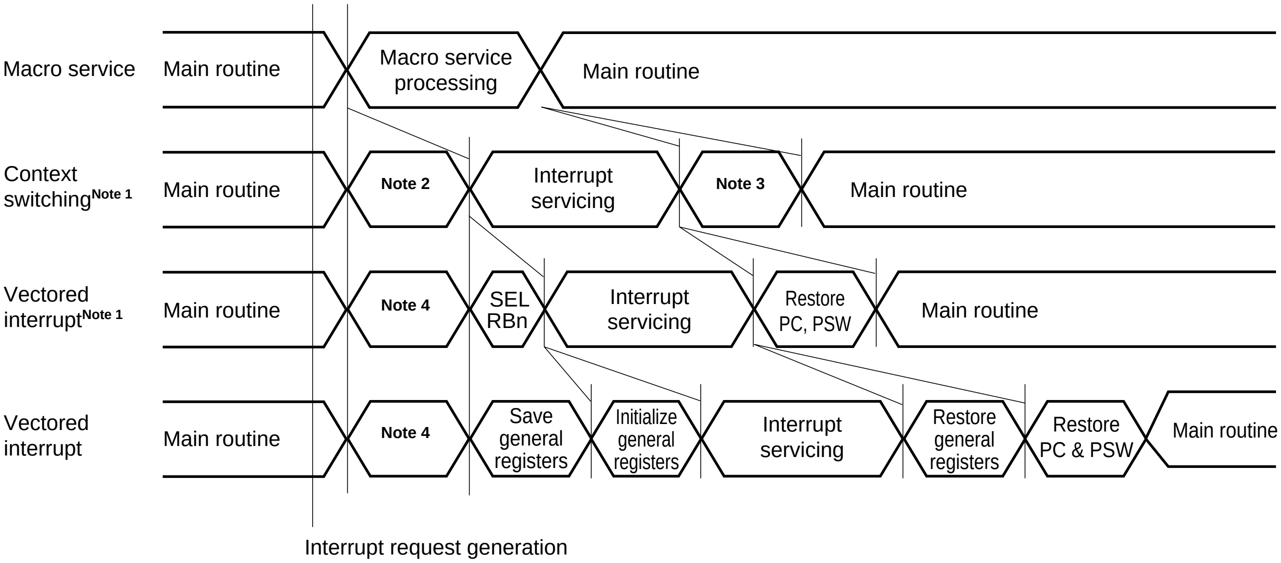
22.8 Macro Service Function

22.8.1 Outline of macro service function

Macro service is one method of servicing interrupts. With a normal interrupt, the program counter (PC) and program status word (PSW) are saved, and the start address of the interrupt service program is loaded into the PC, but with macro service, different processing (mainly data transfers) is performed instead of this processing. This enables interrupt requests to be responded to quickly, and moreover, since transfer processing is faster than processing by a program, the processing time can also be reduced.

Also, since a vectored interrupt is generated after processing has been performed the specified number of times, another advantage is that vectored interrupt programs can be simplified.

Figure 22-16. Differences between Vectored Interrupt and Macro Service Processing



Notes 1. When register bank switching is used, and an initial value has been set in the register beforehand

2. Register bank switching by context switching, saving of PC and PSW

3. Register bank, PC and PSW restoration by context switching

4. PC and PSW saved to the stack, vector address loaded into PC

22.8.2 Types of macro service

Macro service can be used with the 26 kinds of interrupts shown in Table 22-6. There are four kinds of operation, which can be used to suit the application.

Table 22-6. Interrupts for Which Macro Service Can Be Used

Default Priority	Interrupt Request Generation Source	Generating Unit	Macro Service Control Word Address
0	INTP0 (Pin input edge detection)	Edge detection	0FE06H
1	INTP1 (Pin input edge detection)		0FE08H
2	INTP2/INTTM41 (Pin input edge detection/TM4-CR41 match signal)	Edge detection/TM4	0FE0AH
3	INTP3 (Pin input edge detection)	Edge detection	0FE0CH
4	INTP4 (Pin input edge detection)		0FE0EH
5	INTP5/INTTM51 (Pin input edge detection/TM5-CR51 match signal)	Edge detection/TM5	0FE10H
6	INTP6 (Pin input edge detection)	Edge detection	0FE12H
7	INTTM00 (TM0-CR00 match signal)	TM0	0FE14H
8	INTTM01 (TM0-CR01 match signal)		0FE16H
9	INTTM10 (TM1-CR10 match signal)	TM1	0FE18H
10	INTTM11 (TM1-CR11 match signal)		0FE1AH
11	INTTM20 (TM2-CR20 match signal)	TM2	0FE1CH
12	INTTM21 (TM2-CR21 match signal)		0FE1EH
13	INTTM30 (TM3-CR30 match signal)	TM3	0FE20H
14	INTTM31 (TM3-CR31 match signal)		0FE22H
15	INTTM40 (TM4-CR40 match signal)	TM4	0FE24H
16	INTTM42 (TM4-CR42 match signal)		0FE26H
17	INTTM50 (TM5-CR50 match signal)	TM5	0FE28H
18	INTTM52 (TM5-CR52 match signal)		0FE2AH
19	INTTM6 (TM6-CR6 match signal)	TM6	0FE2CH
20	INTTM7 (TM7-CR7 match signal)	TM7	0FE2EH
21	INTSER1 (UART receive error)	UART	0FE30H
22	INTSR1 (UART receive completion)		0FE32H
23	INTST1 (UART transmission completion)		0FE34H
24	INTCS10 (clocked serial interface transmit/receive completion)	CSI	0FE36H
25	INTAD (A/D conversion completion)	A/D converter	0FE38H

Remarks 1. The default priority is a fixed number. It indicates the priority when multiple interrupt requests with the same specified priority are generated simultaneously.

2.

TM	: timer/counter
CR (00, 01, 40, 41, 42, 50)	: capture/compare registers
CR (10, 11, 20, 21, 30, 31, 52, 6, 7)	: compare registers
UART	: asynchronous serial interface
CSI	: clocked serial interface

There are four kinds of macro service, as shown below.

(1) Type A

One byte or one word of data is transferred between a special function register (SFR) and memory each time an interrupt request is generated, and a vectored interrupt request is generated when the specified number of transfers have been performed.

Memory that can be used in the transfers is limited to internal RAM addresses 0FE00H to 0FEFFH when the LOCATION 0 instruction is executed, and addresses 0FFE00H to 0FFEFFH when the LOCATION 0FH instruction is executed.

The specification method is simple and is suitable for low-volume, high-speed data transfers.

(2) Type B

As with type A, one byte or one word of data is transferred between a special function register (SFR) and memory each time an interrupt request is generated, and a vectored interrupt request is generated when the specified number of transfers have been performed.

The SFR and memory to be used in the transfers is specified by the macro service channel (the entire 1-Mbyte memory space can be used).

This is a general version of type A, suitable for large volumes of transfer data.

(3) Type C

Data is transferred from memory to two special function registers (SFR) each time an interrupt request is generated, and a vectored interrupt request is generated when the specified number of transfers have been performed.

With type C macro service, not only are data transfers performed to two locations in response to a single interrupt request, but it is also possible to add output data ring control and a function that automatically adds data to a compare register. The entire 1-Mbyte memory space can be used.

Type C is mainly used with the INTTM30, INTTM6, INTTM50 and INTTM7 interrupts, and is used for stepping motor control, etc., by macro service, with RTBL or RTBH and CR30, CR6, CR50, CR7, or CR1W used as the SFRs to which data is transferred.

Table 22-7. Examples of Main Uses for Type C

Interrupt	Output Data	Compare Register
INTTM30	RTBH0 or RTBL0	CR30
INTTM6	RTBH0	CR6
INTTM50	RTBH1 or RTBL1	CR50
INTTM7	RTBH1	CR7

(4) Counter mode

This mode is to decrement the macro service counter (MSC) when an interrupt occurs and is used to count the division operation of an interrupt and interrupt generation circuit.

When MSC is 0, a vectored interrupt can be generated.

To restart the macro service, MSC must be set again.

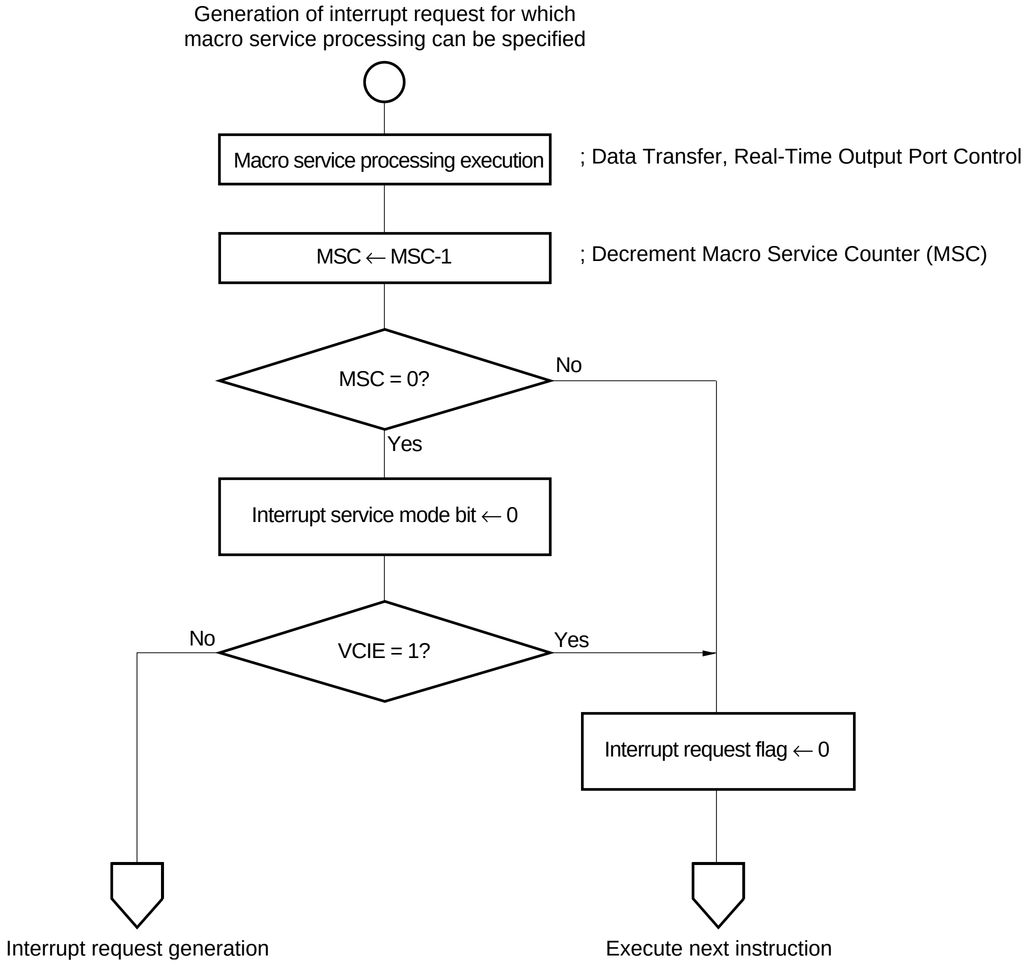
MSC is fixed to 16 bits and cannot be used as an 8-bit counter.

22.8.3 Basic macro service operation

Interrupt requests for which the macro service processing generated by the algorithm shown in Figure 22-10 can be specified are basically serviced in the sequence shown in Figure 22-17.

Interrupt requests for which macro service processing can be specified are not affected by the status of the IE flag, but are disabled by setting an interrupt mask flag in the interrupt mask register (MK0) to 1. Macro service processing can be executed in the interrupt disabled state and during execution of an interrupt service program.

Figure 22-17. Macro Service Processing Sequence


Generation of interrupt request for which macro service processing can be specified
Macro service processing execution
; Data Transfer, Real-Time Output Port Control
MSC ← MSC-1
; Decrement Macro Service Counter (MSC)
MSC = 0?
No
Yes
Interrupt service mode bit ← 0
VCIE = 1?
No
Yes
Interrupt request flag ← 0
Interrupt request generation
Execute next instruction

The macro service type and transfer direction are determined by the value set in the macro service control word mode register. Transfer processing is then performed using the macro service channel specified by the channel pointer according to the macro service type.

The macro service channel is memory that contains the macro service counter that records the number of transfers, the transfer destination and transfer source pointers, and data buffers, and can be located at any address in the range FE00H to FEFFH when the LOCATION 0 instruction is executed, or FFE00H to FFEFFH when the LOCATION 0FH instruction is executed.

22.8.4 Operation at end of macro service

In macro service, processing is performed the number of times specified during execution of another program. Macro service ends when the processing has been performed the specified number of times (when the macro service counter (MSC) reaches 0). Either of two operations may be performed at this point, as specified by the VCIE bit (bit 7) of the macro service mode register for each macro service.

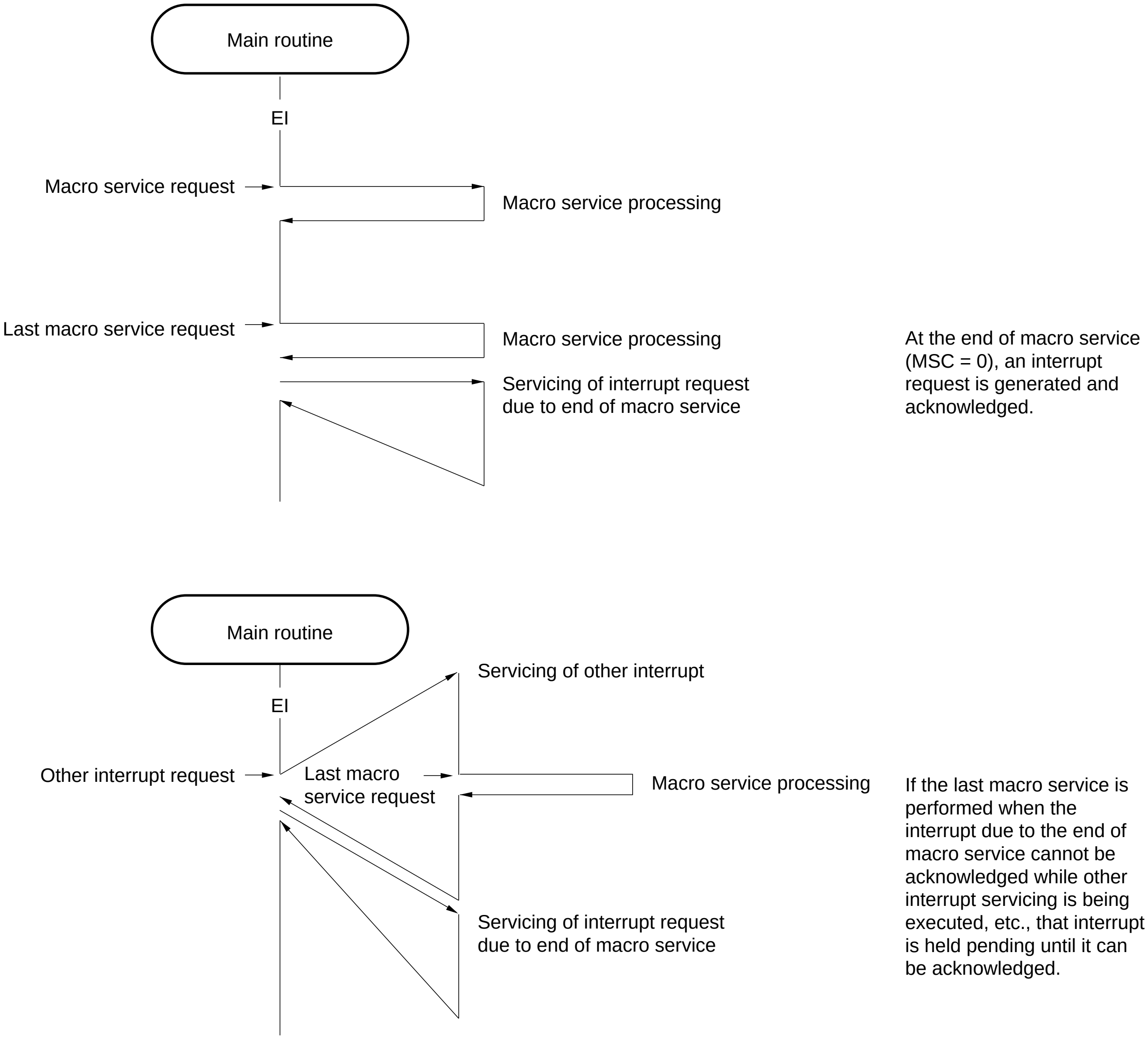
(1) When VCIE bit is 0

In this mode, an interrupt is generated as soon as the macro service ends. Figure 22-18 shows an example of macro service and interrupt acknowledgment operations when the VCIE bit is 0.

This mode is used when a series of operations end with the last macro service processing performed, for instance. It is mainly used in the following cases:

- Asynchronous serial interface receive data buffering (INTSR1)
- A/D conversion result fetch (INTAD)
- Compare register update as the result of a match between a timer register and the compare register (INTTM00, INTTM01, INTTM10, INTTM11, INTTM20, INTTM21, INTTM30, INTTM31, INTTM40, INTTM41, INTTM42, INTTM50, INTTM52, INTTM6, and INTTM7)

Figure 22-18. Operation at End of Macro Service When VCIE = 0


Main routine
EI
Macro service request
Macro service processing
Last macro service request
Macro service processing
Servicing of interrupt request due to end of macro service
Main routine
EI
Servicing of other interrupt
Other interrupt request
Last macro service request
Macro service processing
Servicing of interrupt request due to end of macro service

At the end of macro service (MSC = 0), an interrupt request is generated and acknowledged.

If the last macro service is performed when the interrupt due to the end of macro service cannot be acknowledged while other interrupt servicing is being executed, etc., that interrupt is held pending until it can be acknowledged.

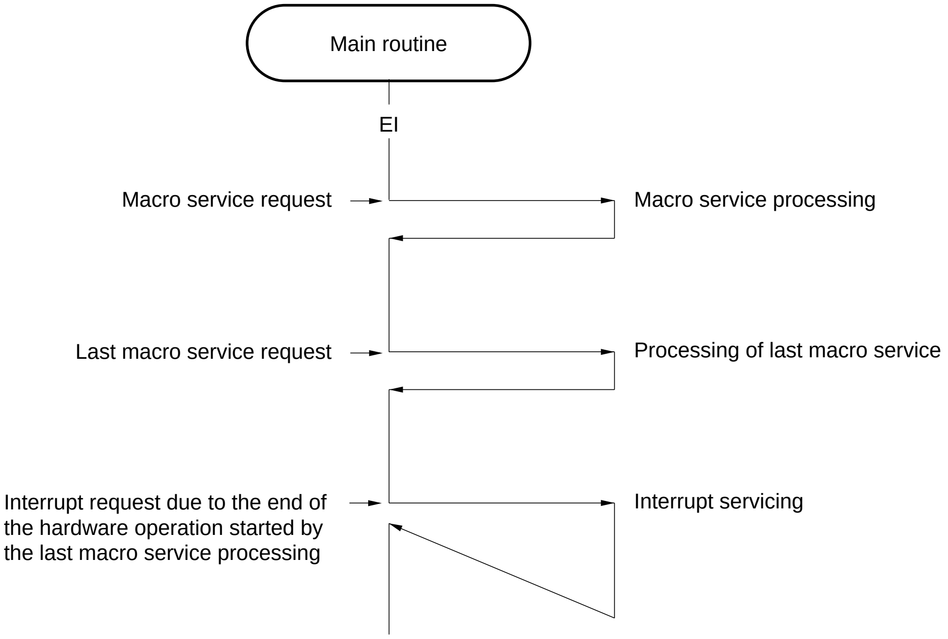
(2) **When VCIE bit is 1**

In this mode, an interrupt is not generated after macro service ends. Figure 22-19 shows an example of macro service and interrupt acknowledgment operations when the VCIE bit is 1.

This mode is used when the final operation is to be started by the last macro service processing performed, for instance. It is mainly used in the following cases:

- Clocked serial interface receive data transfers (INTCSI0)
- Asynchronous serial interface data transfers (INTST1)
- To stop a stepping motor in the case of controlling a stepping motor by means of macro service type C using the real-time output port and timer/counter (INTTM6, INTTM7, INTTM30, and INTTM50) .

Figure 22-19. Operation at End of Macro Service When VCIE = 1


Main routine
EI
Macro service request
Macro service processing
Last macro service request
Processing of last macro service
Interrupt request due to the end of the hardware operation started by the last macro service processing
Interrupt servicing

22.8.5 Macro service control registers

(1) Macro service control word

The μPD784955's macro service function is controlled by the macro service control mode register and macro service channel pointer. The macro service processing mode is set by means of the macro service mode register, and the macro service channel address is indicated by the macro service channel pointer.

The macro service mode register and macro service channel pointer are mapped onto the part of the internal RAM shown in Figure 22-20 for each macro service as the macro service control word.

When macro service processing is performed, the macro service mode register and channel pointer values corresponding to the interrupt requests for which macro service processing can be specified must be set beforehand.

★

Figure 22-20. Macro Service Control Word Format

Reserved word	Address		Cause
ADCHP	0FE39H	Channel Pointer	INTAD
ADMMD	0FE38H	Mode Register	
CSICHP0	0FE37H	Channel Pointer	INTCSI0
CSIMMD0	0FE36H	Mode Register	
STCHP1	0FE35H	Channel Pointer	INTST1
STMMD1	0FE34H	Mode Register	
SRCHP1	0FE33H	Channel Pointer	INTSR1
SRMMD1	0FE32H	Mode Register	
SERCHP1	0FE31H	Channel Pointer	INTSER1
SERMMD1	0FE30H	Mode Register	
TMCHP7	0FE2FH	Channel Pointer	INTTM7
TMMMD7	0FE2EH	Mode Register	
TMCHP6	0FE2DH	Channel Pointer	INTTM6
TMMMD6	0FE2CH	Mode Register	
TMCHP52	0FE2BH	Channel Pointer	INTTM52
TMMMD52	0FE2AH	Mode Register	
TMCHP50	0FE29H	Channel Pointer	INTTM50
TMMMD50	0FE28H	Mode Register	
TMCHP42	0FE27H	Channel Pointer	INTTM42
TMMMD42	0FE26H	Mode Register	
TMCHP40	0FE25H	Channel Pointer	INTTM40
TMMMD40	0FE24H	Mode Register	
TMCHP31	0FE23H	Channel Pointer	INTTM31
TMMMD31	0FE22H	Mode Register	
TMCHP30	0FE21H	Channel Pointer	INTTM30
TMMMD30	0FE20H	Mode Register	
TMCHP21	0FE1FH	Channel Pointer	INTTM21
TMMMD21	0FE1EH	Mode Register	
TMCHP20	0FE1DH	Channel Pointer	INTTM20
TMMMD20	0FE1CH	Mode Register	
TMCHP11	0FE1BH	Channel Pointer	INTTM11
TMMMD11	0FE1AH	Mode Register	
TMCHP10	0FE19H	Channel Pointer	INTTM10
TMMMD10	0FE18H	Mode Register	
TMCHP01	0FE17H	Channel Pointer	INTTM01
TMMMD01	0FE16H	Mode Register	
TMCHP00	0FE15H	Channel Pointer	INTTM00
TMMMD00	0FE14H	Mode Register	
PCHP6	0FE13H	Channel Pointer	INTP6
PMMD6	0FE12H	Mode Register	
PCHP5	0FE11H	Channel Pointer	INTTM51/INTP5
PMMD5	0FE10H	Mode Register	
PCHP4	0FE0FH	Channel Pointer	INTP4
PMMD4	0FE0EH	Mode Register	
PCHP3	0FE0DH	Channel Pointer	INTP3
PMMD3	0FE0CH	Mode Register	
PCHP2	0FE0BH	Channel Pointer	INTTM41/INTP2
PMMD2	0FE0AH	Mode Register	
PCHP1	0FE09H	Channel Pointer	INTP1
PMMD1	0FE08H	Mode Register	
PCHP0	0FE07H	Channel Pointer	INTP0
PMMD0	0FE06H	Mode Register	

(2) Macro service mode register

The macro service mode register is an 8-bit register that specifies the macro service operation. This register is written in internal RAM as part of the macro service control word (refer to **Figure 22-20**).

The format of the macro service mode register is shown in Figure 22-21.

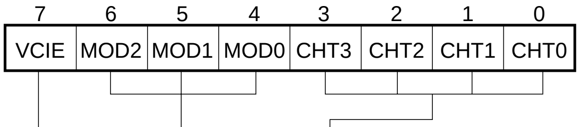
Figure 22-21. Macro Service Mode Register Format (1/2)

7	6	5	4	3	2	1	0
VCIE	MOD2	MOD1	MOD0	CHT3	CHT2	CHT1	CHT0

		CHT0	0	1		0	
		CHT1	0	0		0	
		CHT2	0	0		0	
		CHT3	0	0		1	
MOD2	MOD1	MOD0	Counter Mode	Type A		Type B	
0	0	0	Counter decrement	Data transfer direction Memory → SFR	Data size: 1 byte	Data transfer direction Memory → SFR	Data size: 1 byte
0	0	1		Data transfer direction SFR → memory		Data transfer direction SFR → memory	
0	1	0					
0	1	1					
1	0	0		Data transfer direction Memory → SFR	Data size: 2 bytes	Data transfer direction Memory → SFR	Data size: 2 bytes
1	0	1		Data transfer direction SFR → memory		Data transfer direction SFR → memory	
1	1	0					
1	1	1					

VCIE	Interrupt Request when MSC = 0
0	Generated
1	Not generated (next interrupt servicing is vectored interrupt)

Figure 22-21. Macro Service Mode Register Format (2/2)



<table>
<tr><td></td><td></td><td>CHT0</td><td>0</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td></td><td></td><td>CHT1</td><td>0</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td></td><td></td><td>CHT2</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td></td><td></td><td>CHT3</td><td>1</td><td>1</td><td>1</td><td>1</td></tr>
<tr><td rowspan="3">MOD2</td><td rowspan="3">MOD1</td><td rowspan="3">MOD0</td><td colspan="4">Type C</td></tr>
<tr><td colspan="2">Decrements MPD</td><td colspan="2">Increments MPD</td></tr>
<tr><td>Retains MPT</td><td>Decrements MPT</td><td>Retains MPT</td><td>Increments MPT</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td rowspan="4">Data size for timer specified by MPT: 1 byte</td><td rowspan="2">No automatic addition</td><td colspan="2">No ring control</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td colspan="2">Ring control</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td rowspan="2">Automatic addition</td><td colspan="2">No ring control</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td colspan="2">Ring control</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td rowspan="4">Data size for timer specified by MPT: 2 bytes</td><td rowspan="2">No automatic addition</td><td colspan="2">No ring control</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td colspan="2">Ring control</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td rowspan="2">Automatic addition</td><td colspan="2">No ring control</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td colspan="2">Ring control</td></tr>
</table>

VCIE	Interrupt Request when MSC = 0
0	Generated
1	Not generated (next interrupt processing is vectored interrupt)

(3) Macro service channel pointer

The macro service channel pointer specifies the macro service channel address. The macro service channel can be located in the 256-byte space from FE00H to FEFFH when the LOCATION 0 instruction is executed, or FFE00H to FFEFFH when the LOCATION 0FH instruction is executed, and the high-order 16 bits of the address are fixed. Therefore, the low-order 8 bits of the data stored to the highest address of the macro service channel are set in the macro service channel pointer.

22.8.6 Macro service type A

(1) Operation

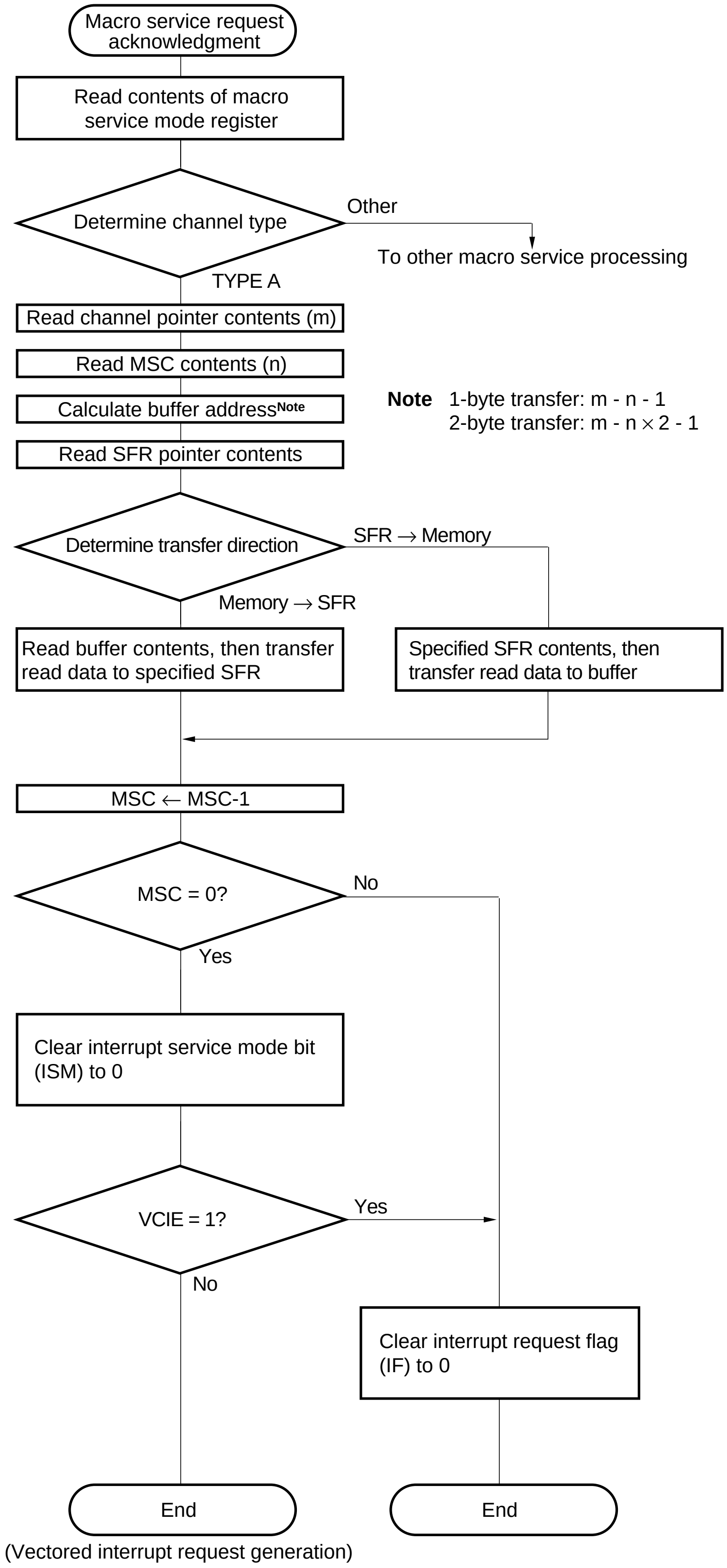
Data transfers are performed between buffer memory in the macro service channel and an SFR specified in the macro service channel.

With type A, the data transfer direction can be selected as memory-to-SFR or SFR-to-memory.

Data transfers are performed the number of times set beforehand in the macro service counter. One macro service processing transfers 8-bit or 16-bit data.

Type A macro service is useful when the amount of data to be transferred is small, as transfers can be performed at high speed.

Figure 22-22. Macro Service Data Transfer Processing Flow (Type A)

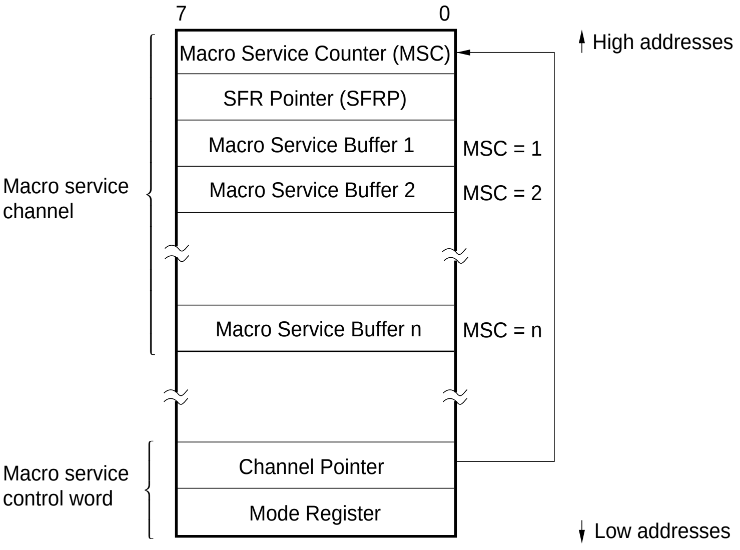

Macro service request acknowledgment
Read contents of macro service mode register
Determine channel type
Other
To other macro service processing
TYPE A
Read channel pointer contents (m)
Read MSC contents (n)
Calculate buffer addressNote
Note 1-byte transfer: m - n - 1
2-byte transfer: m - n × 2 - 1
Read SFR pointer contents
Determine transfer direction
SFR → Memory
Memory → SFR
Read buffer contents, then transfer read data to specified SFR
Specified SFR contents, then transfer read data to buffer
MSC ← MSC-1
MSC = 0?
No
Yes
Clear interrupt service mode bit (ISM) to 0
VCIE = 1?
Yes
No
Clear interrupt request flag (IF) to 0
End
End
(Vectored interrupt request generation)

(2) Macro service channel configuration

The channel pointer and 8-bit macro service counter (MSC) indicate the buffer address in internal RAM (FE00H to FEFFH when the LOCATION 0 instruction is executed, or FFE00H to FFEFFH when the LOCATION 0FH instruction is executed), which is the transfer source or transfer destination (refer to **Figure 22-23**). In the channel pointer, the low-order 8 bits of the address are written to the macro service counter in the macro service channel. The SFR involved with the access is specified by the SFR pointer (SFRP). The low-order 8 bits of the SFR address are written to the SFRP.

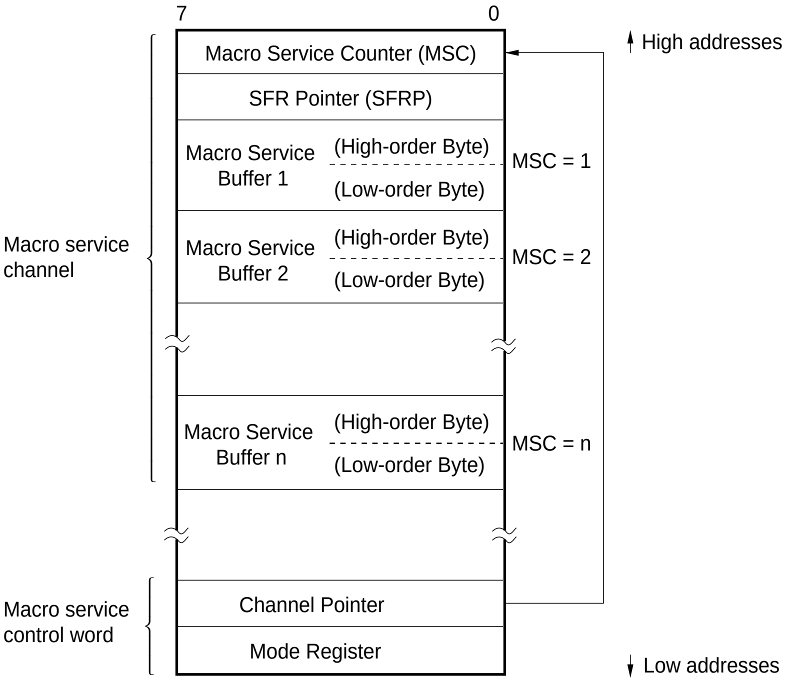
Figure 22-23. Type A Macro Service Channel

(a) 1-byte transfers



Macro service buffer address = (channel pointer) - (macro service counter) - 1

(b) 2-byte transfers

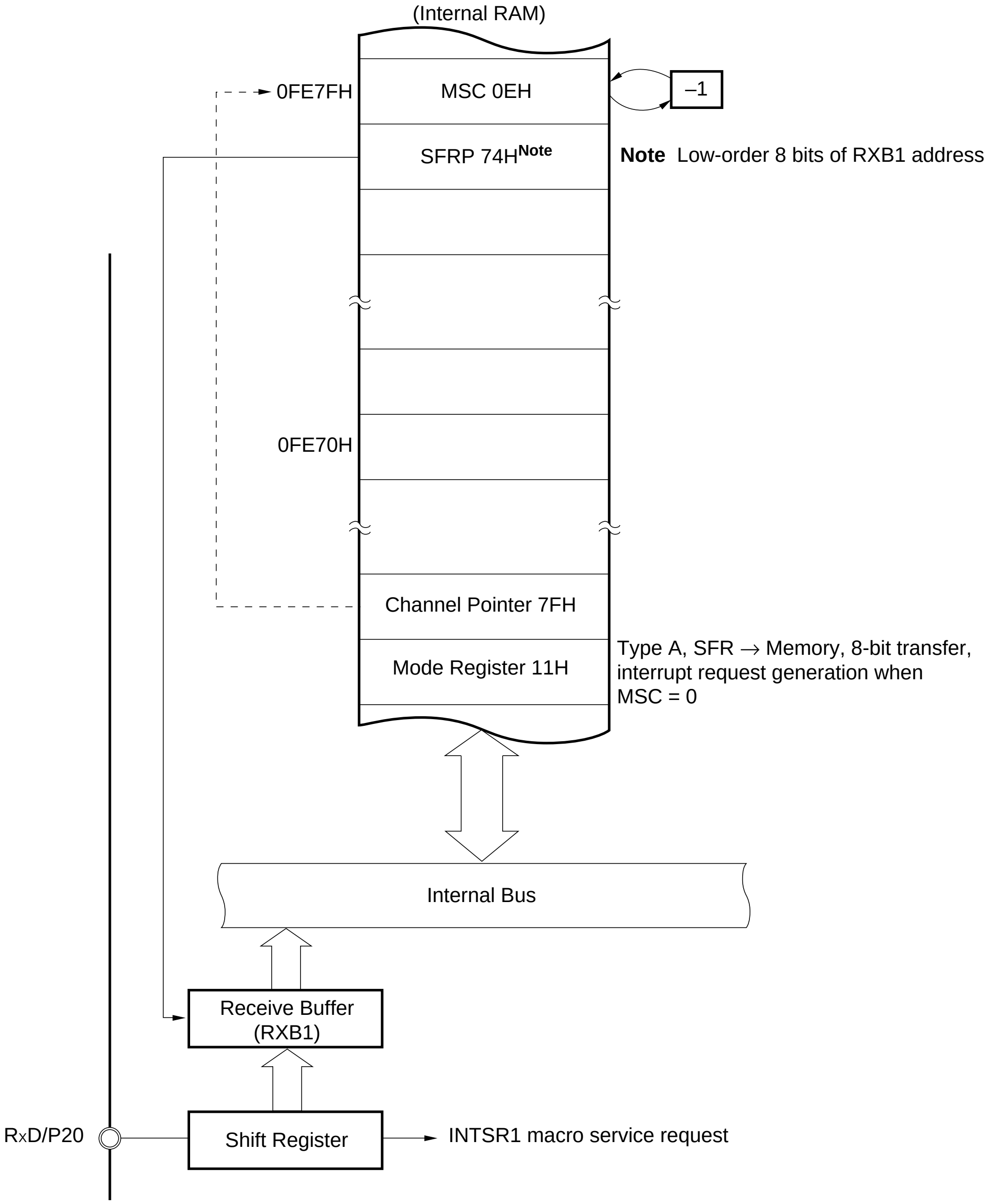


Macro service buffer address = (channel pointer) - (macro service counter) × 2 - 1

(3) Example of use of type A

An example is shown below in which data received via the asynchronous serial interface is transferred to a buffer area in on-chip RAM.

Figure 22-24. Asynchronous Serial Reception


(Internal RAM)
0FE7FH
MSC 0EH
–1
SFRP 74HNote
Note Low-order 8 bits of RXB1 address
0FE70H
Channel Pointer 7FH
Mode Register 11H
Type A, SFR → Memory, 8-bit transfer, interrupt request generation when MSC = 0
Internal Bus
Receive Buffer (RXB1)
RxD/P20
Shift Register
INTSR1 macro service request

Remark Addresses in the figure are the values when the LOCATION 0 instruction is executed. When the LOCATION 0FH instruction is executed, 0F0000H should be added to the values in the figure.

22.8.7 Macro service type B

(1) Operation

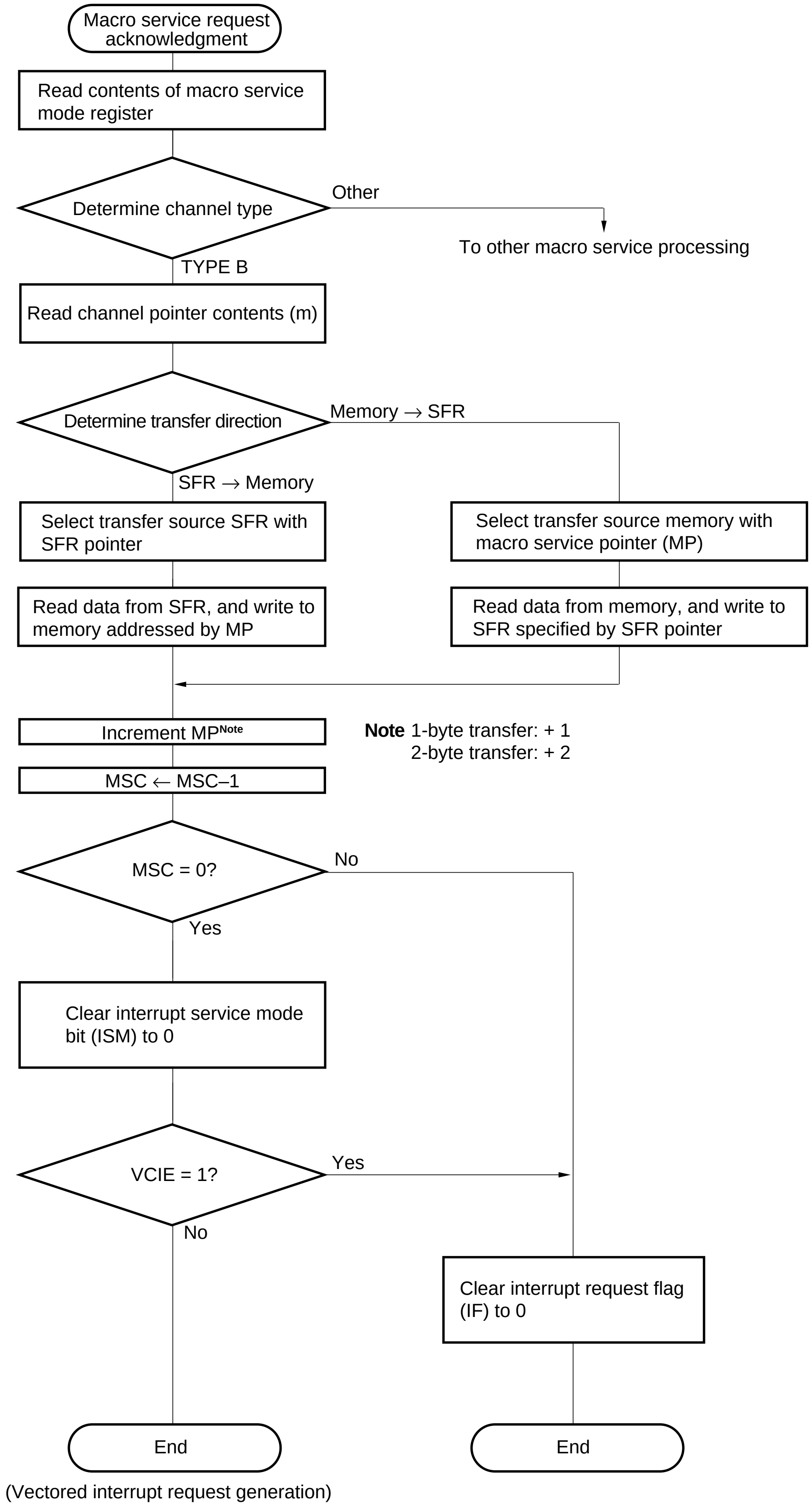
Data transfers are performed between a data area in memory and an SFR specified by the macro service channel.

With type B, the data transfer direction can be selected as memory-to-SFR or SFR-to-memory.

Data transfers are performed the number of times set beforehand in the macro service counter. One macro service processing transfers 8-bit or 16-bit data.

This type of macro service is macro service type A for general purposes and is ideal for processing a large amount of data because up to 64 Kbytes of data buffer area when 8-bit data is transferred or 1 Mbyte of data buffer area when 16-bit data is transferred can be set in any address space.

Figure 22-25. Macro Service Data Transfer Processing Flow (Type B)


Macro service request acknowledgment
Read contents of macro service mode register
Determine channel type
Other
To other macro service processing
TYPE B
Read channel pointer contents (m)
Determine transfer direction
Memory → SFR
SFR → Memory
Select transfer source SFR with SFR pointer
Select transfer source memory with macro service pointer (MP)
Read data from SFR, and write to memory addressed by MP
Read data from memory, and write to SFR specified by SFR pointer
Increment MPNote
Note 1-byte transfer: + 1
2-byte transfer: + 2
MSC ← MSC–1
MSC = 0?
No
Yes
Clear interrupt service mode bit (ISM) to 0
VCIE = 1?
Yes
No
Clear interrupt request flag (IF) to 0
End
End
(Vectored interrupt request generation)

(2) Macro service channel configuration

The macro service pointer (MP) indicates the data buffer area in the 1-Mbyte memory space that is the transfer destination or transfer source.

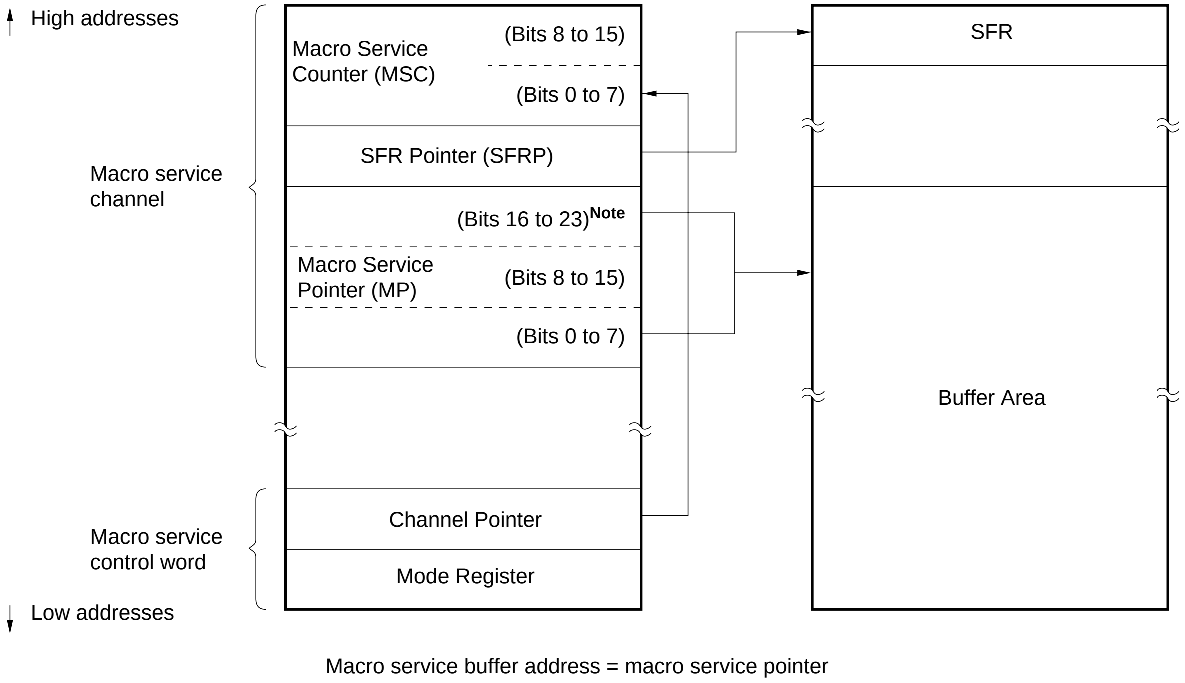
The low-order 8 bits of the SFR that is the transfer destination or transfer source is written to the SFR pointer (SFRP).

The macro service counter (MSC) is a 16-bit counter that specifies the number of data transfers.

The macro service channel that stores the MP, SFRP, and MSC is located in internal RAM space addresses 0FE00H to 0FEFFH when the LOCATION 0 instruction is executed, or 0FFE00H to 0FFEFFH when the LOCATION 0FH instruction is executed.

The macro service channel is indicated by the channel pointer as shown in Figure 22-26. In the channel pointer, the low-order 8 bits of the address are written to the macro service counter in the macro service channel.

Figure 22-26. Type B Macro Service Channel

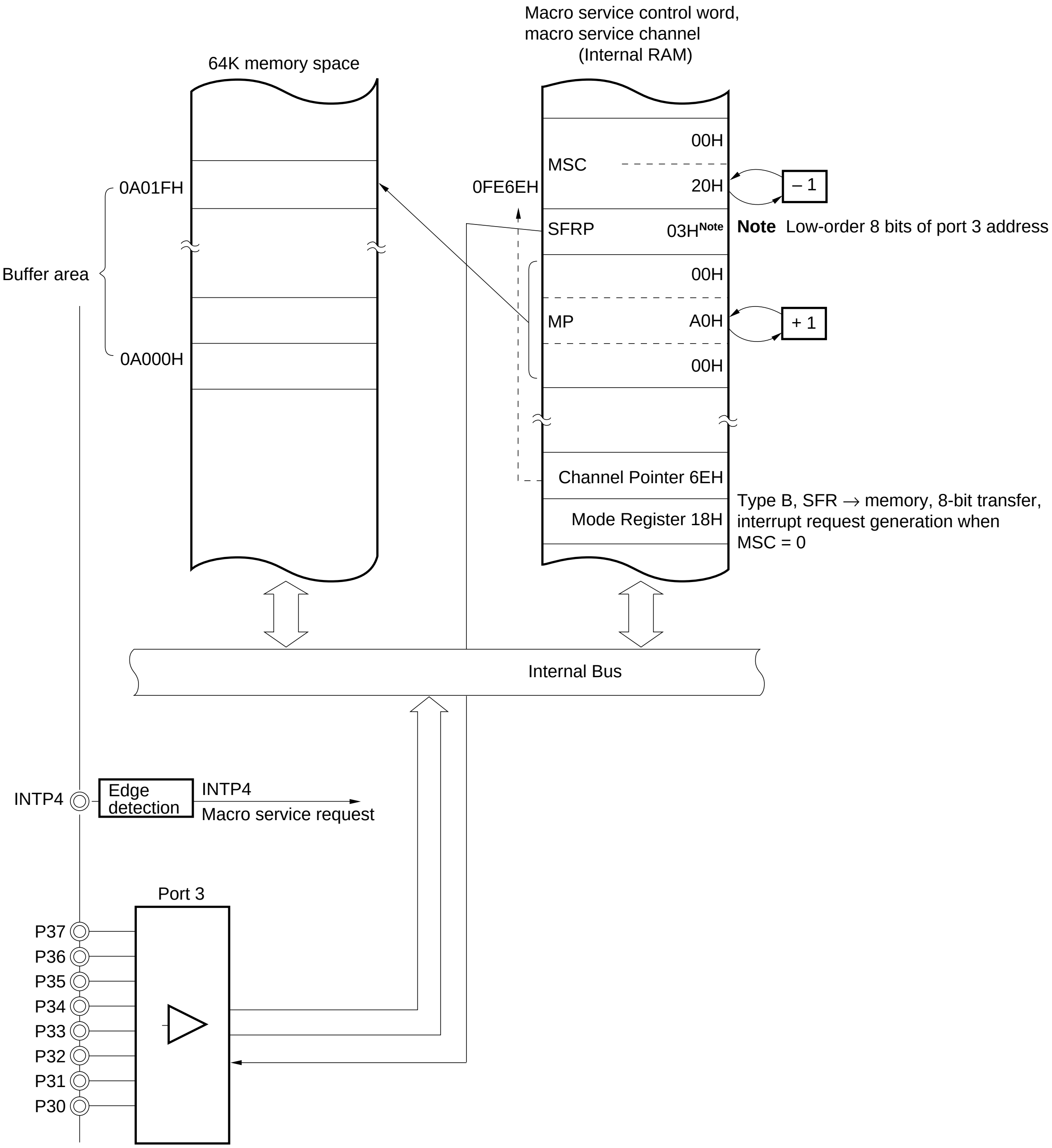


Note Bits 20 to 23 must be set to 0.

(3) Example of use of type B

An example is shown below in which parallel data is input from port 3 in synchronization with an external signal. The INTP4 external interrupt pin is used for synchronization with the external signal.

Figure 22-27. Parallel Data Input Synchronized with External Interrupts

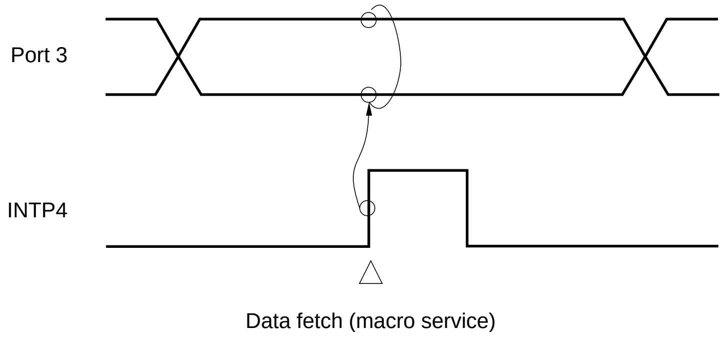


Note Low-order 8 bits of port 3 address

Type B, SFR → memory, 8-bit transfer, interrupt request generation when MSC = 0

Remark Macro service channel addresses in the figure are the values when the LOCATION 0 instruction is executed.
When the LOCATION 0FH instruction is executed, 0F0000H should be added to the values in the figure.

Figure 22-28. Parallel Data Input Timing



22.8.8 Macro service type C

(1) Operation

In type C macro service, data in the memory specified by the macro service channel is transferred to two SFRs, for timer use and data use, specified by the macro service channel in response to a single interrupt request (the SFRs can be freely selected). An 8-bit or 16-bit timer SFR can be selected.

In addition to the basic data transfers described above, type C macro service, the following functions can be added to type C macro service to reduce the size of the buffer area and alleviate the burden on software.

These specifications are made by using the mode register of the macro service control word.

(a) Updating of timer macro service pointer

It is possible to choose whether the timer macro service pointer (MPT) is to be kept as it is or incremented/ decremented. The MPT is incremented or decremented in the same direction as the macro service pointer (MPD) for data.

(b) Updating of data macro service pointer

It is possible to choose whether the data macro service pointer (MPD) is to be incremented or decremented.

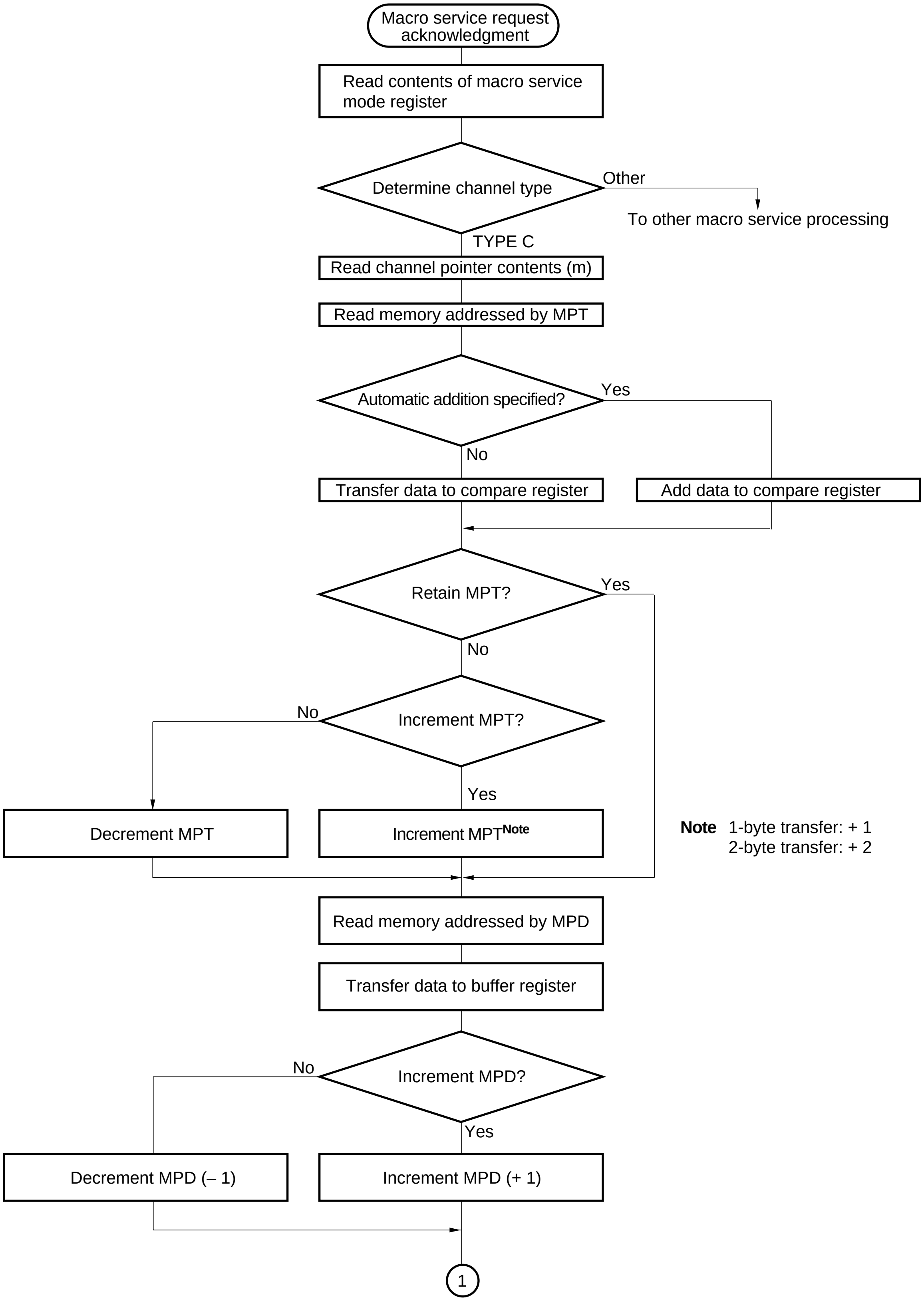
(c) Automatic addition

The current compare register value is added to the data addressed by the timer macro service pointer (MPT), and the result is transferred to the compare register. If automatic addition is not specified, the data addressed by the MPT is simply transferred to the compare register.

(d) Ring control

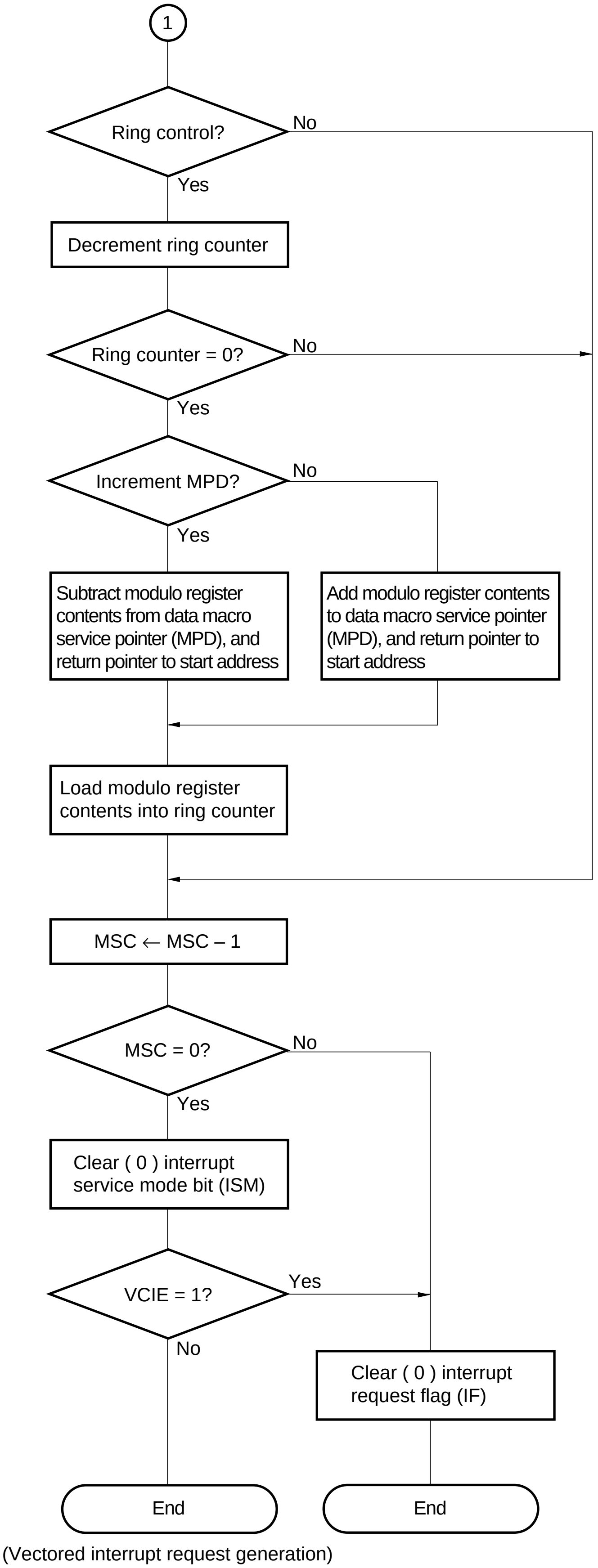
An output data pattern of the length specified beforehand is automatically output repeatedly.

Figure 22-29. Macro Service Data Transfer Processing Flow (Type C) (1/2)


Macro service request acknowledgment
Read contents of macro service mode register
Determine channel type
Other
To other macro service processing
TYPE C
Read channel pointer contents (m)
Read memory addressed by MPT
Automatic addition specified?
Yes
No
Transfer data to compare register
Add data to compare register
Retain MPT?
Yes
No
No
Increment MPT?
Yes
Decrement MPT
Increment MPTNote
Read memory addressed by MPD
Transfer data to buffer register
No
Increment MPD?
Yes
Decrement MPD (– 1)
Increment MPD (+ 1)
1

Note 1-byte transfer: + 1
2-byte transfer: + 2

Figure 22-29. Macro Service Data Transfer Processing Flow (Type C) (2/2)


1
Ring control?
No
Yes
Decrement ring counter
Ring counter = 0?
No
Yes
Increment MPD?
No
Yes
Subtract modulo register contents from data macro service pointer (MPD), and return pointer to start address
Add modulo register contents to data macro service pointer (MPD), and return pointer to start address
Load modulo register contents into ring counter
MSC ← MSC – 1
MSC = 0?
No
Yes
Clear (0) interrupt service mode bit (ISM)
VCIE = 1?
Yes
No
Clear (0) interrupt request flag (IF)
End
End
(Vectored interrupt request generation)

(2) Macro service channel configuration

There are two kinds of type C macro service channel, as shown in Figure 22-30.

The timer macro service pointer (MPT) mainly indicates the data buffer area in the 1-Mbyte memory space to be transferred or added to the timer/counter compare register.

The data macro service pointer (MPD) indicates the data buffer area in the 1-Mbyte memory space to be transferred to the real-time output port.

The modulo register (MR) specifies the number of repeat patterns when ring control is used.

The ring counter (RC) holds the step in the pattern when ring control is used. When initialization is performed, the same value as in the MR is normally set in this counter.

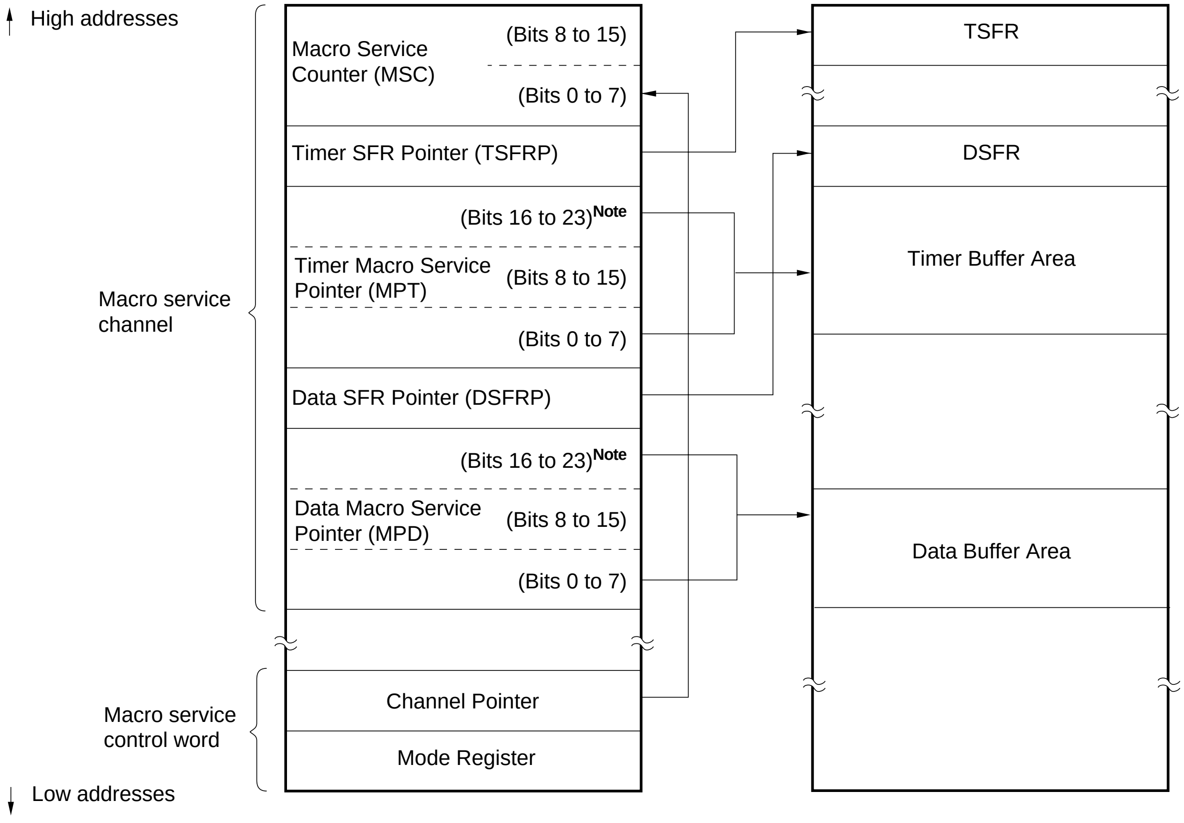
The macro service counter (MSC) is a 16-bit counter that specifies the number of data transfers.

The low-order 8 bits of the SFR that is the transfer destination is written to the timer SFR pointer (TSFRP) and data SFR pointer (DSFRP).

The macro service channel that stores these pointers and counters is located in internal RAM space addresses 0FE00H to 0FEFFH when the LOCATION 0 instruction is executed, or 0FFE00H to 0FFEFFH when the LOCATION 0FH instruction is executed. The macro service channel is indicated by the channel pointer as shown in Figure 22-30. In the channel pointer, the low-order 8 bits of the address are written to the macro service counter in the macro service channel.

Figure 22-30. Type C Macro Service Channel (1/2)

(a) No ring control

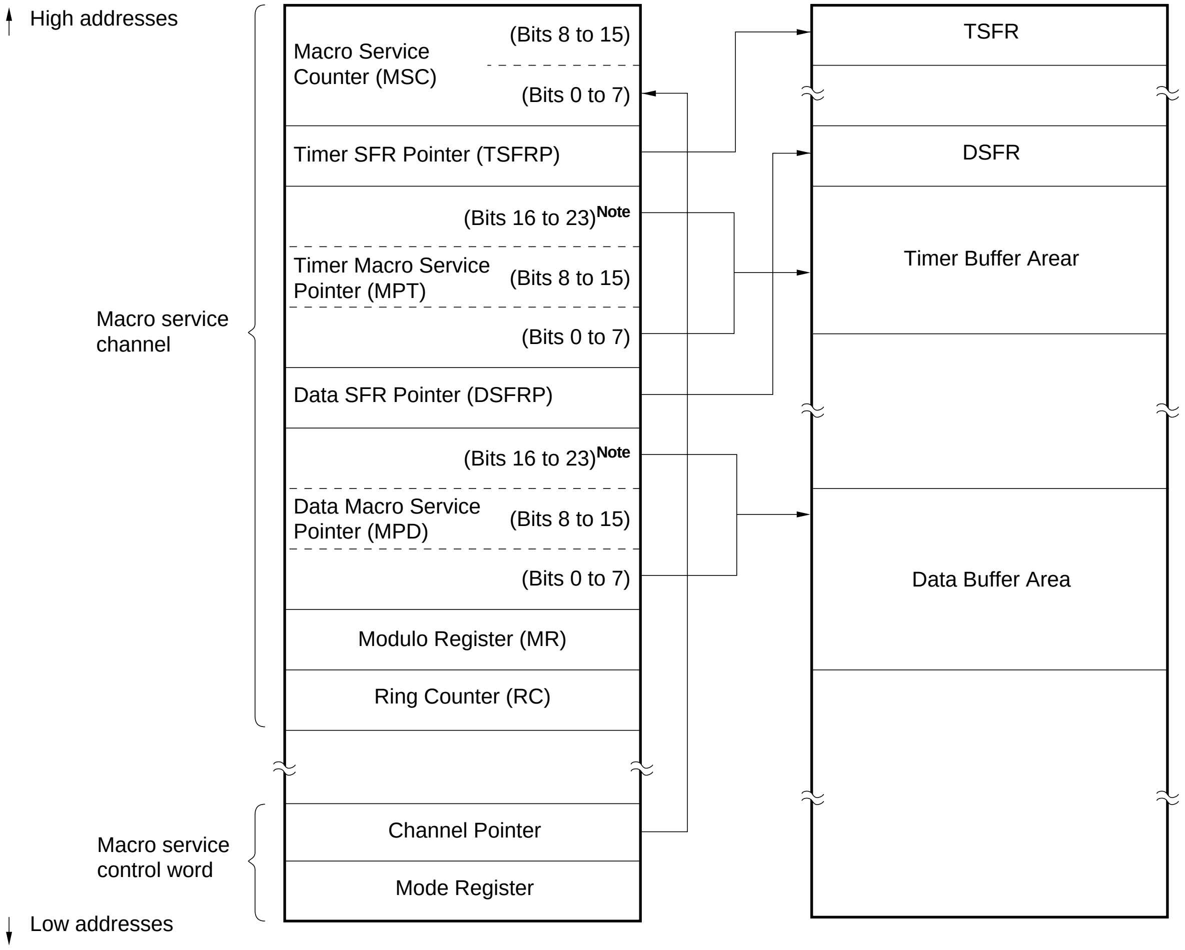


Macro service buffer address = macro service pointer

Note Bits 20 to 23 must be set to 0.

Figure 22-30. Type C Macro Service Channel (2/2)

(b) With ring control



Macro service buffer address = macro service pointer

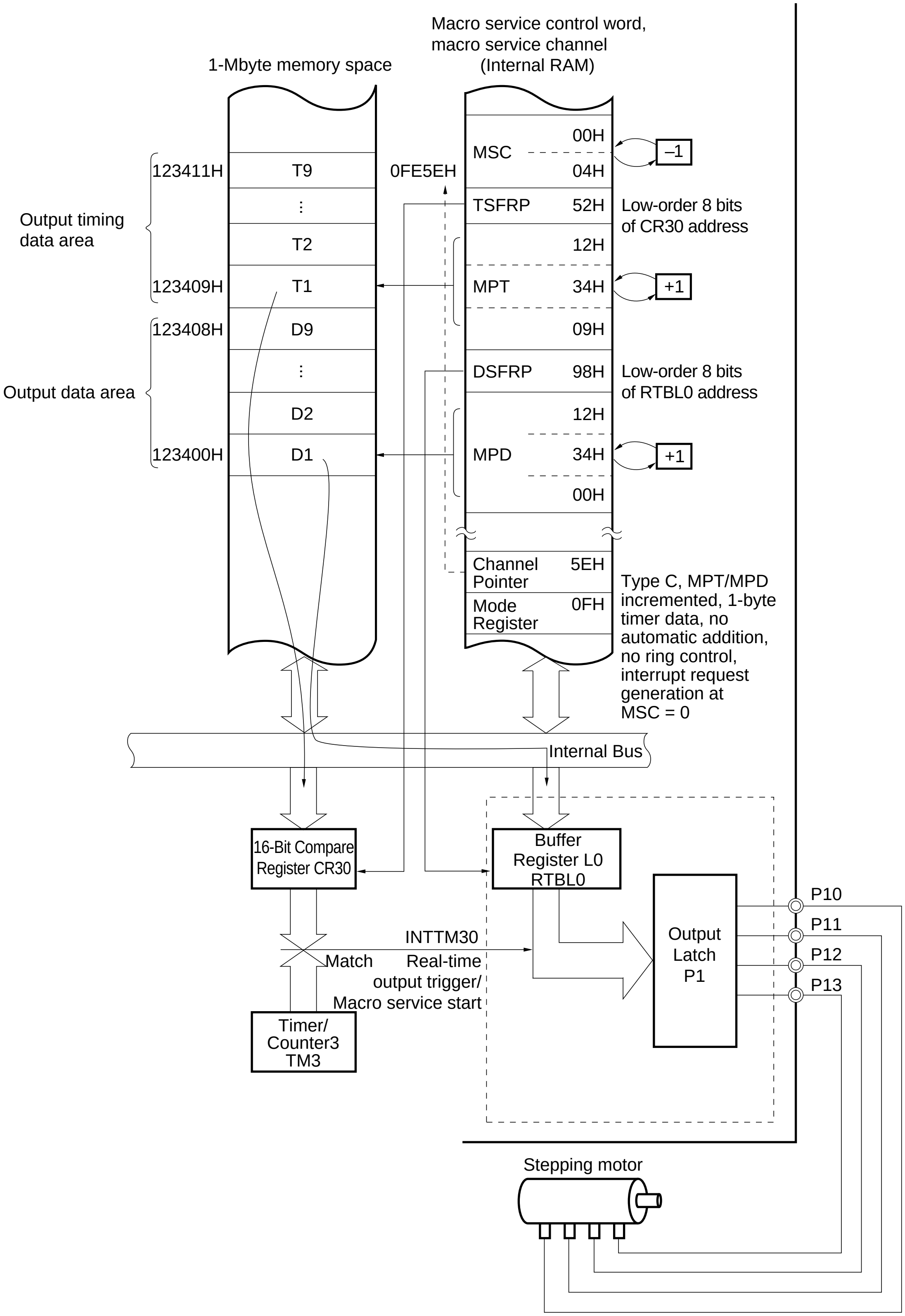
Note Bits 20 to 23 must be set to 0.

(3) Examples of use of type C

(a) Basic operation

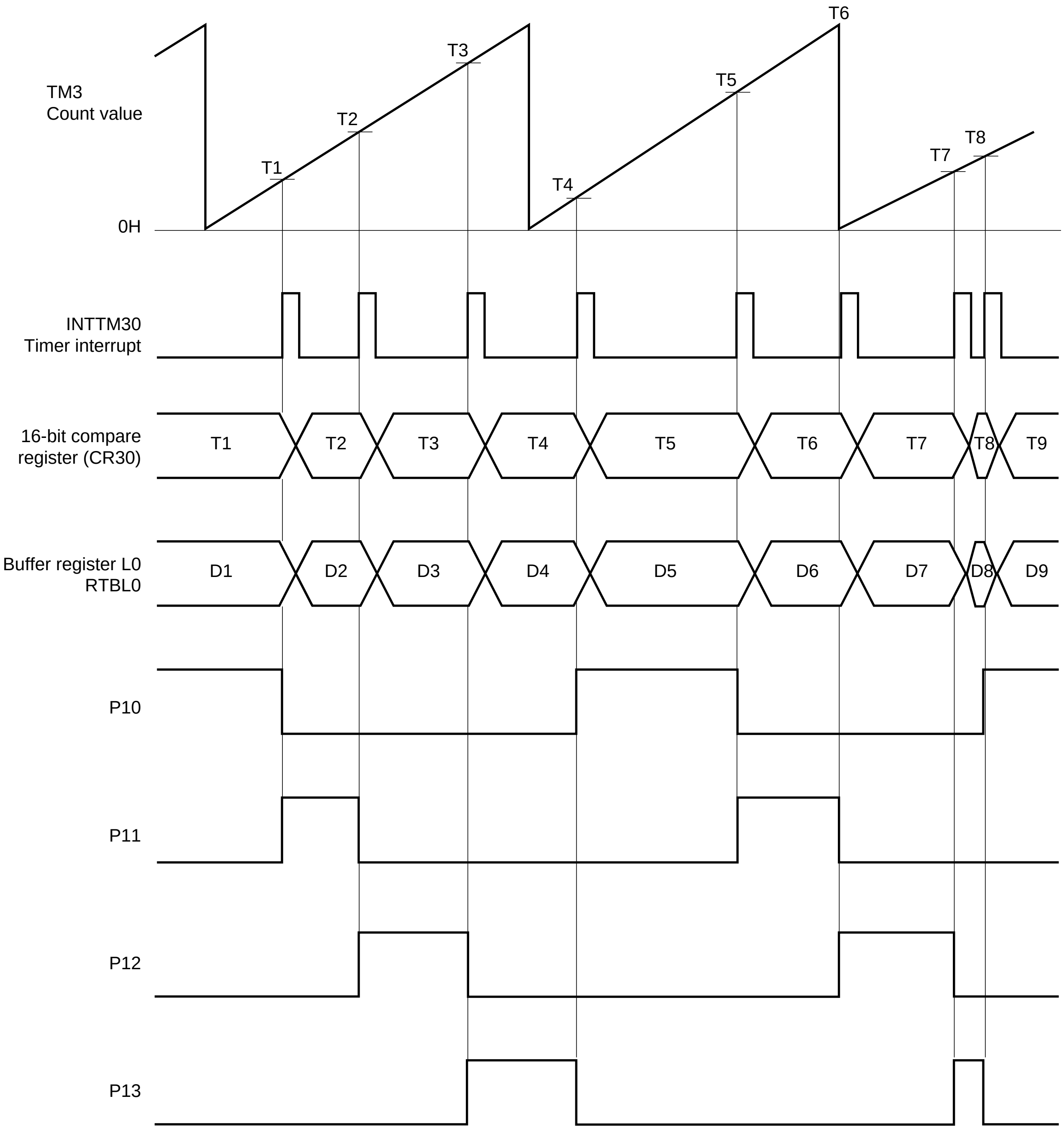
Here we show examples for direct control of output patterns and output intervals to real-time output port 0. Update data is transferred from the two data storage areas set in the 1-Mbyte space beforehand to the real-time output function buffer register 0 (RTBL0) and the 16-bit compare register 30 (CR30).

Figure 22-31. Stepping Motor Open Loop Control by Real-Time Output Port



Remark Internal RAM addresses in the figure are the values when the LOCATION 0 instruction is executed. When the LOCATION 0FH instruction is executed, 0F0000H should be added to the values in the figure.

Figure 22-32. Data Transfer Control Timing



(b) Examples of use of automatic addition control and ring control

(i) Automatic addition control

The output timing data (Δt) specified by the macro service pointer (MPT) is added to the contents of the compare register, and the result is written back to the compare register.

Use of this automatic addition control eliminates the need to calculate the compare register setting value in the program each time.

(ii) Ring control

With ring control, the predetermined output patterns is prepared for one cycle only, and the one-cycle data patterns are output repeatedly in order in ring form.

When ring control is used, only the output patterns for one cycle need be prepared, allowing the size of the data ROM area to be reduced.

The macro service counter (MSC) is decremented each time a data transfer is performed.

With ring control, too, an interrupt request is generated when MSC = 0.

When controlling a stepping motor, for example, the output patterns will vary depending on the configuration of the stepping motor concerned, and the phase excitation method (single-phase excitation, two-phase excitation, etc.), but repeat patterns are used in all cases. Examples of single-phase excitation and 1-2-phase excitation of a 4-phase stepping motor are shown in Figures 22-33 and 22-34.

Figure 22-33. Single-Phase Excitation of 4-Phase Stepping Motor

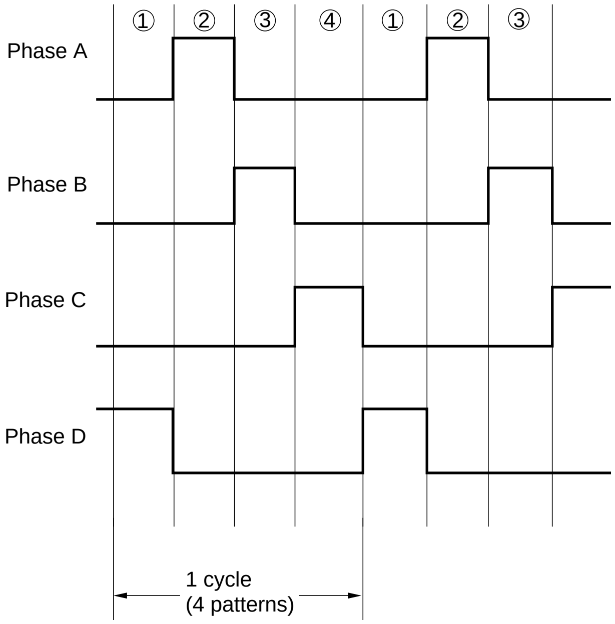


Figure 22-34. 1-2-Phase Excitation of 4-Phase Stepping Motor

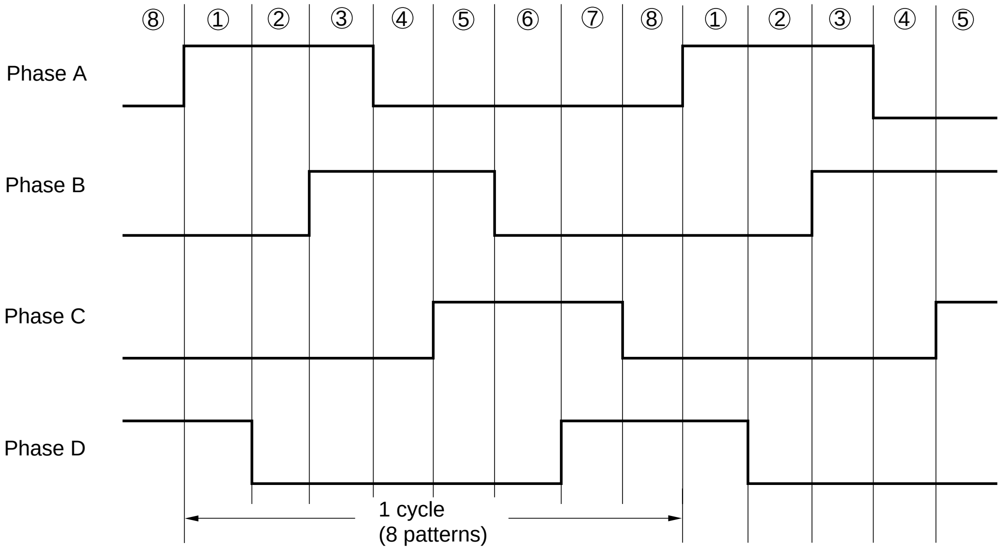
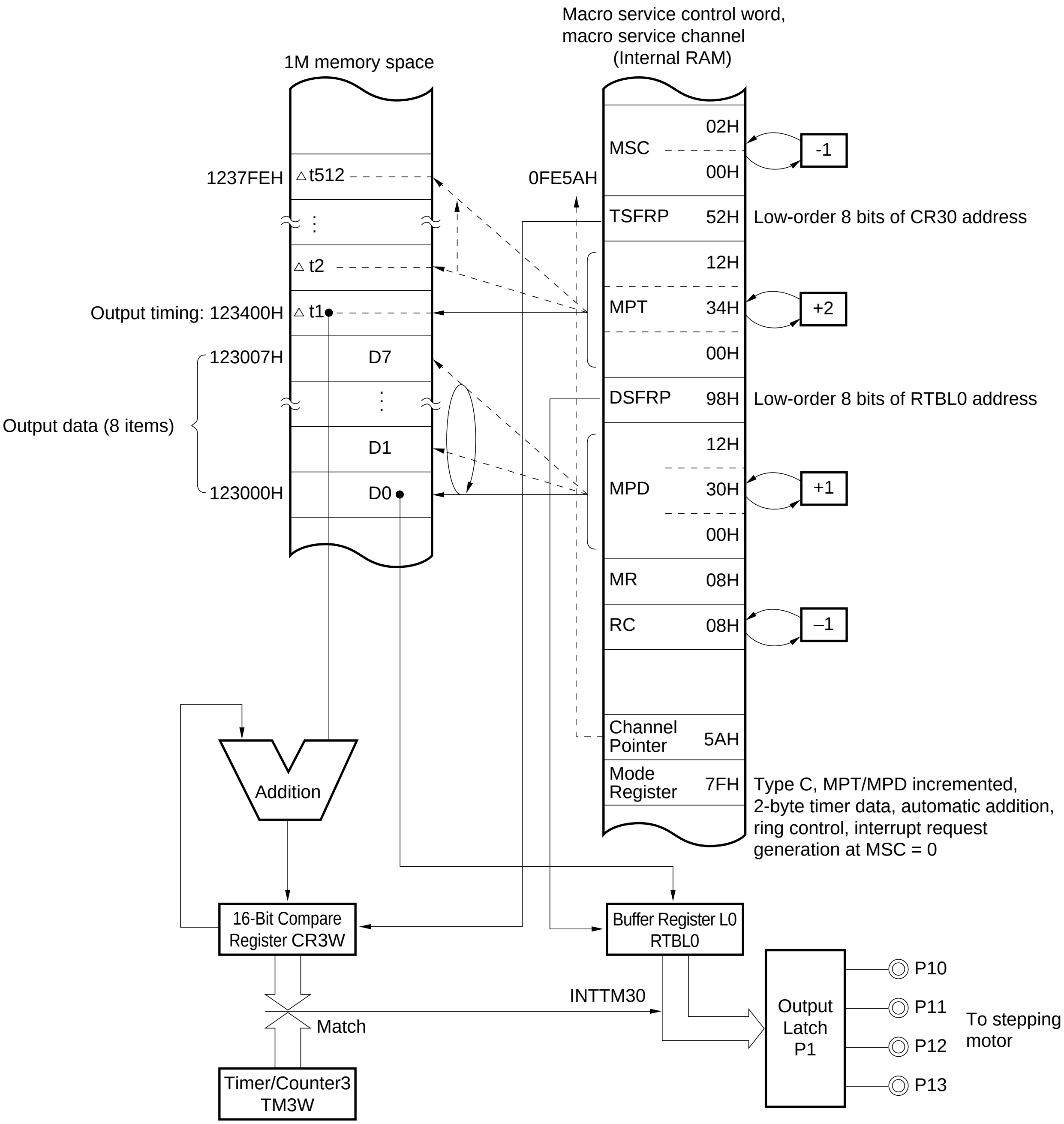


Figure 22-35. Automatic Addition Control + Ring Control Block Diagram 1 (When Output Timing Varies with 1-2-Phase Excitation)



Remark Internal RAM addresses in the figure are the values when the LOCATION 0 instruction is executed. When the LOCATION 0FH instruction is executed, 0F0000H should be added to the values in the figure.

Figure 22-36. Automatic Addition Control + Ring Control Timing Diagram 1 (When Output Timing Varies with 1-2-Phase Excitation)

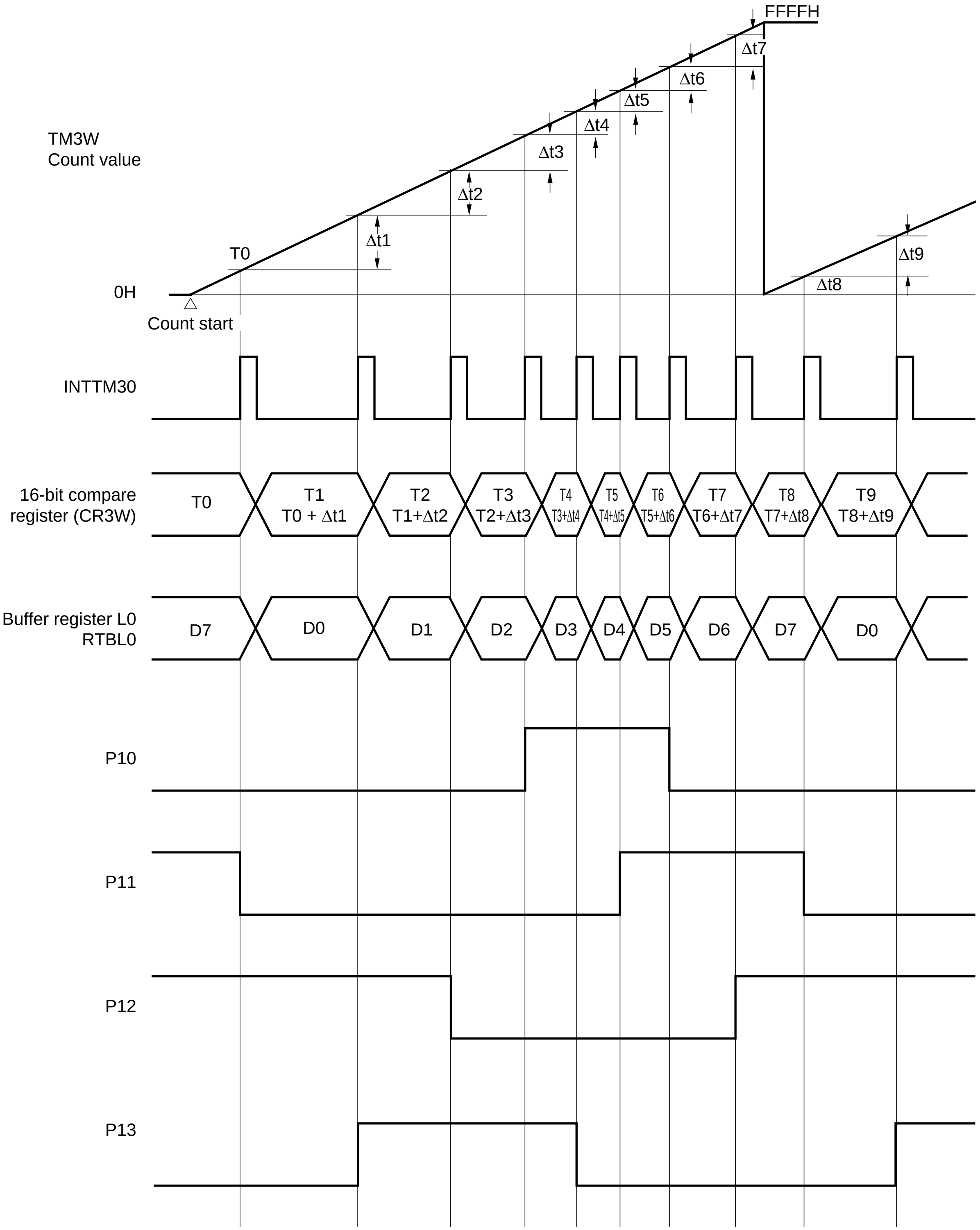
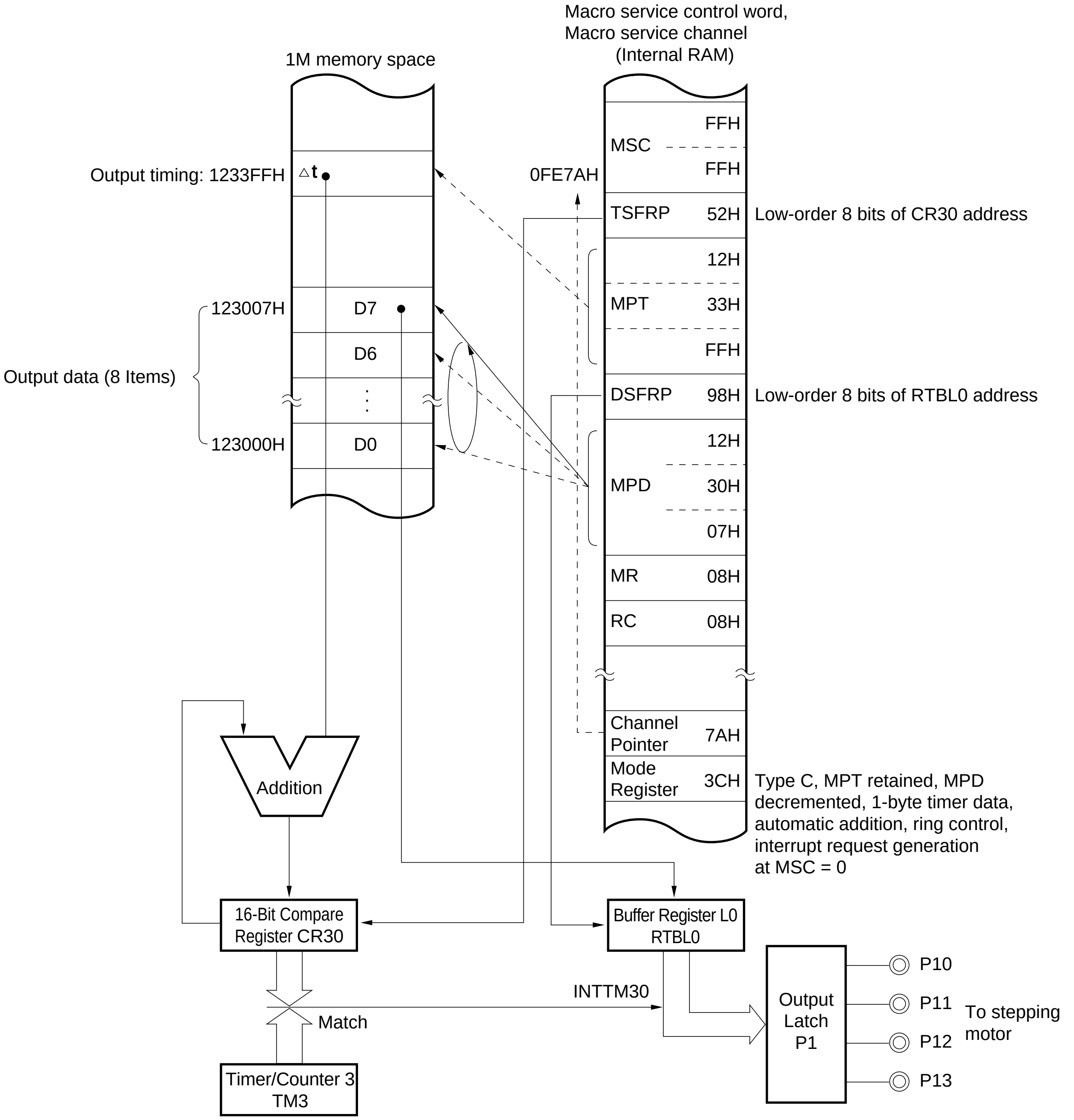
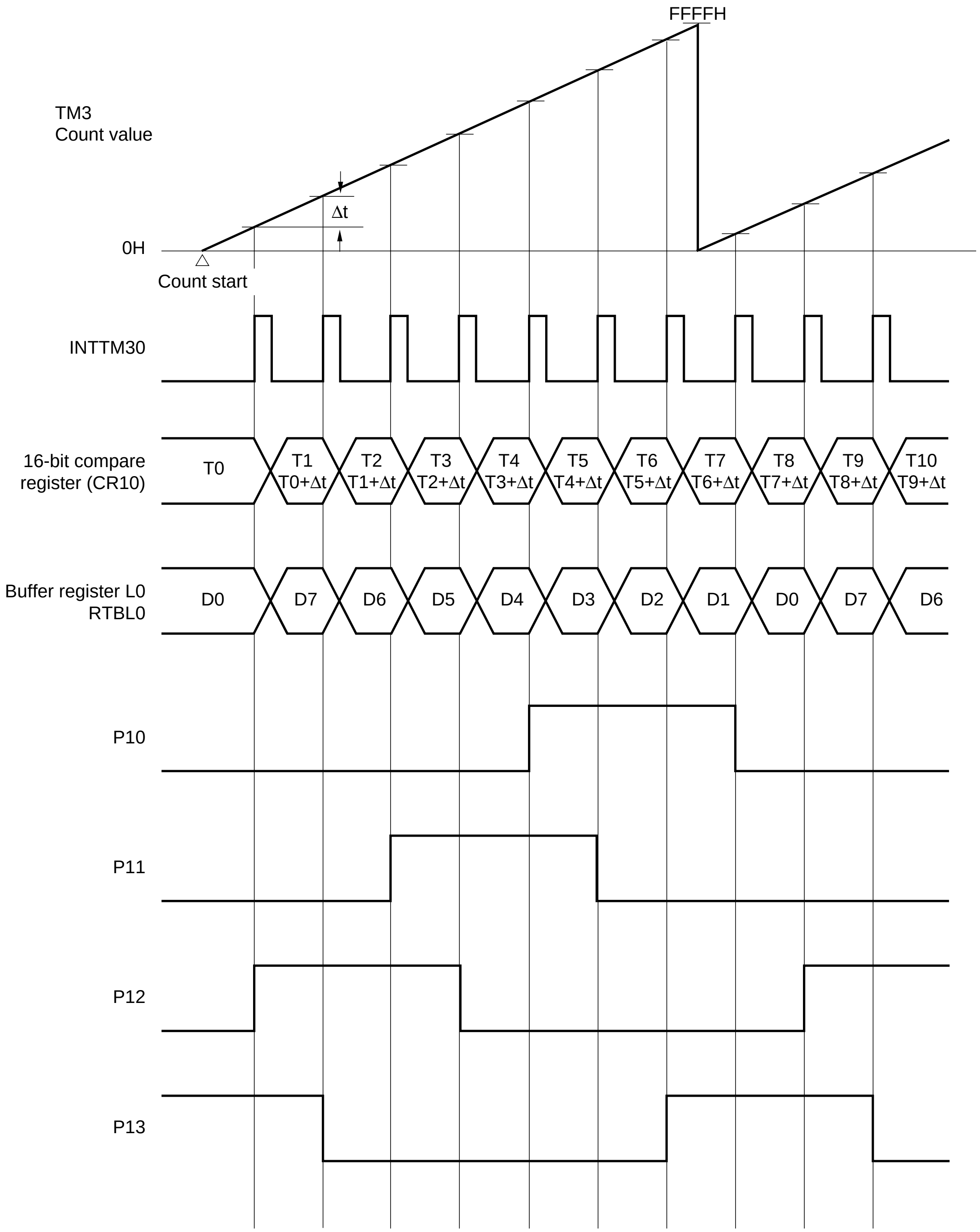


Figure 22-37. Automatic Addition Control + Ring Control Block Diagram 2 (1-2-Phase Excitation Constant-Velocity Operation)



Remark Internal RAM addresses in the figure are the values when the LOCATION 0 instruction is executed. When the LOCATION 0FH instruction is executed, 0F0000H should be added to the values in the figure.

Figure 22-38. Automatic Addition Control + Ring Control Timing Diagram 2 (1-2-Phase Excitation Constant-Velocity Operation)



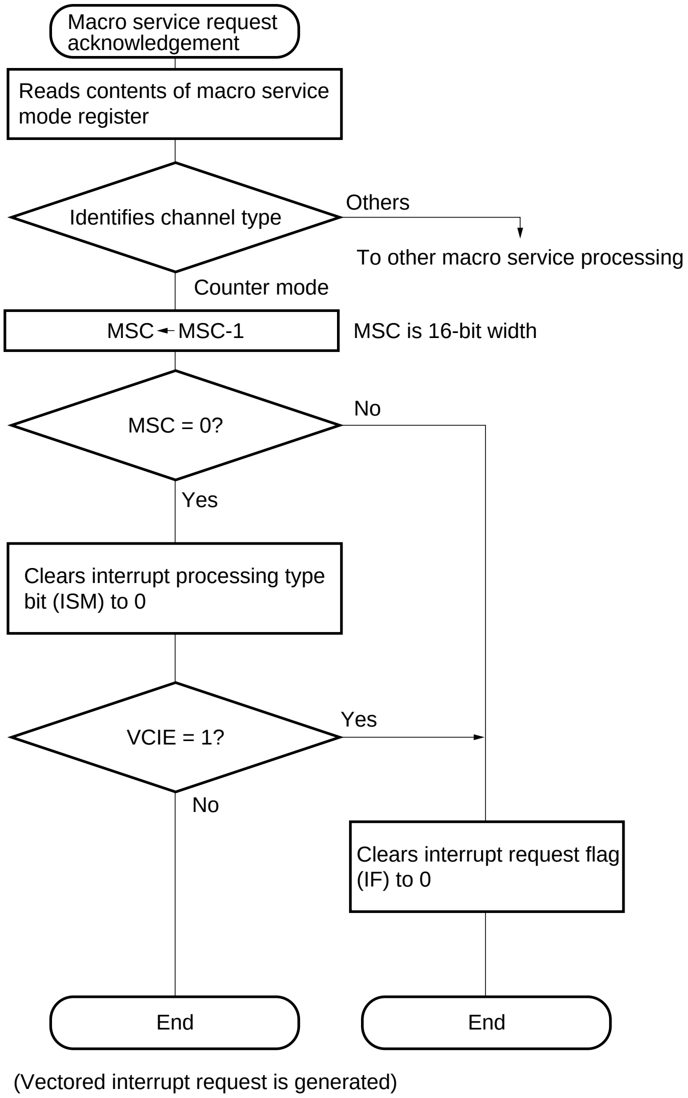
22.8.9 Counter mode

(1) Operation

MSC is decremented the number of times set in advance to the macro service counter (MSC).

Because the number of times an interrupt occurs can be counted, this function can be used as an event counter where the interrupt generation cycle is long.

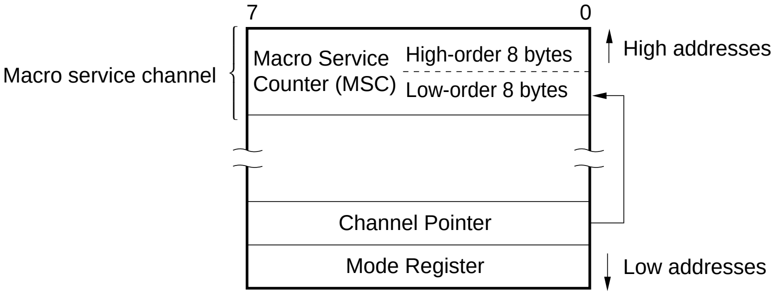
Figure 22-39. Macro Service Data Transfer Processing Flow (Counter Mode)


Macro service request acknowledgement
Reads contents of macro service mode register
Identifies channel type
Others
To other macro service processing
Counter mode
MSC ← MSC-1
MSC is 16-bit width
MSC = 0?
No
Yes
Clears interrupt processing type bit (ISM) to 0
VCIE = 1?
Yes
No
Clears interrupt request flag (IF) to 0
End
End
(Vectored interrupt request is generated)

(2) Configuration of macro service channel

The macro service channel consists of only a 16-bit macro service counter (MSC). The low-order 8 bits of the address of the MSC are written to the channel pointer.

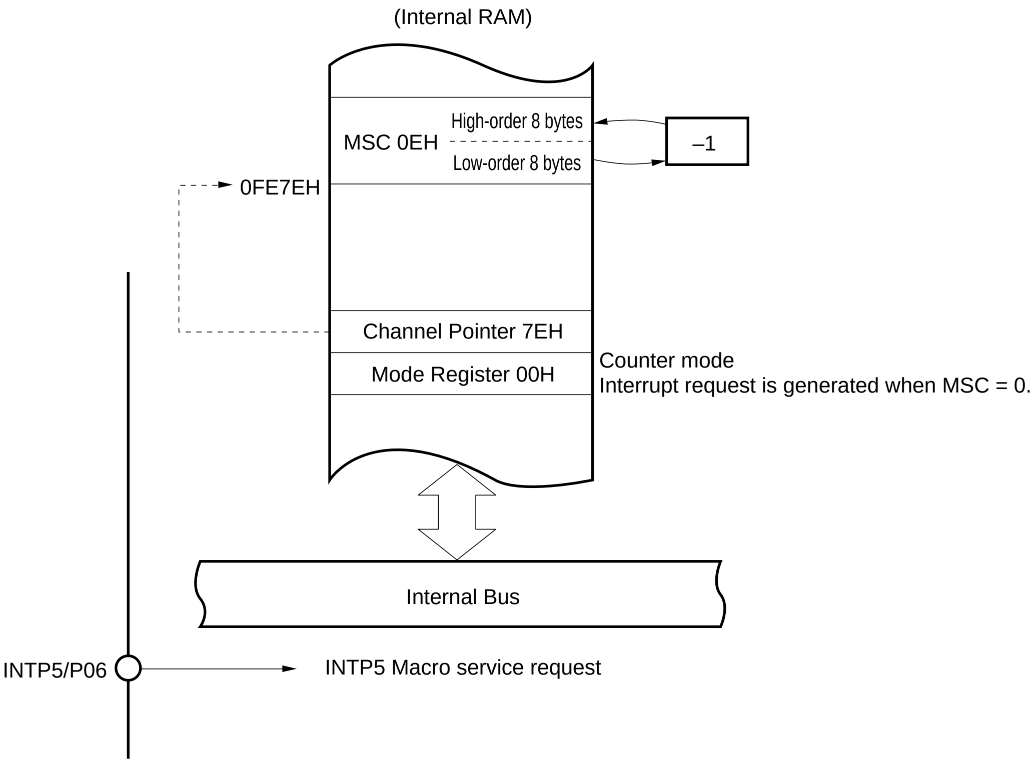
Figure 22-40. Counter Mode



(3) Example of using counter mode

Here is an example of counting the number of edges input to external interrupt pin INTP5.

Figure 22-41. Counting Number of Edges



Remark The internal RAM address in the figure above is the value when the LOCATION 0 instruction is executed. When the LOCATION 0FH instruction is executed, add 0F0000H to this value.

22.9 When Interrupt Requests and Macro Service Are Temporarily Held Pending

When the following instructions are executed, interrupt acknowledgment and macro service processing is held pending for 8 system clock cycles. However, software interrupts are not held pending.

EI
DI
BRK
BRKCS
RETCS
RETCSB !addr16
RETI
RETB
LOCATION 0H or LOCATION 0FH
POP PSW
POPU post
MOV PSWL, A
MOV PSWL, #byte
MOVG SP, #imm 24
Write instruction and bit manipulation instruction (excluding BT and BF) to interrupt control registers[Note], MK0, MK1, IMC, and ISPR.
PSW bit manipulation instruction

(Excluding the BT PSWL.bit, $addr20 instruction, BF PSWL.bit, $addr20 instruction, BT PSWH.bit, $addr20 instruction, BF PSWH.bit, $addr20 instruction, SET1 CY instruction, NOT1 CY instruction, and CLR1 CY instruction)

Note Interrupt control registers: PIC0, PIC1, PIC2, PIC3, PIC4, PIC5, PIC6, TMIC00, TMIC01, TMIC10, TMIC11, TMIC20, TMIC21, TMIC30, TMIC31, TMIC40, TMIC42, TMIC50, TMIC52, TMIC6, TMIC7, SERIC1, SRIC1, STIC1, CSIIC0, ADIC

Cautions 1. When an interrupt related register is polled using a BF instruction, etc., the branch destination of that BR instruction, etc., should not be that instruction. If a program is written in which a branch is made to that instruction itself, all interrupts and macro service requests will be held pending until a condition whereby a branch is not made by that instruction arises.

```
        Bad Example
               ⋮
LOOP :  BF  PIC0.7,  $LOOP          All interrupts and macro service requests are held pend-
                                    ing until PIC0.7 is 1.
              ×××                ←  Interrupts and macro service requests are not serviced
               ⋮                    until after execution of the instruction following the BF
                                    instruction.
      Good Example (1)
               ⋮
LOOP :  NOP
        BF  PIC0.7,  $LOOP       ←  Interrupts and macro service requests are serviced after
               ⋮                    execution of the NOP instruction, so that interrupts are
                                    never held pending for a long period.

      Good Example (2)
               ⋮
LOOP :  BT  PIC0.7,  $NEXT          Using a BTCLR instruction instead of a BT instruction has
                                    the advantage that the flag is cleared (0) automatically.
        BR  $LOOP                ←  Interrupts and macro service requests are serviced after
NEXT :         ⋮                    execution of the BR instruction, so that interrupts are
                                    never held pending for a long period.
```

2. For a similar reason, if problems are caused by a long pending period for interrupts and macro service when instructions to which the above applies are used in succession, a time at which interrupts and macro service requests can be acknowledged should be provided by inserting a NOP instruction, etc., in the series of instructions.

22.10 Instructions Whose Execution Is Temporarily Suspended by an Interrupt Request or Macro Service

Execution of the following instructions is temporarily suspended by an acknowledgeable interrupt request or macro service request, and the interrupt request or macro service request is acknowledged. The suspended instruction is resumed after completion of the interrupt service program or macro service processing.

Temporarily suspended instructions:
MOVM, XCHM, MOVBK, XCHBK
CMPME, CMPMNE, CMPMC, CMPMNC
CMPBKE, CMPBKNE, CMPBKC, CMPBKNC
SACW

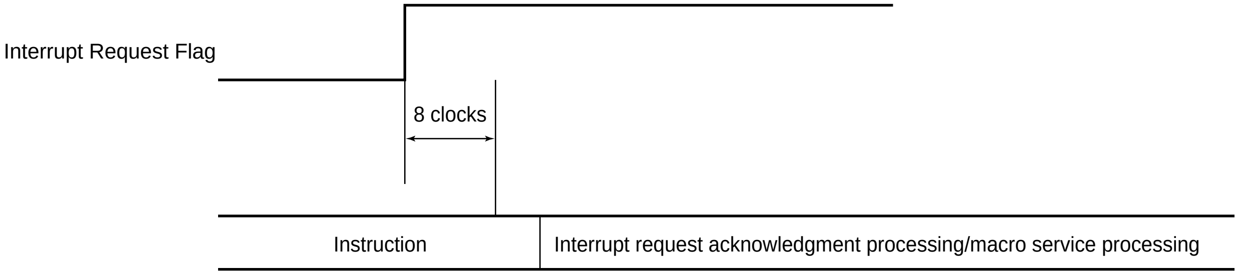
22.11 Interrupt Request and Macro Service Operation Timing

Interrupt requests are generated by hardware. The generated interrupt request sets (1) an interrupt request flag.

When the interrupt request flag is set to 1, a time of 8 clocks (1.00 μs: f_{CLK} = 8 MHz) is taken to determine the priority, etc.

Following this, if acknowledgment of that interrupt or macro service is enabled, interrupt request acknowledgment processing is performed when the instruction being executed ends. If the instruction being executed is one which temporarily defers interrupts and macro service, the interrupt request is acknowledged after the following instruction (refer to **22.9 When Interrupt Requests and Macro Service Are Temporarily Held Pending** for deferred instructions).

Figure 22-42. Interrupt Request Generation and Acknowledgment (Unit: Clock = 1/f_{CLK})


Interrupt Request Flag
8 clocks
Instruction
Interrupt request acknowledgment processing/macro service processing

22.11.1 Interrupt request acknowledge processing time

The time shown in Table 22-8 is required to acknowledge an interrupt request. After the time shown in this table has elapsed, execution of the interrupt servicing program is started.

Table 22-8. Interrupt Request Acknowledge Processing Time

(Unit: Clock = 1/f_{CLK})

Vector Table	IROM						EMEM					
Branch destination	IROM, PRAM			EMEM			PRAM			EMEM		
Stack	IRAM	PRAM	EMEM	IRAM	PRAM	EMEM	IRAM	PRAM	EMEM	IRAM	PRAM	EMEM
Vectored interrupts	26	29	37 + 4n	27	30	38 + 4n	30	33	41 + 4n	31	34	42 + 4n
Context switching	22	–	–	23	–	–	22	–	–	23	–	–

Remarks 1. IROM : internal ROM (with high-speed fetch specified)

PRAM : peripheral RAM of internal RAM (only when LOCATION 0 instruction is executed in the case of branch destination)

IRAM : internal high-speed RAM

EMEM : internal ROM when external memory and high-speed fetch are not specified

2. n is the number of wait states per byte necessary for writing data to the stack (the number of wait states is the sum of the number of address wait states and the number of access wait states).

3. If the vector table is EMEM, and if wait states are inserted in reading the vector table, add 2m to the value of the vectored interrupt in the above table, and add m to the value of context switching, where m is the number of wait states per byte necessary for reading the vector table.

4. If the branch destination is EMEM and if wait states are inserted in reading the instruction at the branch destination, add that number of wait states.

5. If the stack is occupied by PRAM and if the value of the stack pointer (SP) is odd, add 4 to the value in the above table.

6. The number of wait states is the sum of the number of address wait states and the number of access wait states.

22.11.2 Processing time of macro service

Macro service processing time differs depending on the type of the macro service, as shown in Table 22-9.

Table 22-9. Macro Service Processing Time

(Units: Clock = 1/f_{CLK})

Processing Type of Macro Service			Data Area	
			IRAM	Others
Type A	SFR → memory	1 byte	24	–
		2 bytes	25	–
	Memory → SFR	1 byte	24	–
		2 bytes	26	–
Type B	SFR → memory		33	35
	Memory → SFR		33	64
Type C			49	53
Counter mode	MSC ≠ 0		17	–
	USC = 0		25	–

Remarks 1. IRAM: internal high-speed RAM

2. In the following cases in the other data areas, add the number of clocks specified below.

- If the data size is 2 bytes with IROM or PRAM, and the data is located at an odd address: 4 clocks
- If the data size is 1 byte with EMEM: number of wait states for data access
- If the data size is 2 bytes with EMEM: 4 + 2n (where n is the number of wait states per byte)

3. If MSC = 0 with type A, B, or C, add 1 clock.

4. With type C, add the following value depending on the function to be used and the status at that time.

- Ring control: 4 clocks. Adds 7 more clocks if the ring counter is 0 during ring control.

22.12 Restoring Interrupt Function to Initial State

If an inadvertent program loop or system error is detected by means of an operand error interrupt, the watchdog timer, NMI pin input, etc., the entire system must be restored to its initial state. In the μPD784955, interrupt acknowledgment related priority control is performed by hardware. This interrupt acknowledgment related hardware must also be restored to its initial state, otherwise subsequent interrupt acknowledgment control may not be performed normally.

A method of initializing interrupt acknowledgment related hardware in the program is shown below. The only way of performing initialization by hardware is by $\overline{\text{RESET}}$ input.

Example

```
               MOVW   MK0, #0FFFFH    ;  Mask all maskable interrupts
               MOV    MK1L, #0FFH
      IRESL    :
               CMP    ISPR, #0        ;  No interrupt service programs running?
               BZ     $NEXT
               MOVG   SP, #RETVAL     ;  Forcibly change SP location
               RETI                   ;  Forcibly terminate running interrupt service program, return
                                         address = IRESL
      RETVAL   :
               DW     LOWW (IRESL)    ;  Stack data to return to IRESL with RETI instruction
               DB     0
               DB     HIGHW (IRESL)   ;  LOWW & HIGHW are assembler operators for calculating low-
                                         order 16 bits & high-order 16 bits respectively of symbol
      NEXT     :
```

- It is necessary to ensure that a non-maskable interrupt request is not generated via the NMI pin during execution of this program.
- After this, on-chip peripheral hardware initialization and interrupt control register initialization are performed.
- When interrupt control register initialization is performed, the interrupt request flags must be cleared to 0.

22.13 Cautions

(1) The in-service priority register (ISPR) is read-only. Writing to this register may result in malfunction.

(2) The watchdog timer mode register (WDM) can only be written to with a dedicated instruction (MOV WDM, #byte).

(3) The RETI instruction must not be used to return from a software interrupt caused by a BRK instruction.
Use the RETB instruction.

(4) The RETCS instruction must not be used to return from a software interrupt caused by a BRKCS instruction.
Use the RETCSB instruction.

(5) When a maskable interrupt is acknowledged by a vectored interrupt, the RETI instruction must be used to return from the interrupt. Subsequent interrupt related operations will not be performed normally if a different instruction is used.

(6) The RETCS instruction must be used to return from a context switching interrupt. Subsequent interrupt related operations will not be performed normally if a different instruction is used.

(7) Macro service requests are acknowledged and serviced even during execution of a non-maskable interrupt service program. If you do not want macro service processing to be performed during a non-maskable interrupt service program, you should manipulate the interrupt mask register in the non-maskable interrupt service program to prevent macro service generation.

(8) The RETI instruction must be used to return from a non-maskable interrupt. Subsequent interrupt acknowledgment will not be performed normally if a different instruction is used. Refer to **22.12 Restoring Interrupt Function to Initial State** when a program is to be restarted from the initial status after a non-maskable interrupt acknowledgement.

(9) Non-maskable interrupts are always acknowledged, except during non-maskable interrupt service program execution (except when a high non-maskable interrupt request is generated during execution of a low-priority non-maskable interrupt service program) and for a certain period after execution of the special instructions shown in **22.9**. Therefore, a non-maskable interrupt will be acknowledged even when the stack pointer (SP) value is undefined, in particular after reset release, etc. In this case, depending on the value of the SP, it may happen that the program counter (PC) and program status word (PSW) are written to the address of a write-inhibited special function register (SFR) (refer to **Table 3-6** in **3.8 Special Function Registers (SFRs)**), and the CPU becomes deadlocked, or an unexpected signal is output from a pin, or PC and PSW are written to an address in which RAM is not mounted, with the result that the return from the non-maskable interrupt service program in not performed normally and a software upset occurs.
Therefore, the program following $\overline{\text{RESET}}$ release must be as follows.

```
          CSEG  AT 0
          DW    STRT
          CSEG  BASE
  STRT:
          LOCATION 0FH; or LOCATION 0
          MOVG  SP, #imm24
```

(10) When an interrupt related register is polled using a BF instruction, etc., the branch destination of that BR instruction, etc., should not be that instruction. If a program is written in which a branch is made to that instruction itself, all interrupts and macro service requests will be held pending until a condition whereby a branch is not made by that instruction arises.

```
          Bad Example
                :
LOOP:   BF  PIC0.7,  $LOOP           All interrupts and macro service requests are held pending until
                                     PIC0.7 is 1.
               ×××                ←  Interrupts and macro service requests are not serviced until after
                :                    execution of the instruction following the BF instruction.

       Good Example (1)

LOOP:   NOP
        BF  PIC0.7,  $LOOP        ←  Interrupts and macro service requests are serviced after execution of
                                     the NOP instruction, so that interrupts are never held pending for a
                :                    long period.


       Good Example (2)
                :
LOOP:   BT  PIC0.7,  $NEXT           Using a BTCLR instruction instead of a BT instruction has the
                                     advantage that the flag is cleared (0) automatically.
        BR  $LOOP                 ←  Interrupts and macro service requests are serviced after execution of
                                     the BR instruction, so that interrupts are never held pending for a long
                                     period.
NEXT:           :
```

(11) For a similar reason to that given in (10), if problems are caused by a long pending period for interrupts and macro service when instructions to which the above applies are used in succession, a time at which interrupts and macro service requests can be acknowledged should be provided by inserting a NOP instruction, etc., in the series of instructions.

CHAPTER 23 STANDBY FUNCTION

23.1 Structure and Function

The μPD784955 has a standby function that can reduce the system's power consumption. The standby function has the following three modes.

Table 23-1. Standby Function Modes

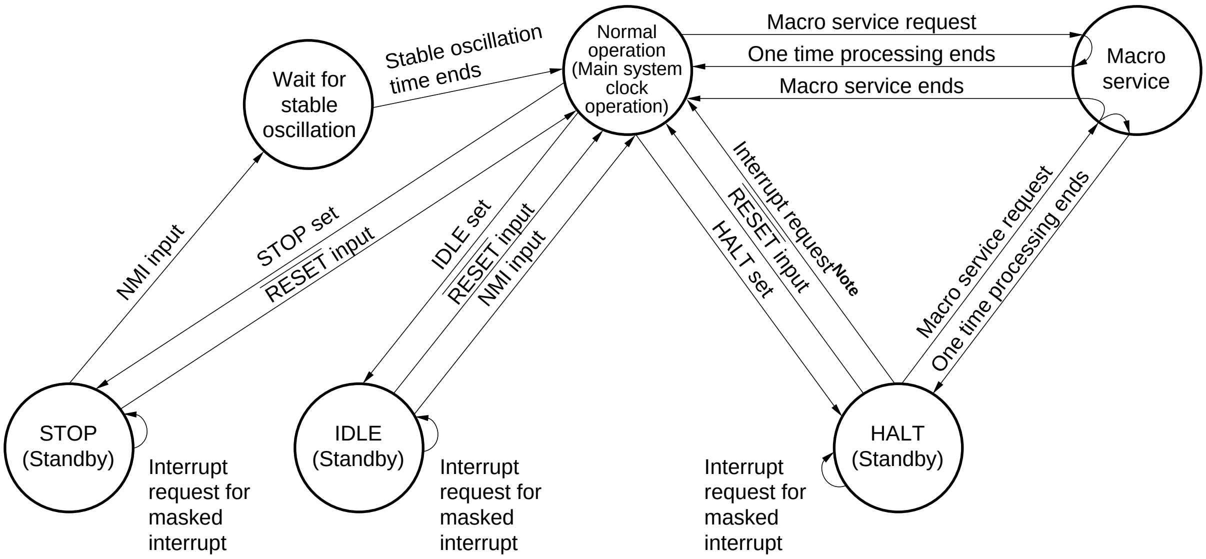
HALT mode	Stops the CPU operating clock. The average power consumption can be reduced by intermittent operation during normal operation.
STOP mode	Stops the main system clock. All of the operations in the chip are stopped, and the extremely low power consumption state of only a leakage current is entered.
IDLE mode	In this mode, the oscillation circuit continues operating while the rest of the system stops. Normal program operation can return to power consumption near that of the STOP mode and for the same time as the HALT mode.

These modes are programmable.

Macro service can be started from the HALT mode. After macro service execution, the device is returned to the HALT mode.

Figure 23-1 shows the standby function state transitions.

Figure 23-1. Standby Function State Transitions



Note Only unmasked interrupt requests

Remark NMI is only valid with external input. The watchdog timer cannot be used for the release of Standby (HALT mode/STOP mode/IDLE mode.)

23.2 Control Registers

(1) Standby control register (STBC)

The STBC register sets the STOP mode and selects the internal system clock.

To prevent the standby mode from accidentally being entered due to a runaway program, this register can only be written by a special instruction. This dedicated instruction, “MOV WDM, #byte”, has a special code configuration (4 bytes), and a write is not performed unless the 3rd and 4th bytes of the operation code are mutual complements.

If the 3rd and 4th bytes of the operation code are not mutual 1’s complements, a write is not performed and an operand error interrupt is generated. In this case, the return address saved in the stack area is the address of the instruction that was the source of the error, and thus the address that was the source of the error can be identified from the return address saved in the stack area.

If recovery from an operand error is simply performed by means of an RETB instruction, an endless loop will result.

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

Other write instructions (i.e., MOV STBC, A; STBC, #byte instruction; SET1 STBC.7) are ignored and nothing happens. In other words, STBC is not written, and an interrupt, such as an operand error interrupt, is not generated.

STBC can always be read by a data transfer instruction.

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

Figure 23-2 shows the STBC format.

Figure 23-2. Standby Control Register (STBC) Format

Address: 0FFC0H After Reset: 00H R/W

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

STP	HLT	Operation Setting Flag
0	0	Normal operating mode
0	1	HALT mode (automatically cleared when the HALT mode is released)
1	0	STOP mode (automatically cleared when the STOP mode is released)
1	1	IDLE mode (automatically cleared when the IDLE mode is released)

Caution Execute the NOP command three times after the standby command (after standby is released). If the interrupt request and standby instruction execution conflict, do not execute the standby instruction but, execute more than one instruction after the standby instruction and then acknowledge the interrupts. An instruction executed before acknowledging the interrupts, would be one that begins execution in less than six clock pulses after the execution of the standby instruction.

Example

```
MOV STBC, #byte
NOP
NOP
NOP
  :
  :
```

(2) Oscillation stable time specification register (OSTS)

The OSTS register sets the oscillation circuit operation and the oscillation stabilization time when the STOP mode is released.

Bits OSTS0 and OSTS1 in OSTS select the oscillation stabilization time when the STOP mode is released. Generally, select an oscillation stabilization time of at least 40 ms when using a crystal resonator and at least 4 ms when using a ceramic resonator.

The time until the stabilization oscillation is affected by the crystal/ceramic resonator that is used and the capacitance of the connected capacitor. Therefore, if you want a short oscillation stabilization time, consult the manufacturer of the crystal/ceramic resonator.

OSTS can be set by a 1-bit or 8-bit memory manipulation instruction.

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

Figure 23-3 shows the OSTS format.

Figure 23-3. Oscillation Stabilization Time Specification Register (OSTS) Format

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

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

OSTS1	OSTS0	Oscillation Stabilization Time Selection
0	0	$2^{19}/f_{CLK}$ (65.5 ms)
0	1	$2^{18}/f_{CLK}$ (32.8 ms)
1	0	$2^{17}/f_{CLK}$ (16.4 ms)
1	1	$2^{16}/f_{CLK}$ (8.2 ms)

Remark Figures in parentheses apply to operation with f_{CLK} = 8 MHz.

23.3 HALT Mode

23.3.1 Settings and operating states of HALT mode

The HALT mode is set by setting the HLT bit in the standby control register (STBC) to 1.

STBC can be written in with 8-bit data by a special instruction. Therefore, the HALT mode is specified by the "MOV STBC, #byte" instruction.

When enable interrupts is set (IE flag in PSW is set to 1), specify three NOP instructions after the HALT mode setting instruction (after the HALT mode is released). If this is not done after the HALT mode is released, multiple instructions may execute before interrupts are acknowledged. Unfortunately, the order relationship between the interrupt service and instruction execution may change. Since problems caused by the changes in the execution order are prevented, the measures described earlier are required.

The operating states in the HALT mode are described next.

Table 23-2. Operating States in the HALT Mode

Item	Operating States
Clock oscillation circuit	Operation
Internal system clock	Operation
CPU	Stop operation [Note]
I/O line	Holds the state before the HALT mode was set.
Peripheral function	Continues operation
Internal RAM	Retention

Note Macro service processing is carried out.

23.3.2 Releasing HALT mode

The HALT mode can be released by the following three sources.

- Non-maskable interrupt request (only possible for NMI pin input.)
- Maskable interrupt request (vectored interrupt, context switching, macro service)
- $\overline{\text{RESET}}$ input

Table 23-3 lists the release source and describes the operation after release. Operations following the canceling of the HALT mode are also shown in Figure 23-4.

Table 23-3. HALT Mode Release and Operation After Release

Release Source	MK[Note 1]	IE[Note 2]	State During Release	Operation After Release
$\overline{\text{RESET}}$ input	×	×	–	Normal reset operation
NMI pin input	×	×	• None while executing a non-maskable interrupt service program • Executing a low-priority non-maskable interrupt service program	Acknowledges interrupt requests
			• Executing the service program for the same request • Executing a high-priority non-maskable interrupt service program	The instruction following the “MOV STBC, #byte” instruction is executed. (The interrupt request that released the HALT mode is saved[Note 3].)
Maskable interrupt request (except for a macro service request)	0	1	• None while executing an interrupt service program • Executing a low-priority maskable interrupt service program • The PRSL bit[Note 4] is cleared to 0 while executing an interrupt service program at priority level 3.	Acknowledges interrupt requests
			• Executing a maskable interrupt service program with the same priority (This excludes executing an interrupt service program in priority level 3 when the PRSL bit[Note 4] is cleared to 0.) • Executing a high-priority interrupt service program	The instruction following “MOV STBC, #byte” is executed. (The interrupt request that released the HALT mode is saved[Note 3].)
	0	0	–	
	1	×	–	Holds the HALT mode
Macro service request	0	×	–	Macro service process execution End condition is not satisfied → End HALT mode condition is satisfied again → When VCIE[Note 5] = 1: HALT mode again When VCIE[Note 5] = 0: Same as a release by the maskable interrupt request
	1	×	–	Holds the HALT mode

Notes 1. Interrupt mask bit in each interrupt request source

2. Interrupt enable flag in the program status word (PSW)

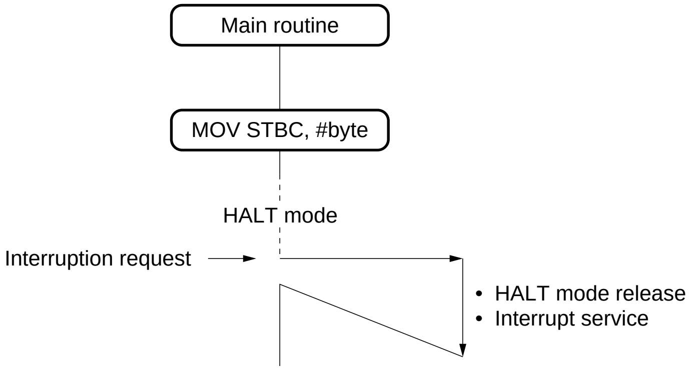
3. The held interrupt request is acknowledged when acknowledgement is enabled.

4. Bit in the interrupt mode control register (IMC)

5. Bit in the macro service mode register of the macro service control word that is in each macro service request source

Figure 23-4. Operations After HALT Mode Release (1/4)

(1) Interrupt after HALT mode



(2) Reset after HALT mode

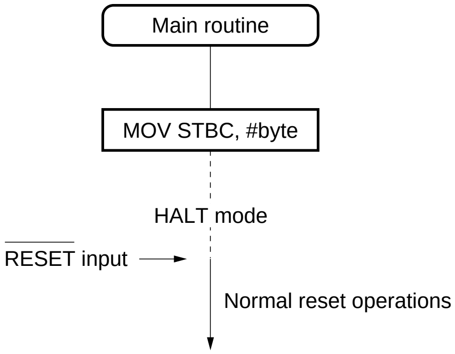
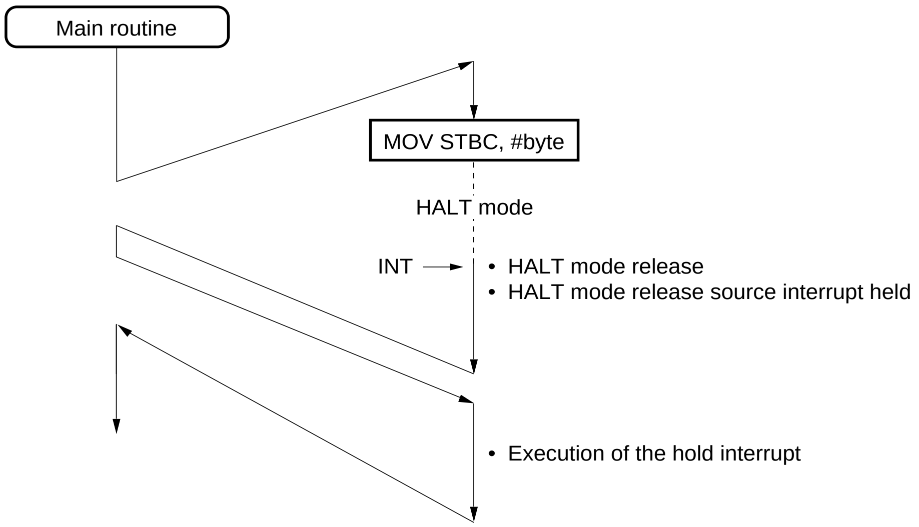


Figure 23-4. Operations After HALT Mode Release (2/4)

(3) HALT mode during interrupt servicing routine whose priority is higher than or equal to release source interrupt


Main routine
MOV STBC, #byte
HALT mode
INT
• HALT mode release
• HALT mode release source interrupt held
• Execution of the hold interrupt

(4) HALT mode during interrupt servicing routine whose priority is lower than release source interrupt

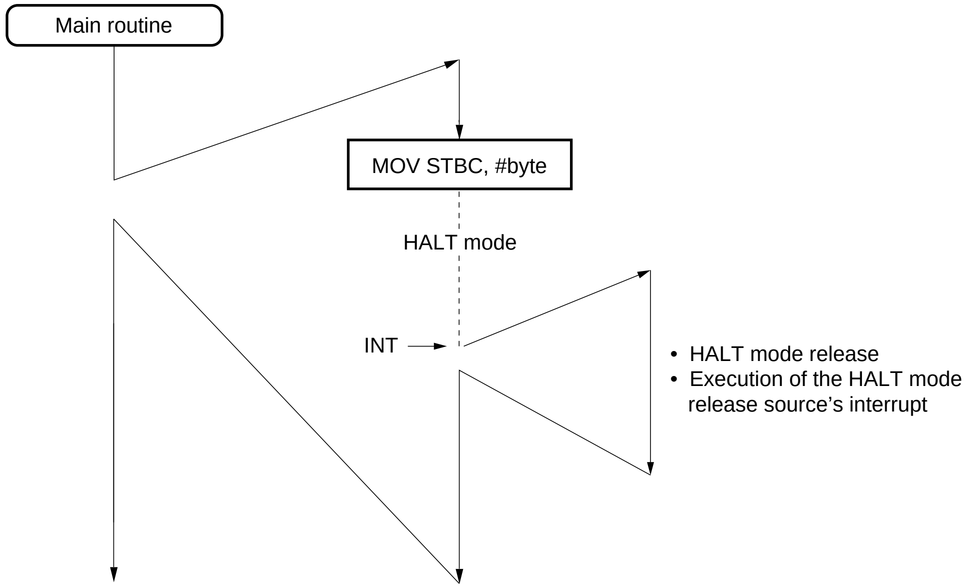
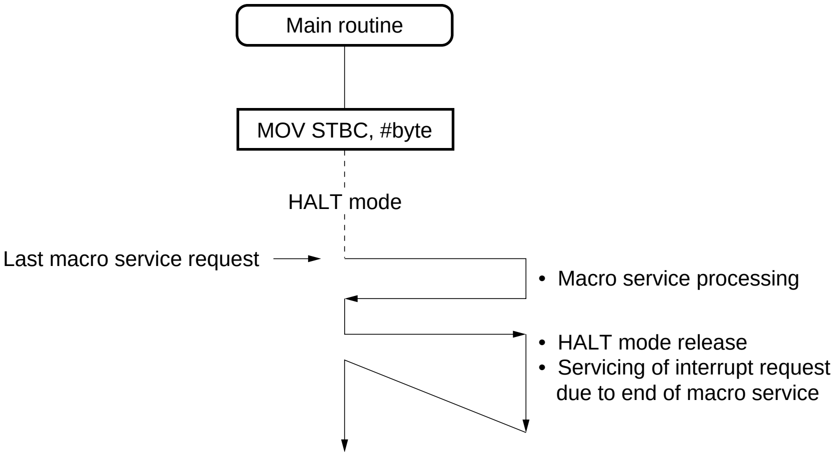


Figure 23-4. Operations After HALT Mode Release (3/4)

(5) Macro service request during HALT mode

(a) Immediately after macro service end condition is satisfied, interrupt request is issued (VCIE=0).


Main routine
MOV STBC, #byte
HALT mode
Last macro service request
• Macro service processing
• HALT mode release
• Servicing of interrupt request due to end of macro service

(b) Macro service end condition is not satisfied, or after macro service end condition is satisfied, interrupt request is not issued (VCIE = 1)

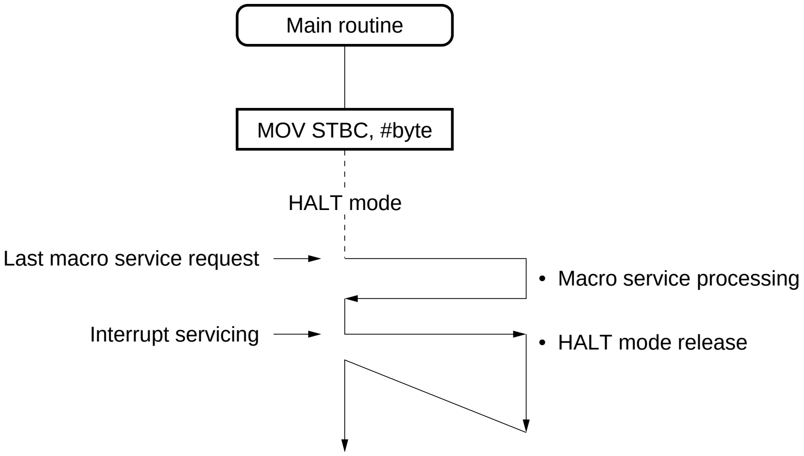
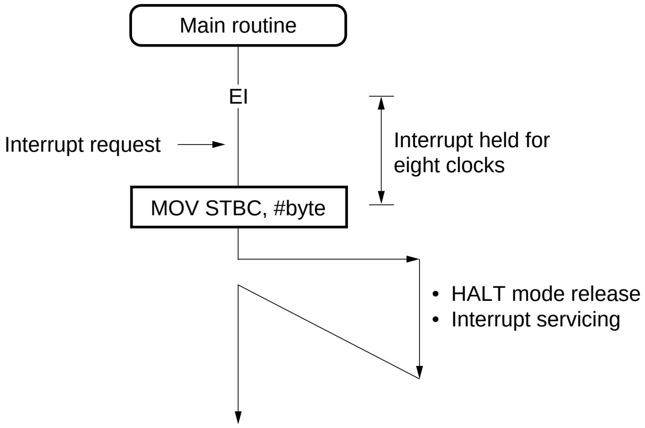
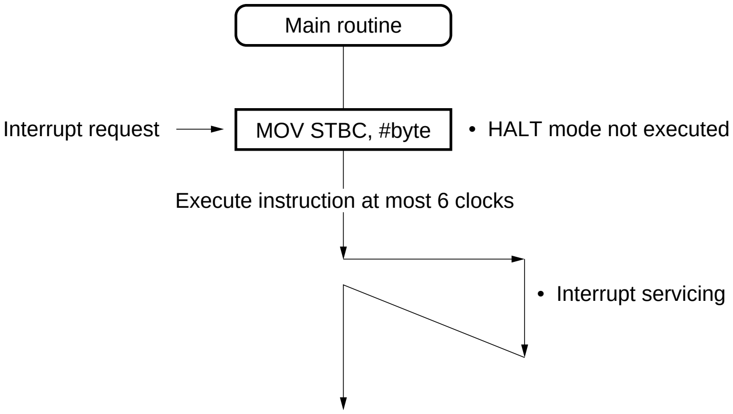


Figure 23-4. Operations After HALT Mode Release (4/4)

(6) HALT mode which the interrupt is held, which is enabled in an instruction that interrupt requests are temporarily held.



(7) Contention between HALT instruction and interrupt.



(1) Released by a non-maskable interrupt

When a non-maskable interrupt is generated, the halt mode is released regardless of the enable state (EI) and disable state (DI) for interrupt acknowledgment.

If the non-maskable interrupt that released the HALT mode can be acknowledged when the HALT mode is released, that non-maskable interrupt is acknowledged, and execution branches to the service program. If it cannot be acknowledged, the instruction following the instruction that set the HALT mode ("MOV STBC, #byte" instruction) is executed. The non-maskable interrupt that released the HALT mode is acknowledged when acknowledgment is enabled. For details about non-maskable interrupt acknowledgment, refer to **22.6 Non-maskable Interrupt Acknowledgment Operation**.

Caution The HALT mode cannot be released with the watchdog timer.

(2) Released by a maskable interrupt request

The HALT mode released by a maskable interrupt request can only be released by an interrupt where the interrupt mask flag is 0.

If an interrupt can be acknowledged when the halt mode is released and the interrupt request enable flag (IE) is set to 1, execution branches to the interrupt service program. If the IE flag is cleared to 0 when acknowledgment is not possible, execution restarts from the next instruction that sets the HALT mode. For details about interrupt acknowledgment, refer to **22.7 Maskable Interrupt Acknowledgment Operation**.

A macro service temporarily releases the HALT mode, performs the one-time processing, and returns again to the HALT mode. If the macro service is only specified several times, the HALT mode is released when the VCIE bit in the macro service mode register in the macro service control word is cleared to 0.

The operation after this release is identical to the release by the maskable interrupt described earlier. Also when the VCIE bit is set to 1, the HALT mode is entered again, and the HALT mode is released by the next interrupt request.

Table 23-4. Releasing HALT Mode by Maskable Interrupt Request

Release Source	MK[Note 1]	IE[Note 2]	State During Release	Operation After Release
Maskable interrupt request (except for a macro service request)	0	1	• None while executing an interrupt service program • Executing a low-priority maskable interrupt service program • The PRSL bit[Note 4] is cleared to 0 while executing an interrupt service program at priority level 3.	Acknowledges interrupt requests
			• Executing a maskable interrupt service program with the same priority (This excludes executing an interrupt service program in priority level 3 when the PRSL bit[Note 4] is cleared to 0.) • Executing a high-priority interrupt service program	The instruction following the “MOV STBC, #byte” instruction is executed. (The interrupt request that released the HALT mode is saved[Note 3].)
	0	0	–	
	1	×	–	Holds the HALT mode
Macro service request	0	×	–	Macro service process execution End condition is not satisfied → HALT mode again End condition is satisfied → When VCIE[Note 5] = 1: HALT mode again When VCIE[Note 5] = 0: Same as a release by a maskable interrupt request
	1	×	–	Holds the HALT mode

Notes 1. Interrupt mask bit in each interrupt request source

2. Interrupt enable flag in the program status word (PSW)

3. The held interrupt request is acknowledged when acknowledgment is enabled.

4. Bit in the interrupt mode control register (IMC)

5. Bit in the macro service mode register of the macro service control word that is in each macro service request source

(3) Released by $\overline{\text{RESET}}$ input

After branching to the reset vector address as in a normal reset, the program executes. However, the contents of the internal RAM hold the value before the HALT mode was set.

23.4 STOP Mode

23.4.1 Settings and operating states of STOP mode

The STOP mode is set by setting the STP bit in the standby control register (STBC) to 1.

STBC can only be written with 8-bit data by a special instruction. Therefore, the STOP mode is set by the “MOV STBC, #byte” instruction.

When enable interrupts is set (IE flag in PSW is set to 1), specify three NOP instructions after the STOP mode setting instruction (after the STOP mode is released). If this is not done after the STOP mode is released, multiple instructions can be executed before interrupts are acknowledged. Unfortunately, the order relationship between the interrupt service and instruction execution may change. Since the problems caused by changes in the execution order are prevented, the measures described earlier are required.

Caution Since an interrupt request signal is used for releasing the standby mode, when there is an interrupt source that sets the interrupt request flag or resets the interrupt mask flag, even though the standby mode is entered, it is immediately released. Therefore, in the STOP mode, the HALT mode is entered immediately after the HALT instruction is executed, and the operating mode returns after waiting only the time set in the oscillation stable time selection register (OSTS).

Next, the operating states during the STOP mode are described.

Table 23-5. Operating States in STOP Mode

Item	Operating States
Clock generation circuit	Stop operations (PLL also stops)
Internal system clock	Halting
CPU	Stop operation
I/O line	Holds the state before the STOP mode was set.
Peripheral function	All operations stop Note
Internal RAM	Retention

Note Even the A/D converter stops its operation, current consumption is not reduced if the A/D converter mode register (ADM0)’s ADCS0 bit is set to 1.

Cautions 1. Clear (set to 0) ADM0’s ADCS0 bit.

2. Don’t enable the STOP mode during external clock input.

23.4.2 Releasing STOP mode

The STOP mode is released by NMI input or $\overline{\text{RESET}}$ input.

Outlines of the release sources and operations following release are shown in Table 23-6. Operations following release of the STOP mode are also shown in Figure 23-5.

Table 23-6. Releasing STOP Mode and Operation After Release

Release Source	MK[Note 1]	ISM[Note 2]	IE[Note 3]	State During Release	Operation After Release
$\overline{\text{RESET}}$ input	×	×	×	–	Normal reset operation
NMI pin input	×	×	×	• None while executing a non-maskable interrupt service program • Executing a low-priority non-maskable interrupt service program	Acknowledges interrupt requests
				• Executing the service program for the NMI pin input • Executing a high-priority non-maskable interrupt service program	The instruction following the “MOV STBC, #byte” instruction is executed. (The interrupt request that released the STOP mode is pending[Note 4].)

Notes 1. Interrupt mask bit in each interrupt request source

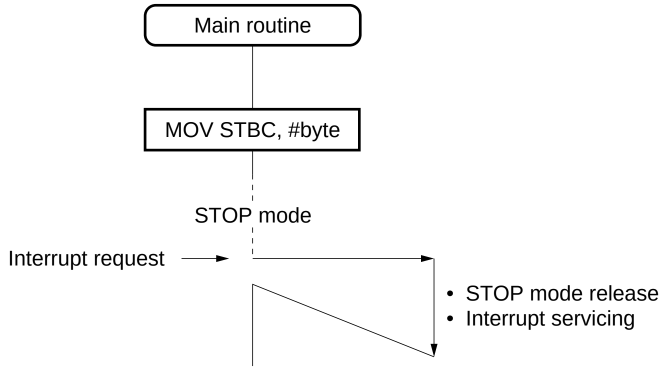
2. Macro service enable flag that is in each interrupt request source

3. Interrupt enable flag in the program status word (PSW)

4. The pending interrupt request is acknowledged when acknowledgment is enabled.

Figure 23-5. Operations After the STOP Mode Has Been Released (1/2)

(1) Interrupt after STOP mode



(2) Reset after STOP mode

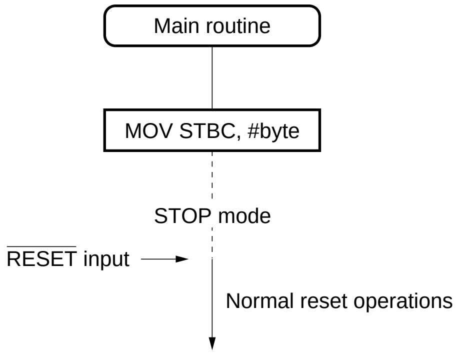
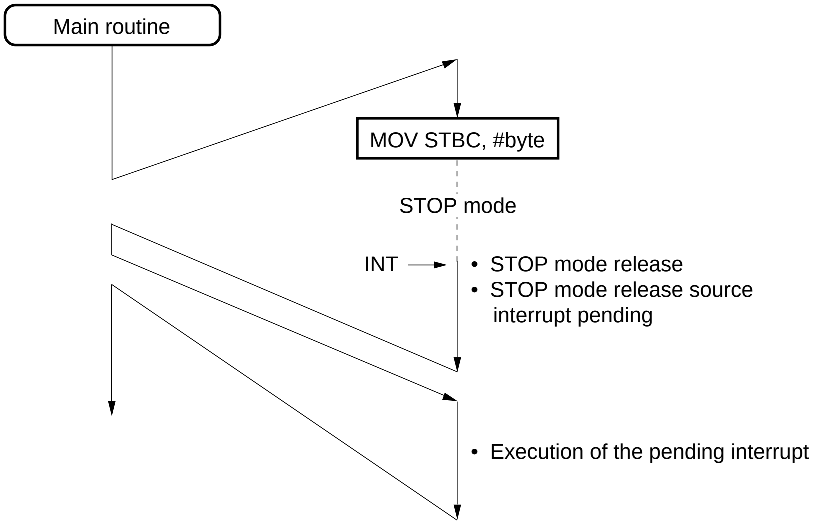
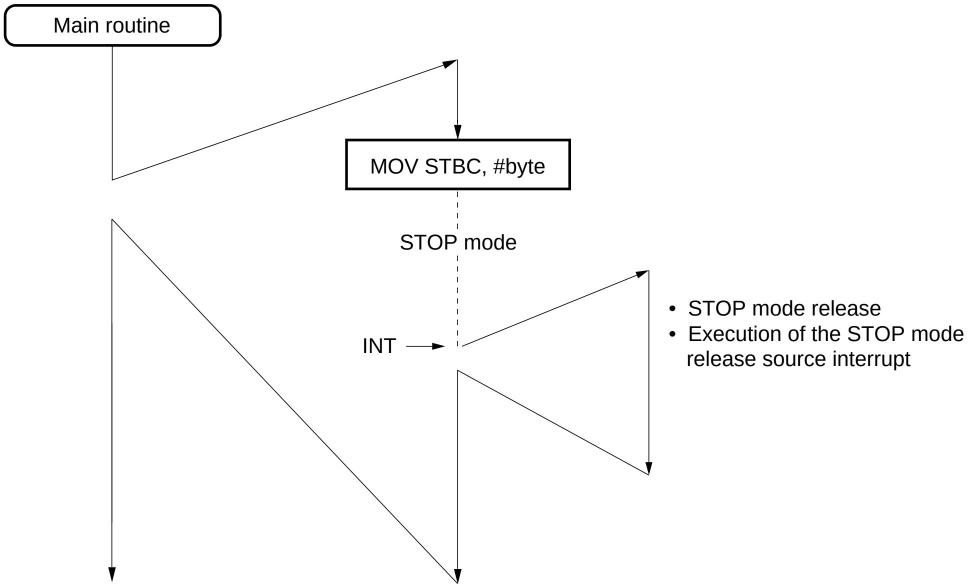


Figure 23-5. Operations After the STOP Mode Has Been Released (2/2)

(3) STOP mode during interrupt servicing routine whose priority is lower than release source interrupt


Main routine
MOV STBC, #byte
STOP mode
INT
• STOP mode release
• STOP mode release source interrupt pending
• Execution of the pending interrupt

(4) STOP mode during interrupt servicing routine whose priority is lower than release source interrupt



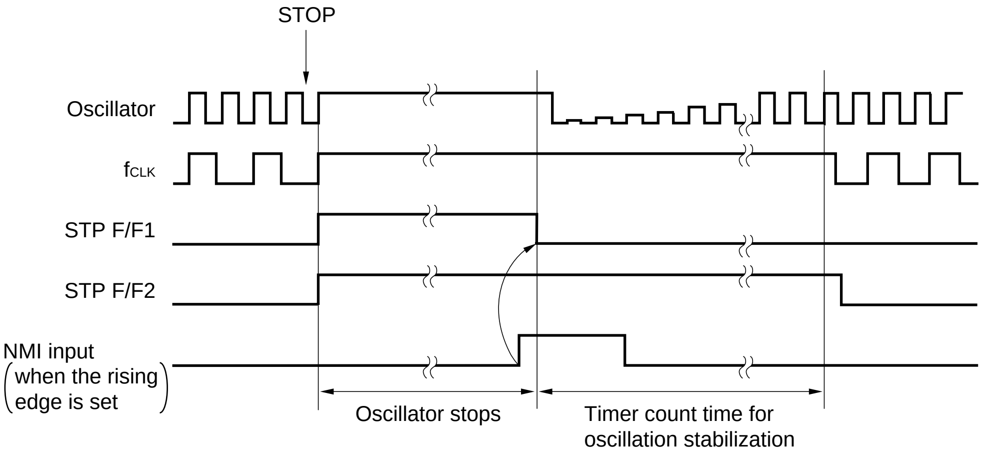
(1) Releasing the STOP mode by NMI input

When the valid edge specified in the external interrupt edge enable registers (EGP0, EGN0) is input by the NMI input, the oscillator starts oscillating again. Then the STOP mode is released after the oscillation stabilization time set in the oscillation stabilization time setting register (OSTS) elapses.

When the STOP mode is released and non-maskable interrupts from the NMI pin input can be acknowledged, execution branches to the NMI interrupt service program. If acknowledgment is disabled (such as when set in the STOP mode in the NMI interrupt service program), execution starts again from the instruction following the instruction that set the STOP mode. When acknowledgment is enabled, execution branches to the NMI interrupt service program (by executing the RETI instruction).

For details about NMI interrupt acknowledgment, refer to **22.6 Non-maskable Interrupt Acknowledgment Operation**.

Figure 23-6. Releasing STOP Mode by NMI Input


STOP
Oscillator
fCLK
STP F/F1
STP F/F2
NMI input
(when the rising edge is set)
Oscillator stops
Timer count time for oscillation stabilization

(2) Releasing the STOP mode by $\overline{\text{RESET}}$ input

When $\overline{\text{RESET}}$ input falls from high to low and the reset condition is entered, the oscillator starts oscillating. Maintain the oscillation stabilization time for the $\overline{\text{RESET}}$ active period. Then, when the $\overline{\text{RESET}}$ rises, normal operation starts.

The difference from the normal reset operation is that the data memory saves the contents before setting the STOP mode.

23.5 IDLE Mode

23.5.1 Settings and operating states of IDLE mode

The IDLE mode is set by setting both bits STP and HLT in the standby control register (STBC) to 1.

STBC can only be written with 8-bit data by using a special instruction. Therefore, the IDLE mode is set by the "MOV STBC, #byte" instruction.

When enable interrupts is set (the IE flag in PSW is set to 1), specify three NOP instructions after the IDLE mode setting instruction (after the IDLE mode is released). If this is not done after the IDLE mode is released, multiple instructions can be executed before interrupts are acknowledged. Unfortunately, the order relationship between the interrupt processing and the instruction execution may change. To prevent the problems caused by the change in the execution order, the measures described earlier are required.

The operating states in the IDLE mode are described next.

Table 23-7. Operating States in IDLE Mode

Item	Operating States
Clock generation circuit	Continues oscillating (PLL also continues to operate)
Internal system clock	Halting
CPU	Stop operation
I/O line	Saves the state before the IDLE mode was set.
Peripheral function	All operations stop **Note**
Internal RAM	Retention

Note Even though the A/D converter stops its operation, current consumption is not reduced if A/D converter mode register (ADM0)'s ADCS0 bit is set to 1.

Caution Clear (set to 0) ADM0's ADCS0 bit.

23.5.2 Releasing IDLE mode

The IDLE mode is released by NMI input or $\overline{\text{RESET}}$ input.

Outlines of the release sources and operations following release are shown in Table 23-8. Operations following release of the IDLE mode are also shown in Figure 23-8.

Table 23-8. Releasing IDLE Mode and Operation After Release

Release Source	MK[Note 1]	ISM[Note 2]	IE[Note 3]	State During Release	Operation After Release
$\overline{\text{RESET}}$ input	×	×	×	–	Normal reset operation
NMI pin input	×	×	×	• None while executing a non-maskable interrupt service program • Executing a low-priority non-maskable interrupt service program	Acknowledges interrupt requests
				• Executing the service program for the NMI pin input • Executing a high-priority non-maskable interrupt service program	Executes the instruction following the “MOV STBC, #byte” instruction (The interrupt request that released the IDLE mode is saved[Note 4].)

Notes 1. Interrupt mask bit in each interrupt request source

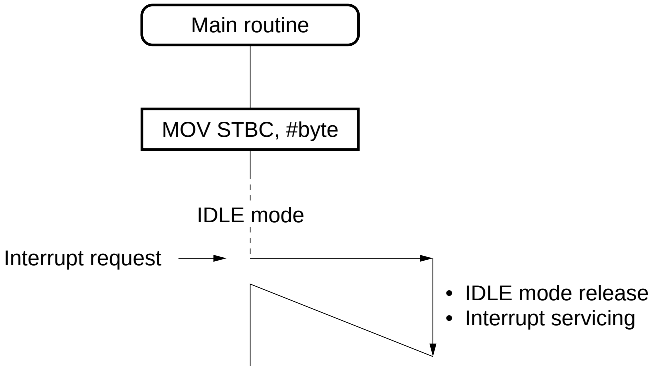
2. Macro service enable flag that is in each interrupt request source

3. Interrupt enable flag in the program status word (PSW)

4. The saved interrupt request is acknowledged when acknowledgment is enabled.

Figure 23-7. Operations After IDLE Mode Release (1/2)

(1) Interrupt after IDLE mode



(2) Reset after IDLE mode

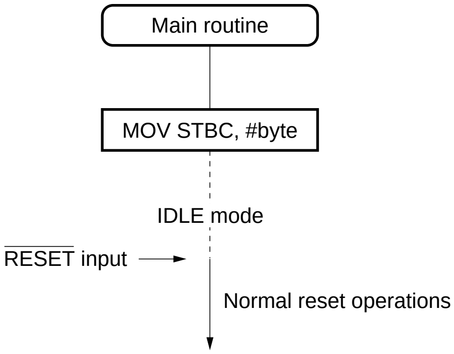
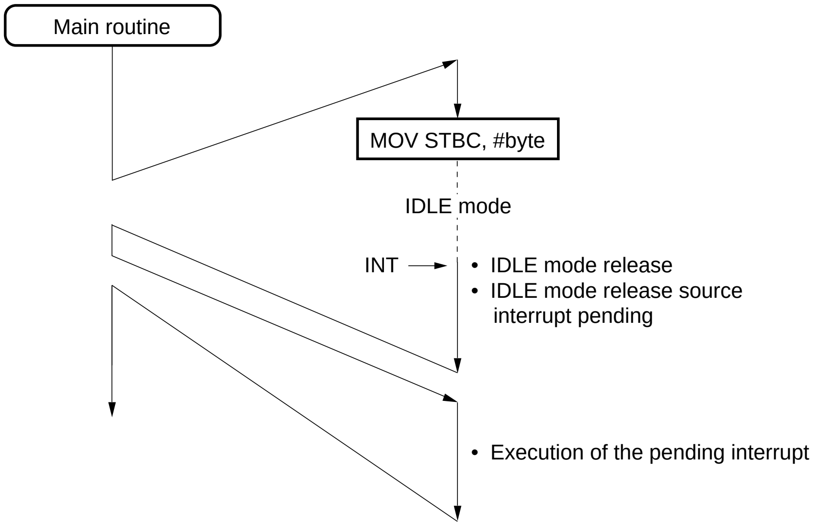
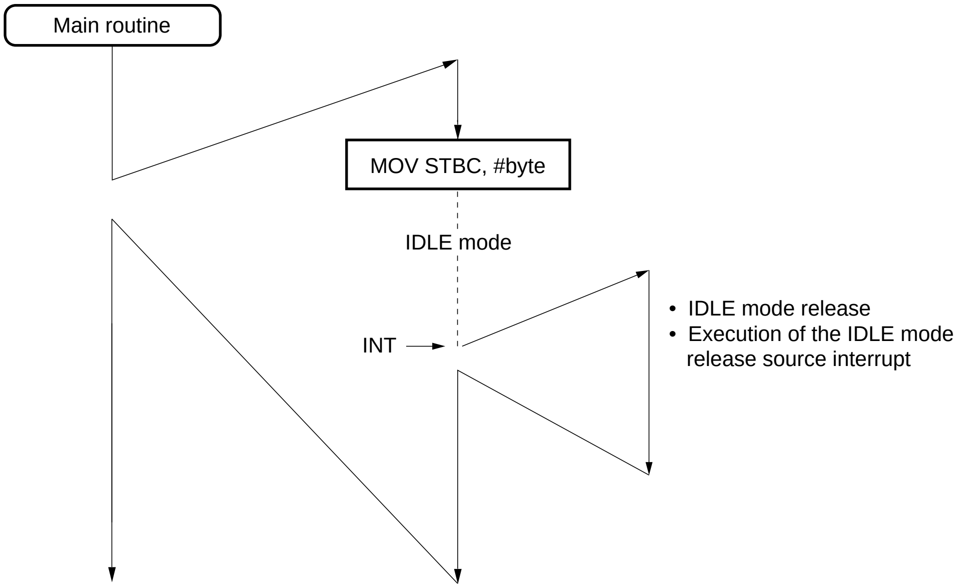


Figure 23-7. Operations After IDLE Mode Release (2/2)

(3) IDLE mode during interrupt servicing routine whose priority is higher than or equal to release source interrupt


Main routine
MOV STBC, #byte
IDLE mode
INT
• IDLE mode release
• IDLE mode release source interrupt pending
• Execution of the pending interrupt

(4) IDLE mode during interrupt servicing routine whose priority is lower than release source interrupt



(1) Releasing the IDLE mode by NMI input

When the valid edge specified in the external interrupt edge enable registers (EGP0, EGN0) is input by the NMI input, the IDLE mode is released.

When the IDLE mode is released and the non-maskable interrupt from the NMI pin input can be acknowledged, execution branches to the NMI interrupt service program. If acknowledgment is disabled (such as when set in the IDLE mode in the NMI interrupt service program), execution starts again from the instruction following the instruction that set the IDLE mode. When acknowledgment is enabled, execution branches to the NMI interrupt service program (by executing the RETI instruction).

For details about NMI interrupt acknowledgment, refer to **22.6 Non-maskable Interrupt Acknowledgment Operation**.

(2) Releasing the IDLE mode by $\overline{\text{RESET}}$ input

When $\overline{\text{RESET}}$ input falls from high to low and the reset condition is entered, the oscillator starts oscillating. Maintain the oscillation stabilization time for the $\overline{\text{RESET}}$ active period. Then, when the $\overline{\text{RESET}}$ rises, normal operation starts.

The difference from the normal reset operation is that the data memory saves the contents before setting the IDLE mode.

23.6 Check Items When Using STOP or IDLE Mode

The checks required to reduce the consumption current when using the STOP mode or IDLE mode are described below.

(1) Is the output level of each output pin appropriate?

The appropriate output level of each pin differs with the circuit in the next stage. Select the output level so that the consumption current is minimized.

- If a high level is output when the input impedance of the circuit in the next stage is low, current flows from the power source to the port, and the consumption current increases. This occurs when the circuit in the next stage is, for example, a CMOS IC. When the power supply is turned off, the input impedance of a CMOS IC becomes low. To reduce the consumption current and not negatively affect the reliability of the CMOS IC, output a low level. If a high level is output, latch-up results when the power supply is applied again.

- Depending on the circuit in the next stage, the consumption current sometimes increases when a low level is input. In this case, output a high level or high impedance to reduce the consumption current.

- When the circuit in the next stage is a CMOS IC, if the output is high impedance when power is supplied to the CMOS IC, the consumption current of the CMOS IC sometimes increases (in this case, the CMOS IC overheats and is sometimes destroyed). In this case, output a suitable level or use a pull-up or pull-down resistor.

The setting method for the output level differs with the port mode.

- Since the output level is determined by the state of the internal hardware when the port is in the control mode, the output level must be set while considering the state of the internal hardware.

- The output level can be set by writing to the output latch of the port and the port mode register by the software when in the port mode.

When the port enters the control mode, the port mode is changed by simply setting the output level.

(2) Is the input level to each input pin appropriate?

Set the voltage level input to each pin within the range from the V_{SS} voltage to the V_{DD} voltage. If a voltage outside of this range is applied, not only does the consumption current increase, but the reliability of the μPD784955 is negatively affected.
In addition, do not increase the middle voltage.

(3) Are internal pullup resistors needed?

Unnecessary pull-up resistors increase the consumption current and are another cause of device latch-up. Set the pull-up resistors to the mode in which they are used only in the required parts.
When the parts needing pull-up resistors and the parts not needing them are mixed together, externally connect the pull-up resistors where they are needed and set the mode in which the internal pull-up resistors are not used.

(4) A/D converter

The current flowing through pin AV_{REF} can be reduced by clearing the ADCS0 bit, that is bit 7 in the A/D converter mode register 0 (ADM0), to 0. Furthermore, if you want to reduce the current, disconnect the current supplied to AV_{REF} by an externally attached circuit.

The AV_{REF} pin must always have the same voltage as the V_{REF} pin. If current is not supplied to the AV_{DD} pin in the STOP mode, not only does the consumption current increase, but the reliability of the μPD784955 is negatively affected.

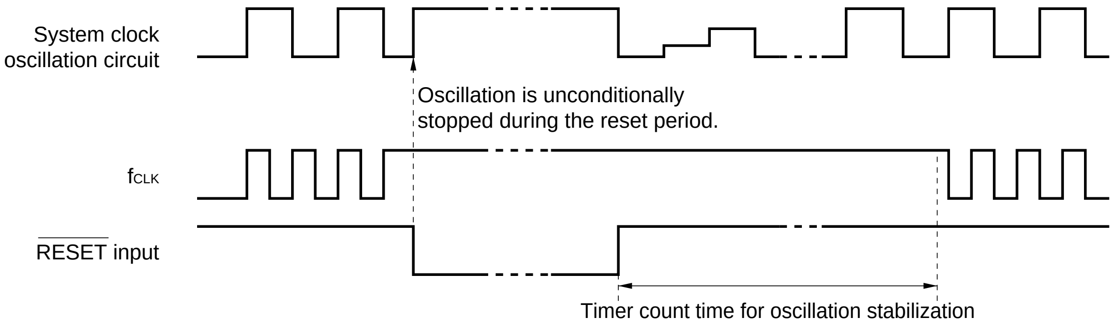
CHAPTER 24 RESET FUNCTION

Inputting the low-level signal to the $\overline{\text{RESET}}$ pin resets the system and places all hardware in the state shown in Table 24-1. The entire system's current consumption can be held down in the reset duration because the system clock stops oscillating unconditionally.

Inputting the high-level signal to the $\overline{\text{RESET}}$ pin clears the reset state, places the reset vector table's contents in the program counter (PC) after the count time for the timer that is used for stabilizing oscillation (65.5 ms: at 8-MHz operation), branches to an address latched in the PC and starts execution of programs from the branch destination address.

A reset can thus be started from any address desired.

Figure 24-1. Oscillation of System Clock in Reset Period


System clock oscillation circuit
Oscillation is unconditionally stopped during the reset period.
f_{CLK}
$\overline{\text{RESET}}$ input
Timer count time for oscillation stabilization

To prevent noise from causing the $\overline{\text{RESET}}$ input signal pin to operate in error, a noise elimination circuit is incorporated to reduce noise by means of analog delay.

Figure 24-2. Accepting Reset Signal

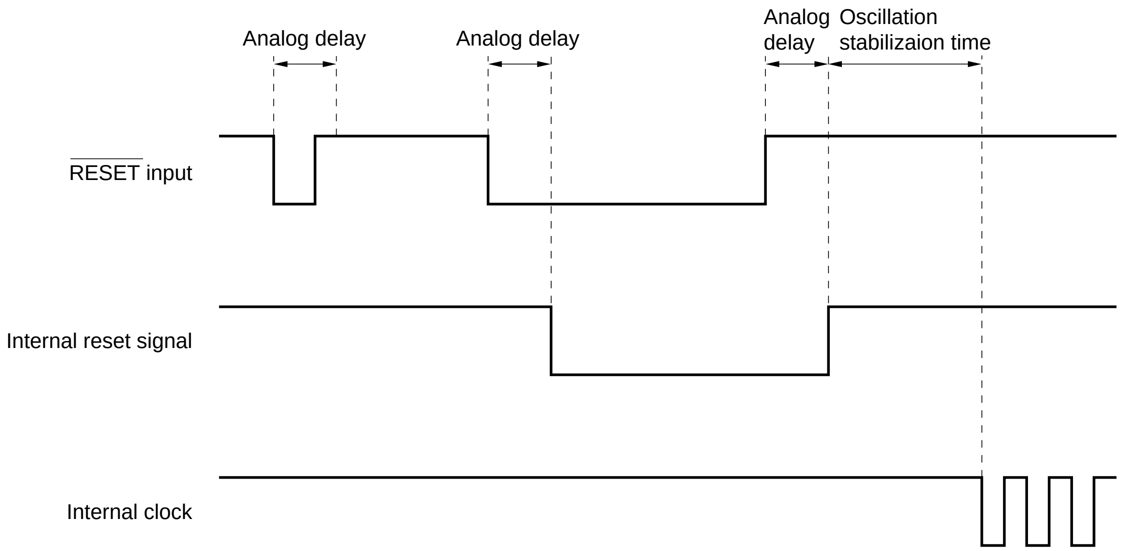


Table 24-1. State After Reset for All Hardware Resets

Hardware	State During Reset ($\overline{\text{RESET}}$ = L)	State After Reset ($\overline{\text{RESET}}$ = H)
System clock oscillation circuit	Oscillation stops	Oscillation starts
Program counter (PC)	Undefined	Set a value in the reset vectored table.
Stack pointer (SP)	Undefined	
Program status word (PSW)	Initialize to 0000H.	
Internal RAM	This is undefined. However, when the standby state is released by a reset, the value is saved before setting standby.	
I/O lines	High impedance	
Other hardware	Initialize to the fixed state[Note].	

Note See **Table 3-6 Special Function Register (SFR) List** when resetting.

CHAPTER 25 μPD78F4956 PROGRAMMING

The μPD78F4956 is a flash memory version of the μPD784955 Subseries.

The μPD78F4956 has on-chip flash memory that allows write, erase, and rewrite of programs in the state in which it is mounted on the substrate. Table 25-1 shows the differences between the flash memory version (μPD78F4956) and the mask ROM versions (μPD784955 and 784953).

Table 25-1. Differences between the μPD78F4956 Mask ROM Versions

Item	μPD78F4956	Mask ROM versions
Internal ROM type	Flash memory	Mask ROM
Internal ROM capacity	64 Kbytes	μPD784953: 24 Kbytes μPD784955: 48 Kbytes
Internal RAM capacity	2,048 bytes	μPD784953: 768 bytes μPD784955: 2,048 bytes
Internal memory size switching register (IMS)	Available	Not available
IC pin	Not available	Available
V_{PP} pin	Available	Not available

Caution There are differences in noise immunity and noise radiation between the PROM and mask ROM versions. When pre-producing an application set with the PROM version and then mass-producing it with the mask ROM version, be sure to conduct sufficient evaluations for the set using commercial samples (not engineering samples) of the mask ROM version.

25.1 Internal Memory Size Switching Register (IMS)

IMS is a register to prevent a certain part of the internal memory from being used by software. By setting the IMS, it is possible to establish a memory map that is the same as that of mask ROM version with a different internal memory (ROM, RAM) with capacity.

IMS is set by an 8-bit memory manipulation instruction.

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

Figure 25-1. Internal Memory Size Switching Register (IMS) Format

Address : 0FFFCH After Reset : FFH W

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

ROM1	ROM0	Internal ROM Capacity Selection
0	0	24 Kbyte
0	1	Setting prohibited
1	0	48 Kbyte
1	1	64 Kbyte

RAM1	RAM0	Internal RAM Capacity Selection
0	0	768 bytes
0	1	Setting prohibited
1	0	
1	1	2,048 bytes

Caution IMS is not available for mask ROM versions (μPD784953 and 784955).

The IMS settings to create the same memory map as mask ROM versions are shown in Table 25-2.

Table 25-2. Internal Memory Size Switching Register (IMS) Settings

Relevant Mask ROM Version	IMS Setting
μPD784953	CCH
μPD784955	EFH

25.2 Programming Flash Memory

Flash memory can be written while mounted on the target system (on-board writing). Connect the dedicated flash programmer (Flashpro II) to the host computer and target system for programming.

Remark The Flashpro II is a product of Naitou Densei Machidaseisakusho Co., Ltd.

25.2.1 Selecting a communication method

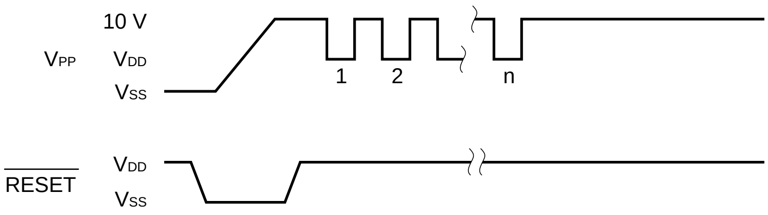
The Flashpro II is used to write data into a flash memory by serial communications. Figure 25-2 shows the format used to select the communication protocol. Each communication protocol is selected with the number of V_{PP} pulses shown in Table 25-3.

Table 25-3. Communication Protocols

Communication Protocol	No. of Channels	Pins Used	No. of V_{PP} Pulses
3-wire serial I/O	1	$\overline{\text{SCK}}$/P27 SO/P26 SI/P25	0
UART	1	TxD/P21 RxD/P20	8

Caution Always select the communication protocol using the number of V_{PP} pulses shown in Table 25-3.

Figure 25-2. Communication Protocol Selection Format



25.2.2 Flash memory programming functions

By transmitting and receiving various commands and data by the selected communication protocol, operations such as writing to the flash memory are performed. Table 25-4 shows the major functions.

Table 25-4. Flash Memory Programming Functions

Function	Description
Batch erase	Erase the entire memory contents.
Block erase	Erase the contents of the specified memory block where one memory block is 16 Kbytes.
Batch blank check	Checks the erase state of the entire memory.
Block blank check	Checks the erase state of the specified block.
Data write	Writes to the flash memory based on the start write address and the number of data written (number of bytes).
Batch verify	Compares the data input to the contents of the entire memory.
Block verify	Compares the data input to the contents of the specified memory block.

Verification for the flash memory entails supplying the data to be verified from an external source via a serial interface, and then outputting the existence of unmatched data to the external source after referencing the blocks or all of the data. Consequently, the flash memory is not equipped with a read function, and it is not possible for third parties to read the contents of the flash memory with the use of the verification function.

25.2.3 Connecting Flashpro II

The connection between the Flashpro II and the μPD78F4956 differs with the communication protocol (3-wire serial I/O or UART). Figures 25-3 and 25-4 are the connection diagrams in each case.

Figure 25-3. Flashpro II Connection in 3-Wire Serial I/O Method

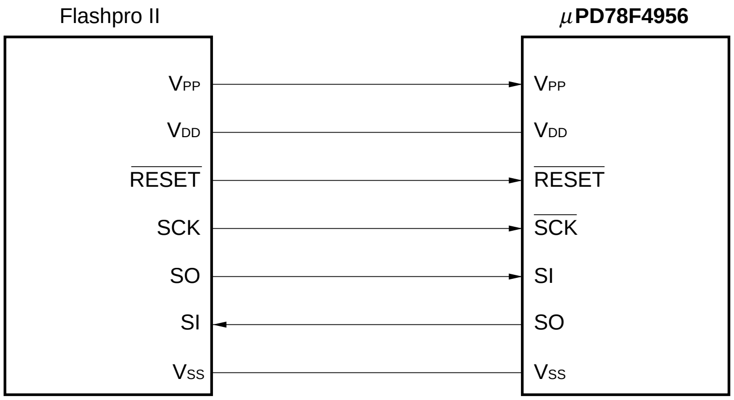
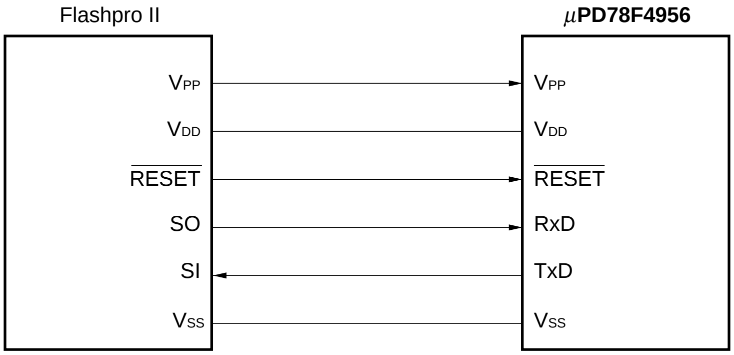


Figure 25-4. Flashpro II Connection in UART Method



[MEMO]

CHAPTER 26 INSTRUCTION OPERATION

26.1 Examples

(1) Operand expression format and description (1/2)

Expression Format	Description
r, r’[Note 1]	X(R0), A(R1), C(R2), B(R3), R4, R5, R6, R7, R8, R9, R10, R11, E(R12), D(R13), L(R14), H(R15)
r1[Note 1]	X(R0), A(R1), C(R2), B(R3), R4, R5, R6, R7
r2	R8, R9, R10, R11, E(R12), D(R13), L(R14), H(R15)
r3	V, U, T, W
rp, rp’[Note 2]	AX(RP0), BC(RP1), RP2, RP3, VP(RP4), UP(RP5), DE(RP6), HL(RP7)
rp1[Note 2]	AX(RP0), BC(RP1), RP2, RP3
rp2	VP(RP4), UP(RP5), DE(RP6), HL(RP7)
rg, rg'	VVP(RG4), UUP(RG5), TDE(RG6), WHL(RG7)
sfr	Special function register symbol (see the special function register table)
sfrp	Special function register symbol (16-bit manipulation register: see the special function register table)
post[Note 2]	AX(RP0), BC(RP1), RP2, RP3, VP(RP4), UP(RP5)/PSW, DE(RP6), HL(RP7) Multiple descriptions are possible. However, UP is restricted to the PUSH/POP instruction, and PSW is restricted to the PUSHU/POPU instruction.
mem	[TDE], [WHL], [TDE+], [WHL+], [TDE–], [WHL–], [VVP], [UUP]: register indirect addressing [TDE+byte], [WHL+byte], [SP+byte], [UUP+byte], [VVP+byte]: based addressing imm24[A], imm24[B], imm24[DE], imm24[HL]: indexed addressing [TDE+A], [TDE+B], [TDE+C], [WHL+A], [WHL+B], [WHL+C], [VVP+DE], [VVP+HL]: based indexed addressing
mem1	Everything under mem except [WHL+] and [WHL–]
mem2	[TDE], [WHL]
mem3	[AX], [BC], [RP2], [RP3], [VVP], [UUP], [TDE], [WHL]

Notes 1. By setting the RSS bit to 1, R4 to R7 can be used as X, A, C, and B. Use this function only when 78K/III Series programs are used.

2. By setting the RSS bit to 1, RP2 and RP3 can be used as AX and BC. Use this function only when 78K/III Series programs are used.

(1) Operand expression format and description (2/2)

Expression Format	Description
Note	
saddr, saddr'	FD20H - FF1FH Immediate data or label
saddr1	FE00H - FEFFH Immediate data or label
saddr2	FD20H - FDFFH, FF00H - FF1FH Immediate data or label
saddrp	FD20H - FF1EH Immediate data or label (when manipulating 16 bits)
saddrp1	FE00H - FEFFH Immediate data or label (when manipulating 16 bits)
saddrp2	FD20H - FDFFH, FF00H - FF1EH Immediate data or label (when manipulating 16 bits)
saddrg	FD20H - FEFDH Immediate data or label (when manipulating 24 bits)
saddrg1	FE00H - FEFDH Immediate data or label (when manipulating 24 bits)
saddrg2	FD20H - FDFFH Immediate data or label (when manipulating 24 bits)
addr24	0H - FFFFFFH Immediate data or label
addr20	0H - FFFFFH Immediate data or label
addr16	0H - FFFFH Immediate data or label
addr11	800H - FFFH Immediate data or label
addr8	0FE00H - 0FEFFH[Note] Immediate data or label
addr5	40H - 7EH Immediate data or label
imm24	24-bit immediate data or label
word	16-bit immediate data or label
byte	8-bit immediate data or label
bit	3-bit immediate data or label
n	3-bit immediate data
locaddr	00H or 0FH

Note When 00H is set by the LOCATION instruction, these addresses become the addresses shown here. When 0FH is set by the LOCATION instruction, the values of the addresses shown here added to F0000H become the addresses.

(2) Operand column symbols

Symbol	Description
+	Auto increment
–	Auto decrement
#	Immediate data
!	16-bit absolute address
!!	24-bit/20-bit absolute address
$	8-bit relative address
$!	16-bit relative address
/	Bit reversal
[ ]	Indirect addressing
[%]	24-bit indirect addressing

(3) Flag column symbols

Symbol	Description
(Blank)	Not changed
0	Clear to zero.
1	Set to one.
×	Set or clear based on the result.
P	Operate with the P/V flag as the parity flag.
V	Operate with the P/V flag as the overflow flag.
R	Restore the previously saved value.

(4) Operation column symbols

Symbol	Description
jdisp8	Two’s complement data (8 bits) of the relative address distance between the head address of the next instruction and the branch address
jdisp16	Two’s complement data (16 bits) of the relative address distance between the head address of the next instruction and the branch address
PC_{HW}	PC bits 16 to 19
PC_{LW}	PC bits 0 to 15

(5) Number of bytes in instruction that has mem in operand

mem Mode	Register Indirect Addressing		Based Addressing	Indexed Addressing	Based Indexed Addressing
No. of bytes	1	2[Note]	3	5	2

Note This becomes a 1-byte instruction only when [TDE], [WHL], [TDE+], [TDE–], [WHL+], or [WHL–] is described in mem in the MOV instruction.

(6) Number of bytes in instruction that has saddr, saddrp, r, or rp in operand

The number of bytes in an instruction that has saddr, saddrp, r, or rp in the operand is described in two parts divided by a slash (/). The following table shows the number of bytes in each one.

Description	No. of Bytes on Left Side	No. of Bytes on Right Side
saddr	saddr2	saddr1
saddrp	saddrp2	saddrp1
r	r1	r2
rp	rp1	rp2

(7) Descriptions of instructions with mem in operand and string instructions

The TDE, WHL, VVP, and UUP (24-bit registers) operands can be described by DE, HL, VP, and UP. However, when DE, HL, VP, and UP are described, they are handled as TDE, WHL, VVP, and UUP (24-bit registers).

26.2 List of Operations

(1) 8-bit data transfer instruction: MOV

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MOV	r, #byte	2/3	r ← byte					
	saddr, #byte	3/4	(saddr) ← byte					
	sfr, #byte	3	sfr ← byte					
	!addr16, #byte	5	(saddr16) ← byte					
	!!addr24, #byte	6	(addr24) ← byte					
	r, r'	2/3	r ← r'					
	A, r	1/2	A ← r					
	A, saddr2	2	A ← (saddr2)					
	r, saddr	3	r ← (saddr)					
	saddr2, A	2	(saddr2) ← A					
	saddr, r	3	(saddr) ← r					
	A, sfr	2	A ← sfr					
	r, sfr	3	r ← sfr					
	sfr, A	2	sfr ← A					
	sfr, r	3	sfr ← r					
	saddr, saddr'	4	(saddr) ← (saddr')					
	r, !addr16	4	r ← (addr16)					
	!addr16, r	4	(addr16) ← r					
	r, !!addr24	5	r ← (addr24)					
	!!addr24, r	5	(addr24) ← r					
	A, [saddrp]	2/3	A ← ((saddrp))					
	A, [%saddrg]	3/4	A ← ((saddrg))					
	A, mem	1-5	A ← (mem)					
	[saddrp], A	2/3	((saddrp)) ← A					
	[%saddrg], A	3/4	((saddrg)) ← A					
	mem, A	1-5	(mem) ← A	×	×	×	×	×
	PSWL, #byte	3	PSW$_L$ ← byte					
	PSWH, #byte	3	PSW$_H$ ← byte	×	×	×	×	×
	PSWL, A	2	PSW$_L$ ← A					
	PSWH, A	2	PSW$_H$ ← A					
	A, PSWL	2	A ← PSW$_L$					
	A, PSWH	2	A ← PSW$_H$					
	r3, #byte	3	r3 ← byte					
	A, r3	2	A ← r3					
	r3, A	2	r3 ← A					

(2) 16-bit data transfer instruction: MOVW

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MOVW	rp, #word	3	rp ← word					
	saddrp, #word	4/5	(saddrp) ← word					
	sfrp, #word	4	sfrp ← word					
	!addr16, #word	6	(addr16) ← word					
	!!addr24, #word	7	(addr24) ← word					
	rp, rp'	2	rp ← rp'					
	AX, saddrp2	2	AX ← (saddrp2)					
	rp, saddrp	3	rp ← (saddrp)					
	saddrp2, AX	2	(saddrp2) ← AX					
	saddrp, rp	3	(saddrp) ← rp					
	AX, sfrp	2	AX ← sfrp					
	rp, sfrp	3	rp ← sfrp					
	sfrp, AX	2	sfrp ← AX					
	sfrp, rp	3	sfrp ← rp					
	saddrp, saddrp'	4	(saddrp) ← (saddrp')					
	rp, !addr16	4	rp ← (addr16)					
	!addr16, rp	4	(addr16) ← rp					
	rp, !!addr24	5	rp ← (addr24)					
	!!addr24, rp	5	(addr24) ← rp					
	AX, [saddrp]	3/4	AX ← ((saddrp))					
	AX, [%saddrg]	3/4	AX ← ((saddrg))					
	AX, mem	2-5	AX ← (mem)					
	[saddrp], AX	3/4	((saddrp)) ← AX					
	[%saddrg], AX	3/4	((saddrg)) ← AX					
	mem, AX	2-5	(mem) ← AX					

(3) 24-bit data transfer instruction: MOVG

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MOVG	rg, #imm24	5	rg ← imm24					
	rg, rg'	2	rg ← rg'					
	rg, !!addr24	5	rg ← (addr24)					
	!!addr24, rg	5	(addr24) ← rg					
	rg, saddrg	3	rg ← (saddrg)					
	saddrg, rg	3	(saddrg) ← rg					
	WHL, [%saddrg]	3/4	WHL ← ((saddrg))					
	[%saddrg], WHL	3/4	((saddrg)) ← WHL					
	WHL, mem1	2-5	WHL ← (mem1)					
	mem1, WHL	2-5	(mem1) ← WHL					

(4) 8-bit data exchange instruction: XCH

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
XCH	r, r'	2/3	r ↔ r'					
	A, r	1/2	A ↔ r´					
	A, saddr2	2	A ↔ (saddr2)					
	r, saddr	3	r ↔ (saddr)					
	r, sfr	3	r ↔ sfr					
	saddr, saddr'	4	(saddr) ↔ (saddr')					
	r, !addr16	4	r ↔ (addr16)					
	r, !!addr24	5	r ↔ (addr24)					
	A, [saddrp]	2/3	A↔ ((saddrp))					
	A, [%saddrg]	3/4	A ↔ ((saddrg))					
	A, mem	2-5	A ↔ (mem)					

(5) 16-bit data exchange instruction: XCHW

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
XCHW	rp, rp'	2	rp ↔ rp'					
	AX, saddrp2	2	AX ↔ (saddrp2)					
	rp, saddrp	3	rp ↔ (saddrp)					
	rp, sfrp	3	rp ↔ sfrp					
	AX, [saddrp]	3/4	AX ↔ ((saddrp))					
	AX, [%saddrg]	3/4	AX ↔ ((saddrg))					
	AX, !addr16	4	AX ↔ (addr16)					
	AX, !!addr24	5	AX ↔ (addr24)					
	saddrp, saddrp'	4	(saddrp) ↔ (saddrp')					
	AX, mem	2-5	AX ↔ (mem)					

(6) 8-bit arithmetic instructions: ADD, ADDC, SUB, SUBC, CMP, AND, OR, XOR

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ADD	A, #byte	2	A, CY ← A + byte	×	×	×	V	×
	r, #byte	3	r, CY ← r + byte	×	×	×	V	×
	saddr, #byte	3/4	(saddr), CY ← (saddr) + byte	×	×	×	V	×
	sfr, #byte	4	sfr, CY ← sfr + byte	×	×	×	V	×
	r, r'	2/3	r, CY ← r + r'	×	×	×	V	×
	A, saddr2	2	A, CY ← A + (saddr2)	×	×	×	V	×
	r, saddr	3	r, CY ← r + (saddr)	×	×	×	V	×
	saddr, r	3	(saddr), CY ← (saddr) + r	×	×	×	V	×
	r, sfr	3	r, CY ← r + sfr	×	×	×	V	×
	sfr, r	3	sfr, CY ← sfr + r	×	×	×	V	×
	saddr, saddr'	4	(saddr), CY ← (saddr) + (saddr')	×	×	×	V	×
	A, [saddrp]	3/4	A, CY ← A + ((saddrp))	×	×	×	V	×
	A, [%saddrg]	3/4	A, CY ← A + ((saddrg))	×	×	×	V	×
	[saddrp], A	3/4	((saddrp)), CY ← ((saddrp)) + A	×	×	×	V	×
	[%saddrg], A	3/4	((saddrg)), CY ← ((saddrg)) + A	×	×	×	V	×
	A, !addr16	4	A, CY ← A + (addr16)	×	×	×	V	×
	A, !!addr24	5	A, CY ← A + (addr24)	×	×	×	V	×
	!addr16, A	4	(addr16), CY ← (addr16) + A	×	×	×	V	×
	!!addr24, A	5	(addr24), CY ← (addr24) + A	×	×	×	V	×
	A, mem	2-5	A, CY ← A + (mem)	×	×	×	V	×
	mem, A	2-5	(mem), CY ← (mem) + A	×	×	×	V	×

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ADDC	A, #byte	2	A, CY ← A + byte + CY	×	×	×	V	×
	r, #byte	3	r, CY ← r + byte + CY	×	×	×	V	×
	saddr, #byte	3/4	(saddr), CY ← (saddr) + byte + CY	×	×	×	V	×
	sfr, #byte	4	sfr, CY ← sfr + byte + CY	×	×	×	V	×
	r, r'	2/3	r, CY ← r + r' + CY	×	×	×	V	×
	A, saddr2	2	A, CY ← A + (saddr2) + CY	×	×	×	V	×
	r, saddr	3	r, CY ← r + (saddr) + CY	×	×	×	V	×
	saddr, r	3	(saddr), CY ← (saddr) + r + CY	×	×	×	V	×
	r, sfr	3	r, CY ← r + sfr + CY	×	×	×	V	×
	sfr, r	3	sfr, CY ← sfr + r + CY	×	×	×	V	×
	saddr, saddr'	4	(saddr), CY ← (saddr) + (saddr') + CY	×	×	×	V	×
	A, [saddrp]	3/4	A, CY ← A + ((saddrp)) + CY	×	×	×	V	×
	A, [%saddrg]	3/4	A, CY ← A + ((saddrg)) + CY	×	×	×	V	×
	[saddrp], A	3/4	((saddrp)), CY ← ((saddrp)) + A + CY	×	×	×	V	×
	[%saddrg], A	3/4	((saddrg)), CY ← ((saddrg)) + A + CY	×	×	×	V	×
	A, !addr16	4	A, CY ← A + (addr16) + CY	×	×	×	V	×
	A, !!addr24	5	A, CY ← A + (addr24) +CY	×	×	×	V	×
	!addr16, A	4	(addr16), CY ← (addr16) + A + CY	×	×	×	V	×
	!!addr24, A	5	(addr24), CY ← (addr24) + A + CY	×	×	×	V	×
	A, mem	2-5	A, CY ← A + (mem) + CY	×	×	×	V	×
	mem, A	2-5	(mem), CY ← (mem) + A + CY	×	×	×	V	×

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
SUB	A, #byte	2	A, CY ← A – byte	×	×	×	V	×
	r, #byte	3	r, CY ← r – byte	×	×	×	V	×
	saddr, #byte	3/4	(saddr), CY ← (saddr) – byte	×	×	×	V	×
	sfr, #byte	4	sfr, CY ← sfr – byte	×	×	×	V	×
	r, r'	2/3	r, CY ← r – r'	×	×	×	V	×
	A, saddr2	2	A, CY ← A – (saddr2)	×	×	×	V	×
	r, saddr	3	r, CY ← r – (saddr)	×	×	×	V	×
	saddr, r	3	(saddr), CY ← (saddr) – r	×	×	×	V	×
	r, sfr	3	r, CY ← r – sfr	×	×	×	V	×
	sfr, r	3	sfr, CY ← sfr – r	×	×	×	V	×
	saddr, saddr'	4	(saddr), CY ← (saddr) – (saddr')	×	×	×	V	×
	A, [saddrp]	3/4	A, CY ← A – ((saddrp))	×	×	×	V	×
	A, [%saddrg]	3/4	A, CY ← A – ((saddrg))	×	×	×	V	×
	[saddrp], A	3/4	((saddrp)), CY ← ((saddrp)) – A	×	×	×	V	×
	[%saddrg], A	3/4	((saddrg)), CY ← ((saddrg)) – A	×	×	×	V	×
	A, !addr16	4	A, CY ← A – (addr16)	×	×	×	V	×
	A, !!addr24	5	A, CY ← A – (addr24)	×	×	×	V	×
	!addr16, A	4	(addr16), CY ← (addr16) – A	×	×	×	V	×
	!!addr24, A	5	(addr24), CY ← (addr24) – A	×	×	×	V	×
	A, mem	2-5	A, CY ← A – (mem)	×	×	×	V	×
	mem, A	2-5	(mem), CY ← (mem) – A	×	×	×	V	×

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
SUBC	A, #byte	2	A, CY ← A – byte – CY	×	×	×	V	×
	r, #byte	3	r, CY ← r – byte – CY	×	×	×	V	×
	saddr, #byte	3/4	(saddr), CY ← (saddr) – byte – CY	×	×	×	V	×
	sfr, #byte	4	sfr, CY ← sfr – byte – CY	×	×	×	V	×
	r, r'	2/3	r, CY ← r – r' – CY	×	×	×	V	×
	A, saddr2	2	A, CY ← A – (saddr2) – CY	×	×	×	V	×
	r, saddr	3	r, CY ← r – (saddr) – CY	×	×	×	V	×
	saddr, r	3	(saddr), CY ← (saddr) – r – CY	×	×	×	V	×
	r, sfr	3	r, CY ← r – sfr – CY	×	×	×	V	×
	sfr, r	3	sfr, CY ← sfr – r – CY	×	×	×	V	×
	saddr, saddr'	4	(saddr), CY ← (saddr) – (saddr') – CY	×	×	×	V	×
	A, [saddrp]	3/4	A, CY ← A – ((saddrp)) – CY	×	×	×	V	×
	A, [%saddrg]	3/4	A, CY ← A – ((saddrg)) – CY	×	×	×	V	×
	[saddrp], A	3/4	((saddrp)), CY ← ((saddrp)) – A – CY	×	×	×	V	×
	[%saddrg], A	3/4	((saddrg)), CY ← ((saddrg)) – A – CY	×	×	×	V	×
	A, !addr16	4	A, CY ← A – (addr16) – CY	×	×	×	V	×
	A, !!addr24	5	A, CY ← A – (addr24) – CY	×	×	×	V	×
	!addr16, A	4	(addr16), CY ← (addr16) – A – CY	×	×	×	V	×
	!!addr24, A	5	(addr24), CY ← (addr24) – A – CY	×	×	×	V	×
	A, mem	2-5	A, CY ← A – (mem) – CY	×	×	×	V	×
	mem, A	2-5	(mem), CY ← (mem) – A – CY	×	×	×	V	×

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
CMP	A, #byte	2	A – byte	×	×	×	V	×
	r, #byte	3	r – byte	×	×	×	V	×
	saddr, #byte	3/4	(saddr) – byte	×	×	×	V	×
	sfr, #byte	4	sfr – byte	×	×	×	V	×
	r, r'	2/3	r – r'	×	×	×	V	×
	A, saddr2	2	A – (saddr2)	×	×	×	V	×
	r, saddr	3	r – (saddr)	×	×	×	V	×
	saddr, r	3	(saddr) – r	×	×	×	V	×
	r, sfr	3	r – sfr	×	×	×	V	×
	sfr, r	3	sfr – r	×	×	×	V	×
	saddr, saddr'	4	(saddr) – (saddr')	×	×	×	V	×
	A, [saddrp]	3/4	A – ((saddrp))	×	×	×	V	×
	A, [%saddrg]	3/4	A – ((saddrg))	×	×	×	V	×
	[saddrp], A	3/4	((saddrp)) – A	×	×	×	V	×
	[%saddrg], A	3/4	((saddrg)) – A	×	×	×	V	×
	A, !addr16	4	A – (addr16)	×	×	×	V	×
	A, !!addr24	5	A – (addr24)	×	×	×	V	×
	!addr16, A	4	(addr16) – A	×	×	×	V	×
	!!addr24, A	5	(addr24) – A	×	×	×	V	×
	A, mem	2-5	A – (mem)	×	×	×	V	×
	mem, A	2-5	(mem) – A	×	×	×	V	×

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
AND	A, #byte	2	A ← A∧byte	×	×		P	
	r, #byte	3	r ← r∧byte	×	×		P	
	saddr, #byte	3/4	(saddr) ← (saddr) ∧byte	×	×		P	
	sfr, #byte	4	sfr ← sfr∧byte	×	×		P	
	r, r'	2/3	r ← r∧r'	×	×		P	
	A, saddr2	2	A ← A∧ (saddr2)	×	×		P	
	r, saddr	3	r ← r∧ (saddr)	×	×		P	
	saddr, r	3	(saddr) ← (saddr) ∧r	×	×		P	
	r, sfr	3	r ← r∧sfr	×	×		P	
	sfr, r	3	sfr ← sfr∧r	×	×		P	
	saddr, saddr'	4	(saddr) ← (saddr) ∧ (saddr')	×	×		P	
	A, [saddrp]	3/4	A ← A∧ ((saddrp))	×	×		P	
	A, [%saddrg]	3/4	A ← A∧ ((saddrg))	×	×		P	
	[saddrp], A	3/4	((saddrp)) ← ((saddrp)) ∧A	×	×		P	
	[%saddrg], A	3/4	((saddrg)) ← ((saddrg)) ∧A	×	×		P	
	A, !addr16	4	A ← A∧ (addr16)	×	×		P	
	A, !!addr24	5	A ← A∧ (addr24)	×	×		P	
	!addr16, A	4	(addr16) ← (addr16) ∧A	×	×		P	
	!!addr24, A	5	(addr24) ← (addr24) ∧A	×	×		P	
	A, mem	2-5	A ← A∧(mem)	×	×		P	
	mem, A	2-5	(mem) ← (mem) ∧A	×	×		P	

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
OR	A, #byte	2	A ← A∨byte	×	×		P	
	r, #byte	3	r ← r∨byte	×	×		P	
	saddr, #byte	3/4	(saddr) ← (saddr) ∨byte	×	×		P	
	sfr, #byte	4	sfr ← sfr∨byte	×	×		P	
	r, r'	2/3	r ← r∨r'	×	×		P	
	A, saddr2	2	A ← A∨ (saddr2)	×	×		P	
	r, saddr	3	r ← r∨ (saddr)	×	×		P	
	saddr, r	3	(saddr) ← (saddr) ∨r	×	×		P	
	r, sfr	3	r ← r∨sfr	×	×		P	
	sfr, r	3	sfr ← sfr∨r	×	×		P	
	saddr, saddr'	4	(saddr) ← (saddr) ∨ (saddr')	×	×		P	
	A, [saddrp]	3/4	A ← A∨ ((saddrp))	×	×		P	
	A, [%saddrg]	3/4	A ← A∨ ((saddrg))	×	×		P	
	[saddrp], A	3/4	((saddrp)) ← ((addrp)) ∨A	×	×		P	
	[%saddrg], A	3/4	((saddrg)) ← ((addrg)) ∨A	×	×		P	
	A, !addr16	4	A ← A∨ (addr16)	×	×		P	
	A, !!addr24	5	A ← A∨ (addr24)	×	×		P	
	!addr16, A	4	(addr16) ← (addr16) ∨A	×	×		P	
	!!addr24, A	5	(addr24) ← (addr24) ∨A	×	×		P	
	A, mem	2-5	A ← A∨ (mem)	×	×		P	
	mem, A	2-5	(mem) ← (mem) ∨A	×	×		P	

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
XOR	A, #byte	2	A ← A ∀ byte	×	×		P	
	r, #byte	3	r ← r ∀ byte	×	×		P	
	saddr, #byte	3/4	(saddr) ← (saddr) ∀ byte	×	×		P	
	sfr, #byte	4	sfr ← sfr ∀ byte	×	×		P	
	r, r'	2/3	r ← r ∀ r'	×	×		P	
	A, saddr2	2	A ← A ∀ (saddr2)	×	×		P	
	r, saddr	3	r ← r ∀ (saddr)	×	×		P	
	saddr, r	3	(saddr) ← (saddr) ∀ r	×	×		P	
	r, sfr	3	r ← r ∀ sfr	×	×		P	
	sfr, r	3	sfr ← sfr ∀ r	×	×		P	
	saddr, saddr'	4	(saddr) ← (saddr) ∀ (saddr')	×	×		P	
	A, [saddrp]	3/4	A ← A ∀ ((saddrp))	×	×		P	
	A, [%saddrg]	3/4	A ← A ∀ ((saddrg))	×	×		P	
	[saddrp], A	3/4	((saddrp)) ← ((saddrp)) ∀ A	×	×		P	
	[%saddrg], A	3/4	((saddrg)) ← ((saddrg)) ∀ A	×	×		P	
	A, !addr16	4	A ← A ∀ (addr16)	×	×		P	
	A, !!addr24	5	A ← A ∀ (addr24)	×	×		P	
	!addr16, A	4	(addr16) ← (addr16) ∀ A	×	×		P	
	!!addr24, A	5	(addr24) ← (addr24) ∀ A	×	×		P	
	A, mem	2-5	A ← A ∀ (mem)	×	×		P	
	mem, A	2-5	(mem) ← (mem) ∀ A	×	×		P	

(7) 16-bit arithmetic instructions: ADDW, SUBW, CMPW

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ADDW	AX, #word	3	AX, CY ← AX + word	×	×	×	V	×
	rp, #word	4	rp, CY ← rp + word	×	×	×	V	×
	rp, rp'	2	rp, CY ← rp + rp'	×	×	×	V	×
	AX, saddrp2	2	AX, CY ← AX + (saddrp2)	×	×	×	V	×
	rp, saddrp	3	rp, CY ← rp + (saddrp)	×	×	×	V	×
	saddrp, rp	3	(saddrp), CY ← (saddrp) + rp	×	×	×	V	×
	rp, sfrp	3	rp, CY ← rp + sfrp	×	×	×	V	×
	sfrp, rp	3	sfrp, CY ← sfrp + rp	×	×	×	V	×
	saddrp, #word	4/5	(saddrp), CY ← (saddrp) + word	×	×	×	V	×
	sfrp, #word	5	sfrp, CY ← sfrp + word	×	×	×	V	×
	saddrp, saddrp'	4	(saddrp), CY ← (saddrp) + (saddrp')	×	×	×	V	×
SUBW	AX, #word	3	AX, CY ← AX – word	×	×	×	V	×
	rp, #word	4	rp, CY ← rp – word	×	×	×	V	×
	rp, rp'	2	rp, CY ← rp – rp'	×	×	×	V	×
	AX, saddrp2	2	AX, CY ← AX – (saddrp2)	×	×	×	V	×
	rp, saddrp	3	rp, CY ← rp – (saddrp)	×	×	×	V	×
	saddrp, rp	3	(saddrp), CY ← (saddrp) – rp	×	×	×	V	×
	rp, sfrp	3	rp, CY ← rp – sfrp	×	×	×	V	×
	sfrp, rp	3	sfrp, CY ← sfrp – rp	×	×	×	V	×
	saddrp, #word	4/5	(saddrp), CY ← (saddrp) – word	×	×	×	V	×
	sfrp, #word	5	sfrp, CY ← sfrp – word	×	×	×	V	×
	saddrp, saddrp'	4	(saddrp), CY ← (saddrp) – (saddrp')	×	×	×	V	×
CMPW	AX, #word	3	AX – word	×	×	×	V	×
	rp, #word	4	rp – word	×	×	×	V	×
	rp, rp'	2	rp – rp'	×	×	×	V	×
	AX, saddrp2	2	AX – (saddrp2)	×	×	×	V	×
	rp, saddrp	3	rp – (saddrp)	×	×	×	V	×
	saddrp, rp	3	(saddrp) – rp	×	×	×	V	×
	rp, sfrp	3	rp – sfrp	×	×	×	V	×
	sfrp, rp	3	sfrp – rp	×	×	×	V	×
	saddrp, #word	4/5	(saddrp) – word	×	×	×	V	×
	sfrp, #word	5	sfrp – word	×	×	×	V	×
	saddrp, saddrp'	4	(saddrp) – (saddrp')	×	×	×	V	×

(8) 24-bit arithmetic instructions: ADDG, SUBG

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ADDG	rg, rg'	2	rg, CY ← rg + rg'	×	×	×	V	×
	rg, #imm24	5	rg, CY ← rg + imm24	×	×	×	V	×
	WHL, saddrg	3	WHL, CY ← WHL + (saddrg)	×	×	×	V	×
SUBG	rg, rg'	2	rg, CY ← rg – rg'	×	×	×	V	×
	rg, #imm24	5	rg, CY ← rg – imm24	×	×	×	V	×
	WHL, saddrg	3	WHL, CY ← WHL – (saddrg)	×	×	×	V	×

(9) Multiply/divide instructions: MULU, MULUW, MULW, DIVUW, DIVUX

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MULU	r	2/3	AX ← A × r					
MULUW	rp	2	AX (high order), rp (low order) ← AX × rp					
MULW	rp	2	AX (high order), rp (low order) ← AX × rp					
DIVUW	r	2/3	AX (quotient), r (remainder) ← AX ÷ r[Note 1]					
DIVUX	rp	2	AXDE (quotient), rp (remainder) ← AXDE ÷ rp[Note 2]					

Notes 1. When r = 0, r ← X, AX ← FFFFH

2. When rp = 0, rp ← DE, AXDE ← FFFFFFFFH

(10) Special arithmetic instructions: MACW, MACSW, SACW

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MACW	byte	3	AXDE ← (B) × (C) + AXDE, B ← B + 2, C ← C + 2, byte ← byte – 1 End if (byte = 0 or P/V = 1)	×	×	×	V	×

MACSW	byte	3	AXDE ← (B) × (C) + AXDE, B ← B + 2,
			C ← C + 2, byte ← byte – 1
			if byte = 0 then End
			if P/V = 1 then if overflow AXDE ← 7FFFFFFFH, End
			if underflow AXDE ← 80000000H, End
				×	×	×	V	×

SACW	[TDE+], [WHL+]	4	AX ← | (TDE) – (WHL) | + AX, TDE ← TDE + 2, WHL ← WHL + 2 C ← C – 1 End if (C = 0 or CY = 1)	×	×	×	V	×

(11) Increment and decrement instructions: INC, DEC, INCW, DECW, INCG, DECG

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
INC	r	1/2	r ← r + 1	×	×	×	V	
	saddr	2/3	(saddr) ← (saddr) + 1	×	×	×	V	
DEC	r	1/2	r ← r – 1	×	×	×	V	
	saddr	2/3	(saddr) ← (saddr) – 1	×	×	×	V	
INCW	rp	2/1	rp ← rp + 1					
	saddrp	3/4	(saddrp) ← (saddrp) + 1					
DECW	rp	2/1	rp ← rp – 1					
	saddrp	3/4	(saddrp) ← (saddrp) – 1					
INCG	rg	2	rg ← rg + 1					
DECG	rg	2	rg ← rg – 1					

(12) Decimal adjust instructions: ADJBA, ADJBS, CVTBW

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ADJBA		2	Decimal Adjust Accumulator after Addition	×	×	×	P	×
ADJBS		2	Decimal Adjust Accumulator after Subtract	×	×	×	P	×
CVTBW		1	X ← A, A ← 00H if A_7 = 0 X ← A, A ← FFH if A_7 = 1					

(13) Shift and rotate instructions: ROR, ROL, RORC, ROLC, SHR, SHL, SHRW, SHLW, ROR4, ROL4

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
ROR	r, n	2/3	$(CY, r_7 \leftarrow r_0, r_{m-1} \leftarrow r_m) \times n$ n = 0 to 7				P	×
ROL	r, n	2/3	$(CY, r_0 \leftarrow r_7, r_{m+1} \leftarrow r_m) \times n$ n = 0 to 7				P	×
RORC	r, n	2/3	$(CY \leftarrow r_0, r_7 \leftarrow CY, r_{m-1} \leftarrow r_m) \times n$ n = 0 to 7				P	×
ROLC	r, n	2/3	$(CY \leftarrow r_7, r_0 \leftarrow CY, r_{m+1} \leftarrow r_m) \times n$ n = 0 to 7				P	×
SHR	r, n	2/3	$(CY \leftarrow r_0, r_7 \leftarrow 0, r_{m-1} \leftarrow r_m) \times n$ n = 0 to 7	×	×	0	P	×
SHL	r, n	2/3	$(CY \leftarrow r_7, r_0 \leftarrow 0, r_{m+1} \leftarrow r_m) \times n$ n = 0 to 7	×	×	0	P	×
SHRW	rp, n	2	$(CY \leftarrow rp_0, rp_{15} \leftarrow 0, rp_{m-1} \leftarrow rp_m) \times n$ n = 0 to 7	×	×	0	P	×
SHLW	rp, n	2	$(CY \leftarrow rp_{15}, rp_0 \leftarrow 0, rp_{m+1} \leftarrow rp_m) \times n$ n = 0 to 7	×	×	0	P	×
ROR4	mem3	2	$A_{3-0} \leftarrow (mem3)_{3-0}, (mem3)_{7-4} \leftarrow A_{3-0}, (mem3)_{3-0} \leftarrow (mem3)_{7-4}$					
ROL4	mem3	2	$A_{3-0} \leftarrow (mem3)_{7-4}, (mem3)_{3-0} \leftarrow A_{3-0}, (mem3)_{7-4} \leftarrow (mem3)_{3-0}$					

(14) Bit manipulation instructions: MOV1, AND1, OR1, XOR1, NOT1, SET1, CLR1

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MOV1	CY, saddr.bit	3/4	CY ← (saddr.bit)					×
	CY, sfr.bit	3	CY ← sfr.bit					×
	CY, X.bit	2	CY ← X.bit					×
	CY, A.bit	2	CY ← A.bit					×
	CY, PSWL.bit	2	CY ← PSW_L.bit					×
	CY, PSWH.bit	2	CY ← PSW_H.bit					×
	CY, !addr16.bit	5	CY ← !addr16.bit					×
	CY, !!addr24.bit	2	CY ← !!addr24.bit					×
	CY, mem2.bit	2	CY ← mem2.bit					×
	saddr.bit, CY	3/4	(saddr.bit) ← CY					
	sfr.bit, CY	3	sfr.bit ← CY					
	X.bit, CY	2	X.bit ← CY					
	A.bit, CY	2	A.bit ← CY					
	PSWL.bit, CY	2	PSW_L.bit ← CY	×	×	×	×	×
	PSWH.bit, CY	2	PSW_H.bit ← CY					
	!addr16.bit, CY	5	!addr16.bit ← CY					
	!!addr24.bit, CY	6	!!addr24.bit ← CY					
	mem2.bit, CY	2	mem2.bit ← CY					

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
AND1	CY, saddr.bit	3/4	CY ← CY∧ (saddr.bit)					×
	CY, /saddr.bit	3/4	CY ← CY∧ $(\overline{\text{saddr.bit}})$					×
	CY, sfr.bit	3	CY ← CY∧ sfr.bit					×
	CY, /sfr.bit	3	CY ← CY∧ $\overline{\text{sfr.bit}}$					×
	CY, X.bit	2	CY ← CY∧ X.bit					×
	CY, /X.bit	2	CY ← CY∧ $\overline{\text{X.bit}}$					×
	CY, A.bit	2	CY ← CY∧ A.bit					×
	CY, /A.bit	2	CY ← CY∧ $\overline{\text{A.bit}}$					×
	CY, PSWL.bit	2	CY ← CY∧ PSW_L.bit					×
	CY, /PSWL.bit	2	CY ← CY∧ $\overline{\text{PSW}_\text{L}\text{.bit}}$					×
	CY, PSWH.bit	2	CY ← CY∧ PSW_H.bit					×
	CY, /PSWH.bit	2	CY ← CY∧ $\overline{\text{PSW}_\text{H}\text{.bit}}$					×
	CY, !addr16.bit	5	CY ← CY∧ !addr16.bit					×
	CY, /!addr16.bit	5	CY ← CY∧ $\overline{\text{!addr16.bit}}$					×
	CY, !!addr24.bit	2	CY ← CY∧ !!addr24.bit					×
	CY, /!!addr24.bit	6	CY ← CY∧ $\overline{\text{!!addr24.bit}}$					×
	CY, mem2.bit	2	CY ← CY∧ mem2.bit					×
	CY, /mem2.bit	2	CY ← CY∧ $\overline{\text{mem2.bit}}$					×
OR1	CY, saddr.bit	3/4	CY ← CY∨ (saddr.bit)					×
	CY, /saddr.bit	3/4	CY ← CY∨ $(\overline{\text{saddr.bit}})$					×
	CY, sfr.bit	3	CY ← CY∨ sfr.bit					×
	CY, /sfr.bit	3	CY ← CY∨ $\overline{\text{sfr.bit}}$					×
	CY, X.bit	2	CY ← CY∨ X.bit					×
	CY, /X.bit	2	CY ← CY∨ $\overline{\text{X.bit}}$					×
	CY, A.bit	2	CY ← CY∨ A.bit					×
	CY, /A.bit	2	CY ← CY∨ $\overline{\text{A.bit}}$					×
	CY, PSWL.bit	2	CY ← CY∨ PSW_L.bit					×
	CY, /PSWL.bit	2	CY ← CY∨ $\overline{\text{PSW}_\text{L}\text{.bit}}$					×
	CY, PSWH.bit	2	CY ← CY∨ PSW_H.bit					×
	CY, /PSWH.bit	2	CY ← CY∨ $\overline{\text{PSW}_\text{H}\text{.bit}}$					×
	CY, !addr16.bit	5	CY ← CY∨ !addr16.bit					×
	CY, /!addr16.bit	5	CY ← CY∨ $\overline{\text{!addr16.bit}}$					×
	CY, !!addr24.bit	2	CY ← CY∨ !!addr24.bit					×
	CY, /!!addr24.bit	6	CY ← CY∨ $\overline{\text{!!addr24.bit}}$					×
	CY, mem2.bit	2	CY ← CY∨ mem2.bit					×
	CY, /mem2.bit	2	CY ← CY∨ $\overline{\text{mem2.bit}}$					×

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
XOR1	CY, saddr.bit	3/4	CY ← CY ∀ (saddr.bit)					×
	CY, sfr.bit	3	CY ← CY ∀ sfr.bit					×
	CY, X.bit	2	CY ← CY ∀ X.bit					×
	CY, A.bit	2	CY ← CY ∀ A.bit					×
	CY, PSWL.bit	2	CY ← CY ∀ PSWL.bit					×
	CY, PSWH.bit	2	CY ← CY ∀ PSWH.bit					×
	CY, !addr16.bit	5	CY ← CY ∀ !addr16.bit					×
	CY, !!addr24.bit	2	CY ← CY ∀ !!addr24.bit					×
	CY, mem2.bit	2	CY ← CY ∀ mem2.bit					×
NOT1	saddr.bit	3/4	(saddr.bit) ← ($\overline{\text{saddr.bit}}$)					
	sfr.bit	3	sfr.bit ← $\overline{\text{sfr.bit}}$					
	X.bit	2	X.bit ← $\overline{\text{X.bit}}$					
	A.bit	2	A.bit ← $\overline{\text{A.bit}}$					
	PSWL.bit	2	PSWL.bit ← $\overline{\text{PSWL.bit}}$	×	×	×	×	×
	PSWH.bit	2	PSWH.bit ← $\overline{\text{PSWH.bit}}$					
	!addr16.bit	5	!addr16.bit ← $\overline{\text{!addr16.bit}}$					
	!!addr24.bit	2	!!addr24.bit ← $\overline{\text{!!addr24.bit}}$					
	mem2.bit	2	mem2.bit ← $\overline{\text{mem2.bit}}$					
	CY	1	CY ← $\overline{\text{CY}}$					×
SET1	saddr.bit	2/3	(saddr.bit) ← 1					
	sfr.bit	3	sfr.bit ← 1					
	X.bit	2	X.bit ← 1					
	A.bit	2	A.bit ← 1					
	PSWL.bit	2	PSWL.bit ← 1	×	×	×	×	×
	PSWH.bit	2	PSWH.bit ← 1					
	!addr16.bit	5	!addr16.bit ← 1					
	!!addr24.bit	2	!!addr24.bit ← 1					
	mem2.bit	2	mem2.bit ←1					
	CY	1	CY ← 1					1
CLR1	saddr.bit	2/3	(saddr.bit) ← 0					
	sfr.bit	3	sfr.bit ← 0					
	X.bit	2	X.bit ← 0					
	A.bit	2	A.bit ← 0					
	PSWL.bit	2	PSWL.bit ← 0	×	×	×	×	×
	PSWH.bit	2	PSWH.bit ← 0					
	!addr16.bit	5	!addr16.bit ← 0					
	!!addr24.bit	2	!!addr24.bit ← 0					
	mem2.bit	2	mem2.bit ←0					
	CY	1	CY ← 0					0

(15) Stack manipulation instructions: PUSH, PUSHU, POP, POPU, MOVG, ADDWG, SUBWG, INCG, DECG

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
PUSH	PSW	1	(SP – 2) ← PSW, SP ← SP – 2					
	sfrp	3	(SP – 2) ← sfrp, SP ← SP – 2					
	sfr	3	(SP – 1) ← sfr, SP ← SP – 1					
	post	2	{(SP – 2) ← post, SP ← SP – 2} × m[Note]					
	rg	2	(SP – 3) ← rg, SP ← SP – 3					
PUSHU	post	2	{(UUP – 2) ← post, UUP ← UUP – 2} × m[Note]					
POP	PSW	1	PSW ← (SP), SP ← SP + 2	R	R	R	R	R
	sfrp	3	sfrp ← (SP), SP ← SP + 2					
	sfr	3	sfr ← (SP), SP ← SP + 1					
	post	2	{post ← (SP), SP ← SP + 2} × m[Note]					
	rg	2	rg ← (SP), SP ← SP + 3					
POPU	post	2	{post ← (UUP), UUP ← UUP + 2} × m[Note]					
MOVG	SP, #imm24	5	SP ← imm24					
	SP, WHL	2	SP ← WHL					
	WHL, SP	2	WHL ← SP					
ADDWG	SP, #word	4	SP ← SP + word					
SUBWG	SP, #word	4	SP ← SP – word					
INCG	SP	2	SP ← SP + 1					
DECG	SP	2	SP ← SP – 1					

Note m is the number of registers specified by post.

(16) Call return instructions: CALL, CALLF, CALLT, BRK, BRKCS, RET, RETI, RETB, RETCS, RETCSB

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
CALL	!addr16	3	(SP – 3) ← (PC + 3), SP ← SP – 3, PC$_{HW}$ ← 0, PC$_{LW}$ ← addr16					
	!!addr20	4	(SP – 3) ← (PC + 4), SP ← SP – 3, PC ← addr20					
	rp	2	(SP – 3) ← (PC + 2), SP ← SP – 3, PC$_{HW}$ ← 0, PC$_{LW}$ ← rp					
	rg	2	(SP – 3) ← (PC + 2), SP ← SP – 3, PC ← rg					
	[rp]	2	(SP – 3) ← (PC + 2), SP ← SP – 3, PC$_{HW}$ ← 0, PC$_{LW}$ ← (rp)					
	[rg]	2	(SP – 3) ← (PC + 2), SP ← SP – 3, PC ← (rg)					
	$!addr20	3	(SP – 3) ← (PC + 3), SP ← SP – 3, PC ← PC + 3 + jdisp16					
CALLF	!addr11	2	(SP – 3) ← (PC + 2), SP ← SP – 3 PC$_{19–12}$ ← 0, PC$_{11}$ ← 1, PC$_{10–0}$ ← addr11					
CALLT	[addr5]	1	(SP – 3) ← (PC + 1), SP ← SP – 3 PC$_{HW}$ ← 0, PC$_{LW}$ ← (addr5)					
BRK		1	(SP – 2) ← PSW, (SP – 1)$_{0–3}$ ←, (PC + 1)$_{HW}$, (SP – 4) ← (PC + 1)$_{LW}$, SP ← SP – 4 PC$_{HW}$ ← 0, PC$_{LW}$ ← (003EH)					
BRKCS	RBn	2	PC$_{LW}$ ← RP2, RP3 ← PSW, RBS2 – 0 ← n, RSS ← 0, IE ← 0, RP3$_{8–11}$ ← PC$_{HW}$, PC$_{HW}$ ← 0					
RET		1	PC ← (SP), SP ← SP + 3					
RETI		1	PC$_{LW}$ ← (SP), PC$_{HW}$ ← (SP + 3)$_{0–3}$, PSW ← (SP + 2), SP ← SP + 4 The flag with the highest priority that is set to 1 in ISPR is cleared to 0.	R	R	R	R	R
RETB		1	PC$_{LW}$ ← (SP), PC$_{HW}$ ← (SP + 3)$_{0–3}$, PSW ← (SP + 2), SP ← SP + 4	R	R	R	R	R
RETCS	!addr16	3	PSW ← RP3, PC$_{LW}$ ← RP2, RP2 ← addr16, PC$_{HW}$ ← RP3$_{8–11}$ The flag with the highest priority that is set to 1 in ISPR is cleared to 0.	R	R	R	R	R
RETCSB	!addr16	4	PSW ← RP3, PC$_{LW}$ ← RP2, RP2 ← addr16, PC$_{HW}$ ← RP3$_{8–11}$	R	R	R	R	R

(17) Unconditional branch instruction: BR

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
BR	!addr16	3	$PC_{HW} \leftarrow 0$, $PC_{LW} \leftarrow$ addr16					
	!!addr20	4	PC $\leftarrow$ addr20					
	rp	2	$PC_{HW} \leftarrow 0$, $PC_{LW} \leftarrow$ rp					
	rg	2	PC $\leftarrow$ rg					
	[rp]	2	$PC_{HW} \leftarrow 0$, $PC_{LW} \leftarrow$ (rp)					
	[rg]	2	PC $\leftarrow$ (rg)					
	$addr20	2	PC $\leftarrow$ PC + 2 + jdisp8					
	$!addr20	3	PC $\leftarrow$ PC + 3 + jdisp16					

(18) Conditional branch instructions: BNZ, BNE, BZ, BE, BNC, BNL, BC, BL, BNV, BPO, BV, BPE, BP, BN, BLT, BGE, BLE, BGT, BNH, BH, BF, BT, BTCLR, BFSET, DBNZ

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
BNZ	$addr20	2	PC ← PC + 2 + jdisp8 if Z = 0					
BNE								
BZ	$addr20	2	PC ← PC + 2 + jdisp8 if Z = 1					
BE								
BNC	$addr20	2	PC ← PC + 2 + jdisp8 if CY = 0					
BNL								
BC	$addr20	2	PC ← PC + 2 + jdisp8 if CY = 1					
BL								
BNV	$addr20	2	PC ← PC + 2 + jdisp8 if P/V = 0					
BPO								
BV	$addr20	2	PC ← PC + 2 + jdisp8 if P/V = 1					
BPE								
BP	$addr20	2	PC ← PC + 2 + jdisp8 if S = 0					
BN	$addr20	2	PC ← PC + 2 + jdisp8 if S = 1					
BLT	$addr20	3	PC ← PC + 3 + jdisp8 if P/V ∀S = 1					
BGE	$addr20	3	PC ← PC + 3 + jidsp8 if P/V ∀S = 0					
BLE	$addr20	3	PC ← PC + 3 + jdisp8 if (P/V ∀S) ∨Z = 1					
BGT	$addr20	3	PC ← PC + 3 + jidsp8 if (P/V ∀S) ∨Z = 0					
BNH	$addr20	3	PC ← PC + 3 + jdisp8 if Z∨CY = 1					
BH	$addr20	3	PC ← PC + 3 + jidsp8 if Z∨CY = 0					
BF	saddr.bit, $addr20	4/5	PC ← PC + 4$^{\text{Note}}$ + jdisp8 if (saddr.bit) = 0					
	sfr.bit, $addr20	4	PC ← PC + 4 + jdisp8 if sfr.bit = 0					
	X.bit, $addr20	3	PC ← PC + 3 + jdisp8 if X.bit = 0					
	A.bit, $addr20	3	PC ← PC + 3 + jdisp8 if A.bit = 0					
	PSWL.bit, $addr20	3	PC ← PC + 3 + jdisp8 if PSW_L.bit = 0					
	PSWH.bit, $addr20	3	PC ← PC + 3 + jdisp8 if PSW_H.bit = 0					
	!addr16.bit, $addr20	6	PC ← PC + 3 + jdisp8 if !addr16.bit = 0					
	!!addr24.bit, $addr20	3	PC ← PC + 3 + jdisp8 if !!addr24.bit = 0					
	mem2.bit, $addr20	3	PC ← PC + 3 + jdisp8 if mem2.bit = 0					

Note This is used when the number of bytes is four. When five, it becomes PC ← PC + 5 + jdisp8.

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
BT	saddr.bit, $addr20	3/4	PC ← PC + $3^{\text{Note 1}}$ + jdisp8 if (saddr.bit) = 1					
	sfr.bit, $addr20	4	PC ← PC + 4 + jdisp8 if sfr.bit = 1					
	X.bit, $addr20	3	PC ← PC + 3 + jdisp8 if X.bit = 1					
	A.bit, $addr20	3	PC ← PC + 3 + jdisp8 if A.bit = 1					
	PSWL.bit, $addr20	3	PC ← PC + 3 + jdisp8 if PSW_L.bit = 1					
	PSWH.bit, $addr20	3	PC ← PC + 3 + jdisp8 if PSW_H.bit = 1					
	!addr16.bit, $addr20	6	PC ← PC + 3 + jdisp8 if !addr16.bit = 1					
	!!addr24.bit, $addr20	3	PC ← PC + 3 + jdisp8 if !!addr24.bit = 1					
	mem2.bit, $addr20	3	PC ← PC + 3 + jdisp8 if mem2.bit = 1					
BTCLR	saddr.bit, $addr20	4/5	{PC ← PC + $4^{\text{Note 2}}$ + jdisp8, (saddr.bit) ← 0} if (saddr.bit = 1)					
	sfr.bit, $addr20	4	{PC ← PC + 4 + jdisp8, sfr.bit ← 0} if sfr. bit = 1					
	X.bit, $addr20	3	{PC ← PC + 3 + jdisp8, X.bit ← 0} if X.bit = 1					
	A.bit, $addr20	3	{PC ← PC + 3 + jdisp8, A.bit ← 0} if A.bit = 1					
	PSWL.bit, $addr20	3	{PC ← PC + 3 + jdisp8, PSW_L.bit ← 0} if PSW_L.bit = 1	×	×	×	×	×
	PSWH.bit, $addr20	3	{PC ← PC + 3 + jdisp8, PSW_H.bit ← 0} if PSW_H.bit = 1					
	!addr16.bit, $addr20	6	{PC ← PC + 3 + jdisp8, !addr16.bit ← 0} if !addr16.bit = 1					
	!!addr24.bit, $addr20	3	{PC ← PC + 3 + jdisp8, !!addr24.bit ← 0} if !!addr24.bit = 1					
	mem2.bit, $addr20	3	{PC ← PC + 3 + jdisp8, mem2.bit ← 0} if mem2.bit = 1					

Notes 1. This is used when the number of bytes is three. When four, it becomes PC ← PC + 4 + jdisp8.

2. This is used when the number of bytes is four. When five, it becomes PC ← PC + 5 + jdisp8.

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
BFSET	saddr.bit, $addr20	4/5	{PC ← PC + 4$^{\text{Note 2}}$ + jdisp8, (saddr.bit) ← 1} if (saddr.bit = 0)					
	sfr.bit, $addr20	4	{PC ← PC + 4 + jdisp8, sfr.bit ← 1} if sfr. bit = 0					
	X.bit, $addr20	3	{PC ← PC + 3 + jdisp8, X.bit ← 1} if X.bit = 0					
	A.bit, $addr20	3	{PC ← PC + 3 + jdisp8, A.bit ← 1} if A.bit = 0					
	PSWL.bit, $addr20	3	{PC ← PC + 3 + jdisp8, PSW$_L$.bit ← 1} if PSW$_L$.bit = 0	×	×	×	×	×
	PSWH.bit, $addr20	3	{PC ← PC + 3 + jdisp8, PSW$_H$.bit ← 1} if PSW$_H$.bit = 0					
	!addr16.bit, $addr20	6	{PC ← PC + 3 + jdisp8, !addr16.bit ← 1} if !addr16.bit = 0					
	!!addr24.bit, $addr20	3	{PC ← PC + 3 + jdisp8, !!addr24.bit ← 1} if !!addr24.bit = 0					
	mem2.bit, $addr20	3	{PC ← PC + 3 + jdisp8, mem2.bit ← 1} if mem2.bit = 0					
DBNZ	B, $addr20	2	B ← B – 1, PC ← PC + 2 + jdisp8 if B ≠ 0					
	C. $addr20	2	C ← C – 1, PC ← PC + 2 + jdisp8 if C ≠ 0					
	saddr, $addr20	3/4	(saddr) ← (saddr) – 1, PC ← PC + 3$^{\text{Note 1}}$ + jdisp8 if (saddr) ≠ 0					

Notes 1. This is used when the number of bytes is three. When four, it becomes PC ← PC + 4 + jdisp8.

2. This is used when the number of bytes is four. When five, it becomes PC ← PC + 5 + jdisp8.

(19) CPU control instructions: MOV, LOCATION, SEL, SWRS, NOP, EI, DI

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MOV	STBC, #byte	4	STBC ← byte					
	WDM, #byte	4	WDM ← byte					
LOCATION	locaddr	4	Specification of the high-order word of the location address of the SFR and internal data area					
SEL	RBn	2	RSS ← 0, RBS2 – 0 ← n					
	RBn, ALT	2	RSS ← 1, RBS2 – 0 ← n					
SWRS		2	RSS ← $\overline{\text{RSS}}$					
NOP		1	No operation					
EI		1	IE ← 1 (Enable interrupt)					
DI		1	IE ← 0 (Disable interrupt)					

(20) String instructions: MOVTBLW, MOVM, XCHM, MOVBK, XCHBK, CMPME, CMPMNE, CMPMC, CMPMNC, CMPBKE, CMPBKNE, CMPBKC, CMPBKNC

Mnemonic	Operand	Bytes	Operation	Flags				
				S	Z	AC	P/V	CY
MOVTBLW	!addr8, byte	4	(addr8 + 2) ← (addr8), byte ← byte – 1, addr8 ← addr8 – 2 End if byte = 0					
MOVM	[TDE+], A	2	(TDE) ← A, TDE ← TDE + 1, C ← C – 1 End if C = 0					
	[TDE–], A	2	(TDE) ← A, TDE ← TDE – 1, C ← C – 1 End if C = 0					
XCHM	[TDE+], A	2	(TDE) ↔ A, TDE ← TDE + 1, C ← C – 1 End if C = 0					
	[TDE–], A	2	(TDE) ↔ A, TDE ← TDE – 1, C ← C – 1 End if C = 0					
MOVBK	[TDE+], [WHL+]	2	(TDE) ← (WHL), TDE ← TDE + 1, WHL ← WHL + 1, C ← C – 1 End if C = 0					
	[TDE–], [WHL–]	2	(TDE) ← (WHL), TDE ← TDE – 1, WHL ← WHL – 1, C ← C – 1 End if C = 0					
XCHBK	[TDE+], [WHL+]	2	(TDE) ↔ (WHL), TDE ← TDE + 1, WHL ← WHL + 1, C ← C – 1 End if C = 0					
	[TDE–], [WHL–]	2	(TDE) ↔ (WHL), TDE ← TDE – 1, WHL ← WHL – 1, C ← C – 1 End if C = 0					
CMPME	[TDE+], A	2	(TDE) – A, TDE ← TDE + 1, C ← C – 1 End if C = 0 or Z = 0	×	×	×	V	×
	[TDE–], A	2	(TDE) – A, TDE ← TDE – 1, C ← C – 1 End if C = 0 or Z = 0	×	×	×	V	×
CMPMNE	[TDE+], A	2	(TDE) – A, TDE ← TDE + 1, C ← C – 1 End if C = 0 or Z = 1	×	×	×	V	×
	[TDE–], A	2	(TDE) – A, TDE ← TDE – 1, C ← C – 1 End if C = 0 or Z = 1	×	×	×	V	×
CMPMC	[TDE+], A	2	(TDE) – A, TDE ← TDE + 1, C ← C – 1 End if C = 0 or CY = 0	×	×	×	V	×
	[TDE–], A	2	(TDE) – A, TDE ← TDE – 1, C ← C – 1 End if C = 0 or CY = 0	×	×	×	V	×
CMPMNC	[TDE+], A	2	(TDE) – A, TDE ← TDE + 1, C ← C – 1 End if C = 0 or CY = 1	×	×	×	V	×
	[TDE–], A	2	(TDE) – A, TDE ← TDE – 1, C ← C – 1 End if C = 0 or CY = 1	×	×	×	V	×
CMPBKE	[TDE+], [WHL+]	2	(TDE) – (WHL), TDE ← TDE + 1, WHL ← WHL + 1, C ← C –1 End if C = 0 or Z = 0	×	×	×	V	×
	[TDE–], [WHL–]	2	(TDE) – (WHL), TDE ← TDE – 1, WHL ← WHL – 1, C ← C –1 End if C = 0 or Z = 0	×	×	×	V	×
CMPBKNE	[TDE+], [WHL+]	2	(TDE) – (WHL), TDE ← TDE + 1, WHL ← WHL + 1, C ← C –1 End if C = 0 or Z = 1	×	×	×	V	×
	[TDE–], [WHL–]	2	(TDE) – (WHL), TDE ← TDE – 1, WHL ← WHL – 1, C ← C –1 End if C = 0 or Z = 1	×	×	×	V	×
CMPBKC	[TDE+], [WHL+]	2	(TDE) – (WHL), TDE ← TDE + 1, WHL ← WHL + 1, C ← C –1 End if C = 0 or CY = 0	×	×	×	V	×
	[TDE–], [WHL–]	2	(TDE) – (WHL), TDE ← TDE – 1, WHL ← WHL – 1, C ← C –1 End if C = 0 or CY = 0	×	×	×	V	×
CMPBKNC	[TDE+], [WHL+]	2	(TDE) – (WHL), TDE ← TDE + 1, WHL ← WHL + 1, C ← C –1 End if C = 0 or CY = 1	×	×	×	V	×
	[TDE–], [WHL–]	2	(TDE) – (WHL), TDE ← TDE – 1, WHL ← WHL – 1, C ← C –1 End if C = 0 or CY = 1	×	×	×	V	×

26.3 Lists of Addressing Instructions

(1) 8-bit instructions (The values enclosed by parentheses are combined to express the A description as r.)
MOV, XCH, ADD, ADDC, SUB, SUBC, AND OR XOR, CMP, MULU, DIVUW, INC, DEC, ROR, ROL, RORC, ROLC, SHR, SHL, ROR4, ROL4, DBNZ, PUSH, POP, MOVM, XCHM, CMPME, CMPMNE, CMPMNC, CMPMC, MOVBK, XCHBK, CMPBKE, CMPBKNE, CMPBKNC, CMPBKC

Table 26-1. 8-Bit Addressing Instructions

Second operand First operand	#byte	A	r r'	saddr saddr'	sfr	!addr16 !!addr24	mem [saddrp] [%saddrg]	r3 PSWL PSWH	[WHL+] [WHL–]	n	None$^{\text{Note 2}}$
A	(MOV) ADD$^{\text{Note 1}}$	(MOV) (XCH) (ADD)$^{\text{Note 1}}$	MOV XCH (ADD)$^{\text{Note 1}}$	(MOV)$^{\text{Note 6}}$ (XCH)$^{\text{Note 6}}$ (ADD)$^{\text{Notes 1, 6}}$	MOV (XCH) (ADD)$^{\text{Note 1}}$	(MOV) (XCH) ADD$^{\text{Note 1}}$	MOV XCH ADD$^{\text{Note 1}}$	MOV	(MOV) (XCH) (ADD)$^{\text{Note 1}}$		
r	MOV ADD$^{\text{Note 1}}$	(MOV) (XCH) (ADD)$^{\text{Note 1}}$	MOV XCH ADD$^{\text{Note 1}}$	MOV XCH ADD$^{\text{Note 1}}$	MOV XCH ADD$^{\text{Note 1}}$	MOV XCH				ROR$^{\text{Note 3}}$	MULU DIVUW INC DEC
saddr	MOV ADD$^{\text{Note 1}}$	(MOV)$^{\text{Note 6}}$ (ADD)$^{\text{Note 1}}$	MOV ADD$^{\text{Note 1}}$	MOV XCH ADD$^{\text{Note 1}}$							INC DEC DBNZ
sfr	MOV ADD$^{\text{Note 1}}$	MOV (ADD)$^{\text{Note 1}}$	MOV ADD$^{\text{Note 1}}$								PUSH POP
!addr16 !!addr24	MOV	MOV ADD$^{\text{Note 1}}$	MOV								
mem [saddrp] [%saddrg]		MOV ADD$^{\text{Note 1}}$									
mem3											ROR4 ROL4
r3 PSWL PSWH	MOV	MOV									
B, C											DBNZ
STBC, WDM	MOV										
[TDE+] [TDE–]		(MOV) (ADD)$^{\text{Note 1}}$ MOVM$^{\text{Note 4}}$							MOVBK$^{\text{Note 5}}$		

Notes 1. ADDC, SUB, SUBC, AND, OR, XOR, and CMP are identical to ADD.

2. There is no second operand, or the second operand is not an operand address.

3. ROL, RORC, ROLC, SHR, and SHL are identical to ROR.

4. XCHM, CMPME, CMPMNE, CMPMNC, and CMPMC are identical to MOVM.

5. XCHBK, CMPBKE, CMPBKNE, CMPBKNC, and CMPBKC are identical to MOVBK.

6. When saddr is saddr2 in this combination, the instruction has a short code length.

(2) 16-bit instructions (The values enclosed by parentheses are combined to express AX description as rp.)
MOVM, XCHW, ADDW, SUBW, CMPW, MULUW, MULW, DIVUX, INCW, DECW, SHRW, SHLW, PUSH, POP, ADDWG, SUBWG, PUSHU, POPU, MOVTBLW, MACW, MACSW, SACW

Table 26-2. 16-Bit Addressing Instructions

Second operand First operand	#word	AX	rp rp'	saddrp saddrp'	sfrp	!addr16 !!addr24	mem [saddrp] [%saddrg]	[WHL+]	byte	n	None[Note 2]
AX	(MOVW) ADDW[Note 1]	(MOVW) (XCHW) (ADD)[Note 1]	(MOVW) (XCHW) (ADDW)[Note 1]	(MOVW)[Note 3] (XCHW)[Note 3] (ADDW)[Notes 1, 3]	MOVW (XCHW) (ADDW)[Note 1]	(MOVW) XCHW	MOVW XCHW	(MOVW) (XCHW)			
rp	MOVW ADDW[Note 1]	(MOVW) (XCHW) (ADDW)[Note 1]	MOVW XCHW ADDW[Note 1]	MOVW XCHW ADDW[Note 1]	MOVW XCHW ADDW[Note 1]	MOVW				SHRW SHLW	MULW[Note 4] INCW DECW
saddrp	MOVW ADDW[Note 1]	(MOVW)[Note 3] (ADDW)[Note 1]	MOVW ADDW[Note 1]	MOVW XCHW ADDW[Note 1]							INCW DECW
sfrp	MOVW ADDW[Note 1]	MOVW (ADDW)[Note 1]	MOVW (ADDW)[Note 1]								PUSH POP
!addr16 !!addr24	MOVW	(MOVW)	MOVW						MOVTBLW		
mem [saddrp] [%saddrg]		MOVW									
PSW											PUSH POP
SP	ADDWG SUBWG										
post											PUSH POP PUSHU POPU
[TDE+]		(MOVW)						SACW			
byte											MACW MACSW

Notes 1. SUBW and CMPW are identical to ADDW.
2. There is no second operand, or the second operand is not an operand address.
3. When saddrp is saddrp2 in this combination, this is a short code length instruction.
4. MULUW and DIVUX are identical to MULW.

(3) 24-bit instructions (The values enclosed by parentheses are combined to express WHL description as rg.)

MOVG, ADDG, SUBG, INCG, DECG, PUSH, POP

Table 26-3. 24-Bit Addressing Instructions

Second operand First operand	#imm24	WHL	rg rg'	saddrg	!!addr24	mem1	[%saddrg]	SP	None[Note]
WHL	(MOVG) (ADDG) (SUBG)	(MOVG) (ADDG) (SUBG)	(MOVG) (ADDG) (SUBG)	(MOVG) ADDG SUBG	(MOVG)	MOVG	MOVG	MOVG	
rg	MOVG ADDG SUBG	(MOVG) (ADDG) (SUBG)	MOVG ADDG SUBG	MOVG	MOVG				INCG DECG PUSH POP
saddrg		(MOVG)	MOVG						
!!addr24		(MOVG)	MOVG						
mem1		MOVG							
[%saddrg]		MOVG							
SP	MOVG	MOVG							INCG DECG

Note There is no second operand, or the second operand is not an operand address.

(4) Bit manipulation instructions

MOV1, AND1, OR1, XOR1, SET1, CLR1, NOT1, BT, BF, BTCLR, BFSET

Table 26-4. Bit Manipulation Instruction Addressing Instructions

Second operand First operand	CY	saddr.bit sfr.bit A.bit X.bit PSWL.bit PSWH.bit mem2.bit !addr16.bit !!addr24.bit	/saddr.bit /sfr.bit /A.bit /X.bit /PSWL.bit /PSWH.bit /mem2.bit /!addr16.bit /!!addr24.bit	None[Note]
CY		MOV1 AND1 OR1 XOR1	AND1 OR1	NOT1 SET1 CLR1
saddr.bit sfr.bit A.bit X.bit PSWL.bit PSWH.bit mem2.bit !addr16.bit !!addr24.bit	MOV1			NOT1 SET1 CLR1 BF BT BTCLR BFSET

Note There is no second operand, or the second operand is not an operand address.

(5) Call return instructions and branch instructions

CALL, CALLF, CALLT, BRK, RET, RETI, RETB, RETCS, RETCSB, BRKCS, BR, BNZ, BNE, BZ, BE, BNC, BNL, BC, BL, BNV, BPO, BV, BPE, BP, BN, BLT, BGE, BLE, BGT, BNH, BH, BF, BT, BTCLR, BFSET, DBNZ

Table 26-5. Call Return Instructions and Branch Instruction Addressing Instructions

Instruction Address Operand	$addr20	$!addr20	!addr16	!!addr20	rp	rg	[rp]	[rg]	!addr11	[addr5]	RBn	None
Basic instructions	BC[Note] BR	CALL BR	CALL BR RETCS RETCSB	CALL BR	CALL BR	CALL BR	CALL BR	CALL BR	CALLF	CALLT	BRKCS	BRK RET RETI RETB
Composite instructions	BF BT BTCLR BFSET DBNZ											

Note BNZ, BNE, BZ, BE, BNC, BNL, BL, BNV, BPO, BV, BPE, BP, BN, BLT, BGE, BLE, BGT, BNH, and BH are identical to BC.

(6) Other instructions

ADJBA, ADJBS, CVTBW, LOCATION, SEL, NOT, EI, DI, SWRS

APPENDIX A MAJOR DIFFERENCES AMONG THE μPD784955 SUBSERIES, μPD784225 SUBSERIES, AND μPD784216 SUBSERIES

Series Name Item			μPD784955 Subseries	μPD784225 Subseries	μPD784216 Subseries
Minimum instruction execution time		When the main system clock is selected	160 ns (at 12.5-MHz operation)	160 ns (at 12.5-MHz operation)	
		When the subsystem clock is selected	—	61 μs (at 32.768-kHz operation	
I/O port		Total	67		86
		CMOS inputs	8		8
		CMOS I/O	59		72
		N-ch open-drain I/O	—		6
	Pins with added functions[Note]	Pins with pull-up resistors	59	57	70
		LED direct drive outputs	32	16	22
		Medium-voltage pins	—		6
Real-time output port			6 bits × 2 channels	8 bits × 1 channel	
Timer/counters			• 16-bit timer/counter × 6 units • 8-bit timer/counter × 2 units	• 16-bit timer/counter × 1 unit • 8-bit timer/counter × 4 units	• 16-bit timer/counter × 1 unit • 8-bit timer/counter × 6 units
Serial interface			• UART × 1 channel • CSI (3-wire serial I/O) × 1 channel	• UART/IOE (3-wire serial I/O) × 2 channels • CSI (3-wire serial I/O) × 1 channel	
External memory expansion function			No	Yes	
Standby function			• HALT/STOP/IDLE mode	• HALT/STOP/IDLE mode • In low power consumption mode: HALT or IDLE mode	
ROM correction			No	Yes	No
Package			• 80-pin plastic QFP (14 × 14 mm)	• 80-pin plastic QFP (14 × 14 mm) • 80-pin plastic TQFP (fine pitch) (12 × 20 mm)	• 100-pin plastic QFP (fine pitch) (14 × 14 mm) • 100-pin plastic QFP (14 × 20 mm)

Note These added functions are valid when these pins are used as I/O pins.

[MEMO]

APPENDIX B DEVELOPMENT TOOLS

The development tool configurations that are required for the development of systems that employ the μPD784955 Subseries are shown in the following pages.

Figure B-1. Development Tool Configuration (1/2)

(1) When using the in-circuit emulator IE-78K4-NS

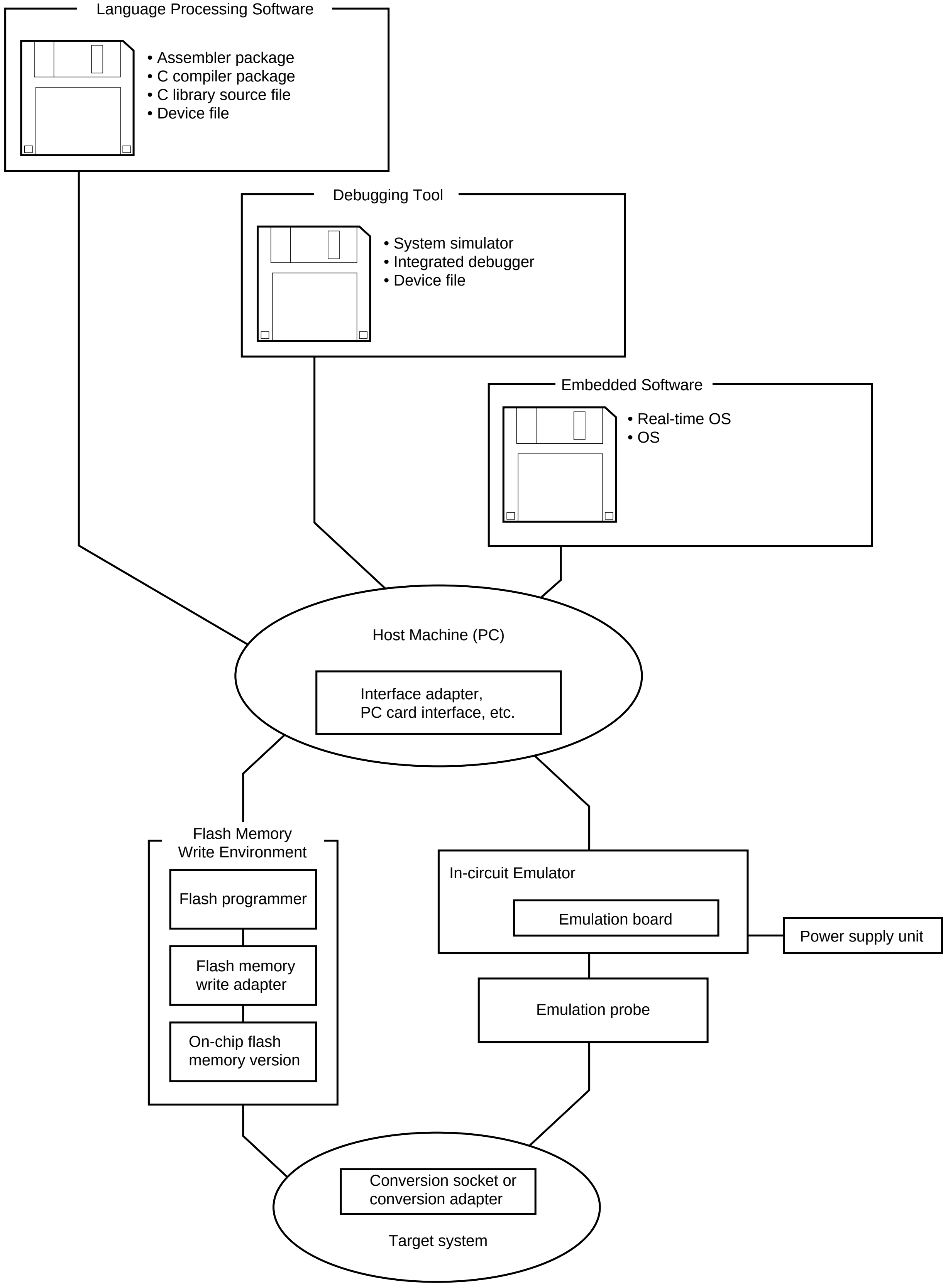
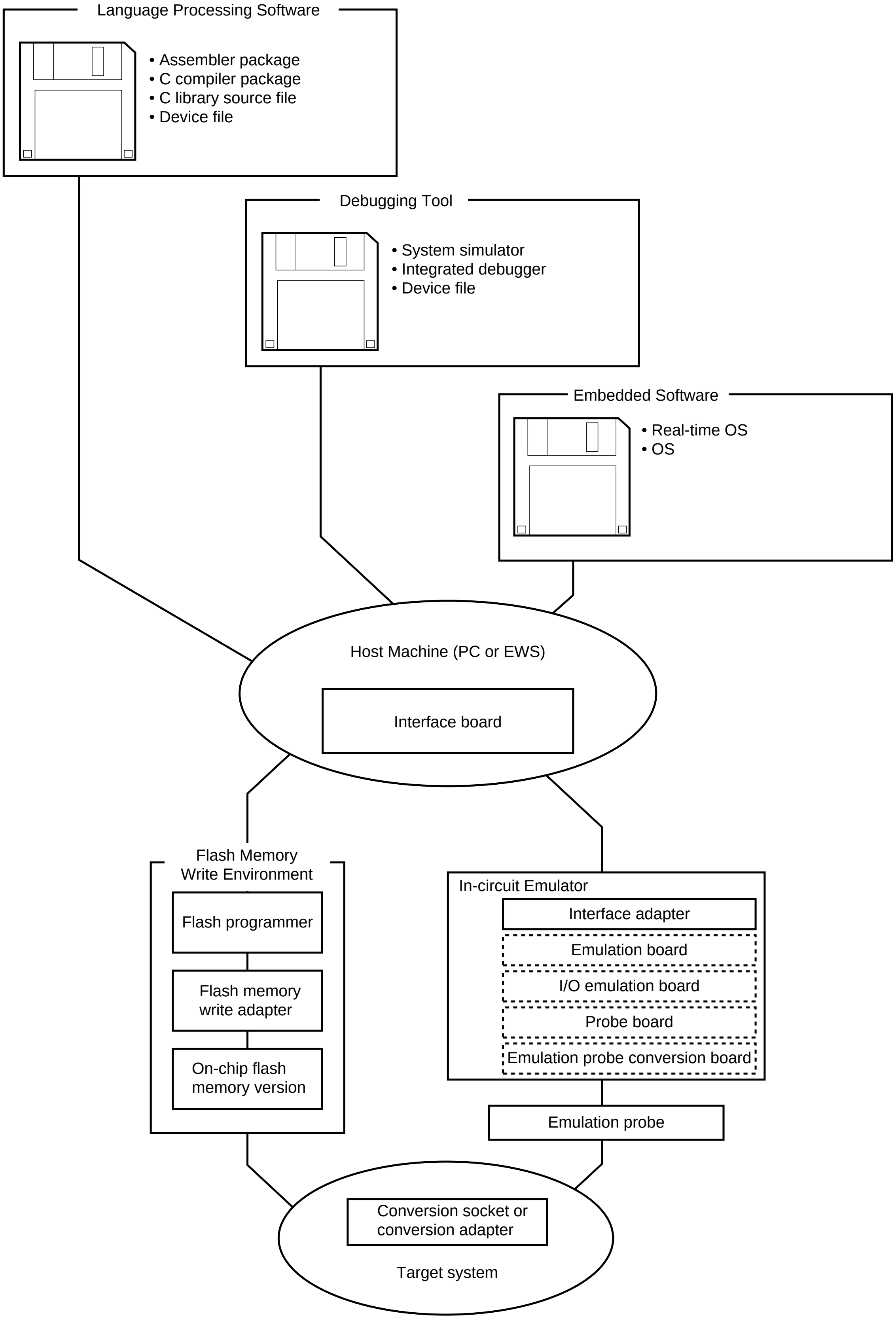


Figure B-1. Development Tool Configuration (2/2)

(2) When using the in-circuit emulator IE-784000-R



Remark Items in broken line boxes differ according to the development environment. Refer to **B.3.1. Hardware**.

B.1 Language Processing Software

RA78K4 Assembler Package	This assembler converts programs written in mnemonics into an object codes executable with a microcontroller. Further, this assembler is provided with functions capable of automatically creating symbol tables and branch instruction optimization. This assembler should be used in combination with an optional device file (DF784956). <Precaution when using RA78K4 in PC environment> This assembler package is a DOS-based application. It can also be used in Windows, however, by using the Project Manager (included in assembler package) on Windows.
	Part Number: μS××××RA78K4
CC78K4 C Compiler Package	This compiler converts programs written in C language into object codes executable with a microcontroller. This compiler should be used in combination with an optional assembler package (RA78K4) and device file (DF784956). <Precaution when using RA78K4 in PC environment> This C compiler package is a DOS-based application. It can also be used in Windows, however, by using the Project Manager (included in assembler package) on Windows.
	Part Number: μS××××CC78K4
DF784956[Note]	This file contains information peculiar to the device. This device file should be used in combination with an optional tool (RA78K4, CC78K4, SM78K4, ID78K4-NS, and ID78K4). Corresponding OS and host machine differ depending on the tool to be used with.
	Part Number: μS××××DF784956
CC78K4-L C Library Source File	This is a source file of functions configuring the object library included in the C compiler package. This file is required to match the object library included in C compiler package to the customer's specifications. Operating environment for the source file is not dependent on the OS.
	Part Number: μS××××CC78K4-L

Note The DF784956 can be used in common with the RA78K4, CC78K4, SM78K4, ID78K4-NS, and ID78K4.

Remark ×××× in the part number differs depending on the host machine and OS used.

*μ*S××××RA78K4
*μ*S××××CC78K4
*μ*S××××DF784956
*μ*S××××CC78K4-L

××××	Host Machine	OS	Supply Medium
AA13	PC-9800 Series	Windows (Japanese version) Notes 1, 2	3.5-inch 2HD FD
AB13	IBM PC/AT™ and compatibles	Windows (Japanese version) Notes 1, 2	3.5-inch 2HC FD
BB13		Windows (English version) Notes 1, 2	
3P16	HP9000 Series 700™	HP-UX (Rel. 9.05)	DAT (DDS)
3K13	SPARCstation™	SunOS (Rel. 4.1.4)	3.5-inch 2HC FD
3K15			1/4-inch CGMT
3R13	NEWS™ (RISC)	NEWS-OS (Rel. 6.1)	3.5-inch 2HC FD

Notes 1. Can be operated in DOS environment.
2. Windows NT™ not supported.

B.2 Flash Memory Writing Tools

Flashpro II (type FL-PR2) Flash programmer	Flash programmer dedicated to microcontrollers with on-chip flash memory.
FA-80GC Flash Memory Writing Adapter	Flash memory writing adapter used connected to the Flashpro II. • FA-80GC : 80-pin plastic QFP (GC-8BT type)

Remark Flashpro II and FA-80GC are products of Naitou Densei Machidaseisakusho Co., Ltd.
Phone: (044) 822-3813 Naitou Densei Machidaseisakusho Co., Ltd.

B.3 Debugging Tools

B.3.1 Hardware (1/2)

(1) When using the in-circuit emulator IE-78K4-NS

★	IE-78K4-NS In-circuit Emulator		The in-circuit emulator serves to debug hardware and software when developing application systems using a 78K/IV Series product. It corresponds to integrated debugger (ID78K4-NS). This emulator should be used in combination with power supply unit, emulation probe, and interface adapter which is required to connect this emulator to the host machine.
	IE-70000-MC-PS-B Power Supply Unit		This adapter is used for supplying power from a receptacle of 100-V to 200-V AC.
★	IE-70000-98-IF-C Interface Adapter		This adapter is required when using the PC-9800 Series computer (except notebook type) as the IE-78K4-NS host machine.
★	IE-70000-CD-IF PC Card Interface		This is PC card and interface cable required when using the PC-9800 Series notebook-type computer as the IE-78K4-NS host machine.
★	IE-70000-PC-IF-C Interface Adapter		This adapter is required when using the IBM PC/AT and its compatible computers as the IE-78K4-NS host machine.
	IE-784956-NS-EM1^{Note} Emulation Board		This board emulates the operations of the peripheral hardware peculiar to a device. It should be used in combination with an in-circuit emulator.
	NP-80GC Emulation Probe		This probe is used to connect the in-circuit emulator to the target system and is designed for 80-pin plastic QFP (GC-8BT type).
		EV-9200GC-80 Conversion Socket (Refer to **Figures B-2** and **B-3**)	This conversion socket connects the NP-80GC to the target system board designed to mount a 80-pin plastic QFP (GC-8BT type).

Note Under development

Remarks 1. NP-80GC is a product of Naitou Densei Machidaseisakusho Co., Ltd.
Phone: (044) 822-3813 Naitou Densei Machidaseisakusho Co., Ltd.

2. EV-9200GC-80 is sold in units of five.

B.3.1 Hardware (2/2)

(2) When using the in-circuit emulator IE-784000-R

IE-784000-R In-circuit Emulator		The IE-784000-R is an in-circuit emulator that can be used in all members of the 78K/IV Series. Use in combination with the separately purchased IE-784000-R-EM and IE-784956-NS-EM1. For debugging, connect to the host machine. Using in conjunction with the mandatory, separately purchased, integrated debugger (ID78K4) and device file, allows debugging on the source program level in C language and structured assembly language. The C0 coverage function provides efficient debugging and program inspection. Connecting with the host machine by either Ethernet™ or a dedicated bus requires a separately purchased interface adapter.
★ IE-70000-98-IF-B or IE-70000-98-IF-C Interface Adapter		This adapter is required when using the PC-9800 Series computer (except notebook type) as the IE-784000-R host machine.
IE-70000-98N-IF Interface Adapter		Interface adapter and cable are required when using a PC-9800 series notebook computer as the IE-784000-R host machine.
★ IE-70000-PC-IF-B or IE-70000-PC-IF-C Interface Adapter		This adapter is required when using the IBM PC/AT and its compatible computers as the IE-78001-R-A host machine.
IE-78000-R-SV3 Interface Adapter		This is adapter and cable required when using an EWS computer as the IE-784000-R host machine, and is used connected to the board in the IE-784000-R. 10Base-5 supports Ethernet, but a commercially available conversion adapter is required for other formats.
IE-784000-R-EM Emulation Board		The emulation board that is used with all units in the 78K/IV Series.
IE-784956-NS-EM1[Note] Emulation Board		Board for emulating peripheral hardware that is inherent to a device.
★	IE-78K4-R-EX2 Emulation Probe Conversion Board	80-pin conversion board required when using the IE-784956-NS-EM1 on the IE-784000-R.
EP-78230GC-R Emulation Probe		This probe is used to connect the in-circuit emulator to the target system and is designed for 80-pin plastic QFP (GC-8BT type).
	EV-9200GC-80 Conversion Socket (Refer to **Figures B-2** and **B-3**)	This conversion socket connects the EP-78243GC-R to the target system board designed to mount a 80-pin plastic QFP (GC-8BT type).

Note Under development

Remark EV-9200GC-80 is sold in units of five.

B.3.2 Software (1/2)

SM78K4 System Simulator	This system simulator is used to perform debugging at C source level or assembler level while simulating the operation of the target system on a host machine. This simulator runs on Windows. Use of the SM78K4 allows the execution of application logical testing and performance testing on an independent basis from hardware development without having to use an in-circuit emulator, thereby providing higher development efficiency and software quality. The SM78K4 should be used in combination with the optional device file (DF784956).
	Part Number: μS××××SM78K4

Remark ×××× in the part number differs depending on the host machine and OS used.

μS××××SM78K4

××××	Host Machine	OS	Supply Medium
AA13	PC-9800 Series	Windows (Japanese version)[Note]	3.5-inch 2HD FD
AB13	IBM PC/AT and compatibles	Windows (Japanese version)[Note]	3.5-inch 2HC FD
BB13		Windows (English version)[Note]	

Note Windows NT not supported.

B.3.2 Software (2/2)

★

ID78K4-NS Integrated Debugger (supporting in-circuit emulator IE-78K4-NS)	This debugger is a control program to debug 78K/IV Series microcontrollers. It adopts a graphical user interface, which is equivalent visually and operationally to Windows or OSF/Motif™. It also has an enhanced debugging function for C language programs, and thus trace results can be displayed on screen in C-language level by using the windows integration function which links a trace result with its source program, disassembled display, and memory display. In addition, by incorporating function modules such as task debugger and system performance analyzer, the efficiency of debugging programs, which run on real-time OSs can be improved. It should be used in combination with the optional device file (DF784956).
ID78K4 Integrated Debugger (supporting in-circuit emulator IE-784000-R)	
	Part Number: µS××××ID78K4-NS, µS××××ID78K4

Remark ×××× in the part number differs depending on the host machine and OS used.

µS××××ID78K4-NS

××××	Host Machine	OS	Supply Medium
AA13	PC-9800 Series	Windows (Japanese version)[Note]	3.5-inch 2HD FD
AB13	IBM PC/AT and compatibles	Windows (Japanese version)[Note]	3.5-inch 2HC FD
BB13		Windows (English version)[Note]	

Note Windows NT not supported.

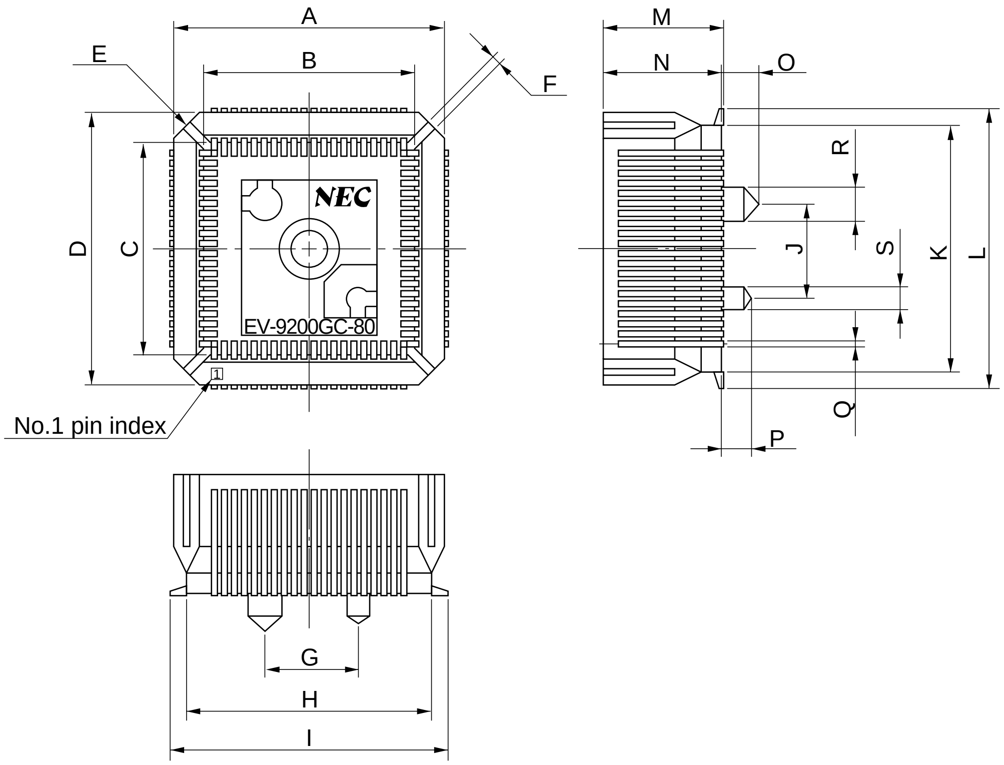
µS××××ID78K4

××××	Host Machine	OS	Supply Medium
AA13	PC-9800 Series	Windows (Japanese version)[Note]	3.5-inch 2HD FD
AB13	IBM PC/AT and compatibles	Windows (Japanese version)[Note]	3.5-inch 2HC FD
BB13		Windows (English version)[Note]	
3P16	HP9000 Series 700	HP-UX (Rel. 9.05)	DAT (DDS)
3K13	SPARCstation	SunOS (Rel. 4.1.4)	3.5-inch 2HC FD
3K15			1/4 inch CGMT
3R13	NEWS (RISC)	NEWS-OS (Rel. 6.1)	3.5-inch 2HC FD

Note Windows NT not supported.

B.4 Conversion Socket Drawing (EV-9200GC-80) and Recommended Footprints

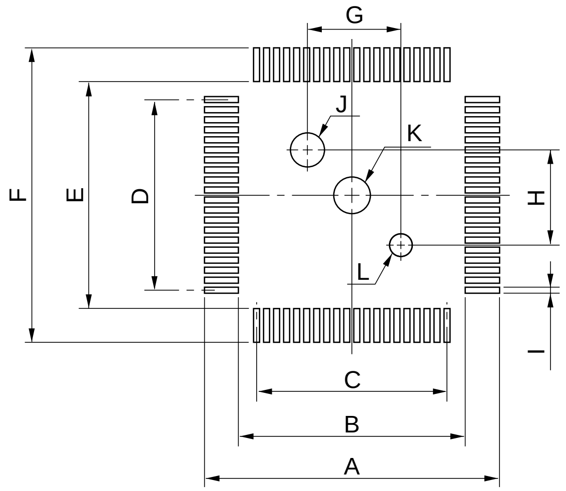
Figure B-2. EV-9200GC-80 Drawing (for reference only)



EV-9200GC-80-G1E

ITEM	MILLIMETERS	INCHES
A	18.0	0.709
B	14.4	0.567
C	14.4	0.567
D	18.0	0.709
E	4-C 2.0	4-C 0.079
F	0.8	0.031
G	6.0	0.236
H	16.0	0.63
I	18.7	0.736
J	6.0	0.236
K	16.0	0.63
L	18.7	0.736
M	8.2	0.323
N	8.0	0.315
O	2.5	0.098
P	2.0	0.079
Q	0.35	0.014
R	ϕ2.3	ϕ0.091
S	ϕ1.5	ϕ0.059

Figure B-3. EV-9200GC-80 Footprints (for reference only)



EV-9200GC-80-P1E

ITEM	MILLIMETERS	INCHES
A	19.7	0.776
B	15.0	0.591
C	0.65±0.02 × 19=12.35±0.05	$0.026^{+0.001}_{-0.002}$ × 0.748=$0.486^{+0.003}_{-0.002}$
D	0.65±0.02 × 19=12.35±0.05	$0.026^{+0.001}_{-0.002}$ × 0.748=$0.486^{+0.003}_{-0.002}$
E	15.0	0.591
F	19.7	0.776
G	6.0±0.05	$0.236^{+0.003}_{-0.002}$
H	6.0±0.05	$0.236^{+0.003}_{-0.002}$
I	0.35±0.02	$0.014^{+0.001}_{-0.001}$
J	ϕ2.36±0.03	$\phi 0.093^{+0.001}_{-0.002}$
K	ϕ2.3	ϕ0.091
L	ϕ1.57±0.03	$\phi 0.062^{+0.001}_{-0.002}$

Caution **Dimensions of mount pad for EV-9200 and that for target device (QFP) may be different in some parts. For the recommended mount pad dimensions for QFP, refer to "SEMICONDUCTOR DEVICE MOUNTING TECHNOLOGY MANUAL" (C10535E).**

[MEMO]

APPENDIX C EMBEDDED SOFTWARE

The following embedded software products are available for efficient program development and maintenance of the μPD784955 Subseries.

Real-Time OS (1/2)

RX78K/IV Real-time OS	RX78K/IV is a real-time OS conforming to the µITRON specifications. Tool (configurator) for generating nucleus of RX78K/IV and plural information tables is supplied. Used in combination with an optional assembler package (RA78K4) and device file (DF784956). <Precaution when using RX78K/IV in PC environment> The real-time OS is a DOS-based application. It should be used in the DOS Prompt when using in Windows.
	Part number: µS××××RX78K4

Caution When purchasing the RX78K/IV, fill in the purchase application form in advance and sign the User Agreement.

Remark ×××× and ΔΔΔ in the part number differ depending on the host machine and OS used.

µS××××RX78K4-ΔΔΔ

ΔΔΔ	Product Outline	Maximum Number for Use in Mass Production
001	Evaluation object	Do not use for mass-produced product.
100K	Mass-production object	0.1 million units
001M		1 million units
010M		10 million units
S01	Source program	Source program for mass-produced object

××××	Host Machine	OS	Supply Medium
AA13	PC-9800 Series	Windows (Japanese version)[Notes 1, 2]	3.5-inch 2HD FD
AB13	IBM PC/AT and compatibles	Windows (Japanese version)[Notes 1, 2]	3.5-inch 2HC FD
BB10		Windows (English version)[Notes 1, 2]	
3P16	HP9000 Series 700	HP-UX (Rel. 9.05)	DAT (DDS)
3K13	SPARCstation	SunOS (Rel. 4.1.4)	3.5-inch 2HC FD
3K15			1/4-inch CGMT
3R13	NEWS (RISC)	NEWS-OS (Rel. 6.1)	3.5-inch 2HC FD

Notes 1. Can also be operated in DOS environment.

2. Windows NT not supported.

Real-Time OS (2/2)

MX78K4 OS	MX78K4 is an OS for μITRON specification subsets. A nucleus for the MX78K4 is also included as a companion product. This manages tasks, events, and time. In the task management, determining the task execution order and switching from task to the next task are performed. <Precaution when using MX78K4 in PC environment> The MX78K4 is a DOS-based application. It should be used in the DOS Prompt when using in Windows.
	Part number: μS××××MX78K4-∆∆∆

Remark ×××× and ∆∆∆ in the part number differ depending on the host machine and OS used.

μS××××MX78K4-∆∆∆

∆∆∆	Product Outline	Maximum Number for Use in Mass Production
001	Evaluation object	Use in preproduction stages.
××	Mass-production object	Use in mass production stages.
S01	Source program	Only the users who purchased mass-production objects are allowed to purchase this program.

××××	Host Machine	OS	Supply Medium
AA13	PC-9800 Series	Windows (Japanese version)Notes 1, 2	3.5-inch 2HD FD
AB13	IBM PC/AT and compatibles	Windows (Japanese version)Notes 1, 2	3.5-inch 2HC FD
BB13		Windows (English version)Notes 1, 2	
3P16	HP9000 Series 700	HP-UX (Rel. 9.05)	DAT (DDS)
3K13	SPARCstation	SunOS (Rel. 4.1.4)	3.5-inch 2HC FD
3K15			1/4-inch CGMT
3R13	NEWS (RISC)	NEWS-OS (Rel. 6.1)	3.5-inch 2HC FD

Notes 1. Can also be operated in DOS environment.

2. Windows NT not supported.

[MEMO]

APPENDIX D REGISTER INDEX

D.1 Register Index (By Register Name)

D.2 Register Index (By Register Symbol)

[MEMO]

★

APPENDIX E REVISION HISTORY

The following shows major revisions up to now.

Edition	Major Revisions from Previous Edition	Revised Chapters
2nd	Addition of document numbers of related documents	**INTRODUCTION**
	Deletion of μPD78F4943 Subseries and addition of μPD784937 Subseries in **78K/IV Series Product Development Diagram** Deletion of the external bus block in **1.4 Block Diagram**	**CHAPTER 1 OVERVIEW**
	Modification of **Table 2-2 Operating Modes of Port 1** Modification of **Table 2-3 Operating Modes of Port 2** Modification of **Table 2-4 Operating Modes of Port 3**	**CHAPTER 2 PIN FUNCTIONS**
	Modification of internal ROM area in **Figure 3-3 μPD78F4956 Memory Map** Deletion of **3.4.3 External SFR area**	**CHAPTER 3 CPU ARCHITECTURE**
	Modification of **Figure 4-1 Block Diagram of Clock Generator** Modification of system clock frequency in **4.5 Clock Generator Operations** f_{XX}→f_{CLK}	**CHAPTER 4 CLOCK GENERATOR**
	Modification of description in **(2) Pull-up resistor option registers (PU0 to PU3, PU9, PUO)** in **5.3 Control Registers**	**CHAPTER 5 PORT FUNCTIONS**
	Modification of **Figure 6-1 Block Diagram of Real-Time Output Port** Modification of the name of external interrupt trigger in **Figure 6-4 Format of Real-Time Output Port Control Register 0 (RTPC0)** INTP3→INTP3TRG Modification of **Figure 6-5 Format of PWM Modulation Control Register 0 (PWMC0)** Modification of the name of external interrupt trigger in **Figure 6-12 Format of Real-Time Output Port Control Register 1 (RTPC1)** INTP5→INTP5TRG Modification of **Figure 6-13 Format of PWM Modulation Control Register 1 (PWMC1)** Modification of **6.6 Cautions**	**CHAPTER 6 REAL-TIME OUTPUT FUNCTIONS**
	Deletion of **(4) Operation of OVF0 flag** in **8.5 Cautions**	**CHAPTER 8 16-BIT TIMER/COUNTER 0**
	Deletion of **(3) Operation of OVF1 flag** in **9.5 Cautions**	**CHAPTER 9 16-BIT TIMER/COUNTER 1**
	Deletion of **(3) Operation of OVF2 flag** in **10.5 Cautions**	**CHAPTER 10 16-BIT TIMER/COUNTER 2**
	Deletion of **(3) Operation of OVF3 flag** in **11.5 Cautions**	**CHAPTER 11 16-BIT TIMER/COUNTER 3**
	Deletion of **(4) Operation of OVF4 flag** in **12.5 Cautions**	**CHAPTER 12 16-BIT TIMER/COUNTER 4**
	Deletion of **(4) Operation of OVF5 flag** in **13.5 Cautions**	**CHAPTER 13 16-BIT TIMER/COUNTER 5**

Edition	Major Revisions from Previous Edition	Revised Chapters
2nd	Addition of Caution 3 in **Figure 14-2 Format of Timer Mode Control Register 6 (TMC6)** Deletion of Caution 2 in **14.4.3 Free running operation of TM6 (PWM output)** Deletion of **(3) TM6 read out during timer operation** in **14.5 Cautions**	**CHAPTER 14 8-BIT TIMER/COUNTER 6**
	Deletion of **(3) TM7 read out during timer operation** in **15.5 Cautions**	**CHAPTER 15 8-BIT TIMER/COUNTER 7**
	Modification of **Figure 17-1 A/D Converter Block Diagram** Deletion of Caution in **17.4.1 Basic operations of A/D converter**	**CHAPTER 17 A/D CONVERTER**
	Modification of **Figure 19-1 Block Diagram in Asynchronous Serial Interface Mode** Modification of expression of baud rate Modification of **Table 19-2 Relationship between 5-Bit Counter Source Clock and m Value** Modification of **Caution** in the case of UART transmission Deletion of the description of selection of external clock in **19.2.3 Standby mode operation**	**CHAPTER 19 ASYNCHRONOUS SERIAL INTERFACE**
	Modification of **Figure 20-1 Block Diagram of Clocked Serial Interface (3-Wire Serial I/O Mode)**	**CHAPTER 20 3-WIRE SERIAL I/O MODE**
	Modification of pins with edge detection function P00 to P07→P00	**CHAPTER 21 EDGE DETECTION FUNCTION**
	Addition of reserved word to **Figure 22-20 Macro Service Control Word Format**	**CHAPTER 22 INTERRUPT FUNCTIONS**
	Deletion of **Note** Under development	**APPENDIX B DEVELOPMENT TOOLS**

NEC

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

Europe
NEC Electronics (Europe) GmbH
Technical Documentation Dept.
Fax: +49-211-6503-274

South America
NEC do Brasil S.A.
Fax: +55-11-6465-6829

Hong Kong, Philippines, Oceania
NEC Electronics Hong Kong Ltd.
Fax: +852-2886-9022/9044

Korea
NEC Electronics Hong Kong Ltd.
Seoul Branch
Fax: 02-528-4411

Taiwan
NEC Electronics Taiwan Ltd.
Fax: 02-719-5951

Asian Nations except Philippines
NEC Electronics Singapore Pte. Ltd.
Fax: +65-250-3583

Japan
NEC Semiconductor Technical Hotline
Fax: 044-548-7900

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	❏	❏	❏	❏
Technical Accuracy	❏	❏	❏	❏
Organization	❏	❏	❏	❏