NEC

User's Manual

# μPD784225, 784225Y Subseries

## 16-/8-Bit Single-Chip Microcontrollers

## Hardware

μPD784224 μPD784224Y
μPD784225 μPD784225Y
μPD78F4225 μPD78F4225Y

Document No. U12697EJ4V0UD00 (4th edition)
Date Published May 2002 N CP(K)

**[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.

License not needed: μPD78F4225, 78F4225Y
The customer must judge the need for license: μPD784224, 784225, 784224Y, 784225Y

Purchase of NEC $I^2C$ components conveys a license under the Philips $I^2C$ Patent Rights to use these components in an $I^2C$ system, provided that the system conforms to the $I^2C$ Standard Specification as defined by Philips.

- **The information in this document is current as of February, 2002. The information is subject to change without notice. For actual design-in, refer to the latest publications of NEC's data sheets or data books, etc., for the most up-to-date specifications of NEC semiconductor products. Not all products and/or types are available in every country. Please check with an NEC sales representative for availability and additional information.**
- No part of this document may be copied or reproduced in any form or by any means without prior written consent of NEC. NEC assumes no responsibility for any errors that may appear in this document.
- NEC does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from the use of NEC semiconductor products listed in this document or any other liability arising from the use of such products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC or others.
- Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of customer's equipment shall be done under the full responsibility of customer. NEC assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.
- While NEC endeavours to enhance the quality, reliability and safety of NEC semiconductor products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC semiconductor products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment, and anti-failure features.
- NEC semiconductor products are classified into the following three quality grades:
  "Standard", "Special" and "Specific". The "Specific" quality grade applies only to semiconductor products developed based on a customer-designated "quality assurance program" for a specific application. The recommended applications of a semiconductor product depend on its quality grade, as indicated below. Customers must check the quality grade of each semiconductor product before using it in a particular application.
  "Standard": Computers, office equipment, communications equipment, test and measurement equipment, audio and visual equipment, home electronic appliances, machine tools, personal electronic equipment and industrial robots
  "Special": Transportation equipment (automobiles, trains, ships, etc.), traffic control systems, anti-disaster systems, anti-crime systems, safety equipment and medical equipment (not specifically designed for life support)
  "Specific": Aircraft, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems and medical equipment for life support, etc.
  The quality grade of NEC semiconductor products is "Standard" unless otherwise expressly specified in NEC's data sheets or data books, etc. If customers wish to use NEC semiconductor products in applications not intended by NEC, they must contact an NEC sales representative in advance to determine NEC's willingness to support a given application.
  (Note)
  (1) "NEC" as used in this statement means NEC Corporation and also includes its majority-owned subsidiaries.
  (2) "NEC semiconductor products" means any semiconductor product developed or manufactured by or for NEC (as defined above).

M8E 00. 4

# Regional Information

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

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

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


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

**NEC do Brasil S.A.**
Electron Devices Division
Guarulhos-SP, Brasil
Tel: 11-6462-6810
Fax: 11-6462-6829

**NEC Electronics (Europe) GmbH**
Duesseldorf, Germany
Tel: 0211-65 03 01
Fax: 0211-65 03 327

- **Sucursal en España**
  Madrid, Spain
  Tel: 091-504 27 87
  Fax: 091-504 28 60

- **Succursale Française**
  Vélizy-Villacoublay, France
  Tel: 01-30-67 58 00
  Fax: 01-30-67 58 99

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

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

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

- **United Kingdom Branch**
  Milton Keynes, UK
  Tel: 01908-691-133
  Fax: 01908-670-290

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

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

**NEC Electronics Shanghai, Ltd.**
Shanghai, P.R. China
Tel: 021-6841-1138
Fax: 021-6841-1137

**NEC Electronics Taiwan Ltd.**
Taipei, Taiwan
Tel: 02-2719-2377
Fax: 02-2719-5951

**NEC Electronics Singapore Pte. Ltd.**
Novena Square, Singapore
Tel: 253-8311
Fax: 250-3583

## Major Revisions in This Edition

| Page | Description |
|---|---|
| | **CHAPTER 1 OVERVIEW** |
| p.31 | Change of **78K/IV SERIES LINEUP** |
| p.38 | Modification of minimum instruction execution time in **1.5 Function List** |
| | **CHAPTER 13 A/D CONVERTER** |
| p.230 | Modification of **Cautions** in **Figure 13-2 Format of A/D Converter Mode Register (ADM)** |
| | **CHAPTER 16 ASYNCHRONOUS SERIAL INTERFACE/3-WIRE SERIAL I/O** |
| p.256 | Addition of **Table 16-2 Serial Interface Operation Mode Settings** |
| | **CHAPTER 17 3-WIRE SERIAL I/O MODE** |
| p.287 | Addition of **Table 17-2 Serial Interface Operation Mode Settings** |
| | **CHAPTER 22 INTERRUPT FUNCTIONS** |
| p.400 | Addition of reserved words in **Figure 22-21 Format of Macro Service Control Word** |
| | **CHAPTER 23 LOCAL BUS INTERFACE FUNCTIONS** |
| p.438 | Modification of **Table 23-3 Settings of Program Wait Control Register 2 (PWC2)** |
| | **CHAPTER 24 STANDBY FUNCTION** |
| p.463 | Modification of **Figure 24-1 Standby Function State Transition** |
| p.551 | Addition of **CHAPTER 29 ELECTRICAL SPECIFICATIONS** |
| p.593 | Addition of **CHAPTER 30 PACKAGE DRAWINGS** |
| p.595 | Addition of **CHAPTER 31 RECOMMENDED SOLDERING CONDITIONS** |
| | **APPENDIX B DEVELOPMENT TOOLS** |
| p.602 | Addition of description on SP78K4 and change of **Remark** in **B.1 Language Processing Software** |
| p.605 | Modification of **Remark** in **B.3.2 Software** |
| p.606 | Addition of **B.4 Cautions on Designing Target System** |
| | **APPENDIX C EMBEDDED SOFTWARE** |
| p.611 | Deletion of MX78K4 description |

**The mark ★ shows major revised points.**

# INTRODUCTION

**Target Readers**

This manual is intended for user engineers who wish to understand the functions of the μPD784225 and 784225Y Subseries and design its application systems.

**Purpose**

This manual describes the hardware functions of the μPD784225 and 784225Y Subseries.

**Organization**

The μPD784225 and 784225Y Subseries User's Manual is divided into two parts: Hardware (this manual) and Instruction.

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

**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**

It is assumed that the readers of this manual have general knowledge about electrical engineering, logic circuits, and microcontrollers.

- **If there are no particular differences in the function**
  The μPD784225 in the μPD784225 Subseries is described as the representative mask ROM version, and the μPD78F4225 is described as the representative flash memory version.

- **If there are differences in the function**
  Each product name is presented and described separately.
  Since μPD784225 Subseries products are described as representative even this case, for information on the operation of μPD784225Y Subseries products, read the sections on the μPD784224Y, 784225Y, and 78F4225Y instead of the μPD784224, 784225, and 78F4225.

- **To understand the overall functions**
  → Read in the order of the 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.

- **For detailed explanations of registers whose names are known**
  → See **APPENDIX D REGISTER INDEX.**

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

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

- **To know the electrical specifications**
  → Refer to **CHAPTER 29 ELECTRICAL SPECIFICATIONS**.

**Differences Between the μPD784225 Subseries and the μPD784225Y Subseries**

The only functional difference between the μPD784225 Subseries and μPD784225Y Subseries is the clocked serial interface. The two subseries share all other functions.

**Caution**

**The clock serial interface is described in the following two chapters.**

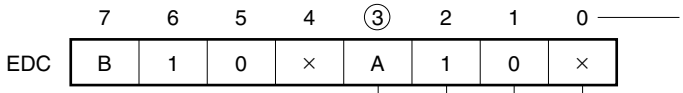
- **CHAPTER 17 3-WIRE SERIAL I/O MODE**
- **CHAPTER 18 $I^2C$ BUS MODE (μPD784225Y SUBSERIES only)**

**For an overview of the serial interface, also read CHAPTER 15.**

**Conventions**

| | |
|---|---|
| Data significance: | Higher digits on the left and lower digits on the right |
| Active low representation: | $\overline{\times\times\times}$ (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 ··· ××××B or ×××× |
| | Decimal ··· ×××× |
| | Hexadecimal ··· ××××H |

**Register Representation**




Bits whose numbers are circled are reserved words in NEC's assembler or are defined as sfr variables by the #pragma sfr directive in the 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 be used. | 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 devices**

| Document Name | Document No. |
|---|---|
| μPD784225, 784225Y Subseries Hardware User's Manual | This manual |
| 78K/IV Series Instruction User's Manual | U10905E |
| 78K/IV Series Software Basics Application Note | U10095E |

**Documents related to development tools (software) (user's manuals)**

| Document Name | | Document No. |
|---|---|---|
| RA78K4 Assembler Package | Operation | U11334E |
| | Language | U11162E |
| | Structured Assembler Preprocessor | U11743E |
| CC78K4 C Compiler | Operation | U11572E |
| | Language | U11571E |
| SM78K4 System Simulator Ver. 1.40 or Later | Reference (Windows™ Based) | U10093E |
| SM78K Series System Simulator Ver. 1.40 or Later | External Part User Open Interface Specifications | U10092E |
| ID78K Series Integrated Debugger Ver. 2.30 or Later | Operation (Windows Based) | U15185E |
| ID78K4 Integrated Debugger Windows Based | Reference | U10440E |
| RX78K4 Real-Time OS | Fundamental | U10603E |
| | Installation | U10604E |
| Project Manager Ver. 3.12 or Later (Windows Based) | | U14610E |

**Documents related to development tools (hardware) (user's manuals)**

| Document Name | Document No. |
|---|---|
| IE-78K4-NS In-Circuit Emulator | U13356E |
| IE-784225-NS-EM1 Emulation Board | U13742E |
| IE-784000-R In-Circuit Emulator | U12903E |

**Documents related to flash memory writing**

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

**Other documents**

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

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

CONTENTS

## LIST OF FIGURES (1/8)

## LIST OF FIGURES (2/8)

## LIST OF FIGURES (3/8)

## LIST OF FIGURES (4/8)

## LIST OF FIGURES (5/8)

## LIST OF FIGURES (6/8)

## LIST OF FIGURES (7/8)

## LIST OF FIGURES (8/8)

## LIST OF TABLES (1/3)

## LIST OF TABLES (2/3)

## LIST OF TABLES (3/3)

# CHAPTER 1 OVERVIEW

The μPD784225 Subseries is a member of the 78K/IV Series, and is an 80-pin general-purpose microcontroller in which the functions of the μPD784216 Subseries have been limited and to which a ROM correction function has been added. The 78K/IV Series includes 8-/16-bit single-chip microcontrollers and provides a high-performance CPU with functions such as 1 MB memory space access.

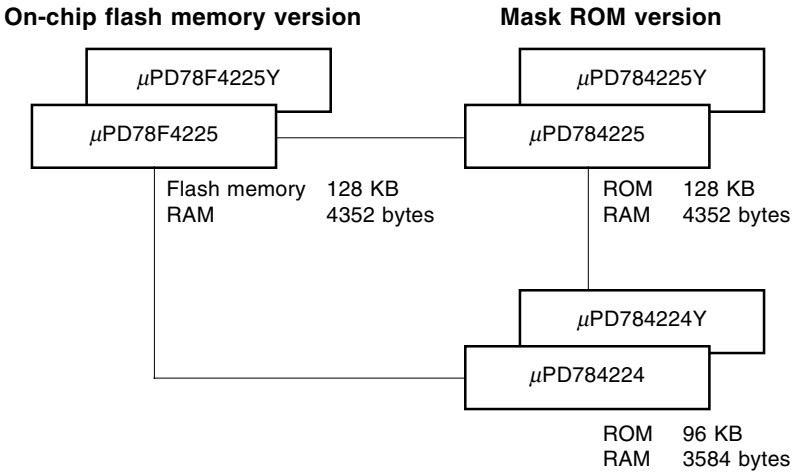
The μPD784225 has a 128 KB ROM and 4,352-byte RAM on chip. In addition, it has a high-performance timer/counter, an 8-bit A/D converter, an 8-bit D/A converter, and an independent 2-channel serial interface.

The μPD784224 is the μPD784225 with a 96 KB mask ROM and a 3,584-byte RAM.

The μPD78F4225 is the μPD784225 with the mask ROM replaced by a flash memory.

The μPD784225Y Subseries is the μPD784225 Subseries with an added $I^2C$ bus control function.

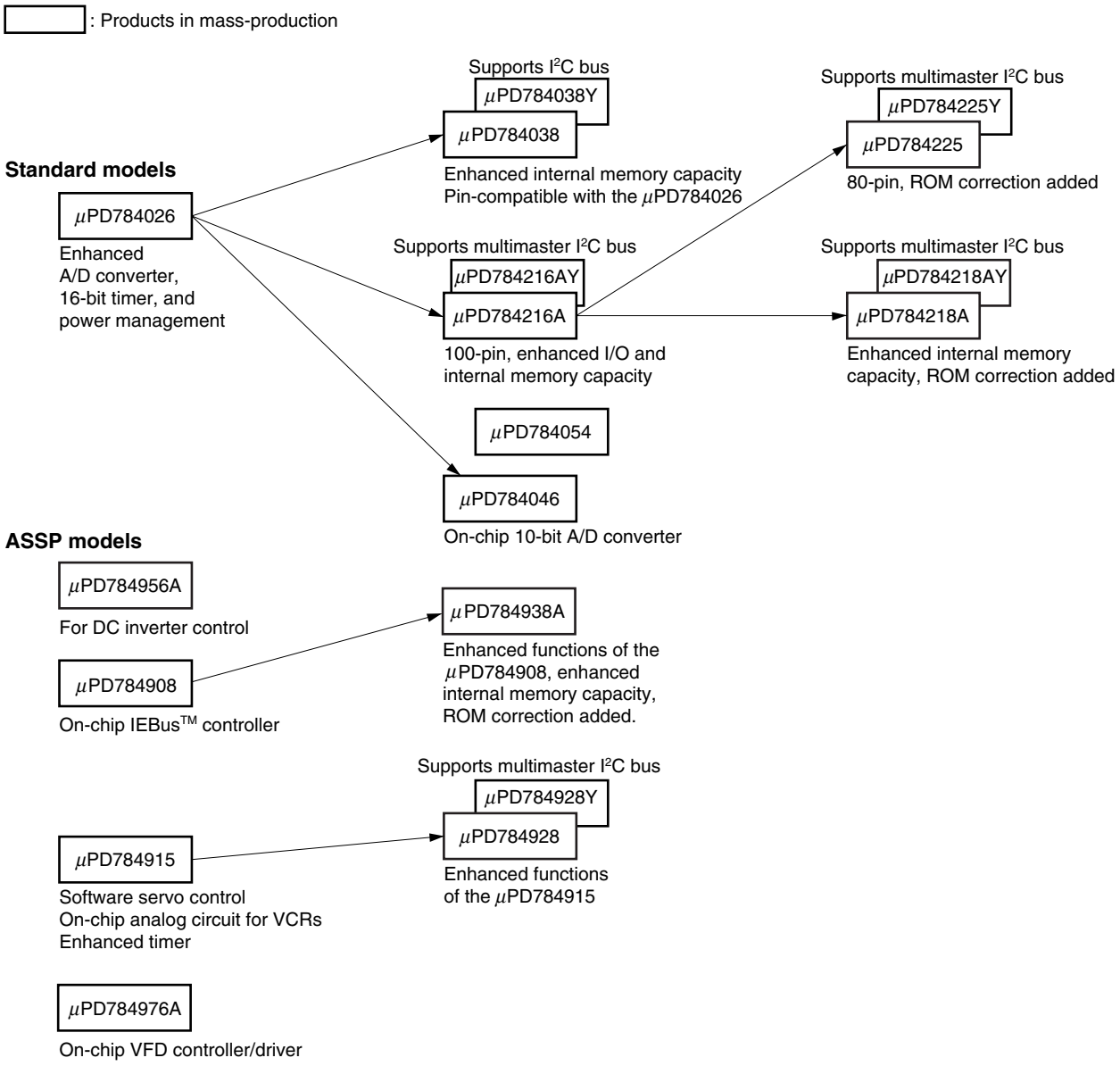
The relationships among the products are shown below.




These products have applications in the following fields.

- Car audio, portable audio, telephones, etc.

## ★ 78K/IV SERIES LINEUP




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

## 1.1 Features

- Inherits the peripheral functions of the μPD780058 Subseries
- Minimum instruction execution time
  - 160 ns (main system clock: @ $f_{XX}$ = 12.5 MHz operation)
  - 61 μs (subsystem clock: @ $f_{XT}$ = 32.768 kHz operation)
- Instruction set suited to control applications
- Interrupt controller (4-level priority)
  - Vectored interrupt servicing, macro service, context switching
- Standby function
  - HALT, STOP, IDLE modes
  - In the low power consumption mode: HALT, IDLE modes (subsystem clock operation)
- On-chip memory: Mask ROM 128 KB (μPD784225)
  96 KB (μPD784224)
  Flash memory 128 KB (μPD78F4225)
  RAM 4,352 bytes (μPD784225, 78F4225)
  3,584 bytes (μPD784224)
- I/O pins: 67
  - Software-programmable pull-up resistors: 57 inputs
  - LED direct drive possible: 16 outputs
- Timers: 16-bit timer/event counter × 1 unit
  8-bit timer/event counter × 2 units
  8-bit timer × 2 units
- Watch timer: 1 channel
- Watchdog timer: 1 channel
- Serial interfaces
  - UART/IOE (3-wire serial I/O): 2 channels (on-chip baud rate generator)
  - CSI/IIC0**Note** (3-wire serial I/O, multimaster supporting $I^2C$ bus**Note**): 1 channel
- A/D converter: 8-bit resolution × 8 channels
- D/A converter: 8-bit resolution × 2 channels
- Real-time output port (by combining with the timer/counters, two stepper motors can be independently controlled.)
- Clock frequency division function
- Clock output function: Select from $f_{XX}$, $f_{XX}/2$, $f_{XX}/2^2$, $f_{XX}/2^3$, $f_{XX}/2^4$, $f_{XX}/2^5$, $f_{XX}/2^6$, $f_{XX}/2^7$, $f_{XT}$
- Buzzer output function: Select from $f_{XX}/2^{10}$, $f_{XX}/2^{11}$, $f_{XX}/2^{12}$, $f_{XX}/2^{13}$
- Power supply voltage: $V_{DD}$ = 1.8 to 5.5 V (μPD784224, 784225, 784224Y, 784225Y)
  $V_{DD}$ = 1.9 to 5.5 V (μPD78F4225, 78F4225Y)

**Note** Only in the μPD784225Y Subseries

## 1.2 Ordering Information

### (1) μPD784225 Subseries

| Part Number | Package | Internal ROM |
|---|---|---|
| μPD784224GC-×××-8BT | 80-pin plastic QFP (14 × 14) | Mask ROM |
| μPD784224GK-×××-9EU | 80-pin plastic TQFP (fine pitch) (12 × 12) | Mask ROM |
| μPD784225GC-×××-8BT | 80-pin plastic QFP (14 × 14) | Mask ROM |
| μPD784225GK-×××-9EU | 80-pin plastic TQFP (fine pitch) (12 × 12) | Mask ROM |
| μPD78F4225GC-8BT | 80-pin plastic QFP (14 × 14) | Flash memory |
| μPD78F4225GK-9EU | 80-pin plastic TQFP (fine pitch) (12 × 12) | Flash memory |

**Note** Under development

**Remark** ××× indicates ROM code suffix.

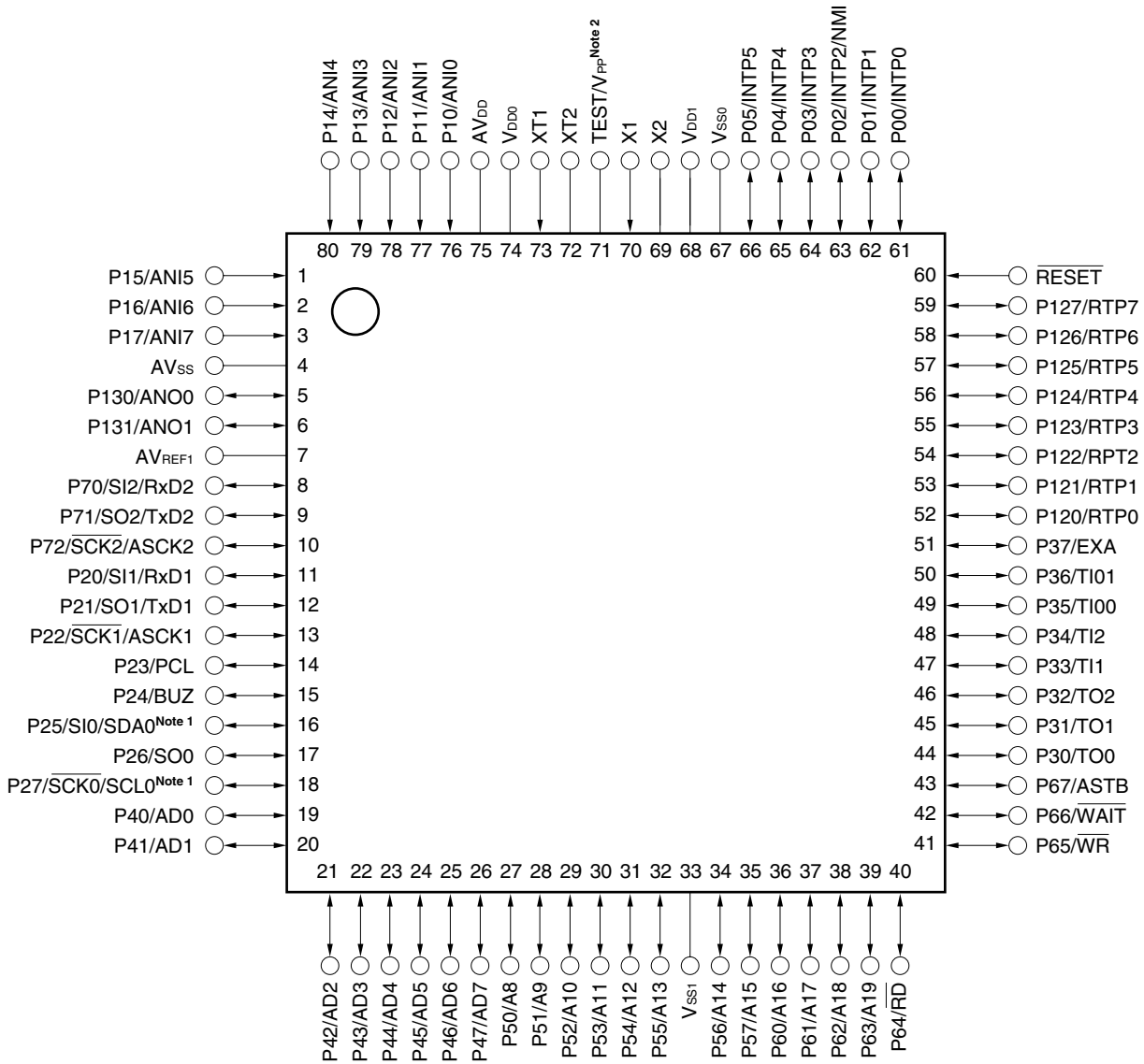
### (2) μPD784225Y Subseries

| Part Number | Package | Internal ROM |
|---|---|---|
| μPD784224YGC-×××-8BT | 80-pin plastic QFP (14 × 14) | Mask ROM |
| μPD784224YGK-×××-9EU | 80-pin plastic TQFP (fine pitch) (12 × 12) | Mask ROM |
| μPD784225YGC-×××-8BT | 80-pin plastic QFP (14 × 14) | Mask ROM |
| μPD784225YGK-×××-9EU | 80-pin plastic TQFP (fine pitch) (12 × 12) | Mask ROM |
| μPD78F4225YGC-8BT | 80-pin plastic QFP (14 × 14) | Flash memory |
| μPD78F4225YGK-9EU | 80-pin plastic TQFP (fine pitch) (12 × 12) | Flash memory |

**Remark** ××× indicates ROM code suffix.

## 1.3 Pin Configuration (Top View)

- **80-pin plastic QFP (14 × 14)**
  μPD784224GC-×××-8BT, 784225GC-×××-8BT, 784224YGC-×××-8BT,
  μPD784225YGC-×××-8BT, 78F4225GC-8BT, 78F4225YGC-8BT
- **80-pin plastic TQFP (fine pitch) (12 × 12)**
  μPD784224GK-×××-9EU, 784225GK-×××-9EU, 784224YGK-×××-9EU,
  μPD784225YGK-×××-9EU, 78F4225GK-9EU, 78F4225YGK-9EU




**Notes 1.** The SDA0 and SCL0 pins are provided only for the μPD784225Y Subseries.

**2.** The VPP pin is provided only in the μPD78F4225 and 78F4225Y.

**Cautions 1. Connect the TEST pin directly to a $V_{SS0}$ or pull down. For the pull-down connection, use of a resistor with a resistance between 470 Ω and 10 kΩ is recommended.**

**2. Connect the $V_{PP}$ pin directly to $V_{SS0}$ or pull down during normal operation. When using a system in which on-chip flash memory is overwritten on board, connect the $V_{PP}$ pin via a pull-down resistor.**

**For the pull-down connection, use of a resistor with a resistance between 470 Ω and 10 kΩ is recommended.**

**3. Connect the $AV_{DD}$ pin to $V_{DD0}$.**

**4. Connect the $AV_{SS}$ pin to $V_{SS0}$.**

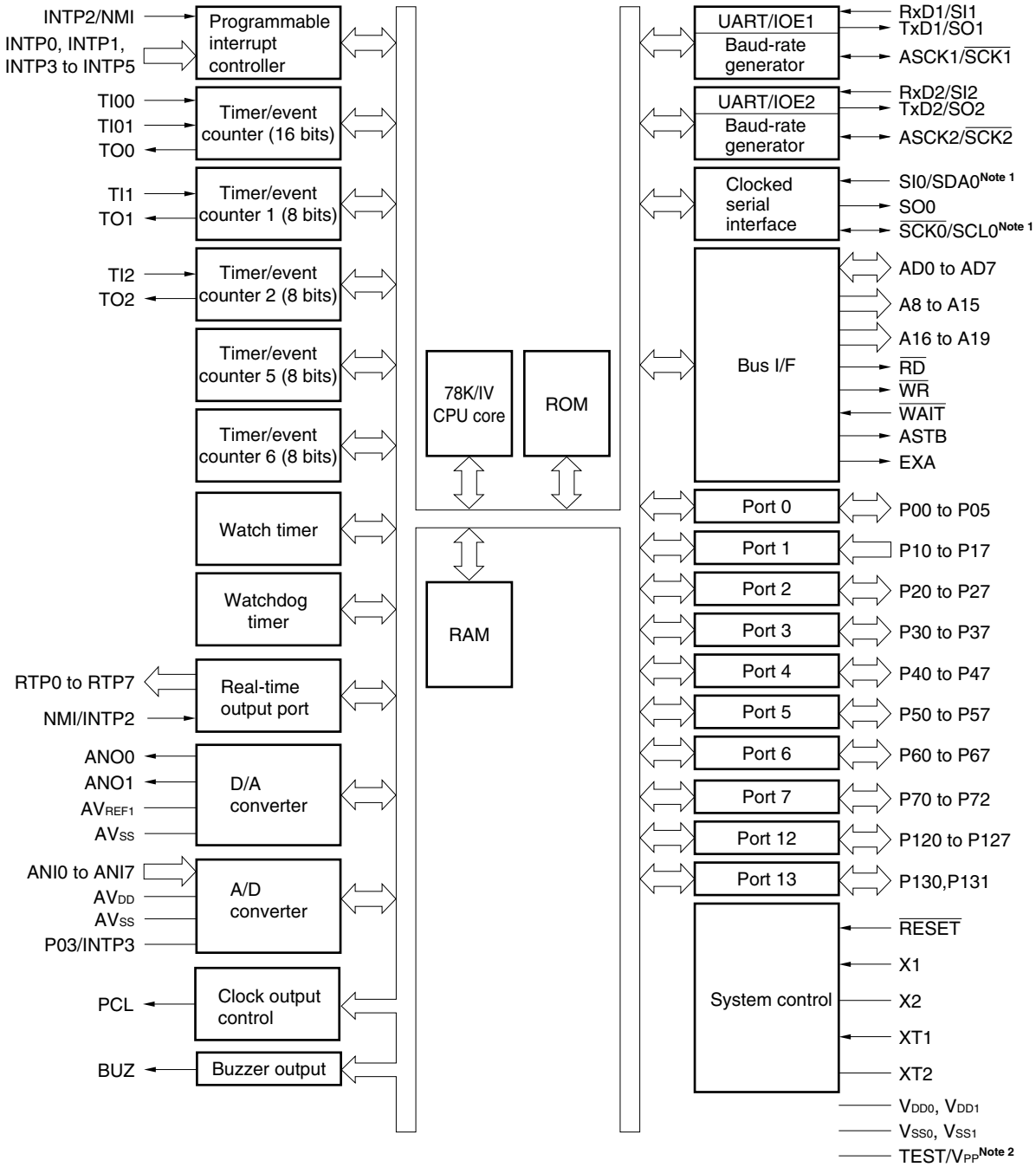
**Remark** When the μPD784225 and 784225Y Subseries are used in applications where the noise generated inside the microcontroller needs to be reduced, the implementation of noise reduction measures, such as supplying voltage to $V_{DD0}$ and $V_{DD1}$ individually and connecting $V_{SS0}$ and $V_{SS1}$ to different ground lines, is recommended. Always use $V_{DD0}$ and $V_{DD1}$, and $V_{SS0}$ and $V_{SS1}$ at the same potential.

| | |
|---|---|
| A8 to A19: | Address bus |
| AD0 to AD7: | Address/data bus |
| ANI0 to ANI7: | Analog input |
| ANO0, ANO1: | Analog output |
| ASCK1, ASCK2: | Asynchronous serial clock |
| ASTB: | Address strobe |
| $AV_{DD}$: | Analog power supply |
| $AV_{REF1}$: | Analog reference voltage |
| $AV_{SS}$: | Analog ground |
| BUZ: | Buzzer clock |
| EXA: | External access status output |
| INTP0 to INTP5: | Interrupt from peripherals |
| NMI: | Non-maskable interrupt |
| P00 to P05: | Port 0 |
| P10 to P17: | Port 1 |
| P20 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 P72: | Port 7 |
| P120 to P127: | Port 12 |
| P130, P131: | Port 13 |
| PCL: | Programmable clock |
| $\overline{RD}$: | Read strobe |
| $\overline{RESET}$: | Reset |
| RTP0 to RTP7: | Real-time output port |
| RxD1, RxD2: | Receive data |
| $\overline{SCK0}$ to $\overline{SCK2}$: | Serial clock |
| SCL0**Note 1**: | Serial clock |
| SDA0**Note 1**: | Serial data |
| SI0 to SI2: | Serial input |
| SO0 to SO2: | Serial output |
| TEST: | Test |
| TI00, TI01, TI1, TI2: | Timer input |
| TO0 to TO2: | Timer output |
| TxD1, TxD2: | Transmit data |
| $V_{DD0}$, $V_{DD1}$: | Power supply |
| $V_{PP}$**Note 2**: | Programming power supply |
| $V_{SS0}$, $V_{SS1}$: | Ground |
| $\overline{WAIT}$: | Wait |
| $\overline{WR}$: | Write strobe |
| X1, X2: | Crystal (Main system clock) |
| XT1, T2: | Crystal (Subsystem clock) |

**Notes 1.** The SDA0 and SCL0 pins are provided only for the μPD784225Y Subseries.

**2.** The $V_{PP}$ pin is provided only in the μPD78F4225 and 78F4225Y.

## 1.4 Block Diagram




**Notes 1.** The SDA0 and SCL0 pins are provided only in the μPD784225Y Subseries.

**2.** The $V_{PP}$ pin is provided only for the μPD78F4225 and 78F4225Y.

**Remark** The internal ROM and RAM capacity varies depending on the product.

## 1.5 Function List

(1/2)

| Item \ Product Name | | μPD784224<br>μPD784224Y | μPD784225<br>μPD784225Y | μPD78F4225<br>μPD78F4225Y |
|---|---|---|---|---|
| Number of basic instructions (mnemonics) | | 113 | | |
| General-purpose registers | | 8 bits × 16 registers × 8 banks or 16 bits × 8 registers × 8 banks (memory mapping) | | |
| ★ Minimum instruction execution time | | • 160 ns (@ 12.5 MHz operation with main system clock)<br>• 61 μs (@ 32.768 kHz operation with subsystem clock) | | |
| Internal memory | ROM | 96 KB<br>(mask ROM) | 128 KB<br>(mask ROM) | 128 KB<br>(flash memory) |
| | RAM | 3,584 bytes | 4,352 bytes | |
| Memory space | | 1 MB of combined program and data space | | |
| I/O ports | Total | 67 | | |
| | CMOS input | 8 | | |
| | CMOS I/O | 59 | | |
| Pins with added functions[Note] | Pins with pull-up resistors | 57 | | |
| | LED direct-drive outputs | 16 | | |
| Real-time output ports | | 4 bits × 2 or 8 bits × 1 | | |
| Timers | | Timer/event counter:<br>(16 bits) | Timer counter × 1<br>Capture/compare register × 2 | Pulse output possible<br>• PPG output<br>• Square wave output<br>• One-shot pulse output |
| | | Timer/event counter 1:<br>(8 bits) | Timer counter × 1<br>Compare register × 1 | Pulse output possible<br>• PWM output<br>• Square wave output |
| | | Timer/event counter 2:<br>(8 bits) | Timer counter × 1<br>Compare register × 1 | Pulse output possible<br>• PWM output<br>• Square wave output |
| | | Timer 5:<br>(8 bits) | Timer counter × 1<br>Compare register × 1 | |
| | | Timer 6:<br>(8 bits) | Timer counter × 1<br>Compare register × 1 | |

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

(2/2)

| Item \ Product Name | | μPD784224<br>μPD784224Y | μPD784225<br>μPD784225Y | μPD78F4225<br>μPD78F4225Y |
|---|---|---|---|---|
| Serial interfaces | | • UART/IOE (3-wire serial I/O): 2 channels (on-chip baud rate generator)<br>• CSI $I^2C$**Note** (3-wire serial I/O, multimaster-supporting $I^2C$ bus**Note**): 1 channel | | |
| A/D converter | | 8-bit resolution × 8 channels | | |
| D/A converter | | 8-bit resolution × 2 channels | | |
| Clock output | | Select from $f_{XX}$, $f_{XX}/2$, $f_{XX}/2^2$, $f_{XX}/2^3$, $f_{XX}/2^4$, $f_{XX}/2^5$, $f_{XX}/2^6$, $f_{XX}/2^7$, $f_{XT}$ | | |
| Buzzer output | | Select from $f_{XX}/2^{10}$, $f_{XX}/2^{11}$, $f_{XX}/2^{12}$, $f_{XX}/2^{13}$ | | |
| Watch timer | | 1 channel | | |
| Watchdog timer | | 1 channel | | |
| Standby | | • HALT, STOP, IDLE modes<br>• In the low power consumption mode (CPU operation by subsystem clock): HALT/IDLE mode | | |
| Interrupts | Hardware sources | 25 (internal: 18, external: 7) | | |
| | Software sources | BRK instruction, BRKCS instruction, operand error | | |
| | Non-maskable | Internal: 1, external: 1 | | |
| | Maskable | Internal: 17, external: 6 | | |
| | | • 4-level programmable priority<br>• Three processing formats: Vectored interrupt, macro service, context switching | | |
| Power supply voltage | | $V_{DD}$ = 1.8 to 5.5 V | | $V_{DD}$ = 1.9 to 5.5 V |
| Package | | • 80-pin plastic TQFP (fine pitch) (12 × 12)<br>• 80-pin plastic QFP (14 × 14) | | |

**Note** Only in the μPD784225Y Subseries

An overview of the timers is shown below. (For details, refer to **CHAPTER 8 16-BIT TIMER/EVENT COUNTER, CHAPTER 9 8-BIT TIMER/EVENT COUNTERS 1, 2,** and **CHAPTER 10 8-BIT TIMERS 5, 6**.)

| Item \ Name | | 16-Bit Timer/ Event Counter | 8-Bit Timer/ Event Counter 1 | 8-Bit Timer/ Event Counter 2 | 8-Bit Timer 5 | 8-Bit Timer 6 |
|---|---|---|---|---|---|---|
| Count width | 8 bits | – | √ | √ | √ | √ |
| | 16 bits | √ | √Note | | √Note | |
| Operation mode | Interval timer | 1ch | 1ch | 1ch | 1ch | 1ch |
| | External event counter | √ | √ | √ | – | – |
| Functions | Timer output | 1ch | 1ch | 1ch | – | – |
| | PPG output | √ | – | – | – | – |
| | PWM output | – | √ | √ | – | – |
| | Square-wave output | √ | √ | √ | – | – |
| | One-shot pulse output | √ | – | – | – | – |
| | Pulse width measurement | 2 inputs | – | – | – | – |
| | Number of interrupt requests | 2 | 1 | 1 | 1 | 1 |

**Note** Can also be used as a 16-bit timer/event counter or 16-bit timer when connected in cascade. When using as a 16-bit timer/event counter, the following functions are available.

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

An overview of the serial interfaces is shown below. (For details, refer to **CHAPTER 16 ASYNCHRONOUS SERIAL INTERFACE/3-WIRE SERIAL I/O, CHAPTER 17 3-WIRE SERIAL I/O MODE,** and **CHAPTER 18 $I^2C$ BUS MODE ($\mu$PD784225Y SUBSERIES ONLY)**.)

| Function | UART1/IOE1 | UART2I/IOE2 | CSI | IIC0 |
|---|---|---|---|---|
| Operation stop mode | √ | √ | √ | – |
| Asynchronous serial interface mode | √ | √ | – | – |
| 3-wire serial I/O mode | – | – | √ (Fixed to MSB) | – |
| $I^2C$ bus mode**Note** | – | – | – | √ **Note** |

**Note** Only in the $\mu$PD784225Y Subseries

## 1.6 Differences Between μPD784225 Subseries Products and μPD784225Y Subseries Products

| Item \ Product Name | μPD784224<br>μPD784224Y | μPD784225<br>μPD784225Y | μPD78F4225<br>μPD78F4225Y |
|---|---|---|---|
| Internal ROM | 96 KB<br>(mask ROM) | 128 KB<br>(mask ROM) | 128 KB<br>(flash memory) |
| Internal RAM | 3,584 bytes | 4,352 bytes | |
| Power supply voltage | $V_{DD}$ = 1.8 to 5.5 V | | $V_{DD}$ = 1.9 to 5.5 V |
| Internal memory size switching register (IMS)**Note** | None | | Provided |
| TEST pin | Provided | | None |
| $V_{PP}$ pin | None | | Provided |

**Note** The internal flash memory capacity and internal RAM capacity can be changed with the internal memory size switching register (IMS).

# CHAPTER 2 PIN FUNCTIONS

## 2.1 Pin Function List

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

| Pin Name | I/O | Alternate Function | Function |
|---|---|---|---|
| P00 | I/O | INTP0 | Port 0 (P0):<br>• 6-bit I/O port<br>• Input/output can be specified in 1-bit units.<br>• Regardless of whether the input or output mode is specified, use of an on-chip pull-up resistor can be specified by a software setting in 1-bit units. |
| P01 | | INTP1 | |
| P02 | | INTP2/NMI | |
| P03 | | INTP3 | |
| P04 | | INTP4 | |
| P05 | | INTP5 | |
| P10 to P17 | Input | ANI0 to ANI7 | Port 1 (P1):<br>• 8-bit input-only port |
| P20 | I/O | RxD1/SI1 | Port 2 (P2):<br>• 8-bit I/O port<br>• Input/output can be specified in 1-bit units.<br>• Regardless of whether the input or output mode is specified, use of an on-chip pull-up resistor can be specified by a software setting in 1-bit units. |
| P21 | | TxD1/SO1 | |
| P22 | | ASCK1/$\overline{\text{SCK1}}$ | |
| P23 | | PCL | |
| P24 | | BUZ | |
| P25 | | SI0/SDA0**Note** | |
| P26 | | SO0 | |
| P27 | | $\overline{\text{SCK0}}$/SCL0**Note** | |
| P30 | I/O | TO0 | Port 3 (P3):<br>• 8-bit I/O port<br>• Input/output can be specified in 1-bit units.<br>• Regardless of whether the input or output mode is specified, use of an on-chip pull-up resistor can be specified by a software setting in 1-bit units. |
| P31 | | TO1 | |
| P32 | | TO2 | |
| P33 | | TI1 | |
| P34 | | TI2 | |
| P35 | | TI00 | |
| P36 | | TI01 | |
| P37 | | EXA | |
| P40 to P47 | I/O | AD0 to AD7 | Port 4 (P4):<br>• 8-bit I/O port<br>• Input/output can be specified in 1-bit units.<br>• For input mode pins, use of on-chip pull-up resistors can be specified for all pins by a software setting.<br>• LEDs can be driven directly. |

**Note** The SDA0 and SCL0 pins are provided only in the $\mu$PD784225Y Subseries.

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

| Pin Name | I/O | Alternate Function | Function |
|---|---|---|---|
| P50 to P57 | I/O | A8 to A15 | Port 5 (P5):<br>• 8-bit I/O port<br>• Input/output can be specified in 1-bit units.<br>• For input mode pins, use of on-chip pull-up resistors can be specified for all pins by a software setting.<br>• LEDs can be driven directly. |
| P60 | I/O | A16 | Port 6 (P6):<br>• 8-bit I/O port<br>• Input/output can be specified in 1-bit units.<br>• For input mode pins, use of on-chip pull-up resistors can be specified for all pins by a software setting. |
| P61 | | A17 | |
| P62 | | A18 | |
| P63 | | A19 | |
| P64 | | $\overline{RD}$ | |
| P65 | | $\overline{WR}$ | |
| P66 | | $\overline{WAIT}$ | |
| P67 | | ASTB | |
| P70 | I/O | RxD2/SI2 | Port 7 (P7):<br>• 3-bit I/O port<br>• Input/output can be specified in 1-bit units.<br>• Regardless of whether the input or output mode is specified, use of an on-chip pull-up resistor can be specified by a software setting in 1-bit units. |
| P71 | | TxD2/SO2 | |
| P72 | | ASCK2/$\overline{SCK2}$ | |
| P120 to P127 | I/O | RTP0 to RTP7 | Port 12 (P12):<br>• 8-bit I/O port<br>• Input/output can be specified in 1-bit units.<br>• Regardless of whether the input or output mode is specified, use of an on-chip pull-up resistor can be specified by a software setting in 1-bit units. |
| P130, P131 | I/O | ANO0, ANO1 | Port 13 (P13):<br>• 2-bit I/O port<br>• Input/output can be specified in 1-bit units. |

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

| Pin Name | I/O | Alternate Function | Function |
|---|---|---|---|
| TI00 | Input | P35 | External count clock input to 16-bit timer counter |
| TI01 | | P36 | Capture trigger signal input to capture/compare register 00 |
| TI1 | | P33 | External count clock input to 8-bit timer counter 1 |
| TI2 | | P34 | External count clock input to 8-bit timer counter 2 |
| TO0 | Output | P30 | 16-bit timer output (TM0) (also used as 14-bit PWM output) |
| TO1 | | P31 | 8-bit timer output (TM1) (also used as 8-bit PWM output) |
| TO2 | | P32 | 8-bit timer output (TM2) (also used as 8-bit PWM output) |
| RxD1 | Input | P20/SI1 | Serial data input (UART1) |
| RxD2 | | P70/SI2 | Serial data input (UART2) |
| TxD1 | Output | P21/SO1 | Serial data output (UART1) |
| TxD2 | | P71/SO2 | Serial data output (UART2) |
| ASCK1 | Input | P22/$\overline{\text{SCK1}}$ | Baud rate clock input (UART1) |
| ASCK2 | | P72/$\overline{\text{SCK2}}$ | Baud rate clock input (UART2) |
| SI0 | Input | P25/SDA0**Note** | Serial data input (3-wire serial I/O0) |
| SI1 | | P20/RxD1 | Serial data input (3-wire serial I/O1) |
| SI2 | | P70/RxD2 | Serial data input (3-wire serial I/O2) |
| SO0 | Output | P26 | Serial data output (3-wire serial I/O0) |
| SO1 | | P21/TxD1 | Serial data output (3-wire serial I/O1) |
| SO2 | | P71/TxD2 | Serial data output (3-wire serial I/O2) |
| SDA0**Note** | I/O | P25/SI0 | Serial data input/output ($I^2C$ bus) |
| $\overline{\text{SCK0}}$ | | P27/SCL0**Note** | Serial clock input/output (3-wire serial I/O0) |
| $\overline{\text{SCK1}}$ | | P22/ASCK1 | Serial clock input/output (3-wire serial I/O1) |
| $\overline{\text{SCK2}}$ | | P72/ASCK2 | Serial clock input/output (3-wire serial I/O2) |
| SCL0**Note** | | P27/$\overline{\text{SCK0}}$ | Serial clock input/output ($I^2C$ bus) |
| NMI | Input | P02/INTP2 | Non-maskable interrupt request input |
| INTP0 | | P00 | External interrupt request input |
| INTP1 | | P01 | |
| INTP2 | | P02/NMI | |
| INTP3 | | P03 | |
| INTP4 | | P04 | |
| INTP5 | | P05 | |

**Note** The SDA0 and SCL0 pins are provided only in the $\mu$PD784225Y Subseries.

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

| Pin Name | I/O | Alternate Function | Function |
|---|---|---|---|
| PCL | Output | P23 | Clock output (for main system clock, subsystem clock trimming) |
| BUZ | | P24 | Buzzer output |
| RTP0 to RTP7 | | P120 to P127 | Real-time output port that outputs data synchronized with the trigger |
| AD0 to AD7 | I/O | P40 to P47 | Lower address/data bus when the memory is externally expanded |
| A8 to A15 | Output | P50 to P57 | Middle address bus when the memory is externally expanded |
| A16 to A19 | | P60 to P63 | Higher address bus when the memory is externally expanded |
| $\overline{RD}$ | | P64 | Strobe signal output for external memory read |
| $\overline{WR}$ | | P65 | Strobe signal output for external memory write |
| $\overline{WAIT}$ | Input | P66 | Wait insertion during external memory access |
| ASTB | Output | P67 | Strobe output that externally latches the address information that is output to ports 4 to 6 in order to access external memory |
| EXA | | P37 | External access status output |
| $\overline{RESET}$ | Input | — | System reset input |
| X1 | | — | Crystal connection for main system clock oscillation |
| X2 | — | | |
| XT1 | Input | — | Crystal connection for subsystem clock oscillation |
| XT2 | — | | |
| ANI0 to ANI7 | Input | P10 to P17 | Analog voltage input for A/D converter |
| ANO0, ANO1 | Output | P130, P131 | Analog voltage output for D/A converter |
| $AV_{REF1}$ | — | — | Reference voltage applied for D/A converter |
| $AV_{DD}$ | | | Positive power supply for A/D converter. Connect to $V_{DD0}$. |
| $AV_{SS}$ | | | Ground for A/D converter and D/A converter. Connect to $V_{SS0}$. |
| $V_{DD0}$ | | | Positive power supply for ports |
| $V_{SS0}$ | | | Ground potential for ports |
| $V_{DD1}$ | | | Positive power supply (excluding ports) |
| $V_{SS1}$ | | | Ground potential (excluding ports) |
| TEST | | $V_{PP}$**Note** | Connect directly to $V_{SS0}$ or pull down (IC test pin). For the pull-down connection, use of a resistor with a resistance between 470 Ω and 10 kΩ is recommended. |
| $V_{PP}$**Note** | | TEST | Flash memory programming mode setting<br>High voltage application pin during program write/verify<br>Connect via a pull-down resistor in flash memory programming mode.<br>For the pull-down connection, use of a resistor with a resistance between 470 Ω and 10 kΩ is recommended. |

**Note** The $V_{PP}$ pin is provided only in the μPD78F4225 and 78F4225Y.

## 2.2 Pin Function Description

**(1) P00 to P05 (Port 0)**

These pins constitute a 6-bit I/O port. In addition to I/O port pins, they also function as external interrupt request inputs. The following operation modes can be specified in 1-bit units.

**(a) Port mode**

These pins function as a 6-bit I/O port. Input or output can be specified in 1-bit units by means of the port 0 mode register. Regardless of whether the input mode or output mode is specified, pull-up resistors can be connected in 1-bit units using pull-up resistor option register 0.

**(b) Control mode**

These pins function as external interrupt request inputs.

**(i) INTP0 to INTP5**

INTP0 to INTP5 are external interrupt request input pins for which the valid edge can be selected (rising edge, falling edge, or both rising and falling edges). The valid edge can be specified by the external interrupt rising edge enable register and the external interrupt falling edge enable register. INTP2 also becomes the external trigger signal input pin of the real-time output port via valid edge input.

**(ii) NMI**

This is the external non-maskable interrupt request input pin. The valid edge can be specified by the external interrupt rising edge enable register and the external interrupt falling edge enable register.

**(2) P10 to P17 (Port 1)**

These pins constitute an 8-bit input-only port. In addition to general-purpose input port pins, they also function as the analog inputs for the A/D converter.
On-chip pull-up resistors are not available.

**(a) Port mode**

These pins function as an 8-bit input-only port.

**(b) Control mode**

These pins function as the analog input pins (ANI0 to ANI7) of the A/D converter. The values are undefined when the pins specified for analog input are read.

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

These pins constitute an 8-bit I/O port. In addition to I/O port pins, they also function as the data I/O, clock I/O, clock output, and buzzer output of the serial interface.

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

**(a) Port mode**

These pins function as an 8-bit I/O port. Input or output can be specified in 1-bit units by means of the port 2 mode register. Regardless of whether the input mode or output mode is specified, pull-up resistors can be connected in 1-bit units using pull-up resistor option register 2.

**(b) Control mode**

These pins function as the data I/O pins, clock I/O pins, clock output pins, and buzzer output pins of the serial interface.

Pins P25 to P27 can be specified as N-channel open drain by the port function control register (PF2) (only in the μPD784225Y Subseries).

**(i) SI0, SI1, SO0, SO1, SDA0[Note]**

These pins are the I/O pins for serial data in the serial interface.

**(ii) $\overline{\text{SCK0}}$, $\overline{\text{SCK1}}$, SCL0[Note]**

These pins are the I/O pins for the serial clock of the serial interface.

**(iii) RxD1, TxD1**

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

**Note** Only in the μPD784225Y Subseries

**(iv) ASCK1**

This is the input pin for the baud rate clock of the asynchronous serial interface.

**(v) PCL**

This is the clock output pin.

**(vi) BUZ**

This is the buzzer output pin.

**(4) P30 to P37 (Port 3)**

These pins constitute an 8-bit I/O port. In addition to I/O port pins, they also function as timer I/O.
The following operation modes can be specified in 1-bit units.

**(a) Port mode**

These pins function as an 8-bit I/O port. Input or output can be specified in 1-bit units by means of the port 3 mode register. Regardless of whether the input mode or output mode is specified, pull-up resistors can be connected in 1-bit units using pull-up resistor option register 3.

**(b) Control mode**

These pins function as timer I/O.

**(i) TI00**

This is the external clock input pin to the 16-bit timer/event counter.
This is also used as the pin for capture trigger signal input to capture/compare registers 00 and 01.

**(ii) TI01**

This is the pin for capture trigger signal input to capture/compare register 00.

**(iii) TI1, TI2**

These are pins for external clock input to the 8-bit timer counter.

**(iv) TO0 to TO2**

These are timer output pins.

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

These pins constitute an 8-bit I/O port. In addition to I/O port pins, they also function as an address/data bus. LEDs can be directly driven.
The following operation modes can be specified in 1-bit units.

**(a) Port mode**

These pins function as an 8-bit I/O port. Input or output can be specified in 1-bit units by means of the port 4 mode register. When used as an input port, pull-up resistors can be connected in 8-bit units with bit 4 (PUO4) of the pull-up resistor option register.

**(b) Control mode**

These pins function as the lower address/data bus pins (AD0 to AD7) when in the external memory expansion mode. If PUO4 = 1, pull-up resistors can be connected.

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

These pins constitute an 8-bit I/O port. In addition to I/O port pins, they also function as an address bus. LEDs can be directly driven.

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

**(a) Port mode**

These pins function as an 8-bit I/O port. Input or output can be specified in 1-bit units by means of the port 5 mode register. When used as an input port, pull-up resistors can be connected in 8-bit units with bit 5 (PUO5) of the pull-up resistor option register.

**(b) Control mode**

These pins function as the middle address bus pins (A8 to A15) in the external memory expansion mode. If PUO5 = 1, pull-up resistors can be connected.

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

These pins constitute an 8-bit I/O port. In addition to I/O port pins, they also function as an address bus and control signal outputs in the external memory expansion mode.

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

**(a) Port mode**

These pins function as an 8-bit I/O port. Input or output can be specified in 1-bit units by means of the port 6 mode register. When used as an input port, pull-up resistors can be connected in 8-bit units with bit 6 (PUO6) of the pull-up resistor option register.

**(b) Control mode**

These pins function as the higher address bus pins (A16 to A19) in the external memory expansion mode. P64 to P67 function as the control signal output pins ($\overline{RD}$, $\overline{WR}$, $\overline{WAIT}$, ASTB) in the external memory expansion mode. If PUO6 = 1 in the external memory expansion mode, pull-up resistors can be connected.

**Caution When external waits are not used in the external memory expansion mode, P66 can be used as an I/O port pin.**

**(8) P70 to P72 (Port 7)**

These pins constitute a 3-bit I/O port. In addition to I/O port pins, they also function as the data I/O and clock I/O of the serial interface.

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

**(a) Port mode**

These pins function as a 3-bit I/O port. Input or output can be specified in 1-bit units by means of the port 7 mode register. Regardless of whether the input mode or output mode is specified, pull-up resistors can be connected in 1-bit units using pull-up resistor option register 7.

**(b) Control mode**

These pins function as data I/O and clock I/O for the serial interface.

**(i) SI2, SO2**

These are the I/O pins for serial data in the serial interface.

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

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

**(iii) RxD2, TxD2**

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

**(iv) ASCK2**

This is the baud rate clock input pin in the asynchronous serial interface.

**(9) P120 to P127 (Port 12)**

These pins constitute an 8-bit I/O port. In addition to I/O port pins, they also function as a real-time output port. The following operation modes can be specified in 1-bit units.

**(a) Port mode**

These pins function as an 8-bit I/O port. Input or output can be specified in 1-bit units by means of the port 12 mode register. Regardless of whether the input mode or output mode is specified, pull-up resistors can be connected in 1-bit units using pull-up resistor option register 12.

**(b) Control mode**

These pins function as a real-time output port (RTP0 to RTP7) that outputs data synchronized with a trigger. When the pins specified as the real-time output port are read, 0 is read.

**(10) P130, P131 (Port 13)**

These pins constitute a 2-bit I/O port. In addition to I/O port pins, they also function as the analog outputs of the D/A converter.

The following operation modes can be specified in 2-bit units.

**(a) Port mode**

These pins function as a 2-bit I/O port. Input or output can be specified in 1-bit units by means of the port 13 mode register. On-chip pull-up resistors are not available.

**(b) Control mode**

These pins function as the analog outputs (ANO0, ANO1) of the D/A converter. The values are undefined when the pins specified as analog outputs are read.

**Caution If the D/A converter uses only one channel when $AV_{REF1} < V_{DD}$, use either of the following processes at pins that are not used for analog output.**

- **Set the port mode register (PM13X) to 1 (input mode) and connect to $V_{SS0}$.**
- **Set the port mode register (PM13X) to 0 (output mode). Set the output latch to 0 and output a low level.**

**(11) $AV_{REF1}$**

This is the reference power input pin of the D/A converter.

If the D/A converter is not used, connect to the $V_{DD0}$ pin.

**(12) $AV_{DD}$**

This is the analog power supply pin of the A/D converter. Even if the A/D converter is not used, always use this pin at the same potential as the $V_{DD0}$ pin. $AV_{DD}$ functions alternately as the reference voltage input pin of the A/D converter.

**(13) $AV_{SS}$**

This is the ground potential pin of the A/D converter. Even if the A/D converter is not used, always use this pin at the same potential as the $V_{SS0}$ pin.

**(14) $\overline{\text{RESET}}$**

This is the active low system reset input pin.

**(15) X1, X2**

These are the crystal resonator connection pins for main system clock oscillation.

When an external clock is supplied, input this clock signal at X1, and its inverted signal at X2.

**(16) XT1, XT2**

These are the crystal resonator connection pins for subsystem clock oscillation.

When an external clock is supplied, input this clock signal at XT1, and its inverted signal at XT2.

**(17) $V_{DD0}$, $V_{DD1}$**

$V_{DD0}$ is the positive power supply pin for the ports.

$V_{DD1}$ is the positive power supply pin for other than the ports and analog pins.

Always use this pin at the same potential as pin $V_{DD0}$ and $V_{DD1}$.

**(18) $V_{SS0}$, $V_{SS1}$**

$V_{SS0}$ is the ground potential pin for the ports.

$V_{SS1}$ is the ground potential pin for other than the ports and analog pins.

Always use this pin at the same potential as pin $V_{SS0}$ and $V_{SS1}$.

**(19) $V_{PP}$ ($\mu$PD78F4225, 78F4225Y only)**

This is the high-voltage application pin when setting the flash memory programming mode and writing or verifying the program.

In the normal operation mode, connect directly to $V_{SS0}$ or pull down.

For the pull-down connection, use of a resistor with a resistance between 470 Ω and 10 kΩ is recommended.

**(20) TEST**

This is the pin used in the IC test. Connect directly to $V_{SS0}$ or pull down.

For the pull-down connection, use of a resistor with a resistance between 470 Ω and 10 kΩ is recommended.

## 2.3 Pin I/O Circuits and Recommended Connection of Unused Pins

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

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

| Pin Name | I/O Circuit Type | I/O | Recommended Connection of Unused Pins |
|---|---|---|---|
| P00/INTP0 | 8-K | I/O | Input: Independently connect to $V_{SS0}$ via a resistor.<br>Output: Leave open. |
| P01/INTP1 | | | |
| P02/INTP2/NMI | | | |
| P03/INTP3 to P05/INTP5 | | | |
| P10/ANI0 to P17/ANI7 | 9 | Input | Connect to $V_{SS0}$ or $V_{DD0}$. |
| P20/RxD1/SI1 | 10-I | I/O | Input: Independently connect to $V_{SS0}$ via a resistor.<br>Output: Leave open. |
| P21/TxD1/SO1 | 10-J | | |
| P22/ASCK1/$\overline{SCK1}$ | 10-I | | |
| P23/PCL | 10-J | | |
| P24/BUZ | | | |
| P25/SDA0[Note]/SI0 | 10-I | | |
| P26/SO0 | 10-J | | |
| P27/SCL0[Note]/$\overline{SCK0}$ | 10-I | | |
| P30/TO0 to P32/TO2 | 8-M | | |
| P33/TI1, P34/TI2 | 8-K | | |
| P35/TI00, P36/TI01 | 8-L | | |
| P37/EXA | 8-M | | |
| P40/AD0 to P47/AD7 | 5-H | | |
| P50/A8 to P57/A15 | | | |
| P60/A16 to P63/A19 | | | |
| P64/$\overline{RD}$ | | | |
| P65/$\overline{WR}$ | | | |
| P66/$\overline{WAIT}$ | | | |
| P67/ASTB | | | |
| P70/RxD2/SI2 | 8-K | | |
| P71/TxD2/SO2 | 8-L | | |
| P72/ASCK2/$\overline{SCK2}$ | 8-K | | |

**Note** The SDA0 and SCL0 pins are provided only in the $\mu$PD784225Y Subseries.

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

| Pin Name | I/O Circuit Type | I/O | Recommended Connection of Unused Pins |
|---|---|---|---|
| P120/RTP0 to P127/RTP7 | 8-K | I/O | Input: Independently connect to $V_{SS0}$ via a resistor.<br>Output: Leave open. |
| P130/ANO0, P131/ANO1 | 12-D | | |
| $\overline{\text{RESET}}$ | 2-G | Input | — |
| XT1 | 16 | | Connect to $V_{SS0}$. |
| XT2 | | — | Leave open. |
| $AV_{REF1}$ | — | | Connect to $V_{DD0}$. |
| $AV_{DD}$ | | | |
| $AV_{SS}$ | | | Connect to $V_{SS0}$. |
| TEST/$V_{PP}$**Note** | | | Connect directly to $V_{SS0}$ or pull down. For the pull-down connection, use of a resistor with a resistance between 470 Ω and 10 kΩ is recommended. |

**Note** The $V_{PP}$ pin is provided only in the μPD78F4225, 78F4225Y.

**Remark** Since the type numbers are unified in the 78K Series, they are not always sequential in each product (not all the circuits are incorporated).

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

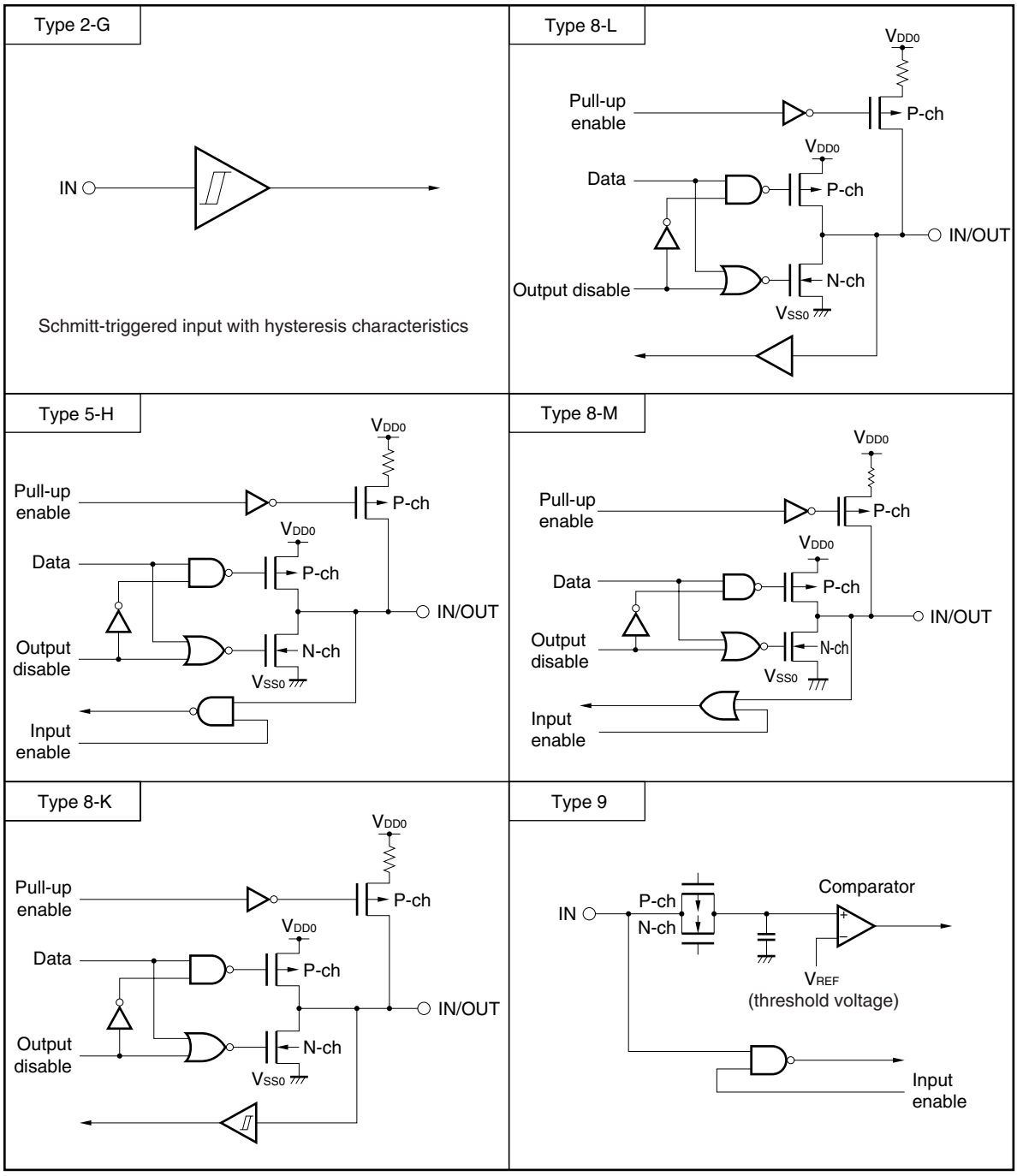
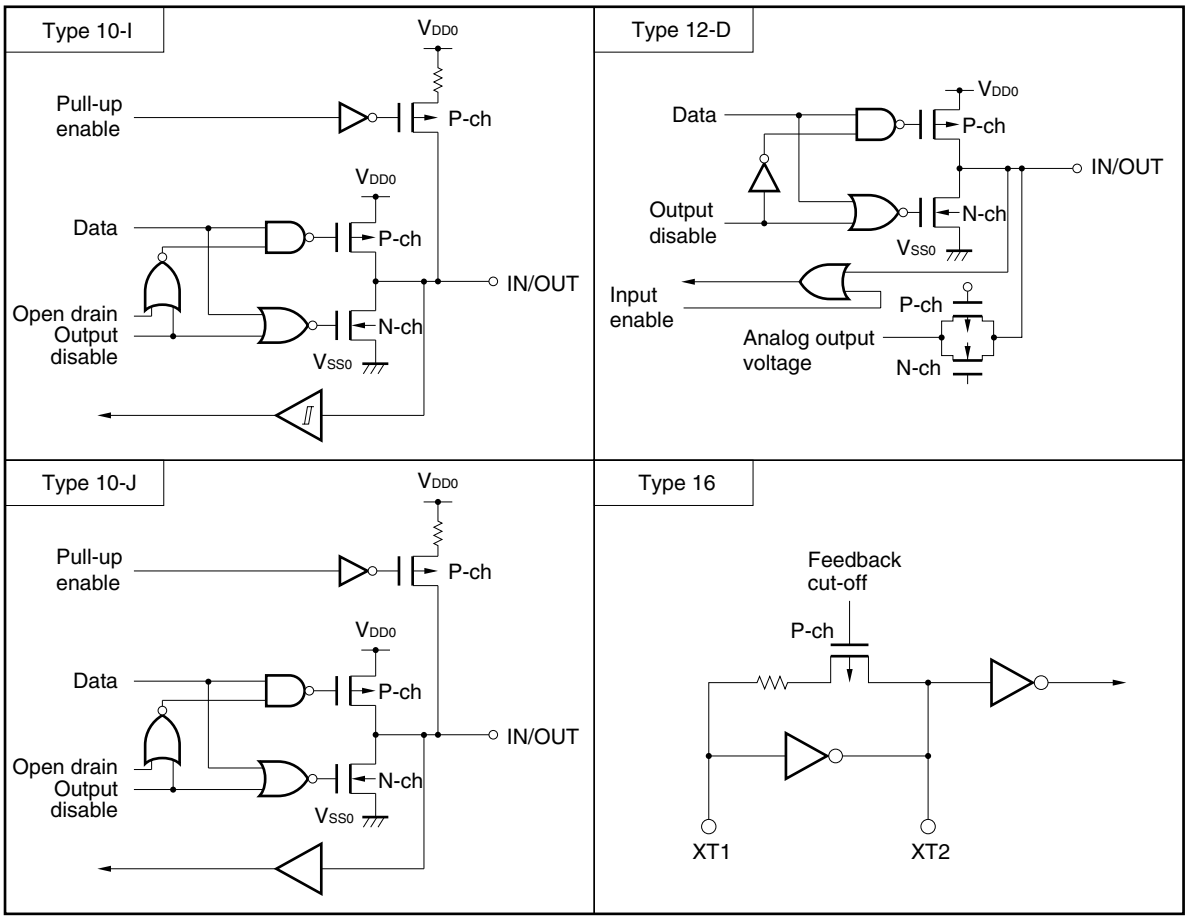


**Figure 2-1. Pin I/O Circuits (2/2)**



# CHAPTER 3 CPU ARCHITECTURE

## 3.1 Memory Space

The $\mu$PD784225 can access a 1 MB space. The mapping of the internal data area differs depending on the LOCATION instruction (special function registers and internal RAM). The LOCATION instruction must always be executed after releasing reset and cannot be used more than once.

The program after releasing reset must be as follows.

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

**(1) When the LOCATION 0H instruction is executed**

- **Internal memory**

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

| Part Number | Internal Data Area | Internal ROM Area |
|---|---|---|
| μPD784224 | 0F100H to 0FFFFH | 00000H to 0F0FFH<br>10000H to 17FFFH |
| μPD784225 | 0EE00H to 0FFFFH | 00000H to 0EDFFH<br>10000H to 1FFFFH |

**Caution The following area, which is overlapped with the internal data area in the on-chip ROM, cannot be used when the LOCATION 0H instruction is executed.**

| Part Number | Use-Prohibited Area |
|---|---|
| μPD784224 | 0F100H to 0FFFFH (3,840 bytes) |
| μPD784225 | 0EE00H to 0FFFFH (4,608 bytes) |

- **External memory**

  External memory is accessed in the external memory expansion mode.

**(2) When the LOCATION 0FH instruction is executed**

- **Internal memory**

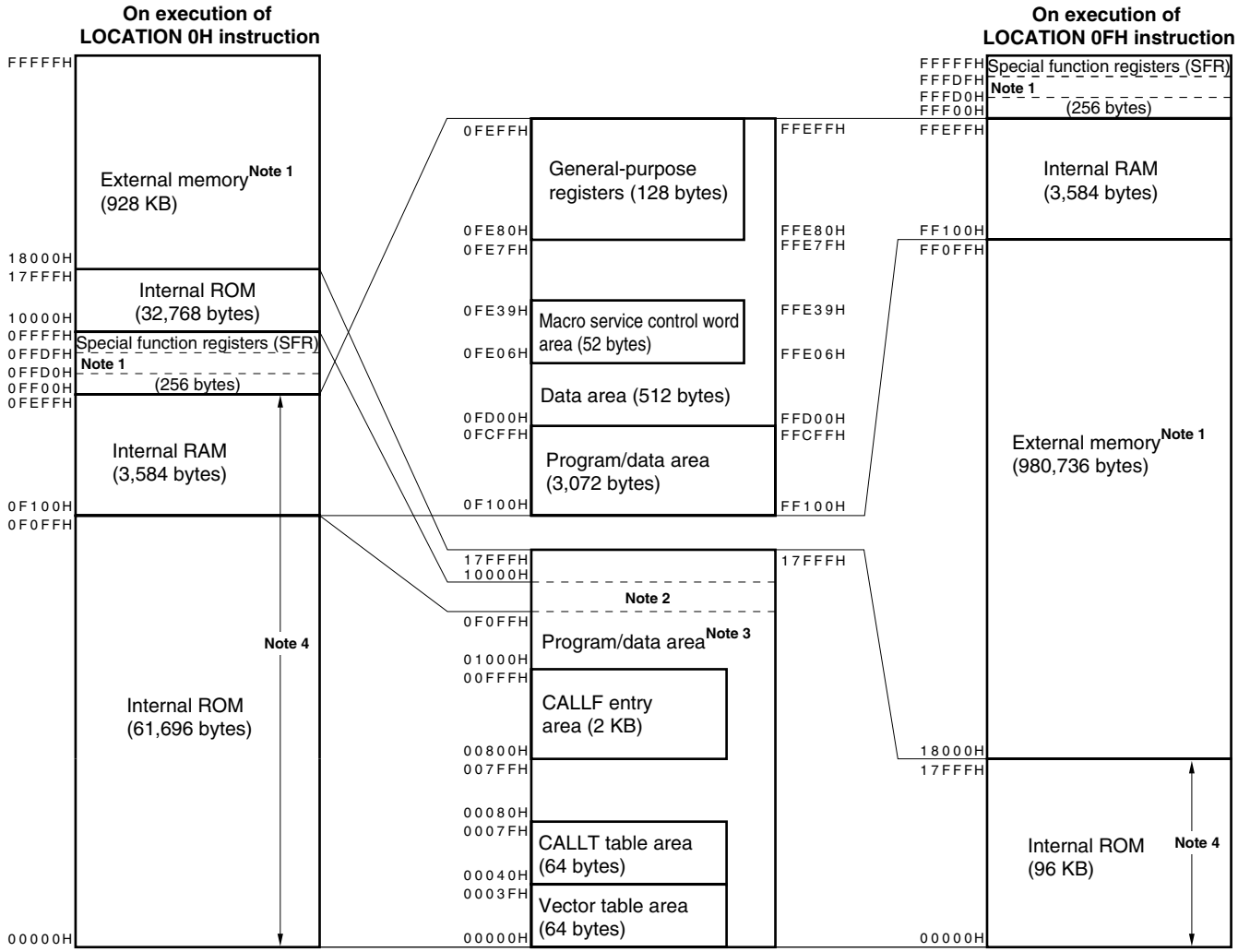
  The internal data area and internal ROM area are as follows.

| Part Number | Internal Data Area | Internal ROM Area |
|---|---|---|
| μPD784224 | FF100H to FFFFFH | 00000H to 17FFFH |
| μPD784225 | FEE00H to FFFFFH | 00000H to 1FFFFH |

- **External memory**

  External memory is accessed in the external memory expansion mode.

**Figure 3-1. μPD784224 Memory Map**



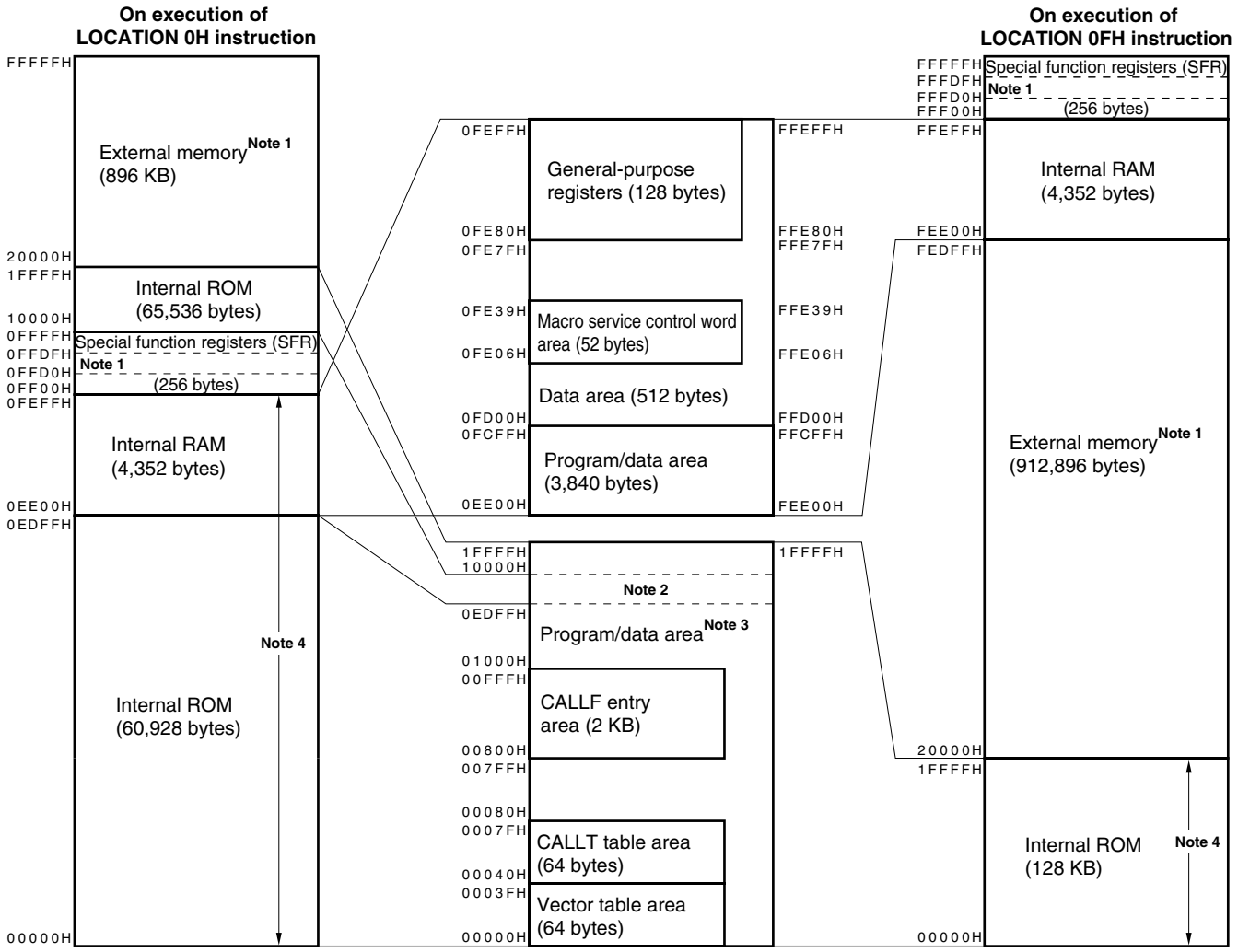

**Notes 1.** Access in the external memory expansion mode.

**2.** The 3,840 bytes in this area can be used as the internal ROM only when the LOCATION 0FH instruction is executed.

**3.** LOCATION 0H instruction execution: 94,464 bytes; LOCATION 0FH instruction execution: 98,304 bytes.

**4.** This is the base area and the entry area based on resets or interrupts. However, the internal RAM is excluded in a reset.

**Figure 3-2. μPD784225 Memory Map**




**Notes 1.** Access in the external memory expansion mode.

**2.** The 4,608 bytes in this area can be used as the internal ROM only when the LOCATION 0FH instruction is executed.

**3.** LOCATION 0H instruction execution: 126,464 bytes; LOCATION 0FH instruction execution: 131,072 bytes.

**4.** This is the base area and the entry area based on resets or interrupts. However, the internal RAM is excluded in a reset.

## 3.2 Internal ROM Area

The following products in the μPD784225 Subseries have on-chip ROM that can store the programs and table data.

If the internal ROM area and internal data area overlap when the LOCATION 0H instruction is executed, the internal data area becomes the access target. The overlapped internal ROM area cannot be accessed.

| Part Number | Internal ROM | Access Space | |
|---|---|---|---|
| | | LOCATION 0H Instruction | LOCATION 0FH Instruction |
| μPD784224 | 96 KB × 8 bits | 00000H to 0F0FFH<br>10000H to 17FFFH | 00000H to 17FFFH |
| μPD784225<br>μPD78F4225 | 128 KB × 8 bits | 00000H to 0EDFFH<br>10000H to 1FFFFH | 00000H to 1FFFFH |

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

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

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

## 3.3 Base Area

The area from 0 to FFFFH becomes the base area. The base area is used for the following.

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

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

When the LOCATION 0H instruction is 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 interrupt requests and $\overline{\text{RESET}}$ input are stored in the vector table area. If context switching is used by each interrupt, the register bank number of the switch destination is also stored in this area.

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

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

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

| Interrupt Source | Vector Table Address | Interrupt Source | Vector Table Address |
|---|---|---|---|
| BRK instruction | 003EH | INTST1 | 001CH |
| Operand error | 003CH | INTSER2 | 001EH |
| NMI | 0002H | INSR2 | 0020H |
| INTWDT (non-maskable) | 0004H | INTCSI2 | |
| INTWDT (maskable) | 0006H | INTST2 | 0022H |
| INTP0 | 0008H | INTTM3 | 0024H |
| INTP1 | 000AH | INTTM00 | 0026H |
| INTP2 | 000CH | INTTM01 | 0028H |
| INTP3 | 000EH | INTTM1 | 002AH |
| INTP4 | 0010H | INTTM2 | 002CH |
| INTP5 | 0012H | INTAD | 002EH |
| INTIIC0[Note] | 0016H | INTTM5 | 0030H |
| INTCSI0 | | INTTM6 | 0032H |
| INTSER1 | 0018H | INTWT | 0038H |
| INTSR1 | 001AH | | |
| INTCSI1 | | | |

**Note** Only in the $\mu$PD784225Y Subseries

### 3.3.2 CALLT instruction table area

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

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

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

### 3.3.3 CALLF instruction entry area

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

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

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

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

## 3.4 Internal Data Area

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

The final address in the internal data area can be set to 0FFFFH (when executing the LOCATION 0H instruction) or FFFFFH (when executing the LOCATION 0FH instruction) by the LOCATION instruction. The address selection of the internal data area by this LOCATION instruction 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 0H 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 μPD784225 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)

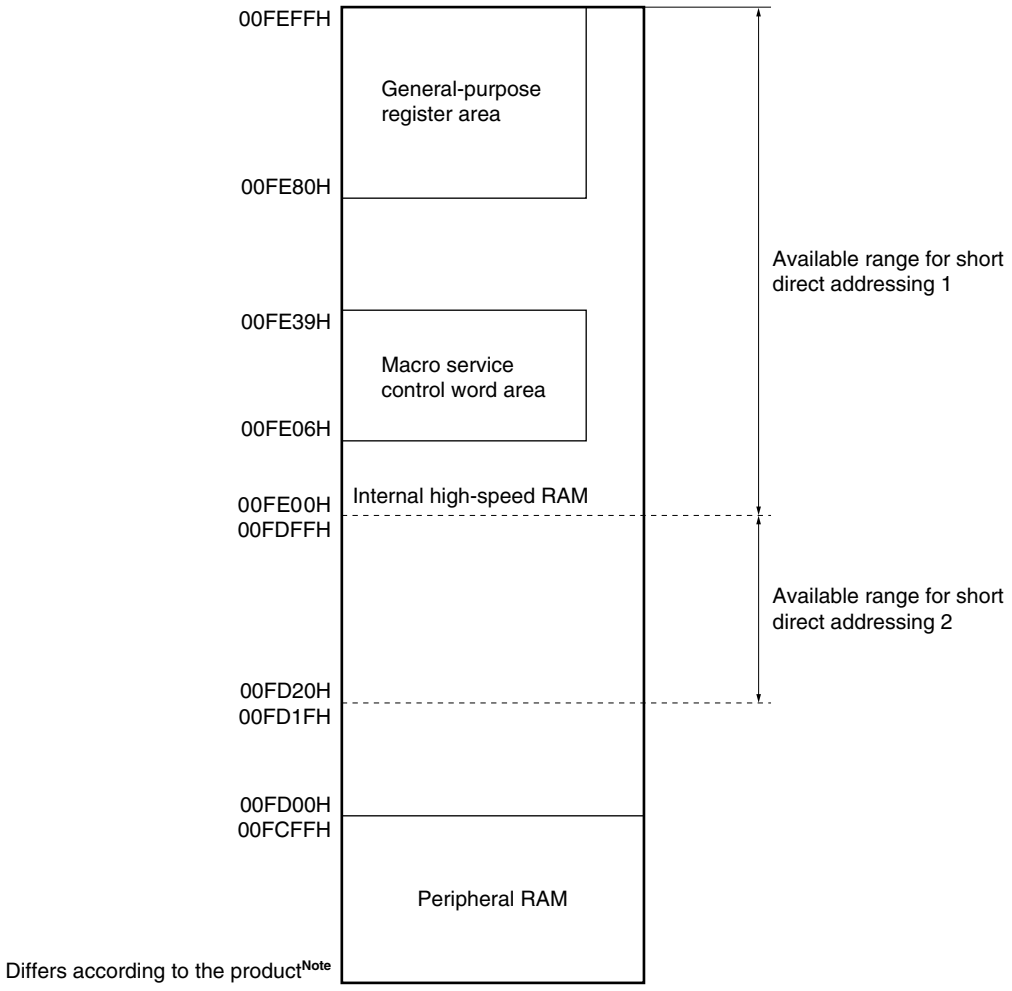
**Table 3-2. Internal RAM Area List**

| Internal RAM / Product Name | Internal RAM Area | | |
|---|---|---|---|
| | | Peripheral RAM: PRAM | Internal High-Speed RAM: IRAM |
| μPD784224 | 3,584 bytes (0F100H to 0FEFFH) | 3,072 bytes (0F100H to 0FCFFH) | 512 bytes (0FD00H to 0FEFFH) |
| μPD784225<br>μPD78F4225 | 4,352 bytes (0EE00H to 0FEFFH) | 3,840 bytes (0EE00H to 0FCFFH) | |

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

Figure 3-3 is the internal RAM memory map.

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




**Note** μPD784224: 00F100H
μPD784225, 78F4225: 00EE00H

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

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

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

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

The following areas are reserved in IRAM.

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

**Remark** The addresses here are the addresses when the LOCATION 0H instruction is executed. When the LOCATION 0FH instruction is executed, 0F0000H is added to the values here.

**(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 registers (SFRs) of the on-chip peripheral hardware are mapped to the area from 0FF00H to 0FFFFH (refer to **Figures 3-1** and **3-2**).

The area from 0FFD0H to 0FFDFH is mapped as the external SFR area. Peripheral I/O externally connected in the external memory expansion mode (set by the memory expansion mode register (MM)) can be accessed.

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

**Remark** The addresses here are the addresses only when the LOCATION 0H instruction is executed. If the LOCATION 0FH instruction is executed, 0F0000H is added to the values here.

### 3.4.3 External SFR area

In the products of the $\mu$PD784225 Subseries, the 16-byte area of the 0FFD0H to 0FFDFH area (during LOCATION 0H instruction execution, or 0FFFD0H to 0FFFDFH area during LOCATION 0FH instruction execution) in the SFR area is mapped as the external SFR area. In the external memory expansion mode, the address bus and address/data bus are used and the externally attached peripheral I/O can be accessed.

Since the external SFR area can be accessed by SFR addressing, the features are that peripheral I/O operations can be simplified; the object size can be reduced; and macro service can be used.

The bus operation when accessing an external SFR area is the same as a normal memory access.

## 3.5 External Memory Space

The external memory space is the memory space that can be accessed based on the setting of the memory expansion mode register (MM). The program and table data can be stored and peripheral I/O devices can be assigned.

## 3.6 μPD78F4225 Memory Mapping

The μPD78F4225 has a 128 KB flash memory and 4,352-byte internal RAM.

The μPD78F4225 has a function (memory size switching function) so that a part of the internal memory is not used by the software.

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

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

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

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

**Figure 3-4. Format of Internal Memory Size Switching Register (IMS)**

Address: 0FFFCH After reset: FFH W

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

| ROM1 | ROM0 | Internal ROM capacity selection |
|---|---|---|
| 0 | 0 | 48 KB |
| 0 | 1 | 64 KB |
| 1 | 0 | 96 KB |
| 1 | 1 | 128 KB |

| RAM1 | RAM0 | Internal RAM capacity selection |
|---|---|---|
| 0 | 0 | 1,536 bytes |
| 0 | 1 | 2,304 bytes |
| 1 | 0 | 3,072 bytes |
| 1 | 1 | 3,840 bytes |

**Caution Mask ROM versions (μPD784224, 784225) do not have an IMS register. Even if a write instruction is executed to IMS in mask ROM versions, the instruction will be invalid.**

Table 3-3 shows the IMS settings that produce the same memory map as the mask ROM versions.

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

| Target Mask ROM Versions | IMS Settings |
|---|---|
| μPD784224 | EEH |
| μPD784225 | FFH |

## 3.7 Control Registers

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

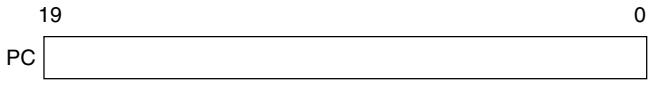
### 3.7.1 Program counter (PC)

This is a 20-bit binary counter that saves address information about the program to be executed next (see **Figure 3-5**).

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

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

**Figure 3-5. Format of Program Counter (PC)**




### 3.7.2 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 higher 8 bit (PSWH) and lower 8-bit (PSWL) units. In addition, bit manipulation instructions can be used to manipulate each flag.

The contents of the PSW are automatically saved on the stack when a vectored interrupt request is acknowledged 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 PR3, 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 3-6. The contents of bits indicated by “-” are undefined when read.

**Figure 3-6. Format of Program Status Word (PSW)**

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

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

Each flag is described below.

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

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

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

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

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

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

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

- **Parity flag action**

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

- **Overflow flag action**

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

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

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

```
   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 acknowledgement.

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

If IE is 1, the interrupt enable state is entered. Enabling the acknowledgment 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 by interrupt acknowledgement.

**(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 executed.

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

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

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

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

This flag indicates that the operation result is 0.

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

**(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 a conditional branch instruction.

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

This is a 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 for program control.

### 3.7.3 Using the RSS bit

Basically, always use the RSS bit fixed to 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 to 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 the RSS bit is used carelessly. Consequently, always set the RSS bit to 0. Use 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 by instructions where the A, X, B, C, and AX registers are directly described in the operand column of the operation list (see **28.2**)

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

- Registers that are used in addressing by instructions that use the A, B, and C registers with 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 the NEC assembler RA78K4, instruction code is generated for any register described by name or for registers set by an RSS quasi directive in the assembler.
When the RSS bit is set or reset, always specify an RSS quasi directive immediately before (or immediately after) that instruction (see the following examples).

**<Program examples>**

- When RSS = 0

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

- When RSS = 1

```
RSS 1          ; RSS quasi directive
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, the 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 quasi directive in the 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 for r, r’, rp, and rp’ in the operand column, an instruction code that conforms to the specification is output. (Instruction code that directly describes A or AX in the operand column is not output.)

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

**(3) Cautions on use**

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

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

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

### 3.7.4 Stack pointer (SP)

This is a 24-bit register that saves the starting address of the stack (LIFO: 00000H to FFFFFFH) (refer to **Figure 3-7**). The stack is used for addressing during subroutine processing or interrupt servicing. Always set the higher four bits to 0.

The contents of the SP are decremented before writing to the stack area and incremented after reading from the stack (refer to **Figures 3-8** and **3-9**).

The 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 acknowledging a subroutine call or interrupt).

**Example** Initializing SP

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

**Figure 3-7. Format of Stack Pointer (SP)**

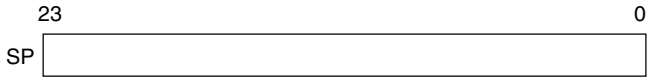


**Figure 3-8. Data Saved to Stack**

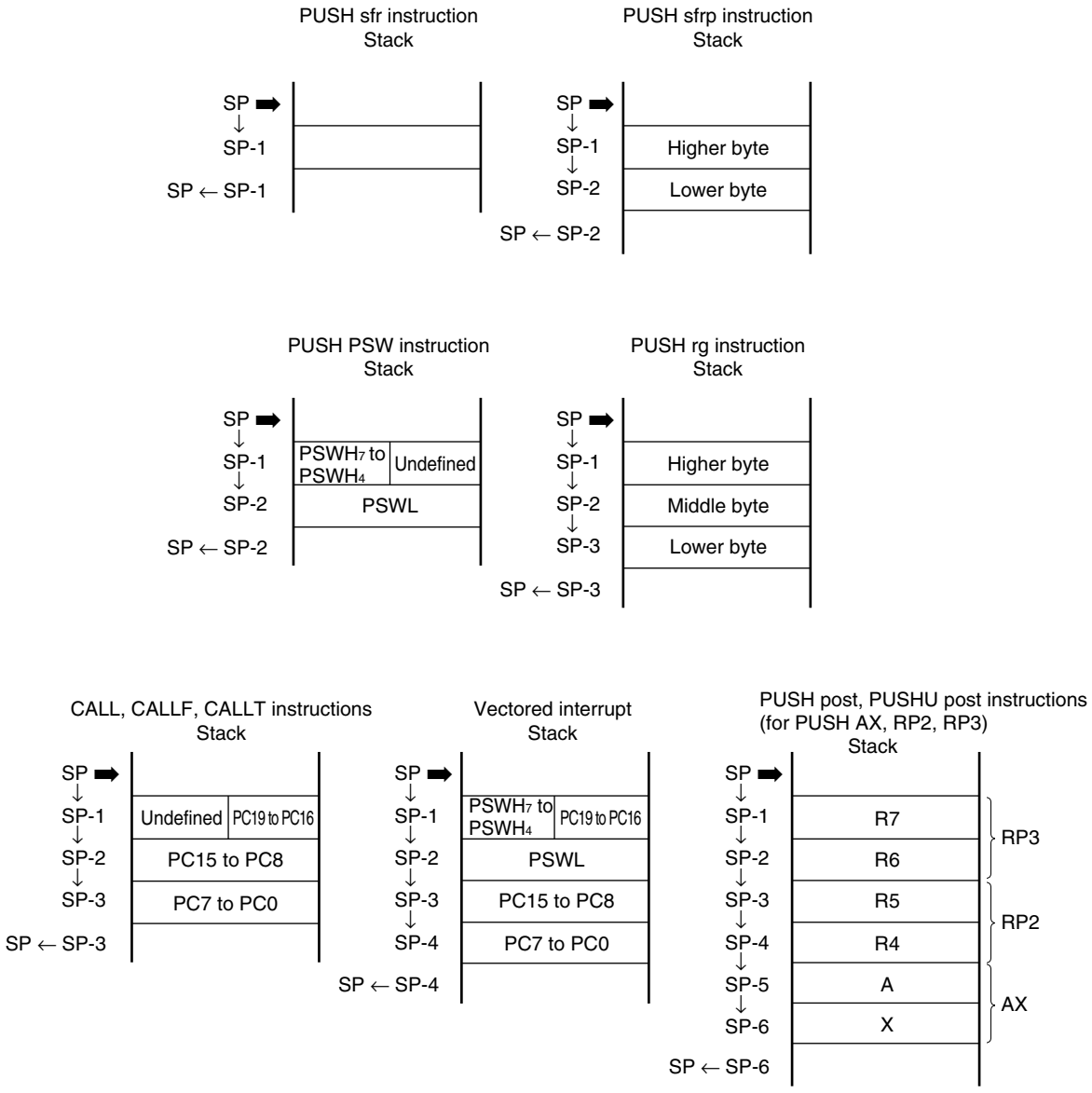
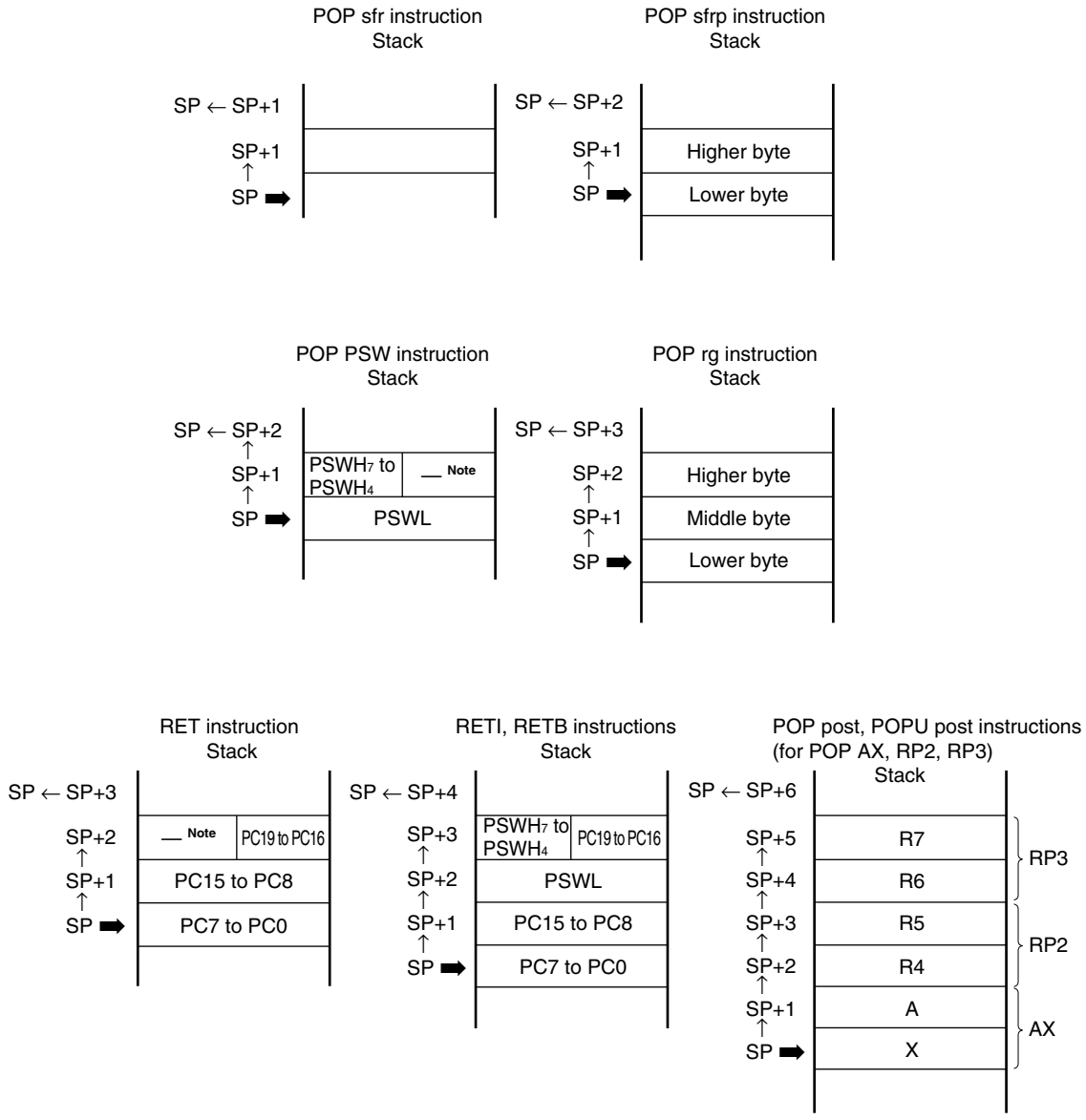


**Figure 3-9. Data Restored from Stack**




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

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

**2. The stack pointer (SP) becomes undefined when $\overline{\text{RESET}}$ is input. In addition, even when the SP is in an undefined state, non-maskable interrupts can be acknowledged. Therefore, when the SP is in an undefined state immediately after the reset is cleared and a request for a non-maskable interrupt is generated, unexpected operations 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
```

## 3.8 General-Purpose Registers

### 3.8.1 Structure

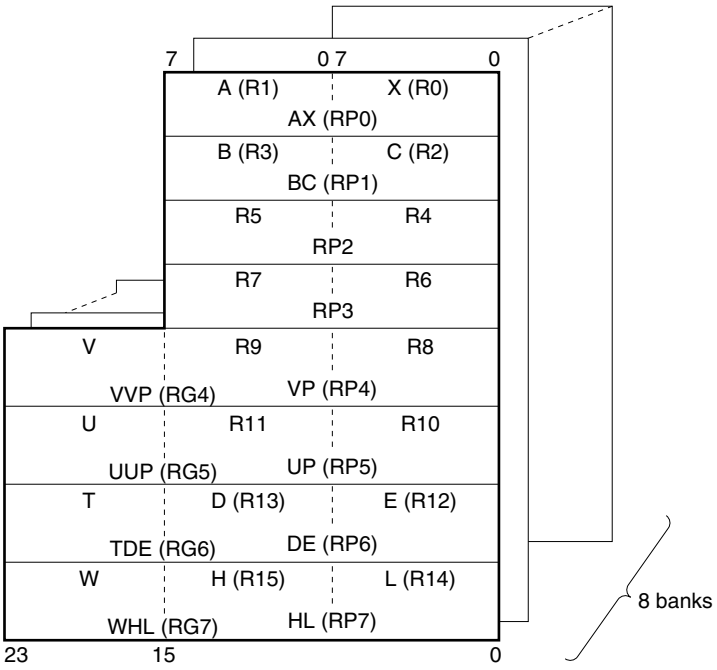
There are sixteen 8-bit general-purpose registers. In addition, two 8-bit general-purpose registers can be combined and used as a 16-bit general-purpose register. Furthermore, four of the 16-bit general-purpose registers can be combined with an 8-bit register for address expansion and used as a 24-bit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

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

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

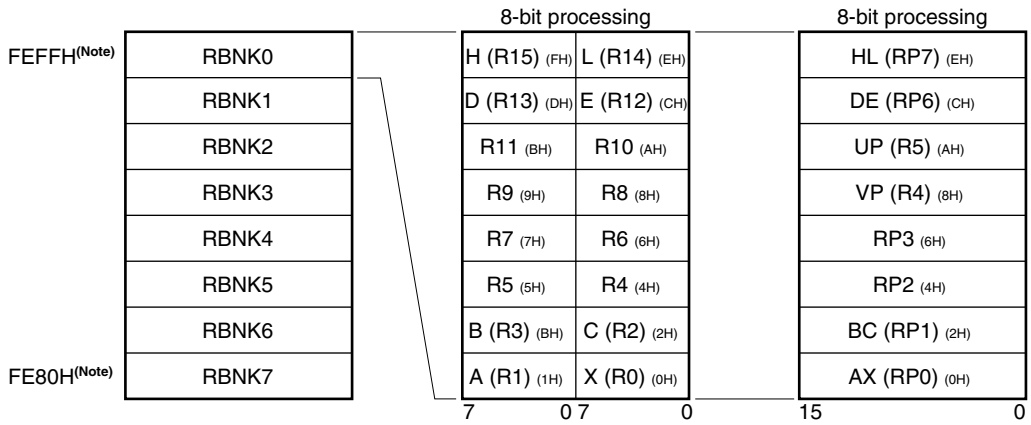
$\overline{\text{RESET}}$ input selects register bank 0. In addition, the register banks that are used in an executing program can be verified by reading the register bank selection flags (RBS0, RBS1, RBS2) in the PSW.

**Figure 3-10. Format of General-Purpose Register**




**Remark** The names in parentheses are the absolute names.

**Figure 3-11. General-Purpose Register Addresses**




**Note** These are the addresses when the LOCATION 0H 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 servicing program are updated, the PSW is automatically saved on the stack when an interrupt is acknowledged 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
        .
        .
        .
```

### 3.8.2 Functions

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

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

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

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

Each register has the unique functions shown below.

**A (R1):**

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

**X (R0):**

- This register can store bit data.

**AX (RP0):**

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

**AXDE:**

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

**B (R3):**

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

**C (R2):**

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

**RP2:**

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

**RP3:**

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

**VVP (RG4):**

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

**UUP (RG5):**

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

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

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

**TDE (RG6):**

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

**WHL (RG7):**

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

In addition to a 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 an 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.

## 3.9 Special Function Registers (SFRs)

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

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

**Caution In this area, do not access an address that is not allocated by an SFR. If erroneously accessed, the μPD784225 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 below.

- Symbol ..... This symbol indicates the on-chip SFR. In the NEC assembler RA78K4, this is a reserved word. In the C compiler CC78K4, it can be used as an sfr variable by a #pragma sfr directive.
- R/W ..... Indicates whether the corresponding SFR can be read or written.
  R/W: Can read/write
  R: Read only
  W: Write only
- Bit Manipulation Unit ..... When the corresponding SFR is manipulated, the appropriate bit manipulation unit is indicated. An SFR that can be manipulated in 16 bits can be described in the sfrp operand. If specified by an address, an even address is described.
  An SFR that can be manipulated in 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 (1/4)**

| Address[Note 1] | Special Function Register (SFR) Name | 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 | R | √ | √ | — | |
| 0FF02H | Port 2 | P2 | R/W | √ | √ | — | |
| 0FF03H | Port 3 | P3 | | √ | √ | — | |
| 0FF04H | Port 4 | P4 | | √ | √ | — | |
| 0FF05H | Port 5 | P5 | | √ | √ | — | |
| 0FF06H | Port 6 | P6 | | √ | √ | — | |
| 0FF07H | Port 7 | P7 | | √ | √ | — | |
| 0FF0CH | Port 12 | P12 | | √ | √ | — | |
| 0FF0DH | Port 13 | P13 | | √ | √ | — | |
| 0FF10H | 16-bit timer counter 0 | TM0 | R | — | — | √ | 0000H |
| 0FF11H | | | | | | | |
| 0FF12H | Capture/compare register 00 (16-bit timer/event counter) | CR00 | R/W | — | — | √ | |
| 0FF13H | | | | | | | |
| 0FF14H | Capture/compare register 01 (16-bit timer/event counter) | CR01 | | — | — | √ | |
| 0FF15H | | | | | | | |
| 0FF16H | Capture/compare control register 0 | CRC0 | | √ | √ | — | 00H |
| 0FF18H | 16-bit timer mode control register 0 | TMC0 | | √ | √ | — | |
| 0FF1AH | 16-bit timer output control register 0 | TOC0 | | √ | √ | — | |
| 0FF1CH | Prescaler mode register 0 | PRM0 | | √ | √ | — | |
| 0FF20H | Port 0 mode register | PM0 | | √ | √ | — | FFH |
| 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 | | √ | √ | — | |
| 0FF27H | Port 7 mode register | PM7 | | √ | √ | — | |
| 0FF2CH | Port 12 mode register | PM12 | | √ | √ | — | |
| 0FF2DH | Port 13 mode register | PM13 | | √ | √ | — | |
| 0FF30H | Pull-up resistor option register 0 | PU0 | | √ | √ | — | 00H |
| 0FF32H | Pull-up resistor option register 2 | PU2 | | √ | √ | — | |
| 0FF33H | Pull-up resistor option register 3 | PU3 | | √ | √ | — | |
| 0FF37H | Pull-up resistor option register 7 | PU7 | | √ | √ | — | |

**Notes 1.** These values are when the LOCATION 0H instruction is executed. When the LOCATION 0FH instruction is executed, F0000H is added to these values.

**2.** Since each port is initialized in the input mode by a reset, 00H is not actually read out. The output latch is initialized to 0.

**Table 3-6. Special Function Register (SFR) List (2/4)**

| Address[Note 1] | Special Function Register (SFR) Name | Symbol | | R/W | Bit Manipulation Unit | | | After Reset |
|---|---|---|---|---|---|---|---|---|
| | | | | | 1 Bit | 8 Bits | 16 Bits | |
| 0FF3CH | Pull-up resistor option register 12 | PU12 | | R/W | √ | √ | — | 00H |
| 0FF40H | Clock output control register | CKS | | | √ | √ | — | |
| 0FF42H | Port function control register 2[Note 2] | PF2 | | | √ | √ | — | |
| 0FF4EH | Pull-up resistor option register 0 | PUO | | | √ | √ | — | |
| 0FF50H | 8-bit timer counter 1 | TM1 | TW1W | R | — | √ | √ | 0000H |
| 0FF51H | 8-bit timer counter 2 | TM2 | | | — | √ | | |
| 0FF52H | Compare register 10 (8-bit timer/event counter 1) | CR10 | CR1W | R/W | — | √ | √ | |
| 0FF53H | Compare register 20 (8-bit timer/event counter 2) | CR20 | | | — | √ | | |
| 0FF54H | 8-bit timer mode control register 1 | TMC1 | TMC1W | | √ | √ | √ | |
| 0FF55H | 8-bit timer mode control register 2 | TMC2 | | | √ | √ | | |
| 0FF56H | Prescaler mode register 1 | PRM1 | PRM1W | | √ | √ | √ | |
| 0FF57H | Prescaler mode register 2 | PRM2 | | | √ | √ | | |
| 0FF60H | 8-bit timer counter 5 | TM5 | TM5W | R | — | √ | √ | |
| 0FF61H | 8-bit timer counter 6 | TM6 | | | — | √ | | |
| 0FF64H | Compare register 50 (8-bit timer 5) | CR50 | CR5W | R/W | — | √ | √ | |
| 0FF65H | Compare register 60 (8-bit timer 6) | CR60 | | | — | √ | | |
| 0FF68H | 8-bit timer mode control register 5 | TMC5 | TMC5W | | √ | √ | √ | |
| 0FF69H | 8-bit timer mode control register 6 | TMC6 | | | √ | √ | | |
| 0FF6CH | Prescaler mode register 5 | PRM5 | PRM5W | | √ | √ | √ | |
| 0FF6DH | Prescaler mode register 6 | PRM6 | | | √ | √ | | |
| 0FF70H | Asynchronous serial interface mode register 1 | ASIM1 | | | √ | √ | — | 00H |
| 0FF71H | Asynchronous serial interface mode register 2 | ASIM2 | | | √ | √ | — | |
| 0FF72H | Asynchronous serial interface status register 1 | ASIS1 | | R | √ | √ | — | |
| 0FF73H | Asynchronous serial interface status register 2 | ASIS2 | | | √ | √ | — | |
| 0FF74H | Transmission shift register 1 | TXS1 | | W | — | √ | — | FFH |
| | Reception buffer register 1 | RXB1 | | R | — | √ | — | |
| 0FF75H | Transmission shift register 2 | TXS2 | | W | — | √ | — | |
| | Reception buffer register 2 | RXB2 | | R | — | √ | — | |
| 0FF76H | Baud rate generator control register 1 | BRGC1 | | R/W | √ | √ | — | 00H |
| 0FF77H | Baud rate generator control register 2 | BRGC2 | | | √ | √ | — | |
| 0FF7AH | Oscillation mode selection register | CC | | | √ | √ | — | |
| 0FF80H | A/D converter mode register | ADM | | | √ | √ | — | |
| 0FF81H | A/D converter input selection register | ADIS | | | √ | √ | — | |
| 0FF83H | A/D conversion result register | ADCR | | R | — | √ | — | Undefined |

**Notes 1.** These are the values when the LOCATION 0H instruction is executed. When the LOCATION 0FH instruction is executed, F0000H is added to this value.

**2.** Only in the μPD784225Y Subseries

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

| Address[Note 1] | Special Function Register (SFR) Name | Symbol | | R/W | Bit Manipulation Unit | | | After Reset |
|---|---|---|---|---|---|---|---|---|
| | | | | | 1 Bit | 8 Bits | 16 Bits | |
| 0FF84H | D/A conversion value setting register 0 | DACS0 | | R/W | √ | √ | — | 00H |
| 0FF85H | D/A conversion value setting register 1 | DACS1 | | | √ | √ | — | |
| 0FF86H | D/A converter mode register 0 | DAM0 | | | √ | √ | — | |
| 0FF87H | D/A converter mode register 1 | DAM1 | | | √ | √ | — | |
| 0FF88H | ROM correction control register | CORC | | | √ | √ | — | |
| 0FF89H | ROM correction address register H | CORAH | | | — | √ | — | |
| 0FF8AH | ROM correction address register L | CORAL | | | — | — | √ | 0000H |
| 0FF8BH | | | | | | | | |
| 0FF8DH | External access status enable register | EXAE | | | √ | √ | — | 00H |
| 0FF90H | Serial operation mode register 0 | CSIM0 | | | √ | √ | — | |
| 0FF91H | Serial operation mode register 1 | CSIM1 | | | √ | √ | — | |
| 0FF92H | Serial operation mode register 2 | CSIM2 | | | √ | √ | — | |
| 0FF94H | Serial I/O shift register 0 | SIO0 | | | — | √ | — | |
| 0FF95H | Serial I/O shift register 1 | SIO1 | | | — | √ | — | |
| 0FF96H | Serial I/O shift register 2 | SIO2 | | | — | √ | — | |
| 0FF98H | Real-time output buffer register L | RTBL | | | — | √ | — | |
| 0FF99H | Real-time output buffer register H | RTBH | | | — | √ | — | |
| 0FF9AH | Real-time output port mode register | RTPM | | | √ | √ | — | |
| 0FF9BH | Real-time output port control register | RTPC | | | √ | √ | — | |
| 0FF9CH | Watch timer mode control register | WTM | | | √ | √ | — | |
| 0FFA0H | External interrupt rising edge enable register 0 | EGP0 | | | √ | √ | — | |
| 0FFA2H | External interrupt falling edge enable register 0 | EGN0 | | | √ | √ | — | |
| 0FFA8H | In-service priority register | ISPR | | R | √ | √ | — | 00H |
| 0FFA9H | Interrupt selection control register | SNMI | | R/W | √ | √ | — | |
| 0FFAAH | Interrupt mode control register | IMC | | | √ | √ | — | 80H |
| 0FFACH | Interrupt mask flag register 0L | MK0L | MK0 | | √ | √ | √ | FFFFH |
| 0FFADH | Interrupt mask flag register 0H | MK0H | | | √ | √ | | |
| 0FFAEH | Interrupt mask flag register 1L | MK1L | MK1 | | √ | √ | √ | |
| 0FFAFH | Interrupt mask flag register 1H | MK1H | | | √ | √ | | |
| 0FFB0H | $I^2C$ bus control register 0[Note 2] | IICC0 | | | √ | √ | — | 00H |
| 0FFB2H | Serial clock prescaler mode register 0[Note 2] | SPRM0 | | | √ | √ | — | |
| 0FFB4H | Slave address register 0[Note 2] | SVA0 | | | — | √ | — | |
| 0FFB6H | $I^2C$ bus status register 0[Note 2] | IICS0 | | R | √ | √ | — | |
| 0FFB8H | Serial shift register 0[Note 2] | IIC0 | | R/W | √ | √ | — | |

**Notes 1.** These are the values when the LOCATION 0H instruction is executed. When the LOCATION 0FH instruction is executed, F0000H is added to this value.

**2.** Only in the μPD784225Y Subseries

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

| Address[Note 1] | Special Function Register (SFR) Name | Symbol | R/W | Bit Manipulation Unit | | | After Reset |
|---|---|---|---|---|---|---|---|
| | | | | 1 Bit | 8 Bits | 16 Bits | |
| 0FFC0H | Standby control register | STBC | R/W | — | √ | — | 30H |
| 0FFC2H | Watchdog timer mode register | WDM | | — | √ | — | 00H |
| 0FFC4H | Memory expansion mode register | MM | | √ | √ | — | 20H |
| 0FFC7H | Programmable wait control register 1 | PWC1 | | √ | √ | — | AAH |
| 0FFC8H | Programmable wait control register 2 | PWC2 | W | — | — | √ | AAAAH |
| 0FFC9H | | | | | | | |
| 0FFCEH | Clock status register | PCS | R | √ | √ | — | 32H |
| 0FFCFH | Oscillation stabilization time specification register | OSTS | R/W | √ | √ | — | 00H |
| 0FFD0H to 0FFDFH | External SFR area | — | | √ | √ | — | — |
| 0FFE0H | Interrupt control register (INTWDTM) | WDTIC | | √ | √ | — | 43H |
| 0FFE1H | Interrupt control register (INTP0) | PIC0 | | √ | √ | — | |
| 0FFE2H | Interrupt control register (INTP1) | PIC1 | | √ | √ | — | |
| 0FFE3H | Interrupt control register (INTP2) | PIC2 | | √ | √ | — | |
| 0FFE4H | Interrupt control register (INTP3) | PIC3 | | √ | √ | — | |
| 0FFE5H | Interrupt control register (INTP4) | PIC4 | | √ | √ | — | |
| 0FFE6H | Interrupt control register (INTP5) | PIC5 | | √ | √ | — | |
| 0FFE8H | Interrupt control register (INTIIC0[Note 2]/INTCSI0) | CSIIC0 | | √ | √ | — | |
| 0FFE9H | Interrupt control register (INTSER1) | SERIC1 | | √ | √ | — | |
| 0FFEAH | Interrupt control register (INTSR1/INTCSI1) | SRIC1 | | √ | √ | — | |
| 0FFEBH | Interrupt control register (INTST1) | STIC1 | | √ | √ | — | |
| 0FFECH | Interrupt control register (INTSER2) | SERIC2 | | √ | √ | — | |
| 0FFEDH | Interrupt control register (INTSR2/INTCSI2) | SRIC2 | | √ | √ | — | |
| 0FFEEH | Interrupt control register (INTST2) | STIC2 | | √ | √ | — | |
| 0FFEFH | Interrupt control register (INTTM3) | TMIC3 | | √ | √ | — | |
| 0FFF0H | Interrupt control register (INTTM00) | TMIC00 | | √ | √ | — | |
| 0FFF1H | Interrupt control register (INTTM01) | TMIC01 | | √ | √ | — | |
| 0FFF2H | Interrupt control register (INTTM1) | TMIC1 | | √ | √ | — | |
| 0FFF3H | Interrupt control register (INTTM2) | TMIC2 | | √ | √ | — | |
| 0FFF4H | Interrupt control register (INTAD) | ADIC | | √ | √ | — | |
| 0FFF5H | Interrupt control register (INTTM5) | TMIC5 | | √ | √ | — | |
| 0FFF6H | Interrupt control register (INTTM6) | TMIC6 | | √ | √ | — | |
| 0FFF9H | Interrupt control register (INTWT) | WTIC | | √ | √ | — | |
| 0FFFCH | Internal memory size switching register[Note 3] | IMS | W | — | √ | — | FFH |

**Notes 1.** These values are when the LOCATION 0H instruction is executed. When the LOCATION 0FH instruction is executed, F0000H is added to these values.

**2.** Only in the μPD784225Y Subseries

**3.** Only in the μPD78F4225 and 78F4225Y

## 3.10 Cautions

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

(2) Special function registers (SFRs)
Do not access an address that is allocated to an SFR in the area from 0FF00H to 0FFFFH[Note]. If mistakenly accessed, the μPD784225 enters the deadlock state. The deadlock state is released only by reset input.

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

(3) Stack pointer (SP) operation
Although the entire 1 MB space can be accessed by stack addressing, the stack cannot be guaranteed in the SFR area and the internal ROM 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 acknowledged. Therefore, the SP enters an undefined state immediately after clearing the reset. When a non-maskable interrupt request is generated, unexpected operations sometimes occur. To minimize these dangers, always describe the following in the program immediately after clearing a reset.

```
RSTVCT    CSEG AT 0
          DW   RSTSTRT

           to

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

# CHAPTER 4 CLOCK GENERATOR

## 4.1 Functions

The clock generator generates the clock to be supplied to the CPU and peripheral hardware. The following two types of system clock oscillators are available.

**(1) Main system clock oscillator**

This circuit oscillates at frequencies of 2 to 12.5 MHz. Oscillation can be stopped by setting the standby control register (STBC) to STOP mode (bit 1 (STP) = 1, bit 0 (HLT) = 0) or by stopping the main system clock (bit 2 of STBC (MCK) = 1) after switching to the subsystem clock.

**(2) Subsystem clock oscillator**

This circuit oscillates at the frequency of 32.768 kHz. Oscillation cannot be stopped. If the subsystem clock oscillator is not used, not using the internal feedback resistor can be set by STBC. This enables the power consumption to be decreased in the STOP mode.

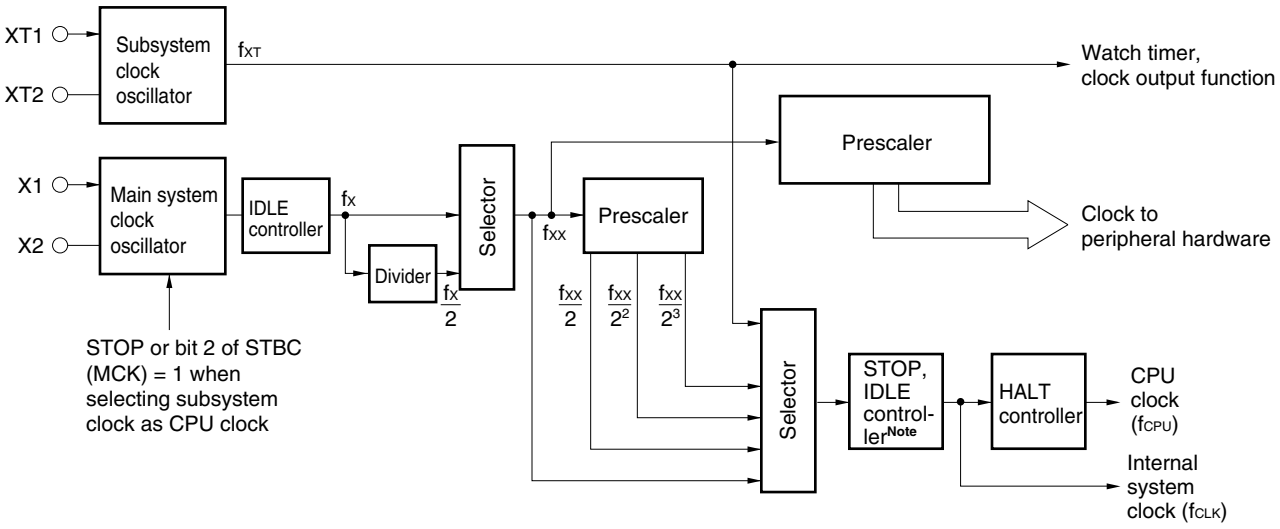
## 4.2 Configuration

The clock generator includes the following hardware.

**Table 4-1. Clock Generator Configuration**

| Item | Configuration |
|---|---|
| Control registers | Standby control register (STBC)<br>Oscillation mode selection register (CC)<br>Clock status register (PCS)<br>Oscillation stabilization time specification register (OSTS) |
| Oscillators | Main system clock oscillator<br>Subsystem clock oscillator |

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




**Note** This controller secures the oscillation stabilization time after releasing STOP mode.

## 4.3 Control Registers

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

This register is used to set the standby mode and select the internal system clock. For details of the standby mode, refer to **CHAPTER 24 STANDBY FUNCTION**.

A write operation can be performed only using dedicated instructions to avoid entering 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 opcode 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 the error. Therefore, the address that caused the 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, a write operation to STBC performed and an interrupt such as an operand error interrupt is not generated. STBC can be read out any time by a data transfer instruction.

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

Figure 4-2 shows the format of STBC.

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

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

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| STBC | SBK | CK2 | CK1 | CK0 | 0 | MCK | STP | HLT |

| SBK | Subsystem clock oscillation control |
|---|---|
| 0 | Use oscillator (internal feedback resistor is used) |
| 1 | Stop oscillator (internal feedback resistor is not used) |

| CK2 | CK1 | CK0 | CPU clock selection |
|---|---|---|---|
| 0 | 0 | 0 | $f_{XX}$ |
| 0 | 0 | 1 | $f_{XX}/2$ |
| 0 | 1 | 0 | $f_{XX}/4$ |
| 0 | 1 | 1 | $f_{XX}/8$ |
| 1 | 1 | 1 | $f_{XT}$ (recommended) |
| 1 | — | — | $f_{XT}$ |

| MCK | Main system clock oscillation control |
|---|---|
| 0 | Use oscillator (internal feedback resistor is used) |
| 1 | Stop oscillator (internal feedback resistor is not used) |

| STP | HLT | Operation specification flag |
|---|---|---|
| 0 | 0 | Normal operation mode |
| 0 | 1 | HALT mode (automatically cleared upon release of HALT mode) |
| 1 | 0 | STOP mode (automatically cleared upon release of STOP mode) |
| 1 | 1 | IDLE mode (automatically cleared upon release of IDLE mode) |

**Cautions 1. When using the STOP mode during external clock input, make sure to set the EXTC bit of the oscillation stabilization time specification register (OSTS) to 1 before setting the STOP mode. If the STOP mode is used during external clock input when the EXTC bit of OSTS has been cleared (0), the μPD784225 may be damaged or its reliability may be impaired.**

**When setting the EXTC bit of OSTS to 1, a clock with the opposite phase of the clock input to the X1 pin must be input to the X2 pin.**

**2. Execute a NOP instruction three times after a standby instruction (after standby release). Otherwise if conflict occurs between standby instruction execution and an interrupt request, the standby instruction is not performed and the interrupt request is acknowledged after the execution of several instructions. The instructions executed before the interrupt request is acknowledged are instructions whose execution is started within 6 clocks following execution of the standby instruction.**

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

**3. When CK2 = 0, the oscillation of the main system clock does not stop even if MCK is set to 1 (refer to 4.5.1 Main system clock operations).**

**Remarks 1.** fxx: Main system clock frequency (fx or fx/2)
fx: Main system clock oscillation frequency
fxT: Subsystem clock oscillation frequency

**2.** ×: Don't care

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

This register specifies whether clock output from the main system clock oscillator with the same frequency as the external clock, or clock output that is half of the original frequency is used to operate the internal circuits. CC is set by a 1-bit or 8-bit memory manipulation instruction.

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

**Figure 4-3. Format of Oscillation Mode Selection Register (CC)**

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

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

| ENMP | Main system clock selection |
|---|---|
| 0 | Half of original oscillation frequency |
| 1 | Through-rate clock mode |

**Cautions 1. If the subsystem clock is selected via the standby control mode register (STBC), the ENMP bit specification becomes invalid.**

**2. The ENMP bit cannot be reset by software. This bit is reset by a system reset.**

### (3) Clock status register (PCS)

This register is a read-only 8-bit register that indicates the CPU clock operation status. By reading bit 2 and bits 4 to 7 of PCS, the relevant bit of the standby control register (STBC) can be read.

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

$\overline{\text{RESET}}$ input sets PCS to 32H.

**Figure 4-4. Format of Clock Status Register (PCS)**

Address: 0FFCEH After reset: 32H R

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PCS | SBK | CK2 | CK1 | CK0 | 0 | MCK | 1 | HLT |

| SBK | Feedback resistor status of subsystem clock |
|---|---|
| 0 | Internal feedback resistor is used. |
| 1 | Internal feedback resistor is not used. |

| CK2 | CK1 | CK0 | CPU clock operating frequency |
|---|---|---|---|
| 0 | 0 | 0 | $f_{XX}$ |
| 0 | 0 | 1 | $f_{XX}/2$ |
| 0 | 1 | 0 | $f_{XX}/4$ |
| 0 | 1 | 1 | $f_{XX}/8$ |
| 1 | 1 | 1 | $f_{XT}$ (recommended) |
| 1 | × | × | $f_{XT}$ |

| MCK | Oscillation status of main system clock |
|---|---|
| 0 | Use oscillator |
| 1 | Stop oscillator |

| CST | CPU clock status |
|---|---|
| 0 | Main system clock operation |
| 1 | Subsystem clock operation |

**Caution [Timing at which bit 0 (CST) changes]**

**The CPU clock does not switch from the main system clock to the subsystem clock immediately after the standby control register (STBC) is set, but switches after synchronization of both clocks (main and subsystem) has been detected. Consequently, CST changes after synchronization detection. This is the same as when switching from subsystem clock to main system clock.**

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

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

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

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

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

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

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

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

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

**Cautions**

1. **When a crystal/ceramic resonator is used, make sure to clear the EXTC bit to 0. If the EXTC bit is set to 1, oscillation stops.**
2. **When using the STOP mode during external clock input, make sure to set the EXTC bit to 1 before setting the STOP mode. If the STOP mode is used during external clock input when the EXTC bit of OSTS has been cleared, the $\mu$PD784225 may be damaged or its reliability may be impaired.**
3. **If the EXTC bit is set to 1 during external clock input, the opposite phase of the clock input to the X1 pin must be input to the X2 pin. If the EXTC bit is set to 1, the $\mu$PD784225 operates only with the clock input to the X2 pin.**

**Remark**

1. Figures in parentheses apply to operation at $f_{XX}$ = 12.5 MHz.
2. ×: Don't care

## 4.4 System Clock Oscillator

### 4.4.1 Main system clock oscillator

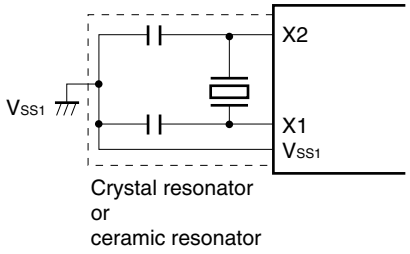
The main system clock oscillator oscillates with a crystal resonator or a ceramic resonator (standard: 12.5 MHz) connected to the X1 and X2 pins.

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

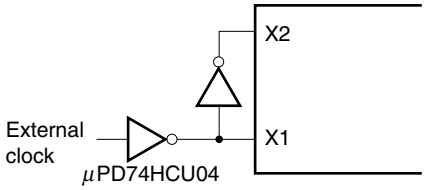
Figure 4-6 shows the external circuit of the main system clock oscillator.

**Figure 4-6. External Circuit of Main System Clock Oscillator**

**(a) Crystal or ceramic oscillation**




**(b) External clock**




**Caution When using a main system clock oscillator, wire as follows in the area enclosed by the broken lines in the figure above to avoid an adverse effect from wiring capacitance (also refer to 4.4.3 Examples of incorrect resonator connection).**

- **Keep the wiring length as short as possible.**
- **Do not cross the wiring with the other signal lines. Do not route the wiring near a signal line through which a high fluctuating current flows.**
- **Always make the ground point of the oscillator capacitor the same potential as $V_{SS1}$. Do not ground the capacitor to a ground pattern through which a high current flows.**
- **Do not fetch signals from the oscillator.**

### 4.4.2 Subsystem clock oscillator

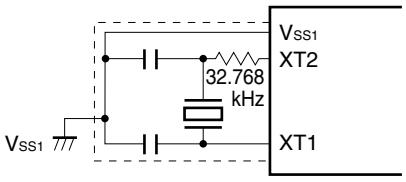
The subsystem clock oscillator oscillates with a crystal resonator (standard: 32.768 kHz) connected to the XT1 and XT2 pins.

External clocks can be input to the main system clock oscillator. In this case, input a clock signal to the XT1 pin and the reverse-phase clock signal to the XT2 pin.

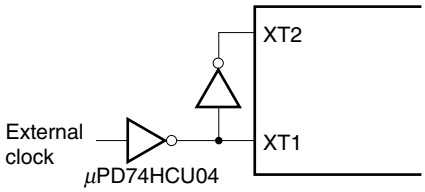
Figure 4-7 shows the external circuit of the subsystem clock oscillator.

**Figure 4-7. External Circuit of Subsystem Clock Oscillator**

**(a) Crystal oscillation**




**(b) External clock**




**Caution When using a subsystem clock oscillator, wire as follows in the area enclosed by the broken lines in the figure above to avoid an adverse effect from wiring capacitance (also refer to 4.4.3 Examples of incorrect resonator connection).**

- **Keep the wiring length as short as possible.**
- **Do not cross the wiring with the other signal lines. Do not route the wiring near a signal line through which a high fluctuating current flows.**
- **Always make the ground point of the oscillator capacitor the same potential as VSS1. Do not ground the capacitor to a ground pattern through which a high current flows.**
- **Do not fetch signals from the oscillator.**

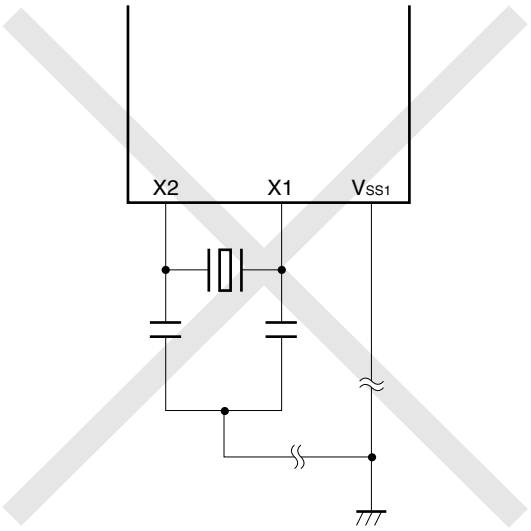
**Take special note of the fact that the subsystem clock oscillator is a circuit with low-level amplification so that current consumption is maintained at low levels.**

### 4.4.3 Examples of incorrect resonator connection

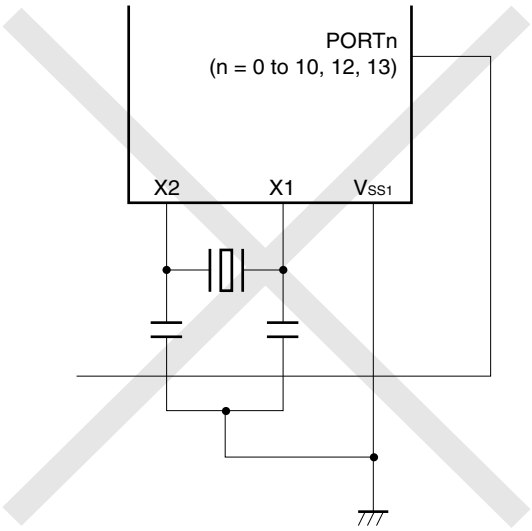
Figure 4-8 shows examples of resonators that are connected incorrectly.

**Figure 4-8. Examples of Incorrect Resonator Connection (1/2)**

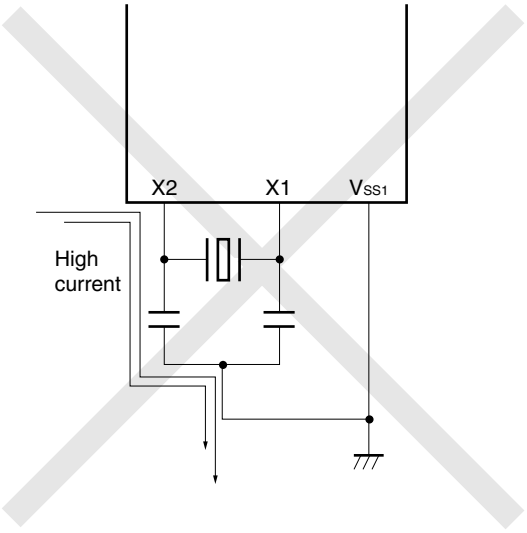
**(a) Wiring of connection circuits is too long**



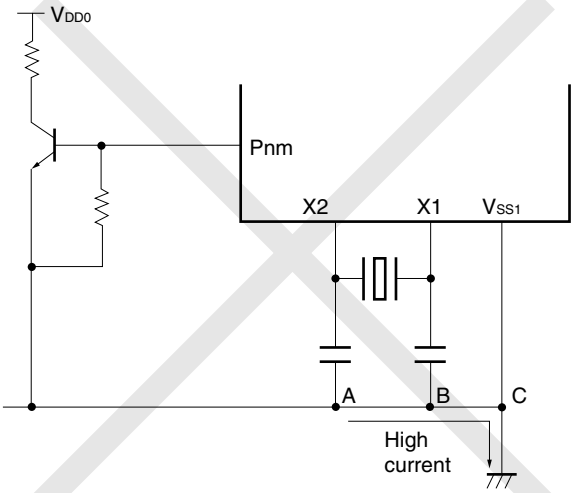

**(b) Signal lines intersect each other**




**(c) Fluctuating high current is too near a signal line**




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

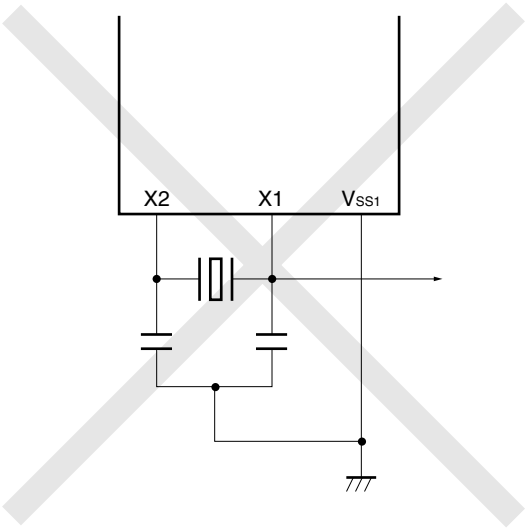



**Remark** When using a subsystem clock, replace X1 and X2 with XT1 and XT2, respectively. Also, insert resistors in series on the XT2 side.

**Caution When XT2 and X1 are wired in parallel, the crosstalk noise of X1 may synergize with XT2, resulting in malfunction. To prevent this from occurring, it is recommended not to wire XT2 and X1 in parallel.**

**Figure 4-8. Examples of Incorrect Resonator Connection (2/2)**

**(e) Signals are fetched**




**Remark** When using a subsystem clock, replace X1 and X2 with XT1 and XT2, respectively. Also, insert resistors in series on the XT2 side.

**Caution When XT2 and X1 are wired in parallel, the crosstalk noise of X1 may synergize with XT2, resulting in malfunction. To prevent this from occurring, it is recommended not to wire XT2 and X1 in parallel.**

### 4.4.4 Frequency divider

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

### 4.4.5 When subsystem clock is not used

If it is not necessary to use the subsystem clock for low power consumption operations and clock operations, connect the XT1 and XT2 pins as follows.

XT1: Connect to $V_{SS1}$
XT2: Leave open

In this state, however, some current may leak via the internal feedback resistor of the subsystem clock oscillator when the main system clock stops. To minimize leakage current, set bit 7 (SBK) of the standby control register (STBC) to 1. In this case also, connect the XT1 and XT2 pins as described above.

## 4.5 Clock Generator Operations

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

- Main system clock ($f_{XX}$)
- Subsystem clock ($f_{XT}$)
- CPU clock ($f_{CPU}$)
- Clock to peripheral hardware

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

(a) Upon generation of the $\overline{\text{RESET}}$ signal, the lowest speed mode of the main system clock (1,280 ns: @ 12.5 MHz operation) is selected (STBC = 30H, CC = 00H). Main system clock oscillation stops while a low level is being applied to the $\overline{\text{RESET}}$ pin.

(b) With the main system clock selected, one of six CPU clock types (80 ns, 160 ns, 320 ns, 640 ns, 1,280 ns: @ 12.5 MHz operation) can be selected by setting STBC and CC.

(c) With the main system clock selected, two standby modes, the STOP mode and the HALT mode, are available. To decrease current consumption in the STOP mode, the subsystem clock feedback resistor can be disconnected to stop the subsystem clock by using bit 7 (SBK) of STBC, when the system does not use the subsystem clock.

(d) STBC can be used to select the subsystem clock to operate the system with low current consumption (30.5 $\mu$s: @ 32.768 kHz operation).

(e) With the subsystem clock selected, main system clock oscillation can be stopped using STBC. The HALT mode can be used. However, the STOP mode cannot be used. (Subsystem clock oscillation cannot be stopped.)

(f) The main system clock is divided and supplied to the peripheral hardware. The subsystem clock is supplied to the 16-bit timer/event counter, the watch timer, and clock output functions only. Thus, the 16-bit timer/event counter (when watch timer output is selected for the count clock during operation with the subsystem clock), the watch function, and the clock output function can also be continued in the standby state. However, since all other peripheral hardware operates with the main system clock, the peripheral hardware (except external input clock operation) also stops if the main system clock is stopped.

### 4.5.1 Main system clock operations

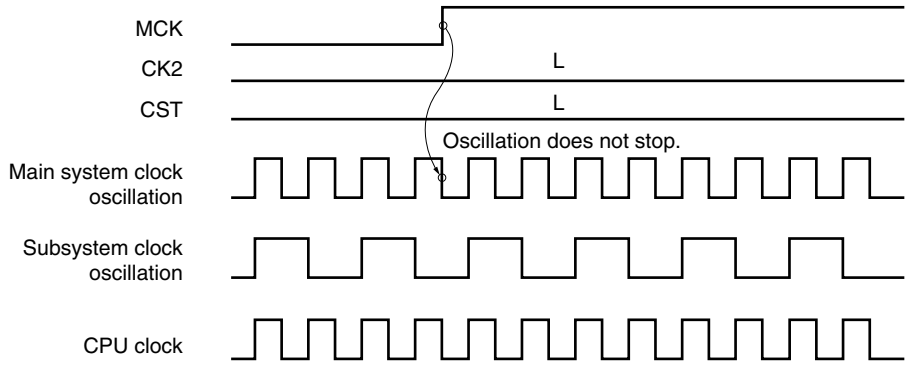
During operation with the main system clock (with bit 6 (CK2) of the standby control register (STBC) set to 0), the following operations are carried out.

(a) Because the operation guaranteed instruction execution speed depends on the power supply voltage, the instruction execution time can be changed by setting bits 4 to 6 (CK0 to CK2) of STBC.

(b) If bit 2 (MCK) of STBC is set to 1 during operation with the main system clock, the main system clock oscillation does not stop. When bit 6 (CK2) of STBC is set to 1 and the operation is switched to subsystem clock operation (CST = 1) after that, the main system clock oscillation stops (refer to **Figure 4-9**).

**Figure 4-9. Main System Clock Stop Function (1/2)**

**(a) Operation when MCK is set after setting CK2 during main system clock operation**




**(b) Operation when MCK is set during main system clock operation**

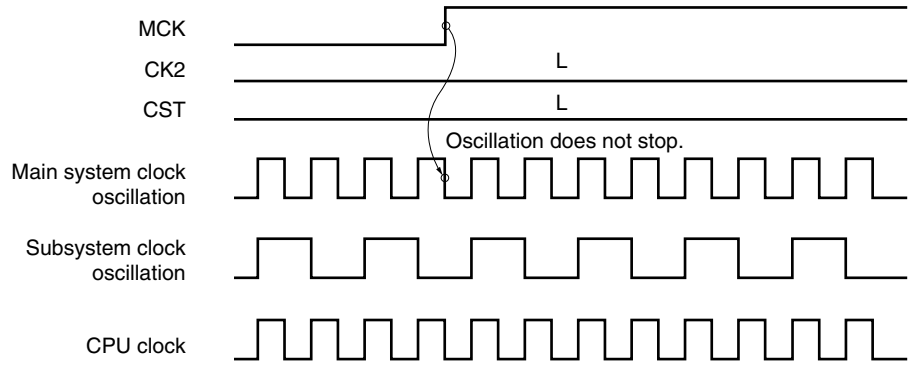
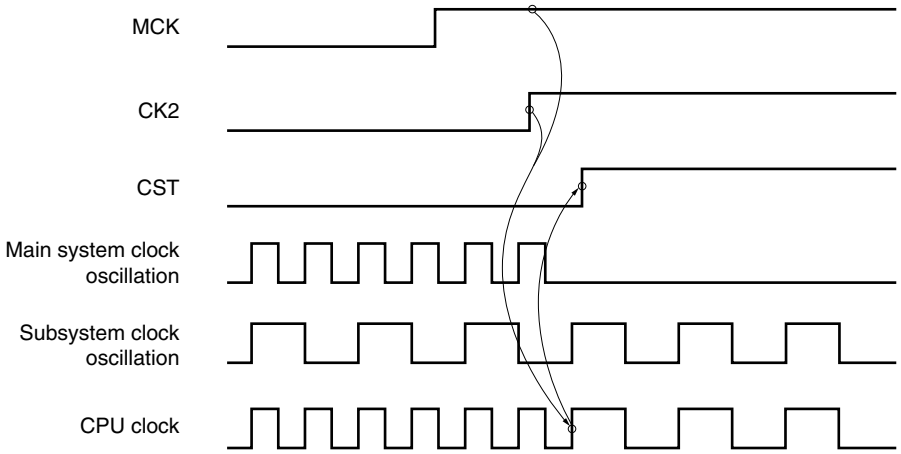


**Figure 4-9. Main System Clock Stop Function (2/2)**

**(c) Operation when CK2 is set after setting MCK during main system clock operation**




### 4.5.2 Subsystem clock operations

During operation with the subsystem clock (with bit 6 (CK2) of the standby control register (STBC) set to 1), the following operations are carried out.

(a) The instruction execution time remains constant (minimum instruction execution time (61 $\mu$s when operated at 32.768 kHz)) irrespective of the setting of bits 4 and 5 (CK0 and CK1) of STBC.

(b) The watchdog timer continues operating.

**Caution Do not set the STOP mode while the subsystem clock is operating.**

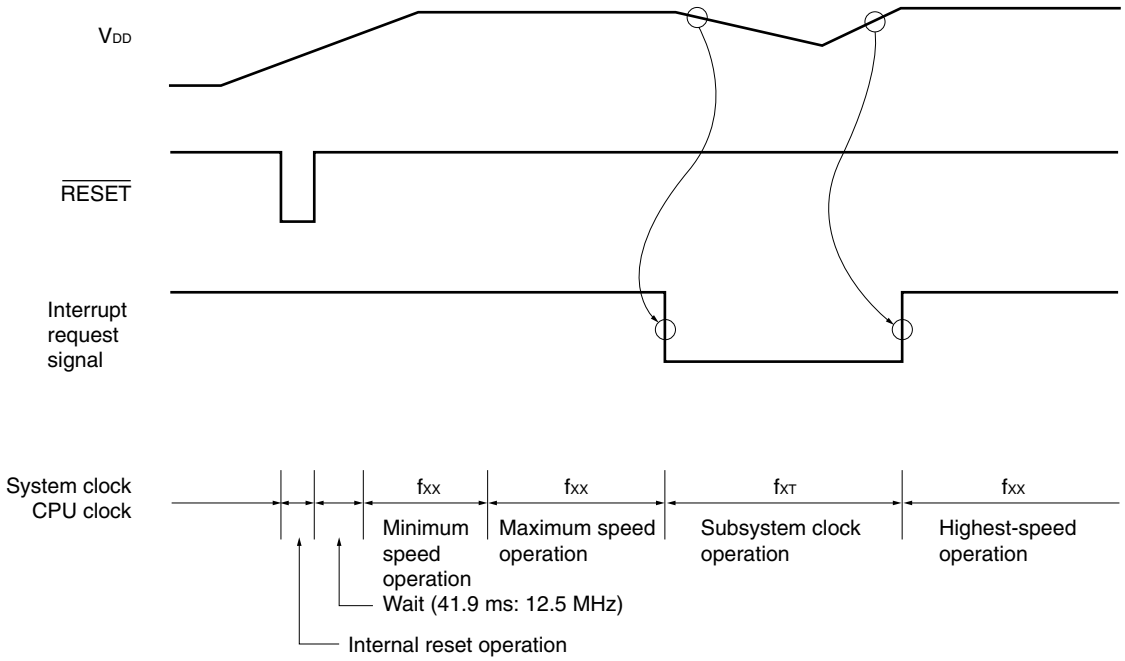
## 4.6 Changing System Clock and CPU Clock Settings

The system clock and CPU clock can be switched by means of bits 4 to 6 (CK0 to CK2) of the standby control register (STBC).

Whether the system is operating on the main system clock or the subsystem clock can be determined by the value of bit 0 (CST) of the clock status register (PCS).

This section describes the procedure for switching between the system clock and the CPU clock.

**Figure 4-10. System Clock and CPU Clock Switching**




(1) The CPU is reset by setting the $\overline{\text{RESET}}$ signal to low level after power application. After that, when reset is released by setting the $\overline{\text{RESET}}$ signal to high level, the main system clock starts oscillating. At this time, the oscillation stabilization time ($2^{19}$/fx) is secured automatically.
After that, the CPU starts operation at the minimum speed of the main system clock (1,280 ns: @ 12.5 MHz operation).

(2) After the lapse of a sufficient time for the $V_{DD}$ voltage to increase to enable operation at maximum speed, STBC and CC are rewritten and maximum-speed operation is carried out.

(3) Upon detection of a decrease in the $V_{DD}$ voltage due to an interrupt, the main system clock is switched to the subsystem clock (which must be in a stable oscillation state).

(4) Upon detection of $V_{DD}$ voltage reset due to an interrupt, bit 2 (MCK) of STBC is set to 0 and oscillation of the main system clock is started. After lapse of the time required for stabilization of oscillation, STBC is rewritten and maximum-speed operation is resumed.

**Caution When the main system clock is stopped and the device is operating on the subsystem clock, wait until the oscillation time has been secured by the program before switching back to the main system clock.**

# CHAPTER 5 PORT FUNCTIONS

## 5.1 Digital I/O Ports

The ports shown in Figure 5-1, which enable a variety of controls, are provided. The function of each port is described in Table 5-1. Connection of on-chip pull-up resistors can be specified for ports 0, 2 to 7, and 12 by a software setting.

**Figure 5-1. Port Configuration**

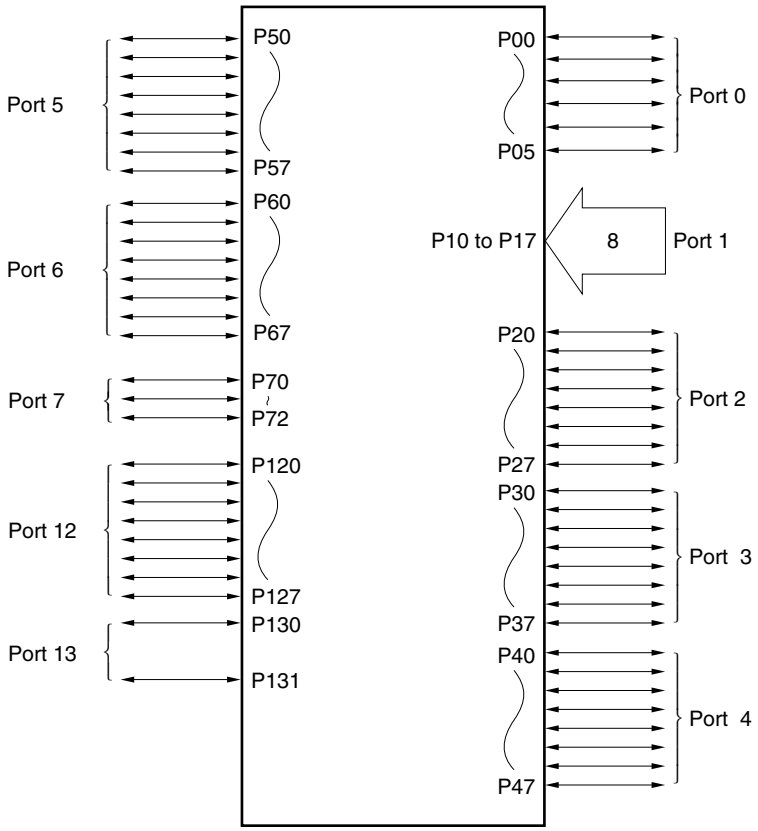


**Table 5-1. Port Functions**

| Port | Pin Name | Function | Specification of Software Pull-up Resistor |
|---|---|---|---|
| Port 0 | P00 to P05 | • Input or output can be specified in 1-bit units | Specifiable in 1-bit units |
| Port 1 | P10 to P17 | • Input port | — |
| Port 2 | P20 to P27 | • Input or output can be specified in 1-bit units | Specifiable in 1-bit units |
| Port 3 | P30 to P37 | • Input or output can be specified in 1-bit units | Specifiable in 1-bit units |
| Port 4 | P40 to P47 | • Input or output can be specified in 1-bit units<br>• Can drive LEDs directly | Specifiable individually for each port |
| Port 5 | P50 to P57 | • Input or output can be specified in 1-bit units<br>• Can drive LEDs directly | Specifiable individually for each port |
| Port 6 | P60 to P67 | • Input or output can be specified in 1-bit units | Specifiable individually for each port |
| Port 7 | P70 to P72 | • Input or output can be specified in 1-bit units | Specifiable in 1-bit units |
| Port 12 | P120 to P127 | • Input or output can be specified in 1-bit units | Specifiable in 1-bit units |
| Port 13 | P130, P131 | • Input or output can be specified in 1-bit units | — |

## 5.2 Port Configuration

The ports include the following hardware.

**Table 5-2. Port Configuration**

| Item | Configuration |
|---|---|
| Control registers | Port mode register (PMm: m = 0, 2 to 7, 12, 13)<br>Pull-up resistor option register (PUO, PUm: m = 0, 2, 3, 7, 12) |
| Ports | Total: 67 (input: 8, I/O: 59) |
| Pull-up resistors | Total: 57 (software control) |

### 5.2.1 Port 0

Port 0 is a 6-bit I/O port with an output latch. The input mode/output mode can be specified for the P00 to P05 pins in 1-bit units using the port 0 mode register. A pull-up resistor can also be connected in 1-bit units via pull-up resistor option register 0, regardless of whether the input mode or output mode is specified.

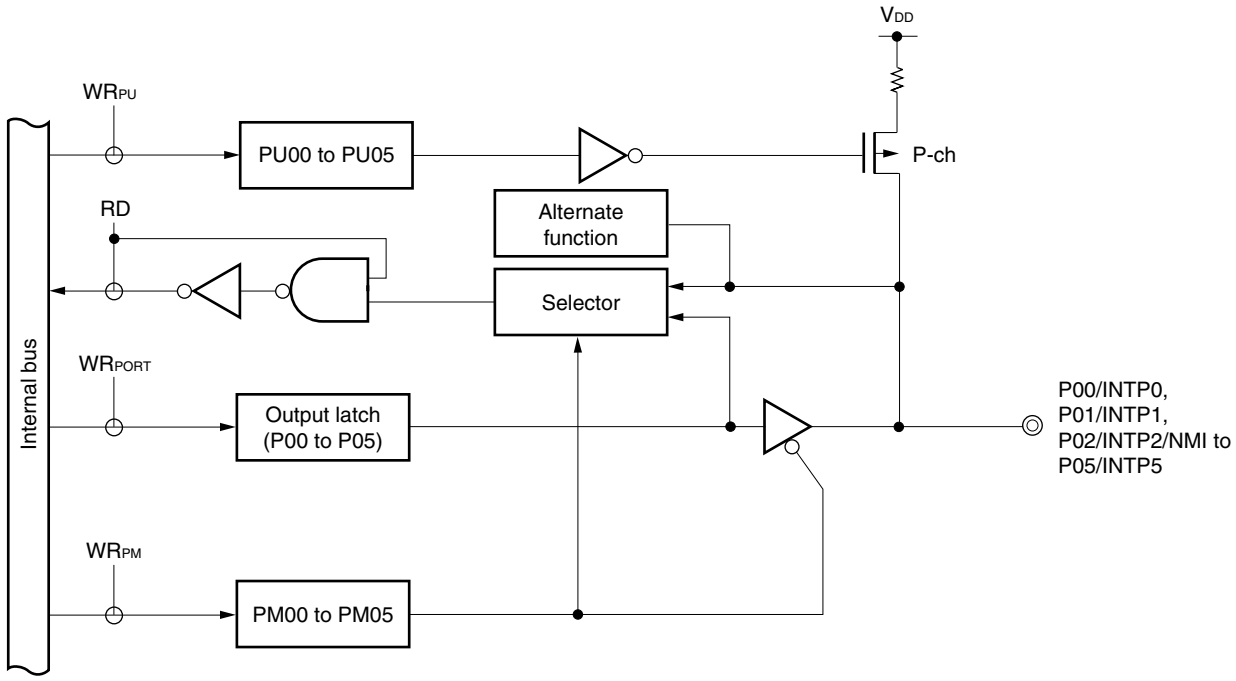
Port 0 also supports external interrupt request input as an alternate function.

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

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

**Caution Even though port 0 is also used as an external interrupt input, when port 0 is not used as an interrupt input pin, be sure to set interrupt disabled by using external interrupt rising edge enable register 0 (EGP0) and external interrupt falling edge enable register 0 (EGN0) or setting the interrupt enable flag (PMKn: n = 0 to 5) to 1. Otherwise, the interrupt request flag is set and unintentional interrupt servicing may be executed when setting ports to output mode and thus changing the output level.**

**Figure 5-2. Block Diagram of P00 to P05**




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

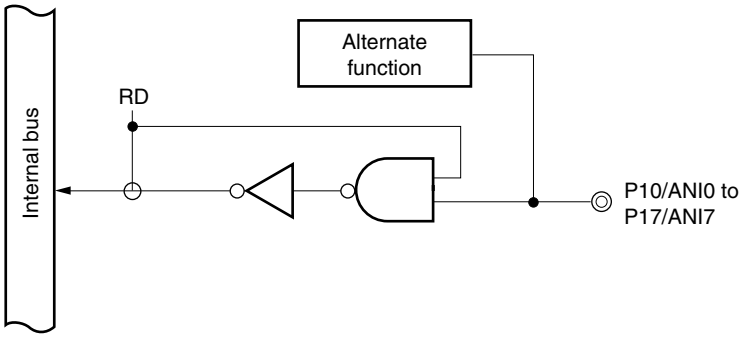
### 5.2.2 Port 1

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

Port 1 supports A/D converter analog input as an alternate function.

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

**Figure 5-3. Block Diagram of P10 to P17**




RD: Port 1 read signal

**Caution Do not execute a read instruction (including bit manipulation instructions) for port 1 when it is used as an analog input port since port 1 can also be used as A/D converter analog input. If middle voltage is input to an analog input pin while the port is being read, the middle voltage will be read, possibly impairing the reliability of the device.**

### 5.2.3 Port 2

Port 2 is an 8-bit I/O port with an output latch. The input mode/output mode can be specified for the P20 to P27 pins in 1-bit units using the port 2 mode register. A pull-up resistor can also be connected in 1-bit units via pull-up resistor option register 2, regardless of whether the input mode or output mode is specified.

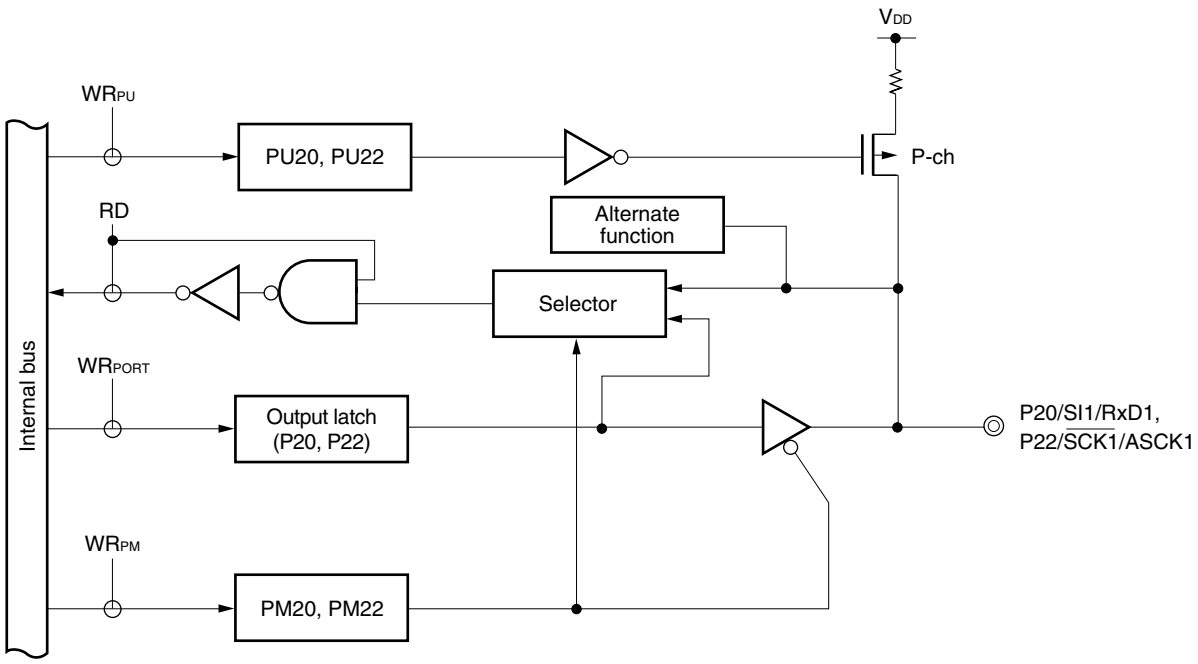
The P25 and P27 pins can be specified as N-ch open-drain using a port function control register (only the $\mu$PD784225Y Subseries).

Port 2 supports serial interface data I/O, clock I/O, clock output, and buzzer output as alternate functions.

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

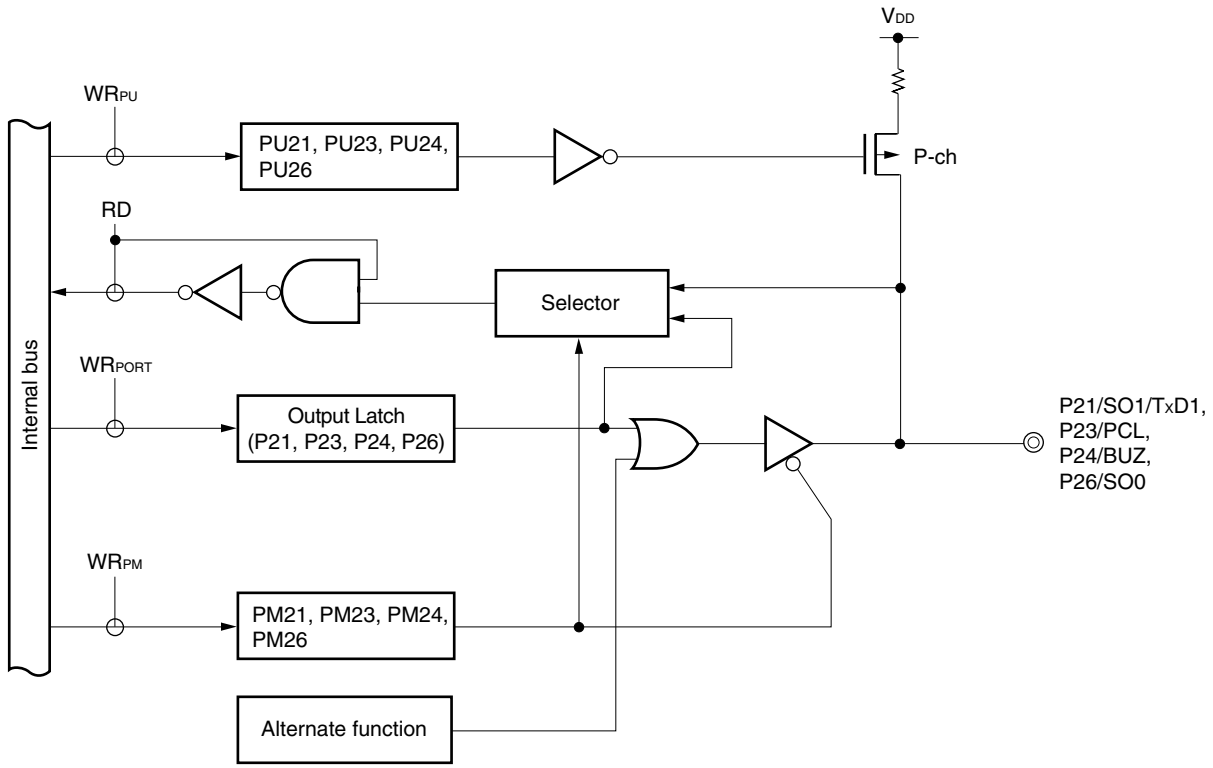
Figures 5-4 to 5-7 show a block diagram of port 2.

**Figure 5-4. Block Diagram of P20 and P22**



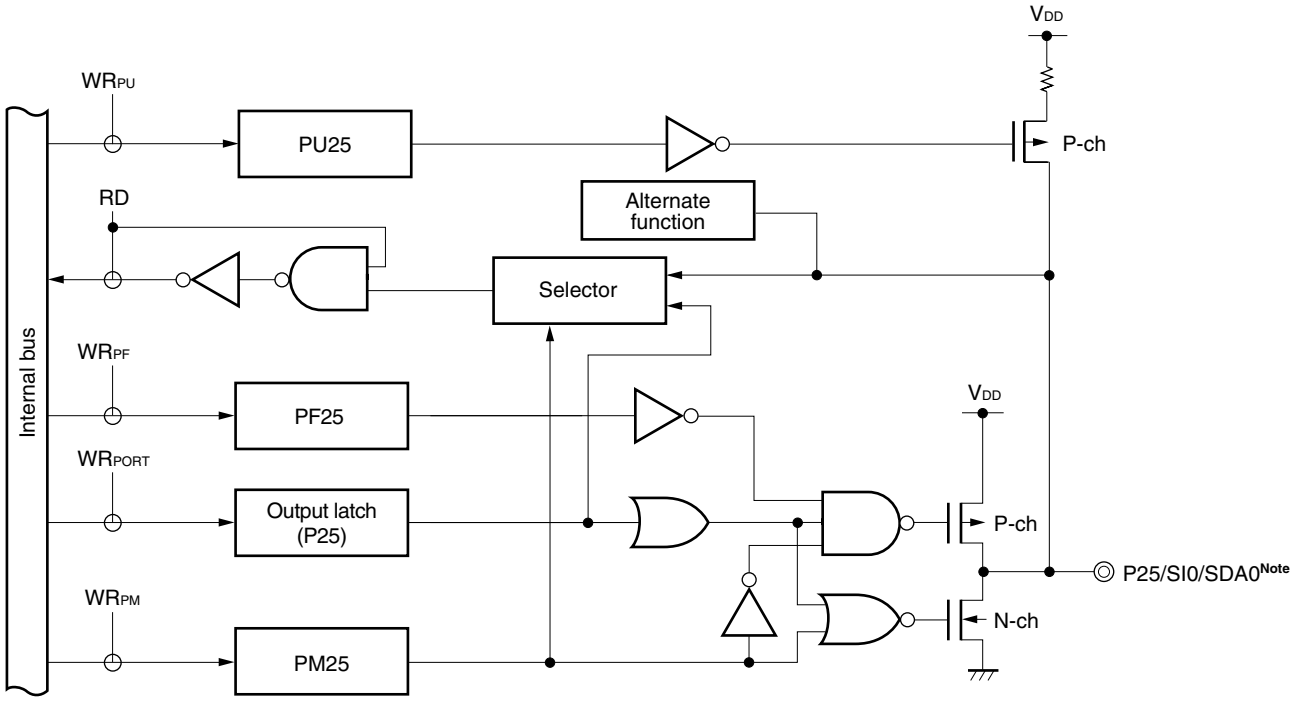

PU: Pull-up resistor option register
PM: Port mode register
RD: Port 2 read signal
WR: Port 2 write signal

**Figure 5-5. Block Diagram of P21, P23 to P24, and P26**




PU: Pull-up resistor option register
PM: Port mode register
RD: Port 2 read signal
WR: Port 2 write signal

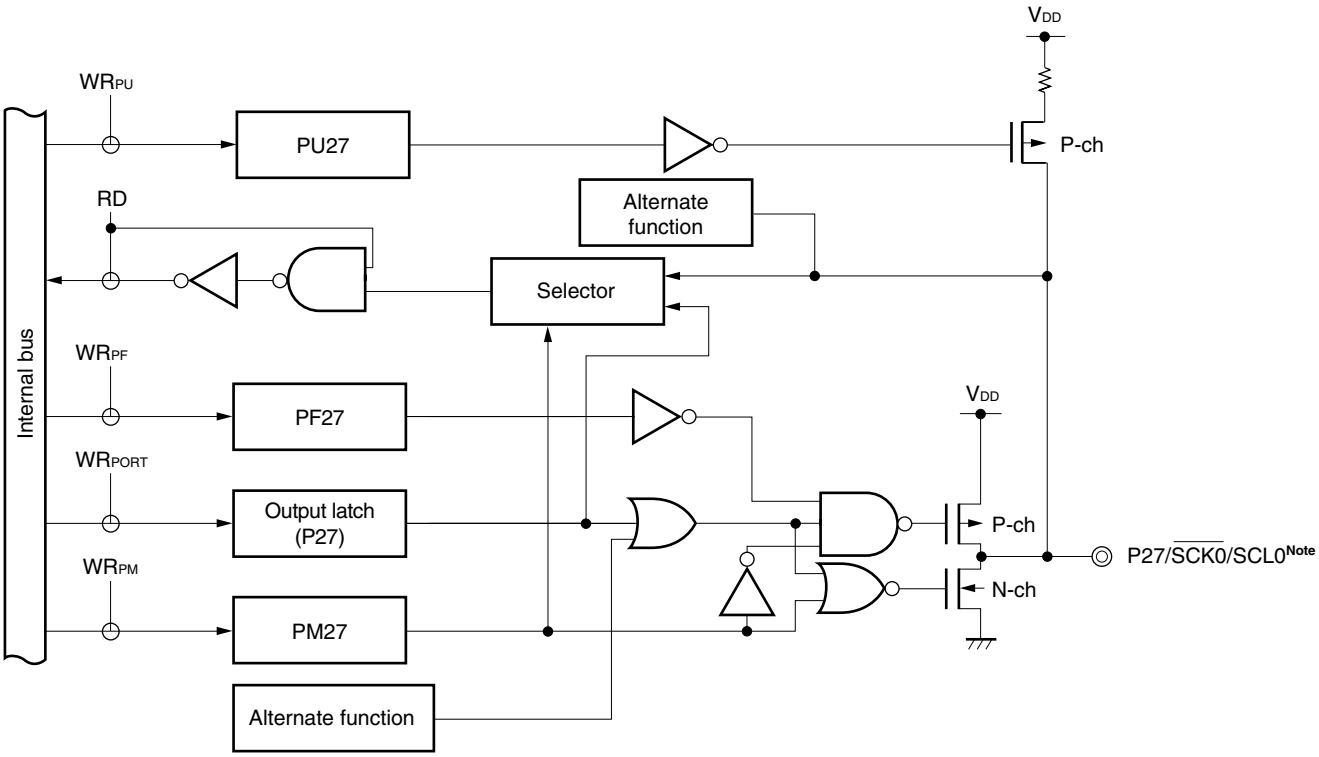
**Figure 5-6. Block Diagram of P25**




**Note** The SDA0 pin applies only to the μPD784225Y Subseries.

PU: Pull-up resistor option register
PF: Port function control register
PM: Port mode register
RD: Port 2 read signal
WR: Port 2 write signal

**Figure 5-7. Block Diagram of P27**




**Note** The SCL0 pin applies only to the $\mu$PD784225Y Subseries.

PU: Pull-up resistor option register
PF: Port function control register
PM: Port mode register
RD: Port 2 read signal
WR: Port 2 write signal

### 5.2.4 Port 3

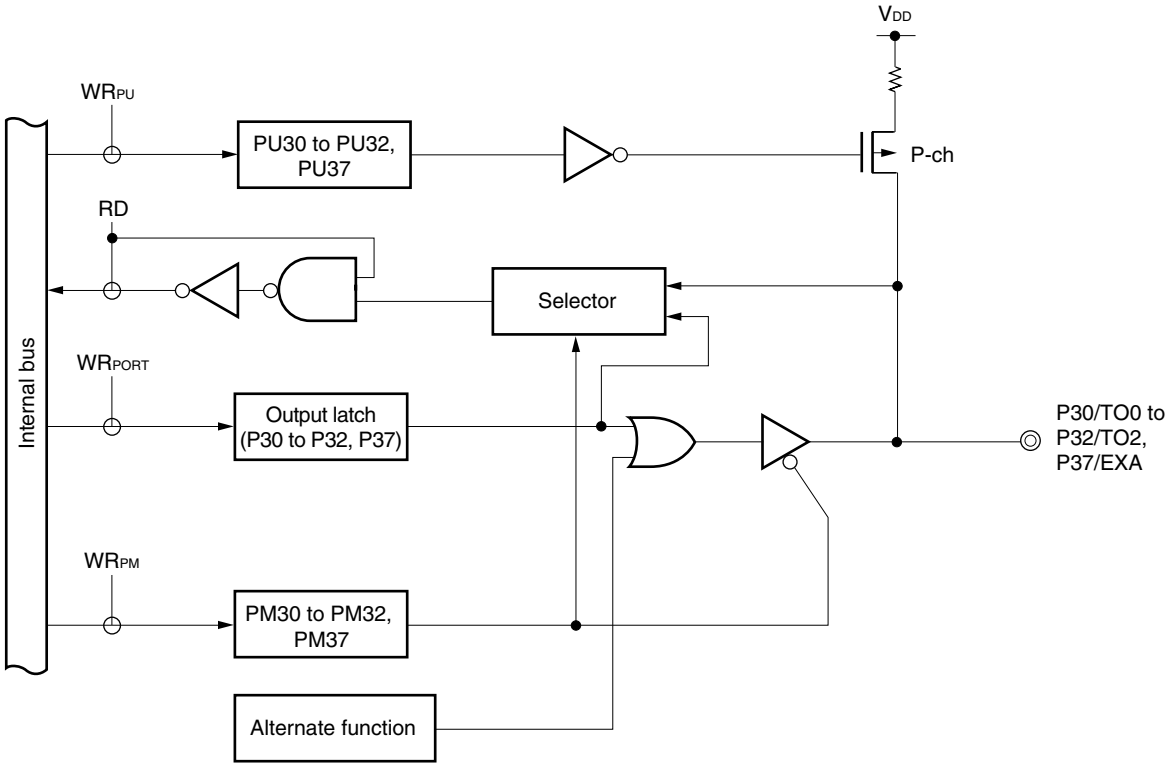
Port 3 is an 8-bit I/O port with an output latch. The input mode/output mode can be specified for the P30 to P37 pins in 1-bit units using the port 3 mode register. A pull-up resistor can also be connected in 1-bits units via pull-up resistor option register 3, regardless of whether the input mode or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

Port 3 supports timer I/O as an alternate function.

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

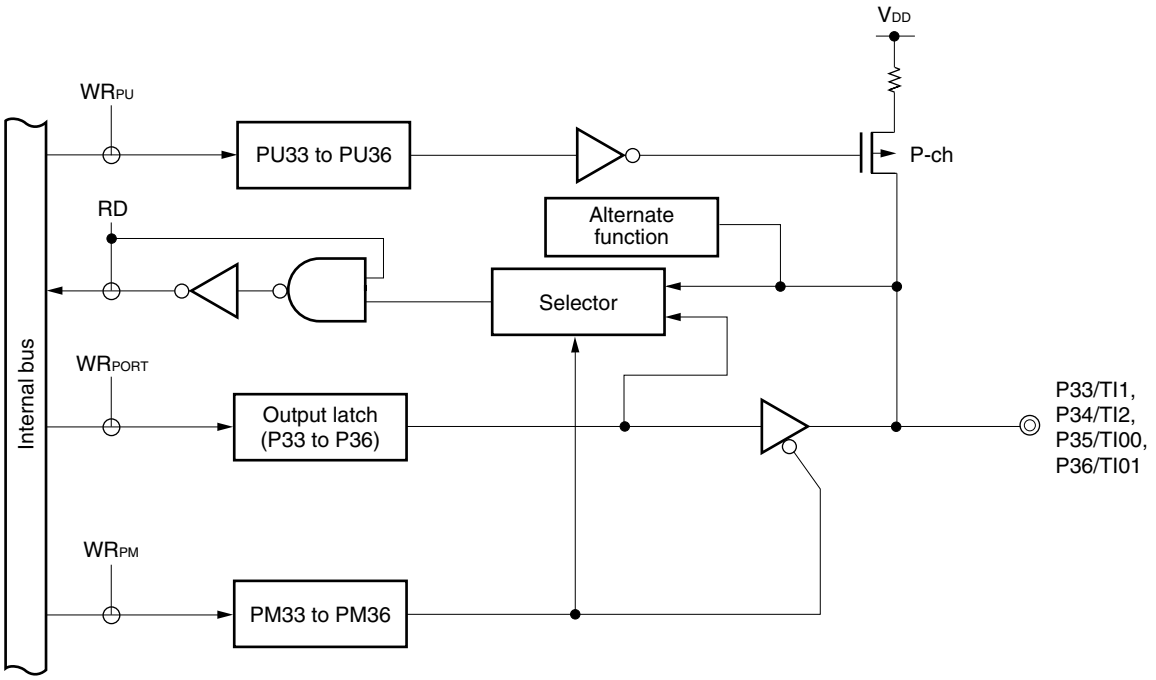
Figures 5-8 and 5-9 show a block diagram of port 3.

**Figure 5-8. Block Diagram of P30 to P32, and P37**




PU: Pull-up resistor option register
PM: Port mode register
RD: Port 3 read signal
WR: Port 3 write signal

**Figure 5-9. Block Diagram of P33 to P36**




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

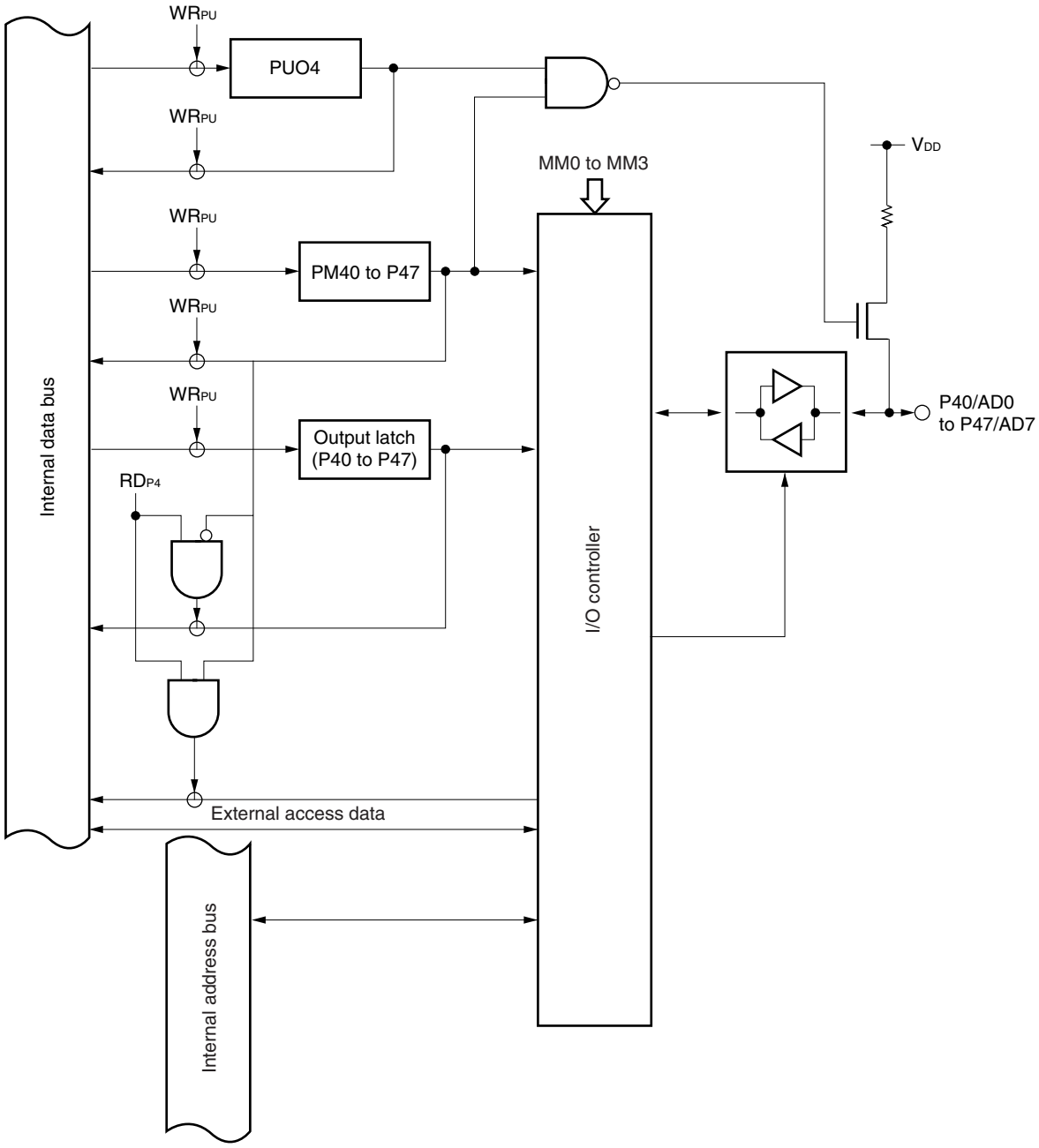
Port 4 can drive LEDs directly.

Port 4 supports the address/data bus function in the external memory expansion mode as an alternate function.

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

Figure 5-10 shows a block diagram of port 4.

**Figure 5-10. 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 I/O port with an output latch. The input mode/output mode can be specified for the P50 to P57 pins in 1-bit units using the port 5 mode register. When the P50 to P57 pins are used as input ports, a pull-up resistor can be connected in 8-bit units via bit 5 (PUO5) of the pull-up resistor option register.

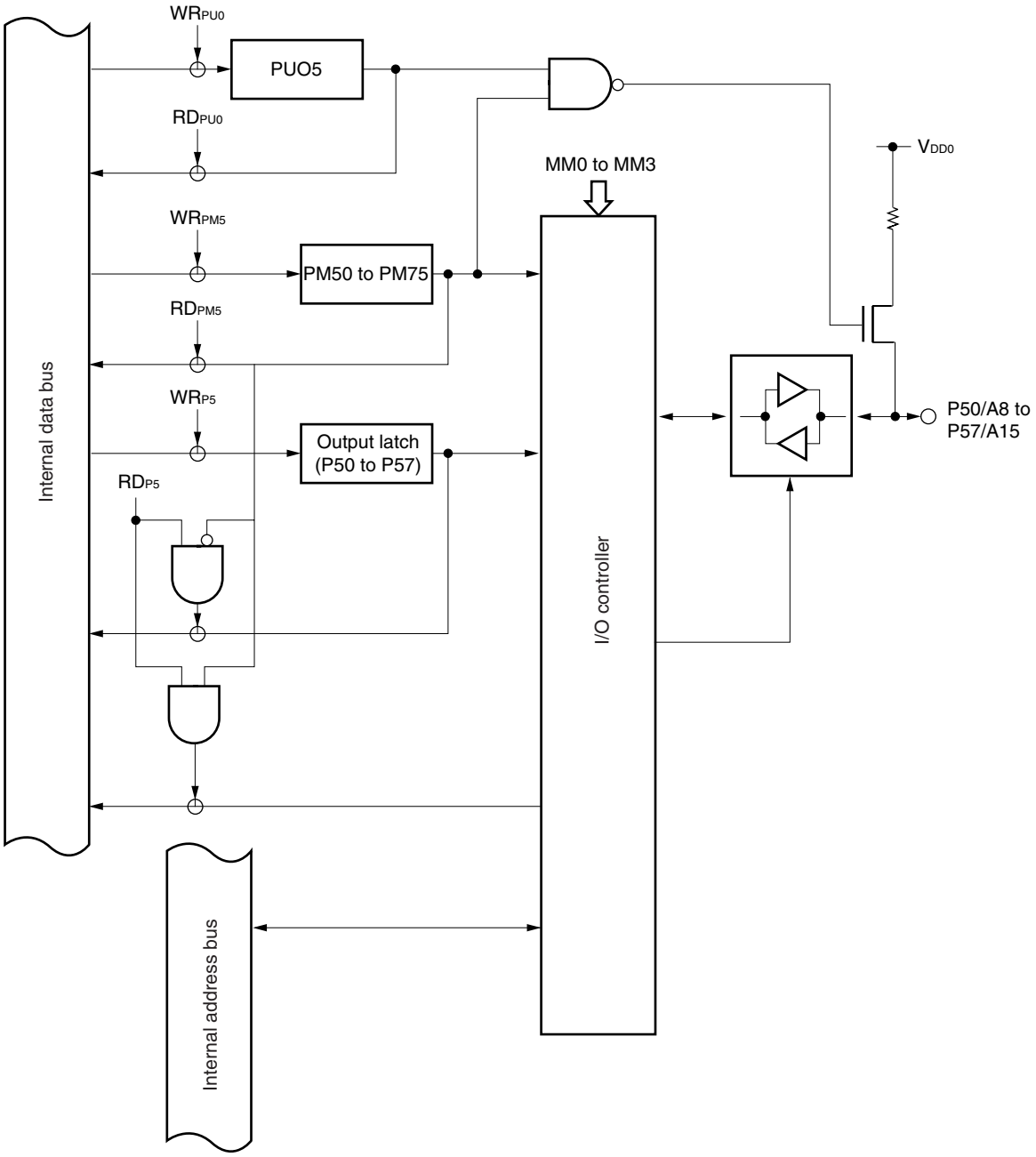
Port 5 can drive LEDs directly.

Port 5 supports the address bus function in the external memory expansion mode as an alternate function.

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

Figure 5-11 shows a block diagram of port 5.

**Figure 5-11. 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

MM0 to MM3: Bits 0 to 3 of the memory expansion mode register (MM)

### 5.2.7 Port 6

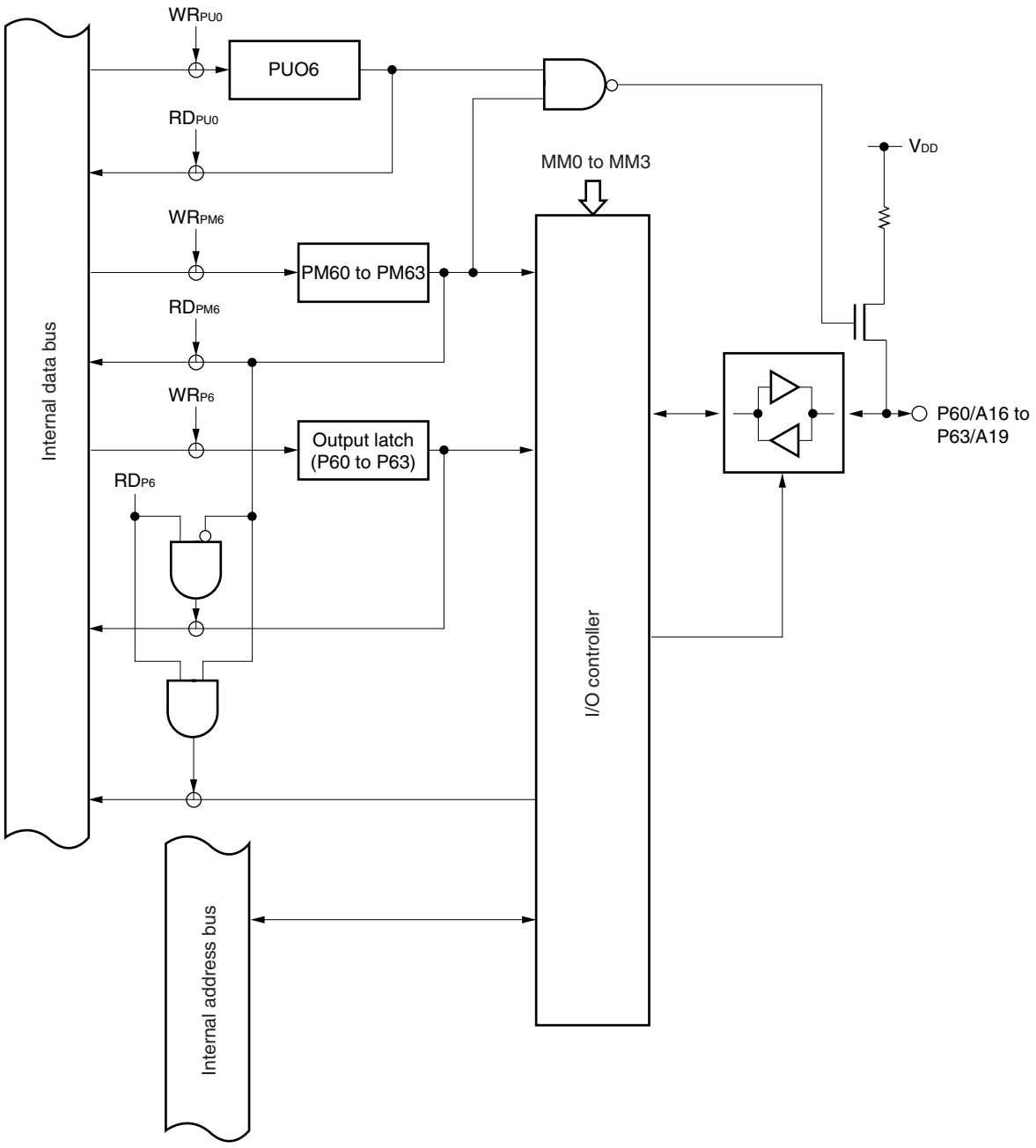
Port 6 is an 8-bit I/O port with an output latch. The input mode/output mode can be specified for the P60 to P67 pins in 1-bit units using the port 6 mode register. When pins P60 to P67 are used as input ports, a pull-up resistor can be connected in 8-bit units via bit 6 (PUO6) of the pull-up resistor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

Port 6 supports the address bus function and the control signal output function in external memory expansion mode as alternate functions.

RESET input sets port 6 to the input mode.

Figures 5-12 to 5-14 show block diagrams of port 6.

**Figure 5-12. Block Diagram of P60 to P63**




PUO: Pull-up resistor option register

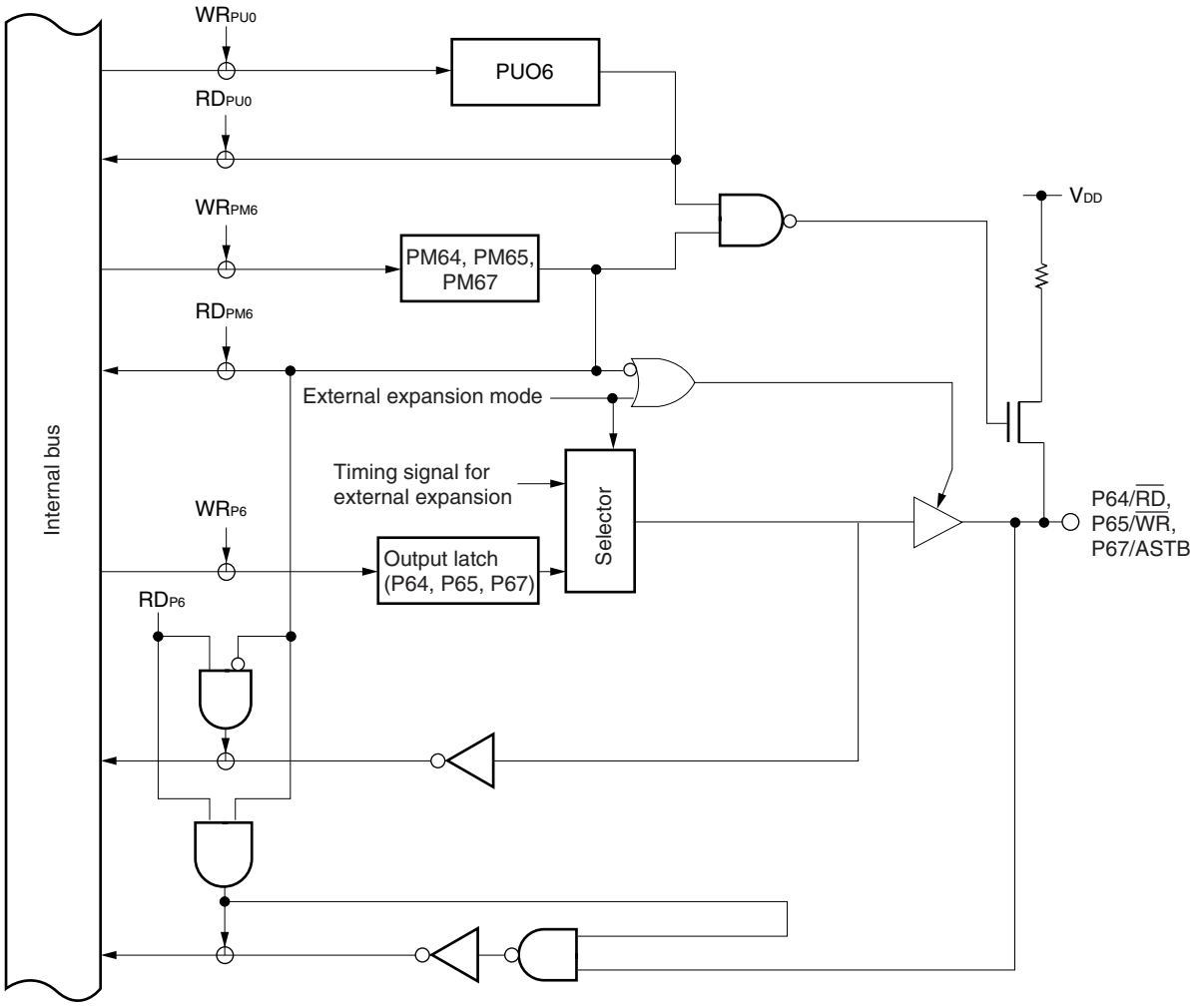
PM: Port mode register

RD: Port 6 read signal

WR: Port 6 write signal

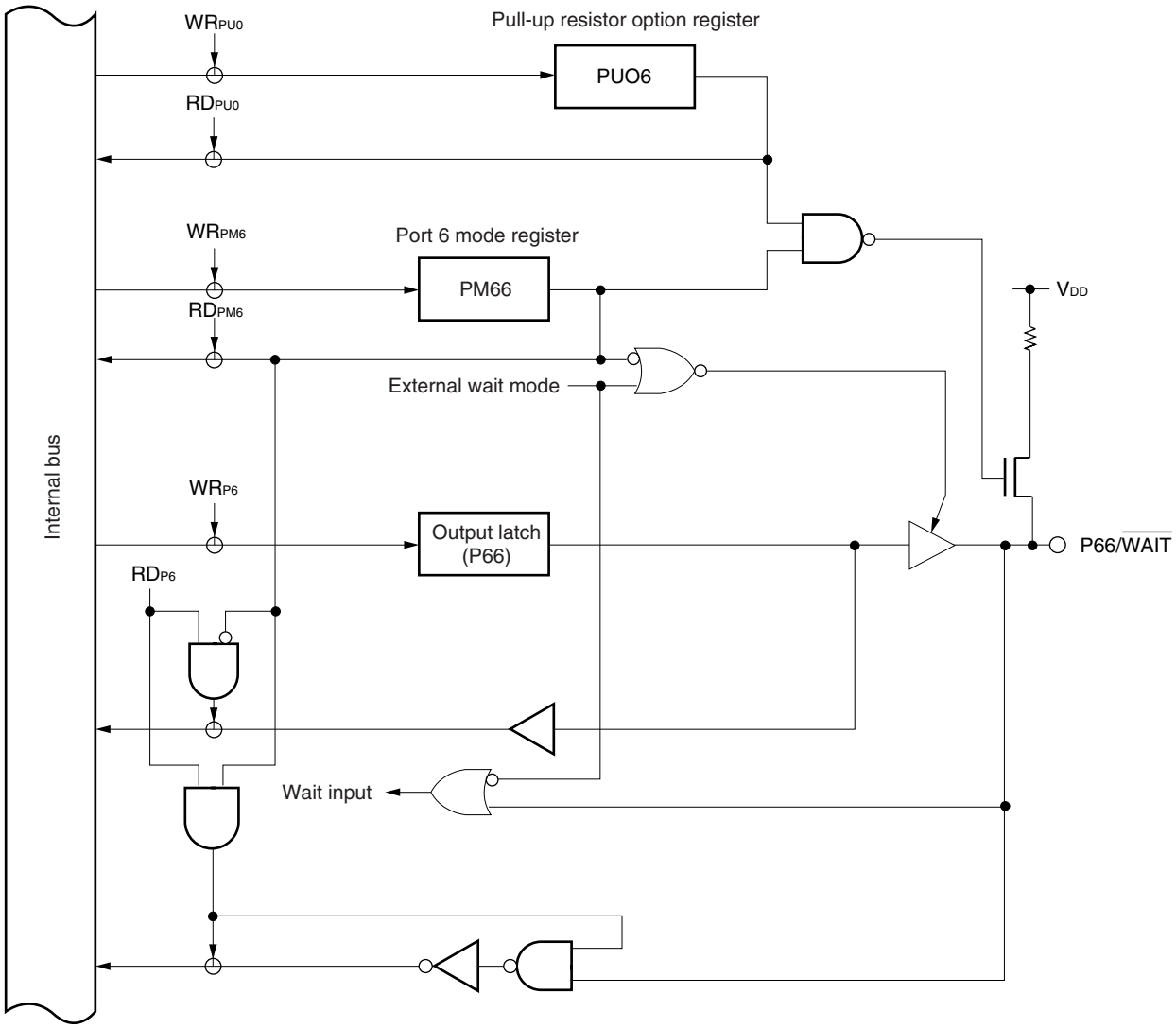
MM0 to MM3: Bits 0 to 3 of the memory expansion mode register (MM)

**Figure 5-13. Block Diagram of P64, P65, and P67**




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

**Figure 5-14. Block Diagram of P66**




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

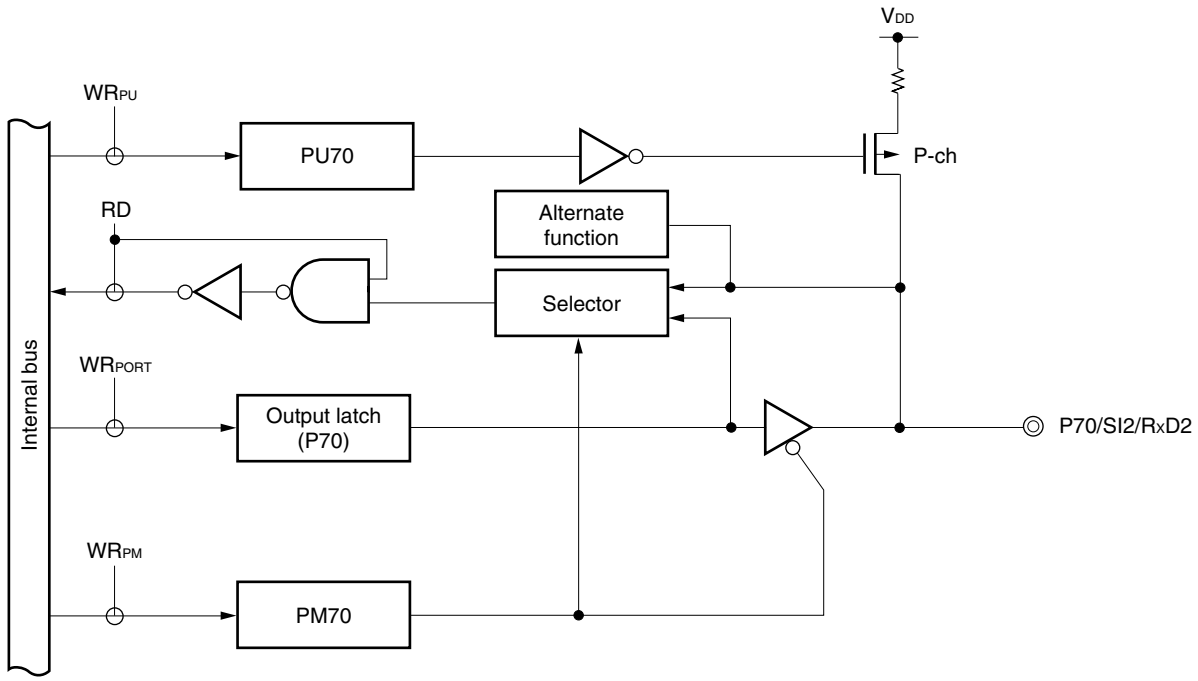
This is a 3-bit I/O port with an output latch. Input mode/output mode can be specified for the P70 to P72 pins in 1-bit units using the port 7 mode register. A pull-up resistor can be connected via pull-up resistor option register 7, regardless of whether the input mode or output mode is specified.

Port 7 supports serial interface data I/O and clock I/O as alternate functions.

$\overline{\text{RESET}}$ input sets port 7 to the input mode.

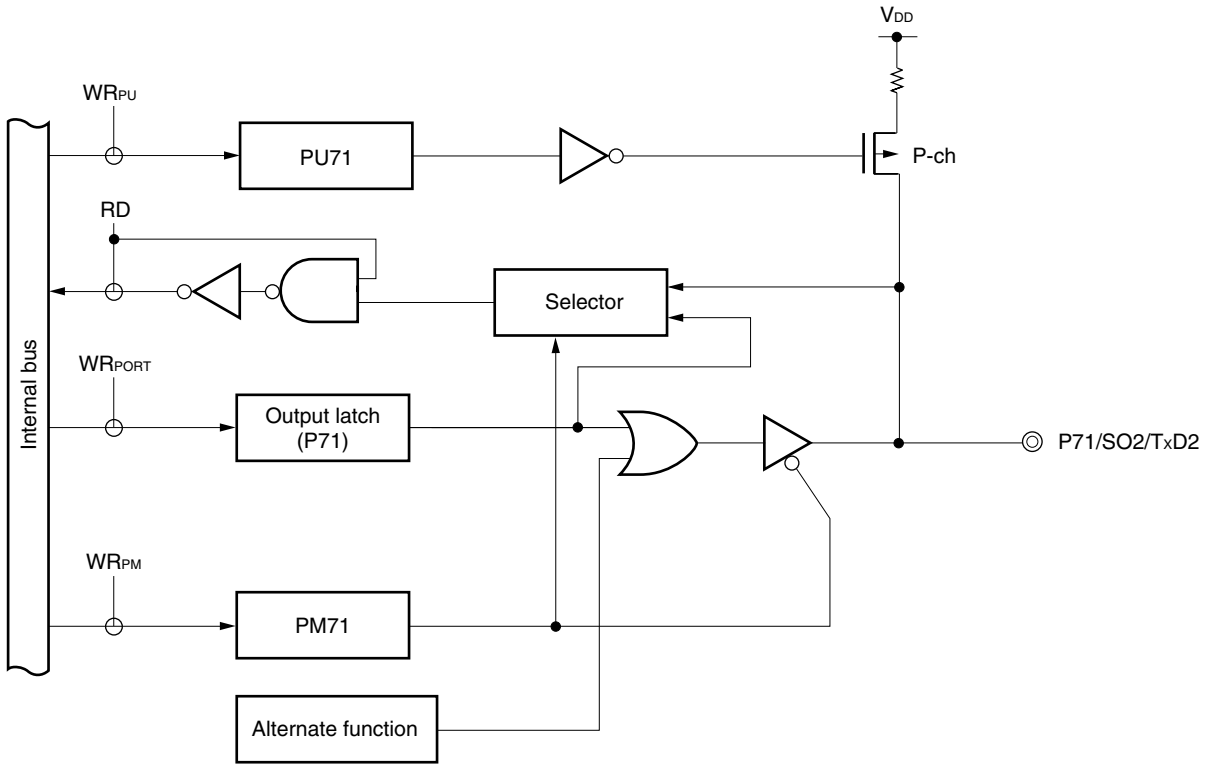
Figures 5-15 to 5-17 show block diagrams of port 7.

**Figure 5-15. Block Diagram of P70**



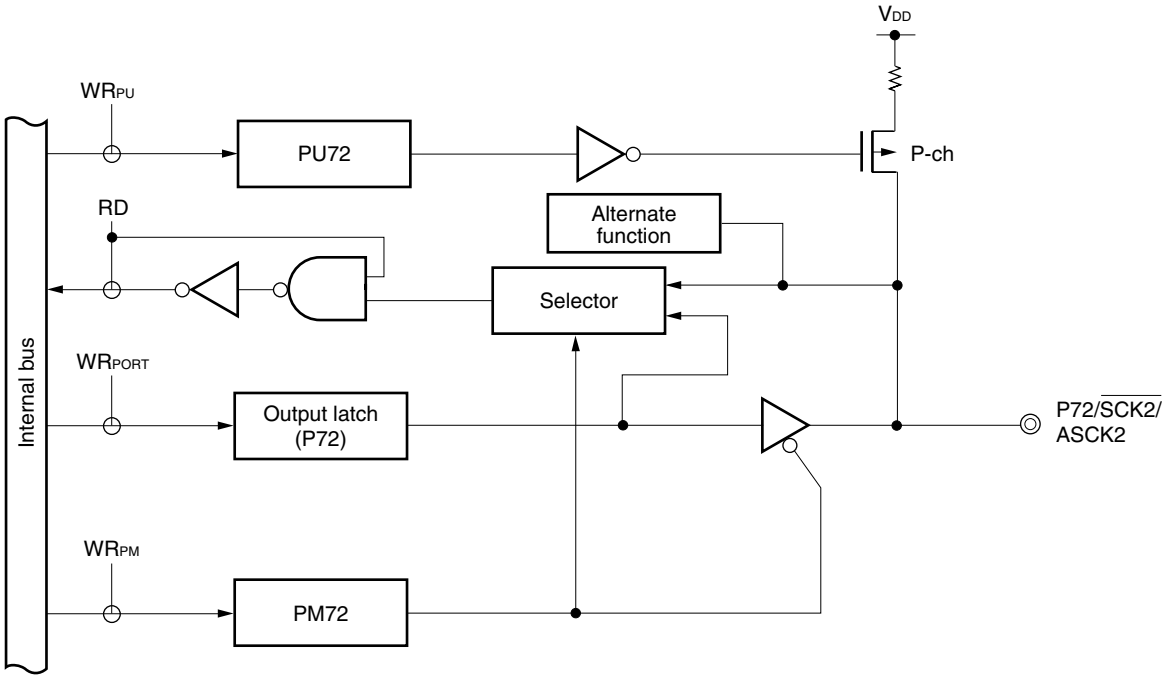

PU: Pull-up resistor option register
PM: Port mode register
RD: Port 7 read signal
WR: Port 7 write signal

**Figure 5-16. Block Diagram of P71**




PU: Pull-up resistor option register
PM: Port mode register
RD: Port 7 read signal
WR: Port 7 write signal

**Figure 5-17. Block Diagram of P72**




PU: Pull-up resistor option register
PM: Port mode register
RD: Port 7 read signal
WR: Port 7 write signal

### 5.2.9 Port 12

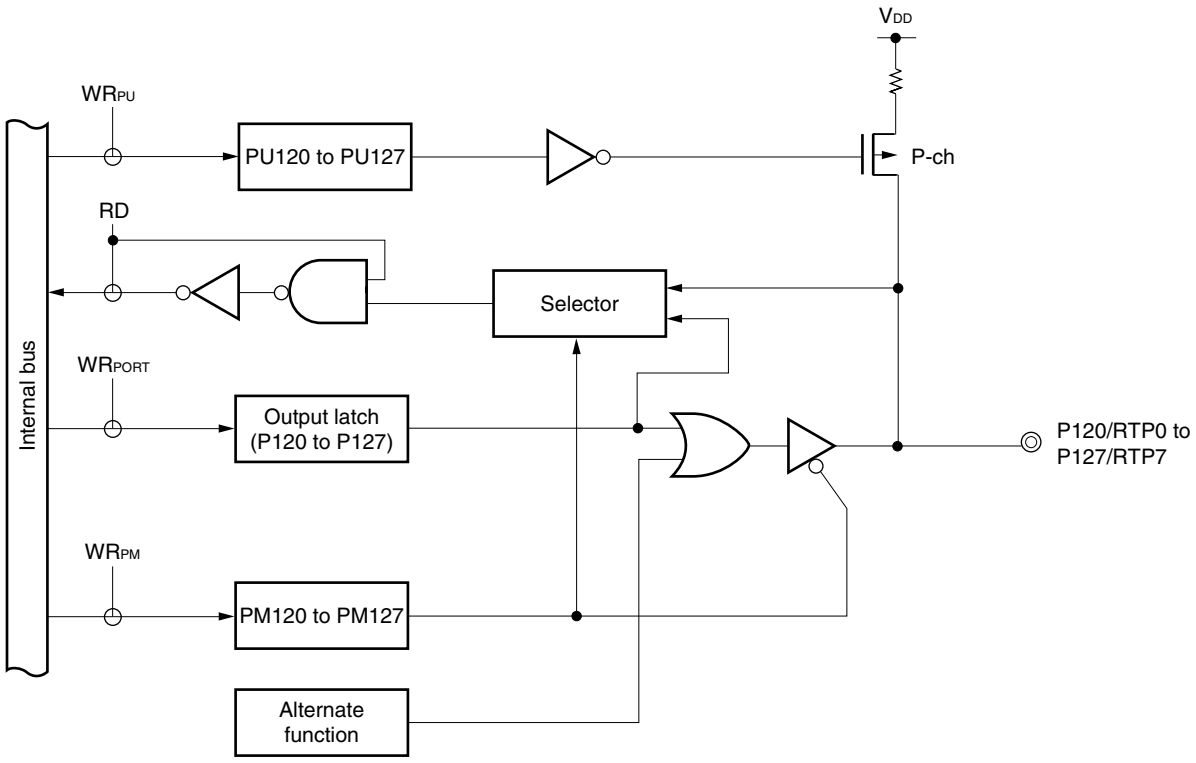
This is an 8-bit I/O port with an output latch. Input mode/output mode can be specified for the P120 to P127 pins in 1-bit units using the port 12 mode register. A pull-up resistor can be connected via pull-up resistor option register 12, regardless of whether the input mode or output mode is specified.

Port 12 supports the real-time output function as an alternate function.

$\overline{\text{RESET}}$ input sets port 12 to the input mode.

Figure 5-18 shows a block diagram of port 12.

**Figure 5-18. Block Diagram of P120 to P127**




PU: Pull-up resistor option register
PM: Port mode register
RD: Port 12 read signal
WR: Port 12 write signal

### 5.2.10 Port 13

This is a 2-bit I/O port with an output latch. The input mode/output mode can be specified for the P130 and P131 pins in 1-bit units using the port 13 mode register. Port 13 does not include pull-up resistors.

Port 13 supports D/A converter analog output as an alternate function.

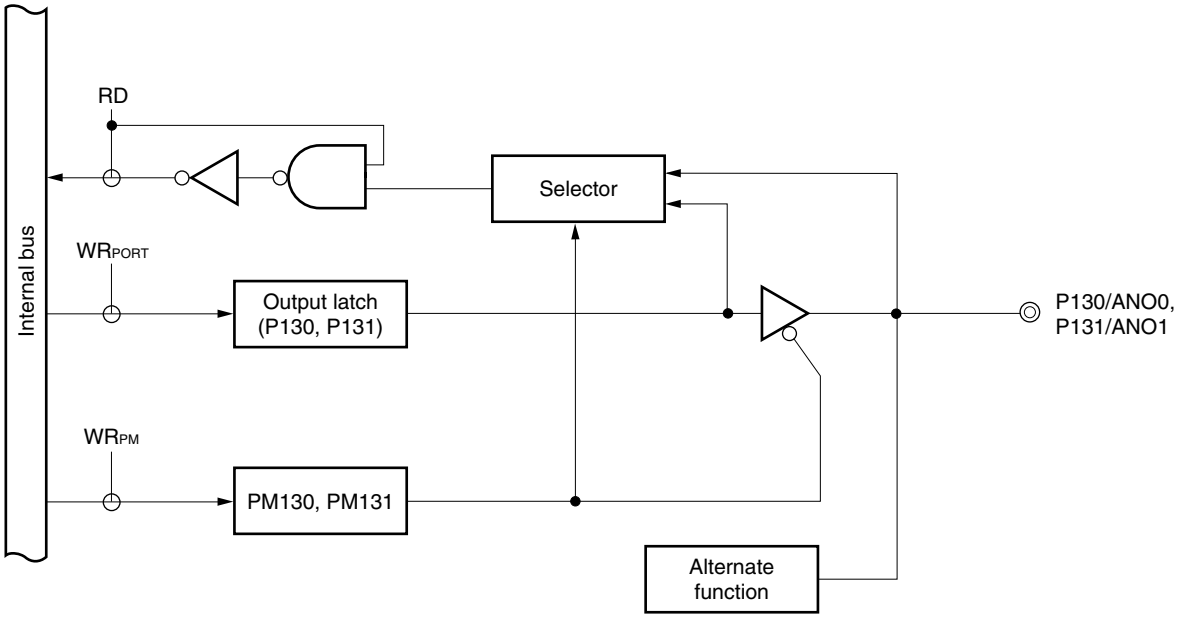
$\overline{\text{RESET}}$ input sets port 13 to the input mode.

Figure 5-19 shows a block diagram of port 13.

**Caution When only one of the D/A converter channels is used with $AV_{REF1} < V_{DD}$, the other pins that are not used as analog outputs must be set as follows.**

- **Set the port mode register (PM13x) to 1 (input mode) and connect the pin to $V_{SS0}$.**
- **Set the port mode register (PM13x) to 0 (output mode) and the output latch to 0 to output a low level from the pin.**

**Figure 5-19. Block Diagram of P130 and P131**




PM: Port mode register
RD: Port 13 read signal
WR: Port 13 write signal

## 5.3 Control Registers

The following three types of registers control the ports.

- Port mode registers (PM0, PM2 to PM7, PM12, PM13)
- Pull-up resistor option registers (PU0, PU2, PU3, PU7, PU12, PUO)
- Port function control register 2 (PF2)[Note]

**Note** Applies only to the μPD784225Y Subseries.

**(1) Port mode registers (PM0, PM2 to PM7, PM12, PM13)**

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

PM0, PM2 to PM7, PM12, and PM13 are set with a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets the 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 Even though port 0 is also used as an external interrupt input, when port 0 is not used as an interrupt input pin, be sure to set interrupt disabled by using external interrupt rising edge enable register 0 (EGP0) and external interrupt falling edge enable register 0 (EGN0) or setting the interrupt enable flag (PMKn: n = 0 to 5) to 1. Otherwise, the interrupt request flag is set and unintentional interrupt servicing may be executed when setting ports to output mode and thus changing the output level.**

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

| Pin Name | Alternate Function | | PM×× | P×× | Pin Name | Alternate Function | | PM×× | P×× |
|---|---|---|---|---|---|---|---|---|---|
| | Name | I/O | | | | Name | I/O | | |
| P00, P01 | INTP0, INTP1 | Input | 1 | × | P35, P36 | TI00, TI01 | Input | 1 | × |
| P02 | INTP2/NMI | Input | 1 | × | P37 | EXA | Output | 0 | 0 |
| P03 to P05 | INTP3 to INTP5 | Input | 1 | × | P40 to P47 | AD0 to AD7 | I/O | ×Note 2 | |
| P10 to P17Note 1 | ANI0 to ANI7 | Input | — | | P50 to P57 | A8 to A15 | Output | ×Note 2 | |
| P20 | RxD1/SI1 | Input | 1 | × | P60 to P63 | A16 to A19 | Output | ×Note 2 | |
| P21 | TxD1/SO1 | Output | 0 | 0 | P64 | $\overline{RD}$ | Output | ×Note 2 | |
| P22 | ASK1 | Input | 1 | × | P65 | $\overline{WR}$ | Output | ×Note 2 | |
| | $\overline{SCK1}$ | Input | 1 | × | P66 | $\overline{WAIT}$ | Input | ×Note 2 | |
| | | Output | 0 | 0 | P67 | ASTB | Output | ×Note 2 | |
| P23 | PCL | Output | 0 | 0 | P70 | RxD2/SI2 | Input | 1 | × |
| P24 | BUZ | Output | 0 | 0 | P71 | TxD2/SO2 | Output | 0 | 0 |
| P25 | SI0 | Input | 1 | × | P72 | ASCK2 | Input | 1 | × |
| | SDA0Note 3 | I/O | 0 | 0 | | $\overline{SCK2}$ | Input | 1 | × |
| P26 | SO0 | Output | 0 | 0 | | | Output | 0 | 0 |
| P27 | $\overline{SCK0}$ | Input | 1 | × | P120 to P127 | RTP0 to RTP7 | Output | 0 | 0 |
| | | Output | 0 | 0 | P130, P131Note 1 | ANO0, ANO1 | Output | 1 | × |
| | SCL0Note 3 | I/O | 0 | 0 | | | | | |
| P30 to P32 | TO0 to TO2 | Output | 0 | 0 | | | | | |
| P33, P34 | TI1, TI2 | Input | 1 | × | | | | | |

**Notes 1.** If the read command is executed for these ports when they are being used as alternate function pins, the data read will be undefined.

**2.** The function is set by the memory expansion mode register (MM) when the P40 to P47, P50 to P57 and P60 to P67 pins are used as alternate function pins.

**3.** The SDA0 and SCL0 pins are only available in the μPD784225Y Subseries.

**Cautions 1. When not using external wait in the external memory expansion mode, the P66 pin can be used as an I/O port.**

**2. Specify the SCL0/P27 and SDA0/P25 pins as N-ch open-drain by setting the port function control register (PF2) when the $I^2C$ bus mode is to be used.**

**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 5-20. Format of Port Mode Register**

Address: 0FF20H, 0FF22H to 0FF27H, 0FF2CH, 0FF2DH After reset: FFH R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PM0 | 1 | 1 | PM05 | PM04 | PM03 | PM02 | PM01 | PM00 |
| PM2 | PM27 | PM26 | PM25 | PM24 | PM23 | PM22 | 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 |
| PM7 | 1 | 1 | 1 | 1 | 1 | PM72 | PM71 | PM70 |
| PM12 | PM127 | PM126 | PM125 | PM124 | PM123 | PM122 | PM121 | PM120 |
| PM13 | 1 | 1 | 1 | 1 | 1 | 1 | PM131 | PM130 |

| PMxn | Pxn pin I/O mode specification<br>(x = 0: n = 0 to 5<br>x = 2 to 6, 12: n = 0 to 7<br>x = 7: n = 0 to 2<br>x = 13: n = 0, 1) |
|---|---|
| 0 | Output mode (output buffer on) |
| 1 | Input mode (output buffer off) |

**(2) Pull-up resistor option registers (PU0, PU2, PU3, PU7, PU12, PUO)**

These registers are used to set whether to use an on-chip pull-up resistor at each port or not in 1-bit or 8-bit units. PUn (n = 0, 2, 3, 7, 12) can specify pull-up resistor connection at each port pin. PUO can specify pull-up resistor connection at ports 4, 5, and 6. 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. Ports 1 and 13 do not incorporate pull-up resistors.**

**2. For ports 4, 5, and 6, a pull-up resistor can be connected in external memory expansion mode.**

**Figure 5-21. Format of Pull-Up Resistor Option Register**

Address: 0FF30H, 0FF32H, 0FF33H, 0FF37H, 0FF3CH After reset: 00H R/W

| Symbol | ⑦ | ⑥ | ⑤ | ④ | ③ | ② | ① | ⓪ |
|---|---|---|---|---|---|---|---|---|
| PU0 | 0 | 0 | PU05 | PU04 | PU03 | PU02 | PU01 | PU00 |
| PU2 | PU27 | PU26 | PU25 | PU24 | PU23 | PU22 | PU21 | PU20 |
| PU3 | PU37 | PU36 | PU35 | PU34 | PU33 | PU32 | PU31 | PU30 |
| PU7 | 0 | 0 | 0 | 0 | 0 | PU72 | PU71 | PU70 |
| PU12 | PU127 | PU126 | PU125 | PU124 | PU123 | PU122 | PU121 | PU120 |

| PUxn | Pxn pin pull-up resistor specification (x = 0: n = 0 to 5; x = 2, 3, 12: n = 0 to 7; x = 7: n = 0 to 2) |
|---|---|
| 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 |

**Caution Connecting pull-up resistors unnecessarily may increase the current consumption or latch up other devices, so specify a mode whereby pull-up resistors are only connected to the required parts. If required and not-required parts exist together, externally connect pull-up resistors to the required parts and set the mode that specifies not to connect on-chip pull-up resistors.**

**(3) Port function control register 2 (PF2)**

This register specifies N-ch open drain for pins P25 and P27.

PF2 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets PF2 to 00H.

**Caution Only the μPD784225Y Subseries incorporates PF2. When using the $I^2C$ bus mode (serial interface), make sure to specify N-ch open drain for the P25 and P27 pins.**

**Figure 5-22. Format of Port Function Control Register 2 (PF2)**

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

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PF2 | PF27 | 0 | PF25 | 0 | 0 | 0 | 0 | 0 |

| PF2n | P2n pin N-ch open drain specification (n = 5, 7) |
|---|---|
| 0 | Don't set N-ch open drain |
| 1 | Set N-ch open drain |

## 5.4 Operations

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

### 5.4.1 Writing to I/O port

**(1) Output mode**

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

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

**(2) Input mode**

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

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

**Caution In the case of 1-bit memory manipulation 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 I/O port

**(1) Output mode**

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

**(2) Input mode**

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

### 5.4.3 Operations on I/O port

**(1) Output mode**

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

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

**(2) Input mode**

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

**Caution In the case of 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 FUNCTION

## 6.1 Function

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

Since jitter-free signals can be output by using the real-time output port, this operation is ideal for the control of stepper motors, etc.

The port mode or real-time output mode can be specified in 1-bit units.

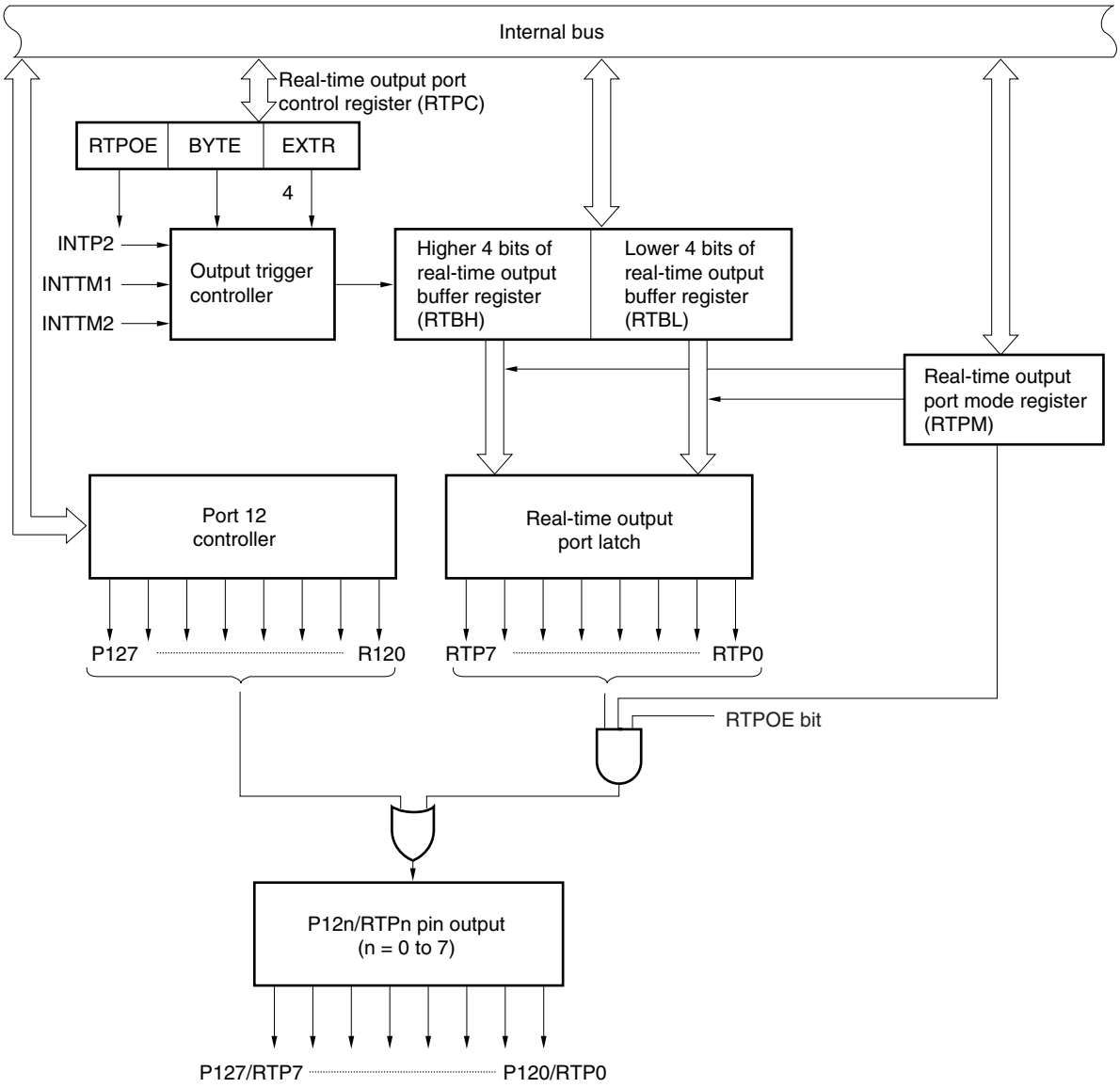
## 6.2 Configuration

The real-time output port includes the following hardware.

**Table 6-1. Configuration of Real-Time Output Port**

| Item | Configuration |
|---|---|
| Registers | Real-time output buffer registers (RTBL, RTBH) |
| Control registers | Real-time output port mode register (RTPM)<br>Real-time output port control register (RTPC) |

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



- **Real-time output buffer registers (RTBL, RTBH)**

  These 4-bit registers save the output data beforehand. RTBL and RTBH are mapped to independent addresses in the special function register (SFR) area as shown in Figure 6-2.

  When the 4 bit × 2-channel operation mode is specified, RTBL and RTBH can be independently set with data. In addition, if the addresses of both RTBL and RTBH are specified, the data in both registers can be read together.

  When the 8-bit × 1-channel operation mode is specified, writing 8-bit data to either RTBL or RTBH can set data in either register. In addition, if the addresses of either RTBL and RTBH are specified, the data in both can be read together.

  Table 6-2 lists the operations for manipulating RTBL and RTBH.

**Figure 6-2. Configuration of Real-Time Output Buffer Register**

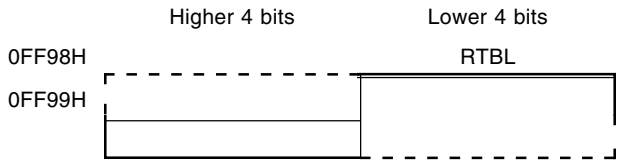



**Table 6-2. Operation for Manipulating Real-Time Output Buffer Registers**

| Operation Mode | Manipulated Register | Reading[Note 1] | | Writing[Note 2] | |
|---|---|---|---|---|---|
| | | Higher 4 Bits | Lower 4 Bits | Higher 4 Bits | Lower 4 Bits |
| 4 bits × 2 channels | RTBL | RTBH | RTBL | Invalid | RTBL |
| | RTBH | RTBH | RTBL | RTBH | Invalid |
| 8 bits × 1 channel | RTBL | RTBH | RTBL | RTBH | RTBL |
| | RTBH | RTBH | RTBL | RTBH | RTBL |

**Notes 1.** Only the bits specified in the real-time output port mode can be read. When 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 RTBL and RTBH until the real-time output trigger is generated.

## 6.3 Control Registers

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

- Real-time output port mode register (RTPM)
- Real-time output port control register (RTPC)

### (1) Real-time output port mode register (RTPM)

This register sets the real-time output port mode and port mode selection in 1-bit units.

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

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

**Figure 6-3. Format of Real-Time Output Port Mode Register (RTPM)**

Address: 0FF9AH After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RTPM | RTPM7 | RTPM6 | RTPM5 | RTPM4 | RTPM3 | RTPM2 | RTPM1 | RTPM0 |

| RTPMm | Real-time output port selection (m = 0 to 7) |
|---|---|
| 0 | Port mode |
| 1 | Real-time output mode |

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

**(2) Real-time output port control register (RTPC)**

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

Table 6-3 shows the relationship between the operation mode and output trigger of the real-time output port.

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

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

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

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

| Symbol | ⑦ | 6 | ⑤ | ④ | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RTPC | RTPOE | 0 | BYTE | EXTR | 0 | 0 | 0 | 0 |

| RTPOE | Real-time output port operation control |
|---|---|
| 0 | Operation disabled |
| 1 | Operation enabled**Note** |

| BYTE | Real-time output port operation mode |
|---|---|
| 0 | 4 bits × 2 channels |
| 1 | 8 bits × 1 channel |

| EXTR | Real-time output control by INTP2 |
|---|---|
| 0 | INTP2 not set as real-time output trigger |
| 1 | INTP2 set as real-time output trigger |

**Note** When real-time output operation is enabled (RTPOE = 1), the values of the real-time output buffer registers (RTBH and RTBL) are transferred to the real-time output port output latch.

**Caution When INTP2 is specified as an output trigger, specify the valid edge using external interrupt rising edge enable register 0 (EGP0) and external interrupt falling edge enable register 0 (EGN0).**

**Table 6-3. Operation Modes and Output Triggers of Real-Time Output Port**

| BYTE | EXTR | Operation Mode | RTBH → Port Output | RTBL → Port Output |
|---|---|---|---|---|
| 0 | 0 | 4 bits × 2 channels | INTTM2 | INTTM1 |
| 0 | 1 | | INTTM1 | INTP2 |
| 1 | 0 | 8 bits × 1 channel | INTTM1 | |
| 1 | 1 | | INTP2 | |

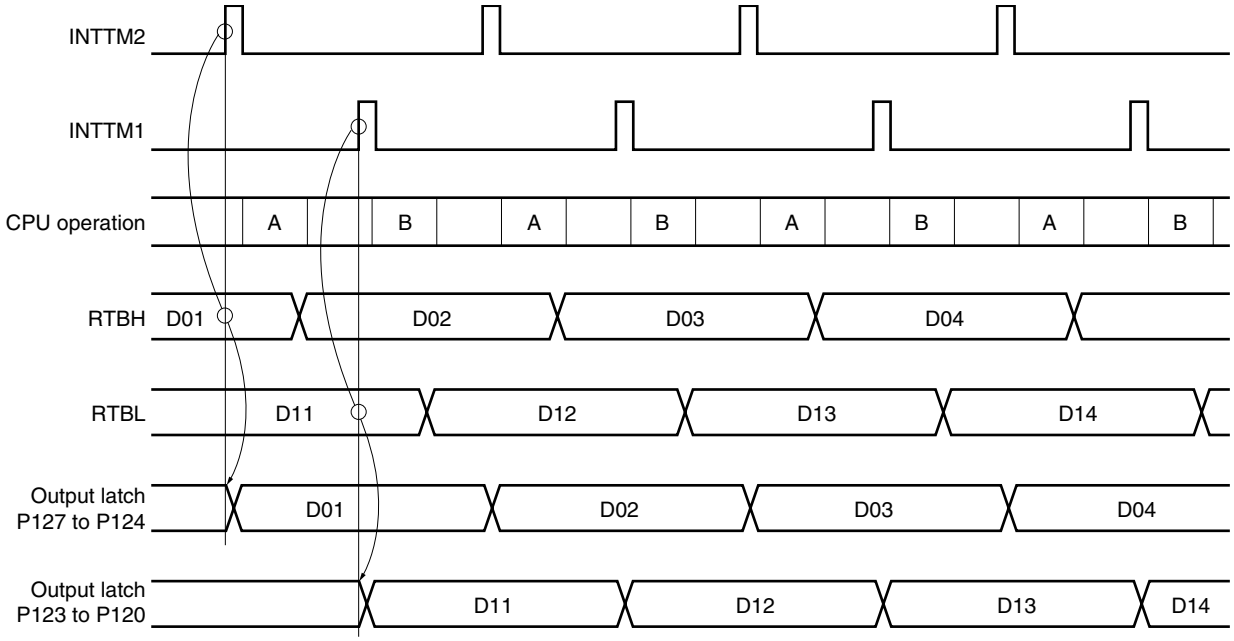
## 6.4 Operation

When real-time output is enabled by bit 7 (RTPOE) = 1 in the real-time output port control register, data in the real-time output buffer register (RTBH, RTBL) is transferred to the output latch synchronized with the generation of the selected transfer trigger (set by EXTR and BYTE[Note]). Based on the setting of the real-time output port mode register (RTPM), only the transferred data for the bits specified in the real-time output port are output from bits RTP0 to RTP7. A port pin set in the port mode by RTPM can be used as a general-purpose I/O port pin.

When the real-time output operation is disabled by RTPOE = 0, RTP0 to RTP7 output 0 regardless of the RTPM setting.

**Note** EXTR: Bit 4 of the real-time output port control register (RTPC)
BYTE: Bit 5 of the real-time output port control register (RTPC)

**Figure 6-5. Example of Operation Timing of Real-Time Output Port (EXTR = 0, BYTE = 0)**




A: Software processing by INTTM2 (RTBH write)
B: Software processing by INTTM1 (RTBL write)

## 6.5 Usage of Real-Time Output Function

(1) Disabling the real-time output operation
Set bit 7 (RTPOE) = 0 in the real-time output port control register (RTPC).

(2) Initial settings
- Set 0 in the output latch (since the output latch and real-time output are configured as a logical AND).
- Set the port to output mode.
- Set the initial value in the real-time output buffer registers (RTBH, RTBL).

(3) Enable real-time output operation.
RTPOE = 1

(4) After generating the selected transfer trigger, the RTBH and RTBL values are output from the pin. Set the next real-time output value in RTBH and RTBL by using trigger interrupt servicing, or by some other means.

(5) Subsequently, the next real-time output values are sequentially set in RTBH and RTBL by the interrupt servicing for the selected trigger.

## 6.6 Cautions

For the initial setting, set bit 7 (RTPOE) in the real-time output port control register (RTPC) to 0 to disable the real-time output operation.

# CHAPTER 7 TIMER OVERVIEW

The μPD784225/784225Y Subseries includes one on-chip 16-bit timer/event counter, two on-chip 8-bit timer/event counters, and two 8-bit timers.

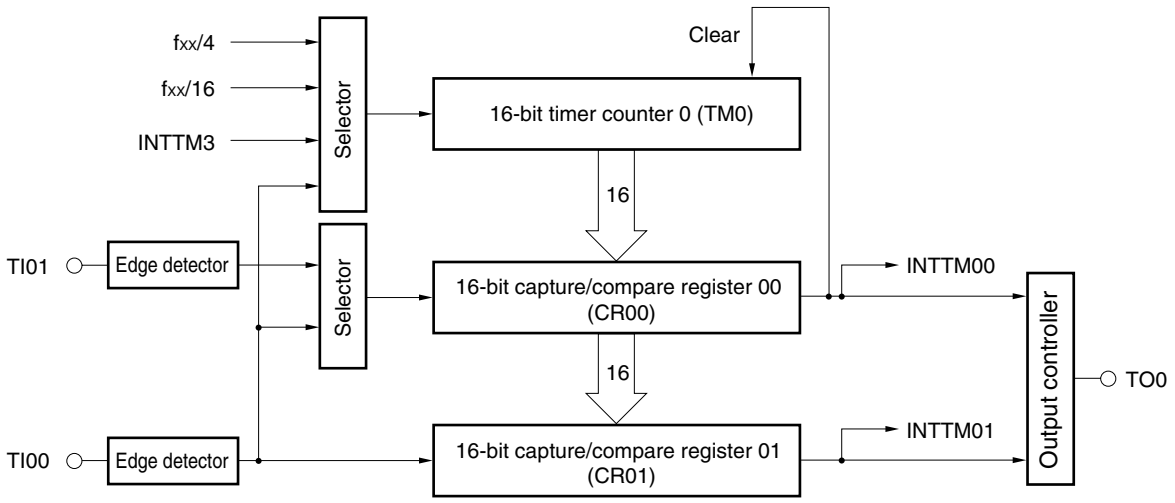
Since a total of six interrupt requests are supported, these timer/event counters can function as six timer/event counter units.

**Table 7-1. Timer Operation**

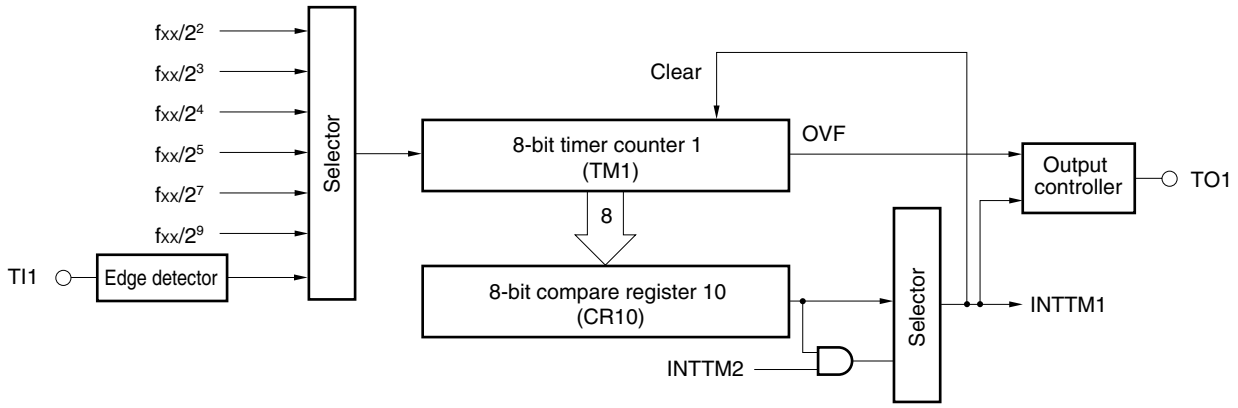
| Item \ Name | | 16-Bit Timer/ Event Counter | 8-Bit Timer/ Event Counter 1 | 8-Bit Timer/ Event Counter 2 | 8-Bit Timer 5 | 8-Bit Timer 6 |
|---|---|---|---|---|---|---|
| Count width | 8 bits | — | √ | √ | √ | √ |
| | 16 bits | √ | √ | | √ | |
| Operation mode | Interval timer | 1 ch | 1 ch | 1 ch | 1 ch | 1 ch |
| | External event counter | √ | √ | √ | — | — |
| Function | Timer output | 1 ch | 1 ch | 1 ch | — | — |
| | PPG output | √ | — | — | — | — |
| | PWM output | — | √ | √ | — | — |
| | Square-wave output | √ | √ | √ | — | — |
| | One-shot pulse output | √ | — | — | — | — |
| | Pulse width measurement | 2 inputs | — | — | — | — |
| | No. of interrupt requests | 2 | 1 | 1 | 1 | 1 |

**Figure 7-1. Block Diagram of Timer (1/2)**

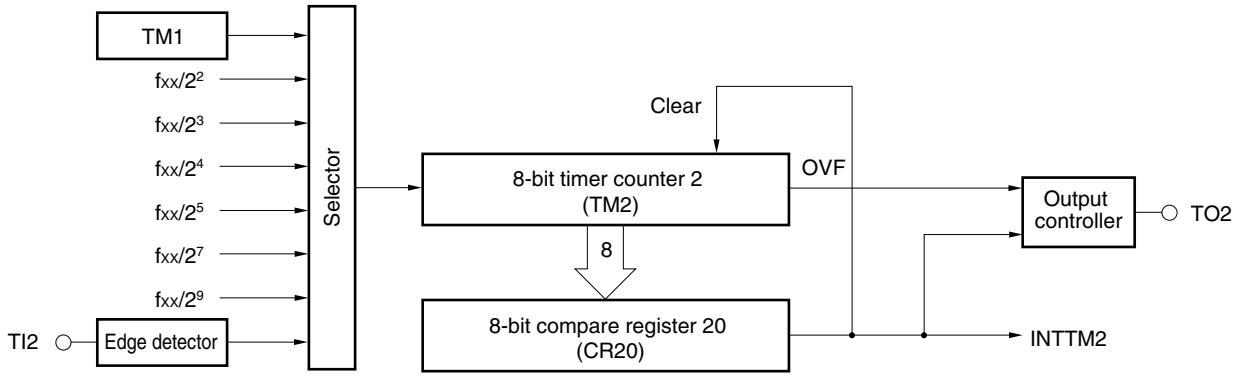
**16-bit timer/event counter**




**8-bit timer/event counter 1**



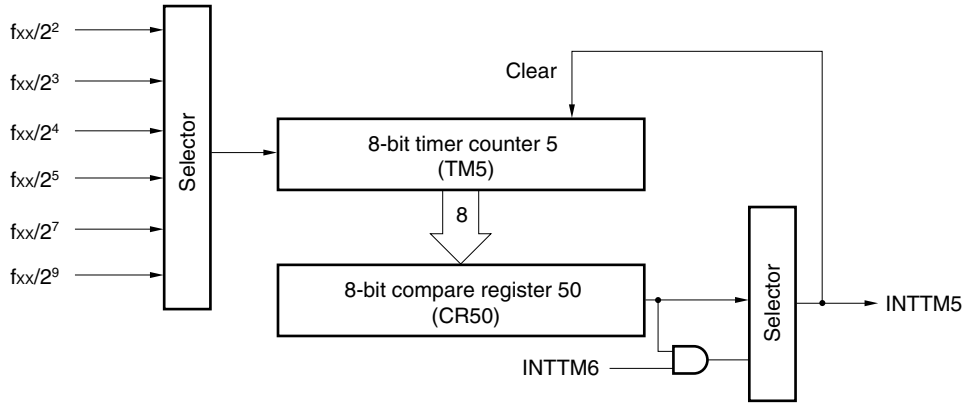

**8-bit timer/event counter 2**



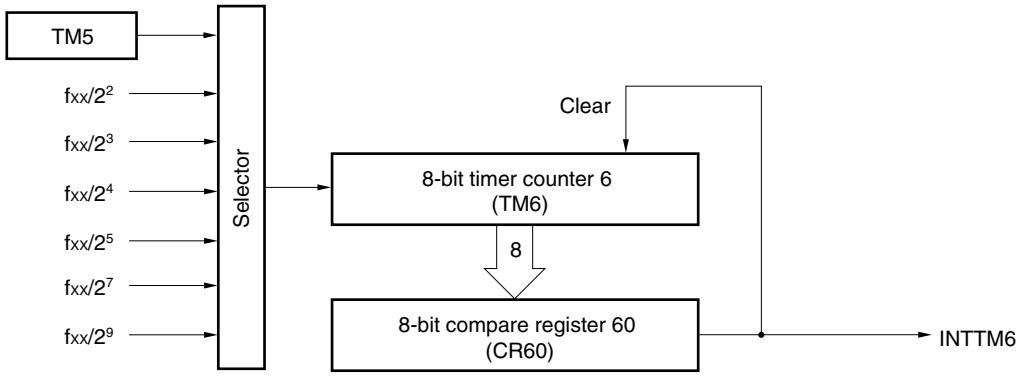

**Remark** OVF: Overflow flag

**Figure 7-1. Block Diagram of Timer (2/2)**

**8-bit timer 5**




**8-bit timer 6**



# CHAPTER 8 16-BIT TIMER/EVENT COUNTER

## 8.1 Function

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

- Interval timer
- PPG output
- Pulse width measurement
- External event counter
- Square-wave output
- One-shot pulse output

**(1) Interval timer**

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

**(2) PPG output**

The 16-bit timer/event counter can output a square wave whose frequency and output pulse width can be freely set.

**(3) Pulse width measurement**

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

**(4) External event counter**

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

**(5) Square wave output**

The 16-bit timer/event counter can output a square wave with any frequency.

**(6) One-shot pulse output**

The 16-bit timer/event counter can output a one-shot pulse with any output pulse width.

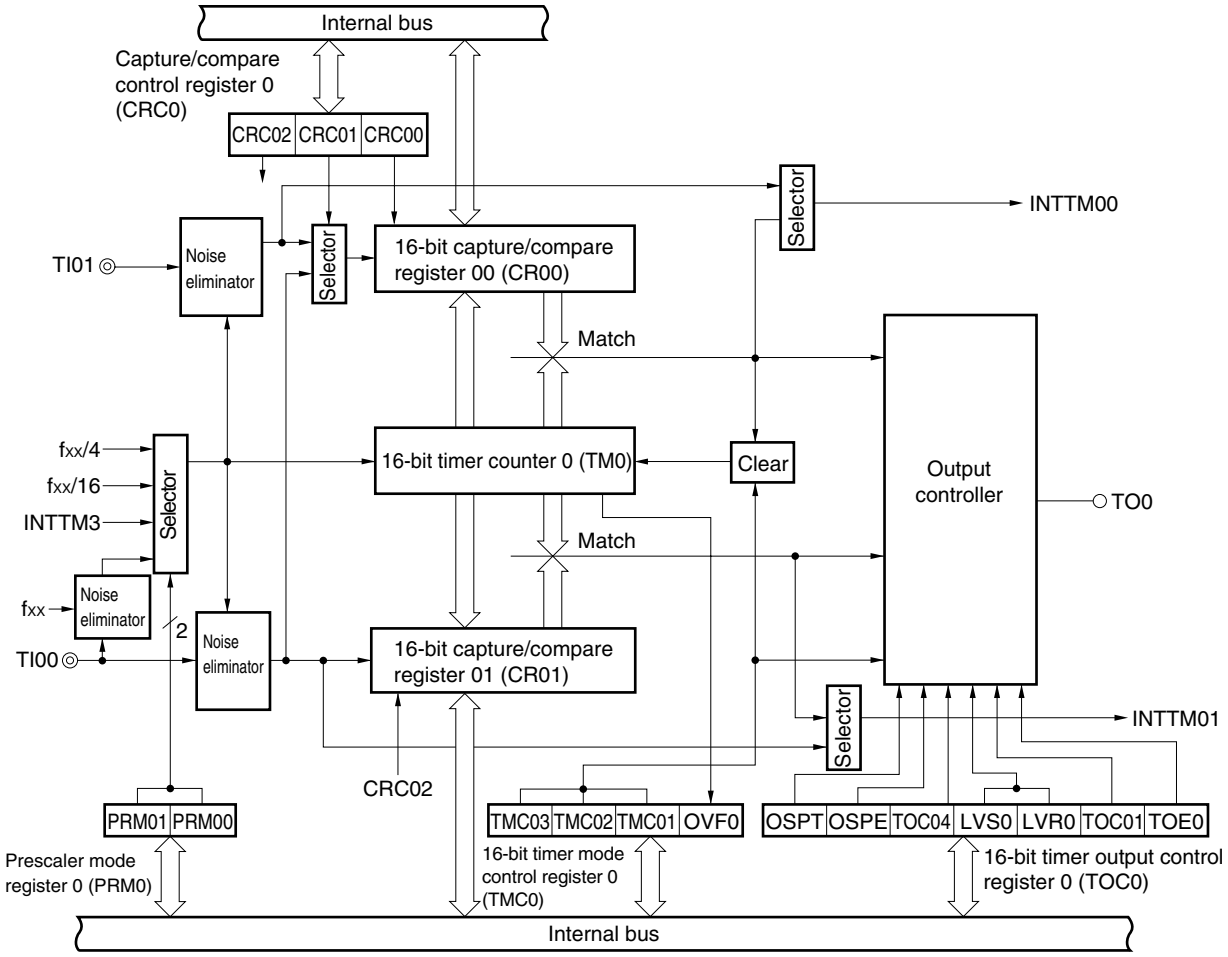
## 8.2 Configuration

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

**Table 8-1. Configuration of 16-Bit Timer/Event Counter**

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

**Figure 8-1. Block Diagram of 16-Bit Timer/Event Counter**



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

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

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

<1> $\overline{\text{RESET}}$ is input .
<2> TMC03 and TMC02 are cleared.
<3> Valid edge of TI00 is input in the clear & start mode entered by inputting valid edge of TI00.
<4> Match between TM0 and CR00 in the clear & start mode entered on match between TM0 and CR00.
<5> If bit 6 of TOC0 (OSPT) is set or the valid edge of TI00 is input in the one-shot pulse output mode.

**(2) Capture/compare register 00 (CR00)**

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

- **When using CR00 as compare register**

  The value set to CR00 is always compared with the count value of 16-bit timer counter 0 (TM0). When the values of the two match, an interrupt request (INTTM00) is generated. When TM00 is used as an interval timer, CR00 can also be used as a register that holds the interval time.

- **When using CR00 as capture register**

  The valid edge of the TI00 or TI01 pin can be selected as a capture trigger. The valid edges for TI00 and TI01 are set with prescaler mode register 0 (PRM0).

  Tables 8-2 and 8-3 show the conditions that apply when the capture trigger is specified as the valid edge of the TI00 pin and the valid edge of the TI01 pin respectively.

**Table 8-2. Valid Edge of TI00 Pin and Capture Trigger of CR00**

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

**Table 8-3. Valid Edge of TI01 Pin and Capture Trigger of CR00**

| ES11 | ES10 | Valid Edge of TI01 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 sets CR00 to 0000H.

**Caution Set any value other than 0000H in CR00. When using the register as an event counter, a one-pulse count operation is not possible.**

**(3) Capture/compare register 01 (CR01)**

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

- **When using CR01 as compare register**

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

- **When using CR01 as capture register**

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

  Table 8-4 shows the conditions that apply when the capture trigger is specified as the valid edge of the TI00 pin.

**Table 8-4. Valid Edge of TI00 Pin and Capture Trigger of CR01**

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

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

$\overline{\text{RESET}}$ input sets CR01 to 0000H.

**Caution Set any value other than 0000H in CR01. When using the register as an event counter, a one-pulse count operation is not possible.**

## 8.3 Control Registers

The following four registers control the 16-bit timer/event counter.

- 16-bit timer mode control register 0 (TMC0)
- Capture/compare control register 0 (CRC0)
- 16-bit 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 16-bit timer counter 0 (TM0).

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

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

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

**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 | TMC01 | OVF0 |

| TMC03 | TMC02 | TMC01 | Selection of operation mode/ clear mode | Selection of TO0 output timing | Generation of interrupt |
|---|---|---|---|---|---|
| 0 | 0 | 0 | Operation stop (TM0 is cleared to 0). | Not affected | Not generated. |
| 0 | 0 | 1 | | | |
| 0 | 1 | 0 | Free-running mode | Match between TM0 and CR00 or match between TM0 and CR01 | Generated on match between TM0 and CR00 and match between TM0 and CR01. |
| 0 | 1 | 1 | | Match between TM0 and CR00, match between TM0 and CR01, or valid edge of TI00 | |
| 1 | 0 | 0 | Clears and starts at valid edge of TI00. | Match between TM0 and CR00 or match between TM0 and CR01 | |
| 1 | 0 | 1 | | Match between TM0 and CR00, match between TM0 and CR01, or valid edge of TI00 | |
| 1 | 1 | 0 | Clears and starts on match between TM0 and CR00. | Match between TM0 and CR00 or match between TM0 and CR01 | |
| 1 | 1 | 1 | | Match between TM0 and CR00, match between TM0 and CR01, or valid edge of TI00 | |

| 0VF0 | Detection of overflow of 16-bit timer counter 0 |
|---|---|
| 0 | Overflow |
| 1 | No overflow |

**Cautions 1. Before changing the clear mode and TO0 output timing, be sure to stop the timer operation (reset TMC02 and TMC03 to 0, 0).**
**The valid edge of the TI00 pin is selected by using prescaler mode register 0 (PRM0).**
**2. When a mode in which the timer is cleared and started on a match between TM0 and CR00, the OVF0 flag is set to 1 when the count value of TM0 changes from FFFFH to 0000H with CR00 set to FFFFH.**
**3. The software trigger (bit 6 (OSPT) of 16-bit timer output control register 0 (TOC0) = 1) and the external trigger (TI00 input) are always valid in one-shot pulse output mode.**
**If the software trigger is used in one-shot pulse output mode, the TI00 pin cannot be used as a general-purpose port pin. Therefore, fix the TI00 pin to either high level or low level.**

**Remark** TO0: Output pin of the 16-bit timer/event counter
TI00: Input pin of the 16-bit timer/event counter
TM0: 16-bit timer counter 0
CR00: Compare register 00
CR01: Compare register 01

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

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

CRC0 is set 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: FF16H After reset: 04H R/W

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

| 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 TI01. |
| 1 | Captured in reverse phase of valid edge of TI00. |

| CRC00 | Selection of operation mode of CR00 |
|---|---|
| 0 | Operates as compare register. |
| 1 | Operates as capture register. |

**Cautions 1. Before setting CRC0, be sure to stop the timer operation.**

**2. When the mode in which the timer is cleared and started on a match between 16-bit timer counter 0 (TM0) and CR00 is selected by 16-bit timer mode control register 0 (TMC0), do not specify CR00 as a capture register.**

**(3) 16-bit timer output control register 0 (TOC0)**

This register controls the operation of the 16-bit timer/event counter output controller by setting or resetting via timer-output level software, enabling or disabling reverse output, enabling or disabling output of the 16-bit timer/event counter, enabling or disabling the one-shot pulse output operation, and selecting an output trigger for a one-shot pulse by software.

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

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

Figure 8-4 shows the format of TOC0.

**Figure 8-4. Format of 16-Bit Timer Output Control Register 0 (TOC0)**

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

| Symbol | 7 | ⑥ | ⑤ | 4 | ③ | ② | 1 | ⓪ |
|---|---|---|---|---|---|---|---|---|
| TOC0 | 0 | OSPT | OSPE | TOC04 | LVS0 | LVR0 | TOC01 | TOE0 |

| OSPT | Output trigger control of one-shot pulse by software |
|---|---|
| 0 | One-shot pulse output disabled |
| 1 | One-shot pulse output enabled |

| OSPE | Controls of one-shot pulse output operation |
|---|---|
| 0 | Successive pulse output |
| 1 | One-shot pulse output |

| TOC04 | Timer output control on match between CR01 and TM0 |
|---|---|
| 0 | Inversion disabled |
| 1 | Inversion enabled |

| LVS0 | LVR0 | Timer output control by software |
|---|---|---|
| 0 | 0 | Not affected |
| 0 | 1 | Reset (0) |
| 1 | 0 | Set (1) |
| 1 | 1 | Setting prohibited |

| TOC01 | Timer output control on match between CR00 and TM0 and valid edge of TI00 |
|---|---|
| 0 | Inversion disabled |
| 1 | Inversion enabled |

| TOE0 | Output control of 16-bit timer/event counter |
|---|---|
| 0 | Output disabled (output is fixed to 0 level) |
| 1 | Output enabled |

**Cautions 1. Before setting TOC0, be sure to stop the timer operation.**

**2. LVS0 and LVR0 are 0 when read after data has been set to them.**

**3. OSPT is 0 when read because it is automatically cleared after data has been set.**

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

This register selects the count clock of the 16-bit timer/event counter and the valid edge of the TI00 and TI01 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 TI01 |
|---|---|---|
| 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 TI00 |
|---|---|---|
| 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 | $f_{XX}/4$ (3.13 MHz) |
| 0 | 1 | $f_{XX}/16$ (781 kHz) |
| 1 | 0 | INTTM3 (timer output for clock) |
| 1 | 1 | Valid edge of TI00 |

**Caution When selecting the valid edge of TI00 as the count clock, do not specify the valid edge of TI00 to clear and start the timer and as a capture trigger. Also, ensure that the count clock frequency is $f_{XX}/4$ or lower.**

**Remark** Figures in parentheses apply to operation at $f_{XX}$ = 12.5 MHz

## 8.4 Operation

### 8.4.1 Operation as interval timer (16 bits)

The 16-bit timer/counter operates as an interval timer when 16-bit timer mode control register 0 (TMC0) and capture/compare control register 0 (CRC0) are set as shown in Figure 8-6.

In this case, the 16-bit timer/event counter repeatedly generates an interrupt at the time interval specified by the count value set in advance to 16-bit capture/compare register 00 (CR00).

When the count value of 16-bit timer counter 0 (TM0) matches the set value of CR00, the value of TM0 is cleared to 0, and the timer continues counting. At the same time, an interrupt request signal (INTTM00) is generated.

The count clock of the 16-bit timer/event counter can be selected by bits 0 and 1 (PRM00 and PRM01) of prescaler mode register 0 (PRM0).

**Figure 8-6. Control Register Settings When Timer 0 Operates as Interval Timer**

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

| | | | | | TMC03 | TMC02 | TMC01 | OVF0 |
|---|---|---|---|---|---|---|---|---|
| TMC0 | 0 | 0 | 0 | 0 | 1 | 1 | 0/1 | 0 |

Clears and starts on match between TM0 and CR00.

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

| | | | | | | CRC02 | CRC01 | CRC00 |
|---|---|---|---|---|---|---|---|---|
| CRC0 | 0 | 0 | 0 | 0 | 0 | 0/1 | 0/1 | 0 |

CR00 as compare register

**Remark** 0/1: When these bits are reset to 0 or set to 1, other functions can be used together with the interval timer function. For details, refer to **Figures 8-2** and **8-3**.

**Figure 8-7. Configuration of Interval Timer**

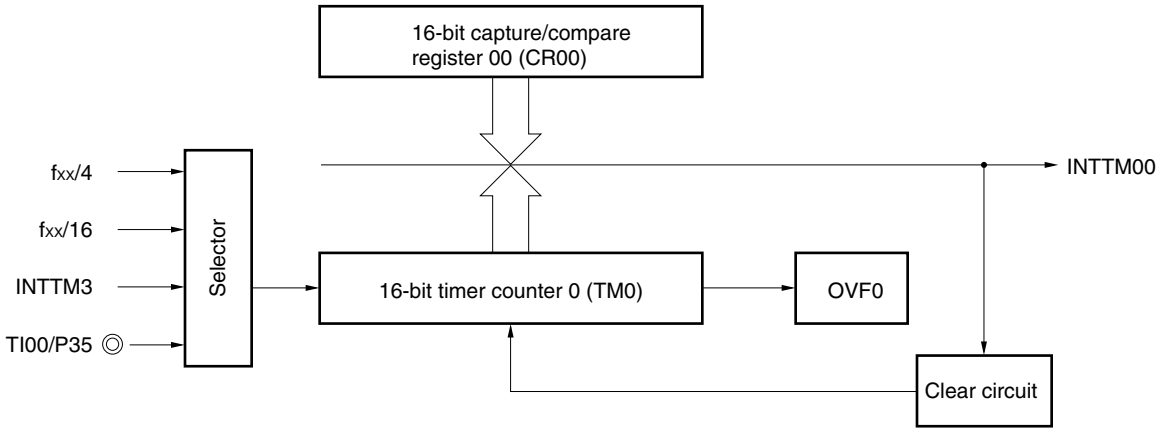
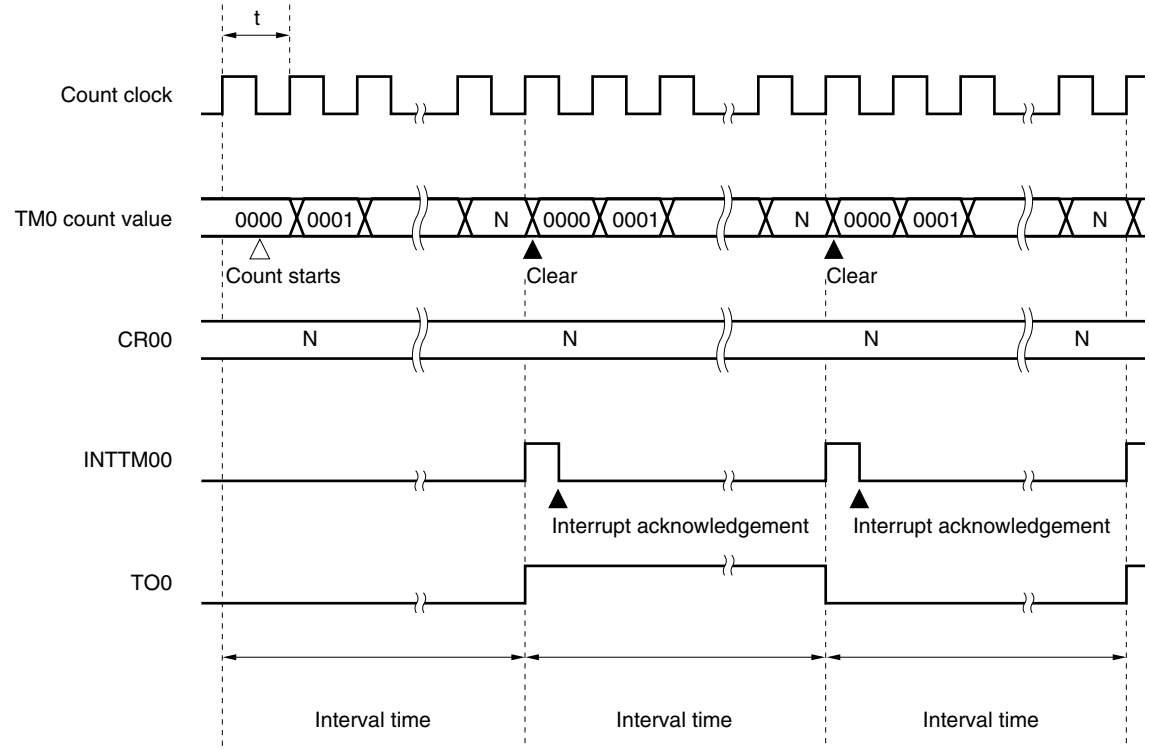



**Figure 8-8. Timing of Interval Timer Operation**




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

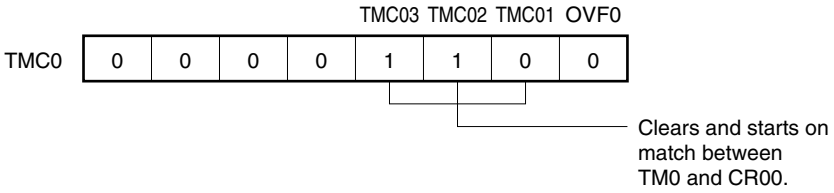
### 8.4.2 PPG output operation

The 16-bit timer/event counter can be used for PPG (Programmable Pulse Generator) output by setting 16-bit timer mode control register 0 (TMC0) and capture/compare control register 0 (CRC0) as shown in Figure 8-9.

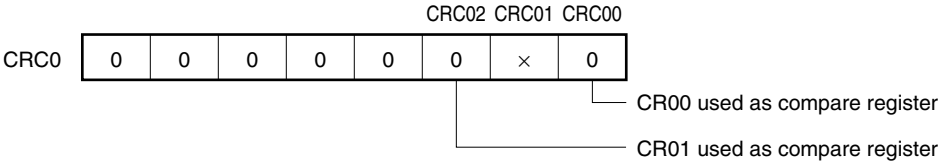
The PPG output function outputs a rectangular wave with a cycle specified by the count value set in advance to 16-bit capture/compare register 00 (CR00) and a pulse width specified by the count value set in advance to 16-bit capture/compare register 01 (CR01).

**Figure 8-9. Control Register Settings in PPG Output Operation**

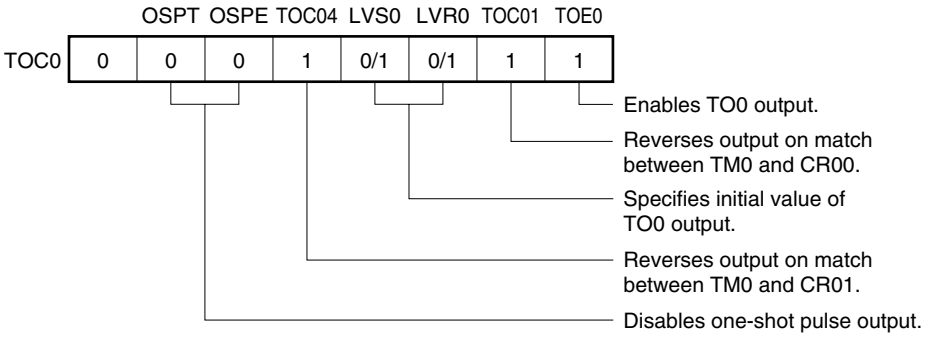
**(a) 16-bit timer mode control register 0 (TMC0)**




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




**(c) 16-bit timer output control register 0 (TOC0)**




**Remark** ×: Don't care

**Caution Make sure that CR00 and CR01 are set to 0000H ≤ CR01 < CR00 ≤ FFFFH.**

### 8.4.3 Pulse width measurement

16-bit timer counter 0 (TM0) can be used to measure the pulse widths of the signals input to the TI00/P35 and TI01/P36 pins.

Measurement can be carried out with TM0 used as a free-running counter or by restarting the timer in synchronization with the edge of the signal input to the TI00/P35 pin.

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

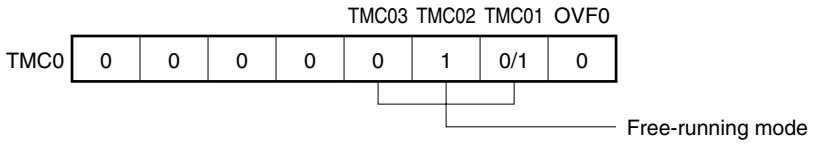
If the edge specified by prescaler mode register 0 (PRM0) is input to the TI00/P35 pin when 16-bit timer counter 0 (TM0) is used as a free-running counter (refer to **Figure 8-10**), the value of TM0 is loaded to 16-bit capture/compare register 01 (CR01), and an external interrupt request signal (INTTM01) is set.

The edge is specified by using bits 4 and 5 (ES00 and ES01) of prescaler mode register 0 (PRM0). The rising edge, falling edge, or both the rising and falling edges can be selected.

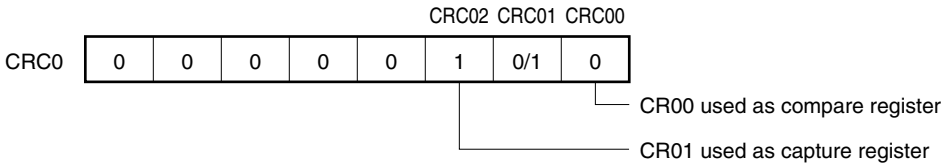
The valid edge is detected through sampling at the count clock cycle selected by prescaler mode register 0 (PRM0), and the capture operation is not performed until the valid level is detected twice. Therefore, noise with a short pulse width can be eliminated.

**Figure 8-10. Control Register Settings for Pulse Width Measurement with Free-Running Counter and One Capture Register**

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




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




**Remark** 0/1: When these bits are reset to 0 or set to 1, other functions can be used together with the pulse width measurement function. For details, refer to **Figures 8-2** and **8-3**.

**Figure 8-11. Configuration for Pulse Width Measurement with Free-Running Counter**

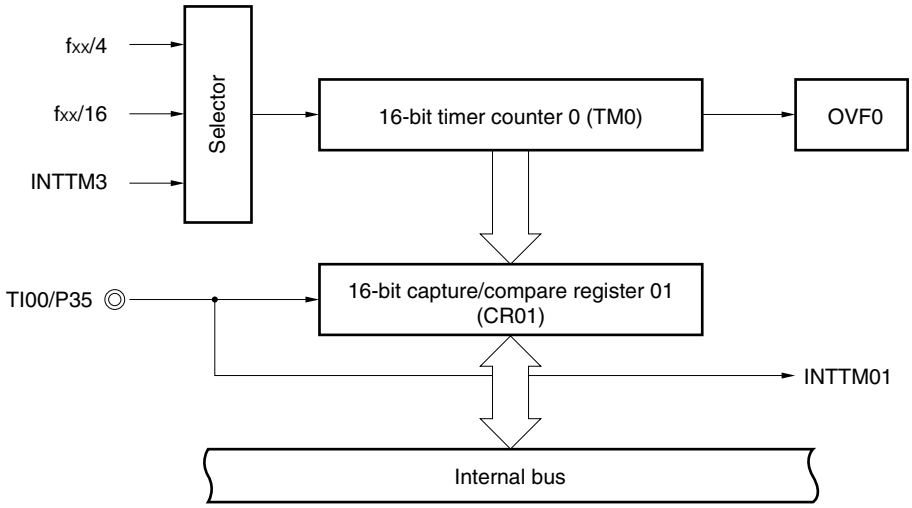
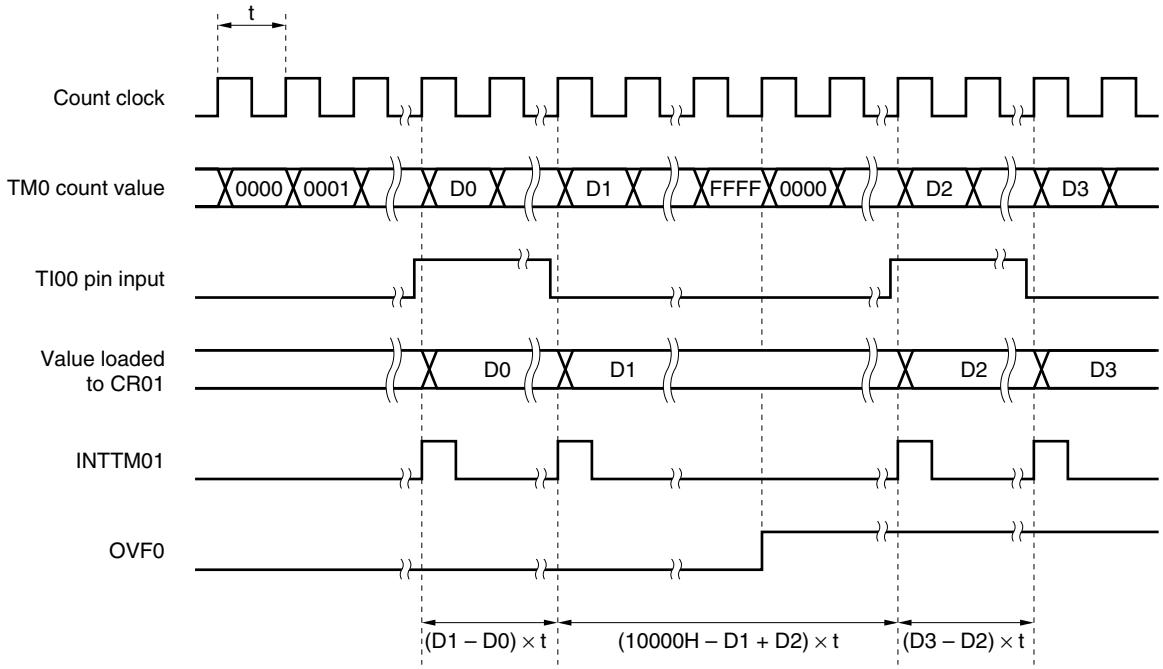



**Figure 8-12. Timing of Pulse Width Measurement with Free-Running Counter and One Capture Register (with Both Edges Specified)**




**Caution For simplification purposes, delay due to noise elimination is not taken into consideration in the capture operation by TI00 pin input and in the interrupt request generation timing in the above figure. For a more accurate picture, refer to Figure 8-14 CR01 Capture Operation with Rising Edge Specified.**

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

The pulse widths of the two signals respectively input to the TI00/P35 and TI01/P36 pins can be measured when 16-bit timer counter 0 (TM0) is used as a free-running counter (refer to **Figure 8-13**).

When the edge specified by bits 4 and 5 (ES00 and ES01) of prescaler mode register 0 (PRM0) is input to the TI00/P35 pin, the value of the TM0 is loaded to 16-bit capture/compare register 01 (CR01) and an external interrupt request signal (INTTM01) is set.

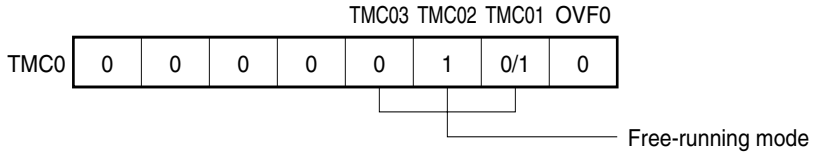
When the edge specified by bits 6 and 7 (ES10 and ES11) is input to the TI01/P36 pin, the value of TM0 is loaded to 16-bit capture/compare register 00 (CR00), and an external interrupt request signal (INTTM00) is set.

The rising, falling, or both rising and falling edges can be specified for the TI00/P35 and TI01/P36 pins.

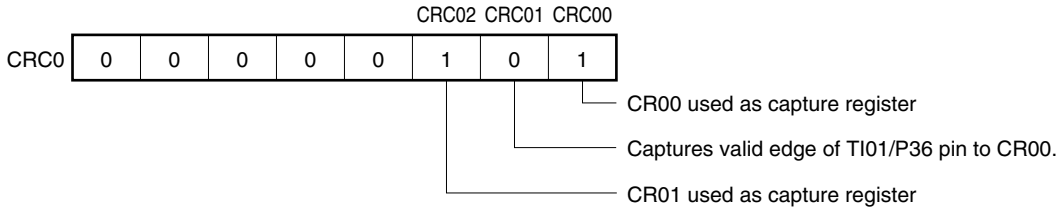
The valid edge of TI00/P35 pin and TI01/P36 pin is detected through sampling at the count clock cycle selected by prescaler mode register 0 (PRM0), and the capture operation is not performed until the valid level is detected twice. Therefore, noise with a short pulse width can be eliminated.

**Figure 8-13. Control Register Settings for Measurement of Two Pulse Widths with Free-Running Counter**

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




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




**Remark** 0/1: When these bits are reset to 0 or set to 1, other functions can be used together with the pulse width measurement function. For details, refer to **Figures 8-2** and **8-3**.

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

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

**Figure 8-14. CR01 Capture Operation with Rising Edge Specified**

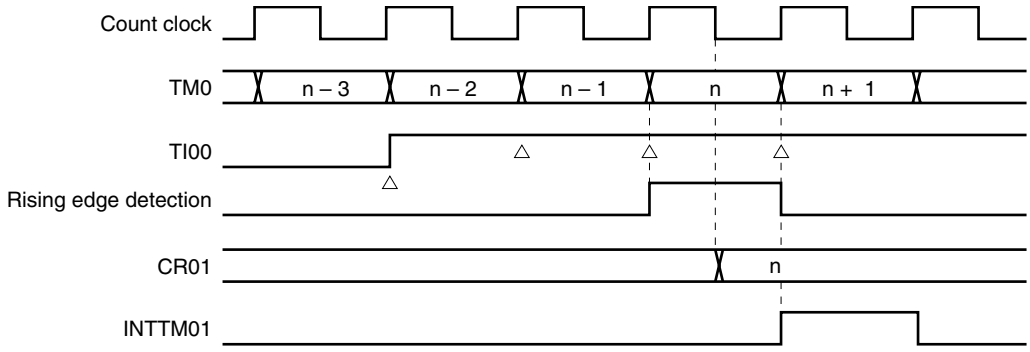
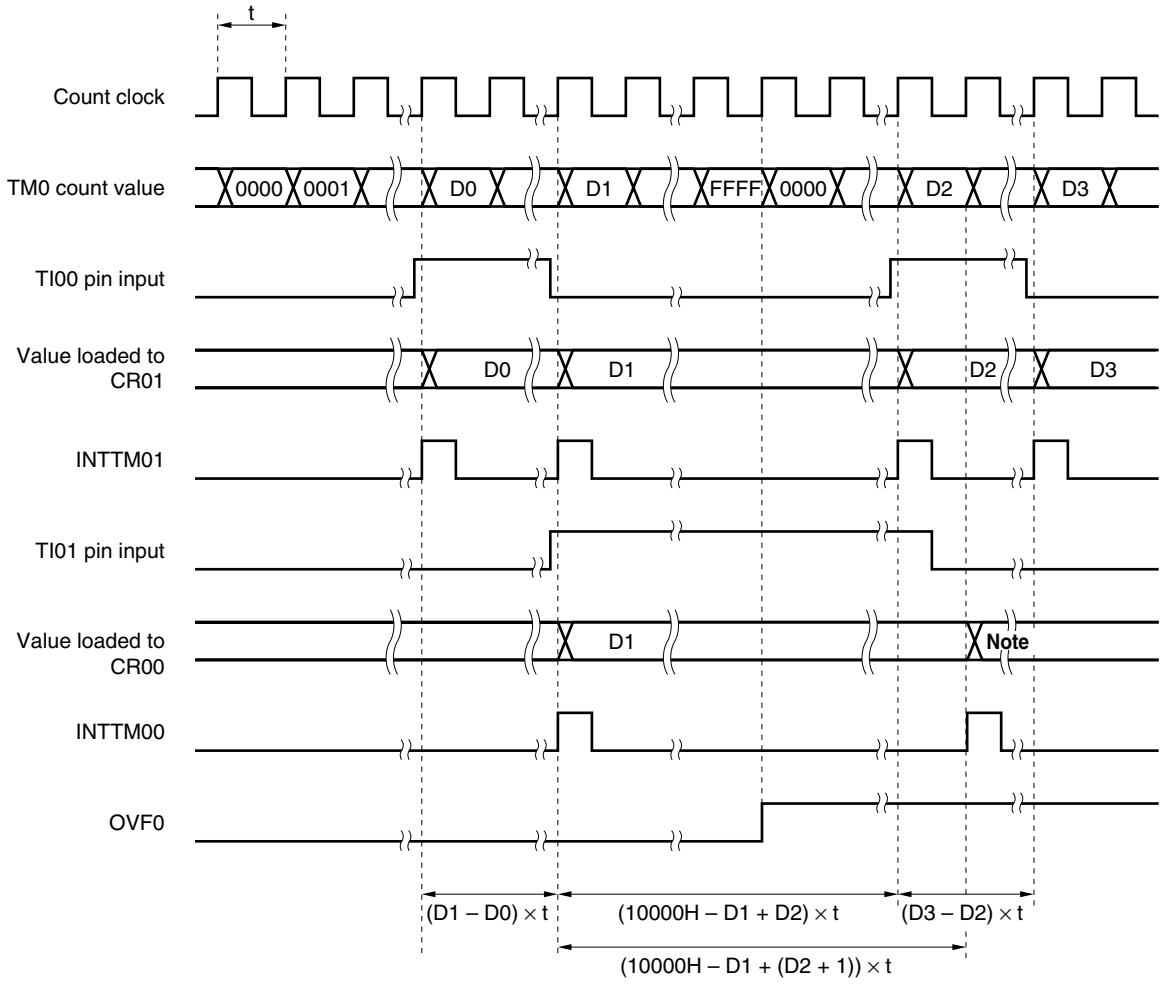



**Figure 8-15. Timing of Pulse Width Measurement with Free-Running Counter (with Both Edges Specified)**




**Note** D2 + 1

**Caution For simplification purposes, delay due to noise elimination is not taken into consideration in the capture operation by TI00 and TI01 pin input and in the interrupt request generation timing in the above figure. For a more accurate picture, refer to Figure 8-14 CR01 Capture Operation with Rising Edge Specified.**

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

When 16-bit timer counter 0 (TM0) is used as a free-running counter (refer to **Figure 8-16**), the pulse width of the signal input to the TI00/P35 pin can be measured.

When the edge specified by bits 4 and 5 (ES00 and ES01) of prescaler mode register 0 (PRM0) is input to the TI00/P35 pin, the value of TM0 is loaded to 16-bit capture/compare register 01 (CR01), and an external interrupt request signal (INTTM01) is set.

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

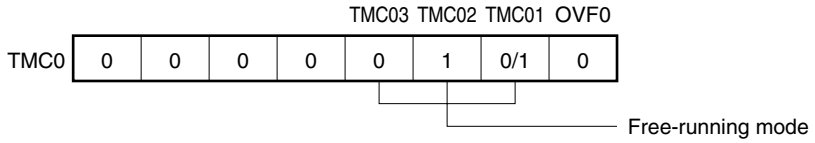
The edge of the TI00/P35 pin is specified by bits 4 and 5 (ES00 and ES01). The rising or falling edge can be specified.

The valid edge of TI00/P35 pin is detected through sampling at the count clock cycle selected by prescaler mode register 0 (PRM0), and the capture operation is not performed until the valid level is detected twice. Therefore, noise with a short pulse width can be eliminated.

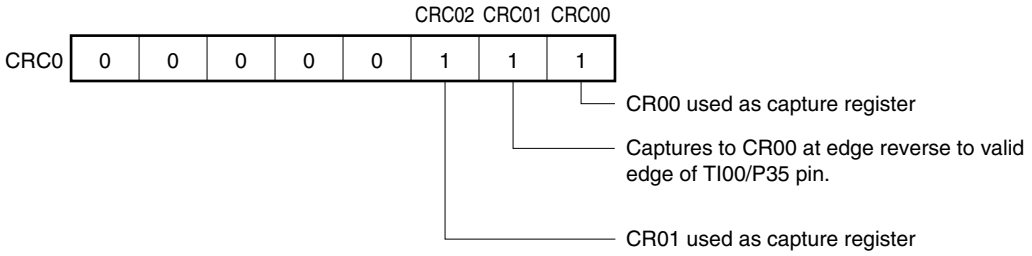
**Caution If the valid edge of the TI00/P35 pin is specified to be both the rising and falling edges, capture/compare register 00 (CR00) cannot perform its capture operation.**

**Figure 8-16. Control Register Settings for Pulse Width Measurement with Free-Running Counter and Two Capture Registers**

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

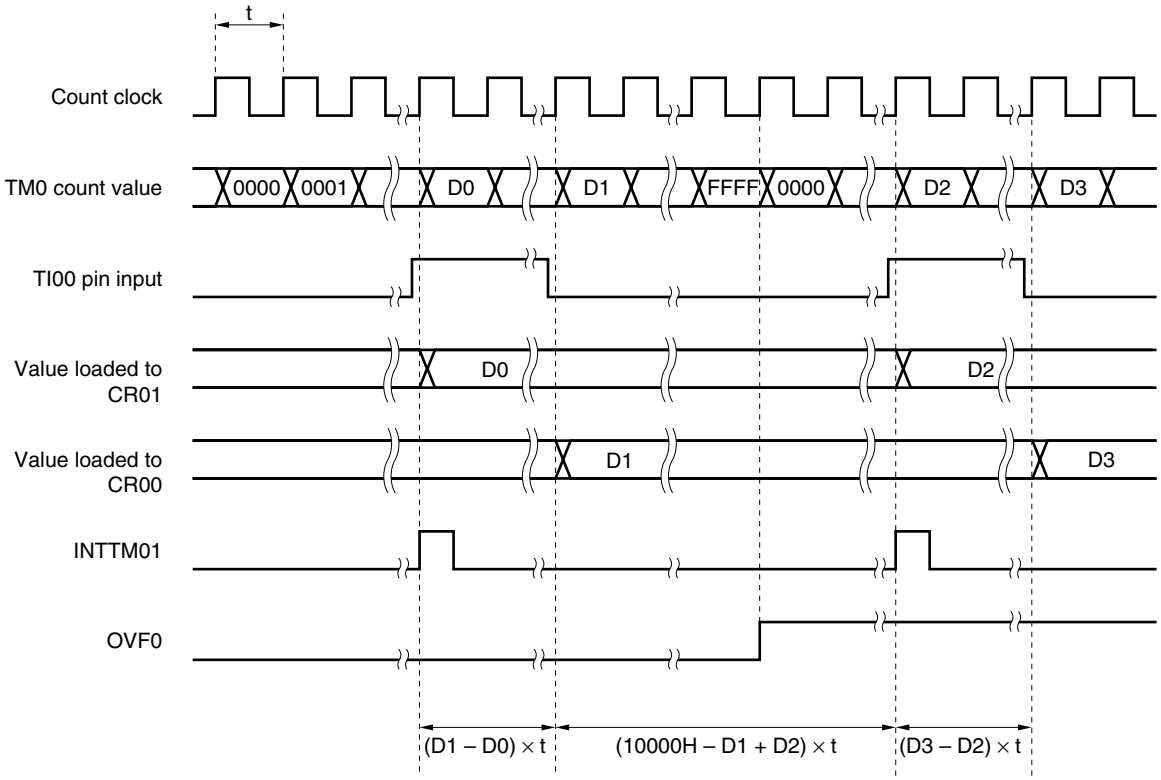



**(b) Capture/compare control register 0 (CRC0)**




**Remark** 0/1: When these bits are reset to 0 or set to 1, other functions can be used together with the pulse width measurement function. For details, refer to **Figures 8-2** and **8-3**.

**Figure 8-17. Timing of Pulse Width Measurement with Free-Running Counter and Two Capture Registers (with Rising Edge Specified)**




**Caution For simplification purposes, delay due to noise elimination is not taken into consideration in the capture operation by TI00 pin input and in the interrupt request generation timing in the above figure. For a more accurate picture, refer to Figure 8-14 CR01 Capture Operation with Rising Edge Specified.**

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

When the valid edge of the TI00/P35 pin is detected, the pulse width of the signal input to the TI00/P35 pin can be measured by clearing 16-bit timer counter 0 (TM0) once and then resuming counting after loading the count value of TM0 to 16-bit capture/compare register 01 (CR01) (Refer to **Figure 8-18**).

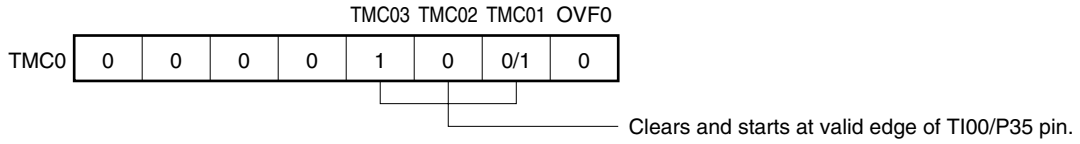
The edge of the TI00/P35 pin is specified by bits 4 and 5 (ES00 and ES01) of prescaler mode register 0 (PRM0). The rising or falling edge can be specified.

The valid edge is detected through sampling at the count clock cycle selected by prescaler mode register 0 (PRM0), and the capture operation is not performed until the valid level is detected twice. Therefore, noise with a short pulse width can be eliminated.

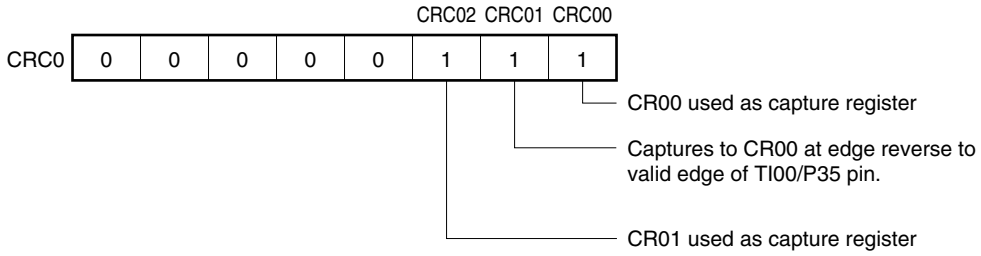
**Caution If the valid edge of the TI00/P35 pin is specified to be both the rising and falling edges, capture/compare register 00 (CR00) cannot perform its capture operation.**

**Figure 8-18. Control Register Settings for Pulse Width Measurement by Restarting**

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

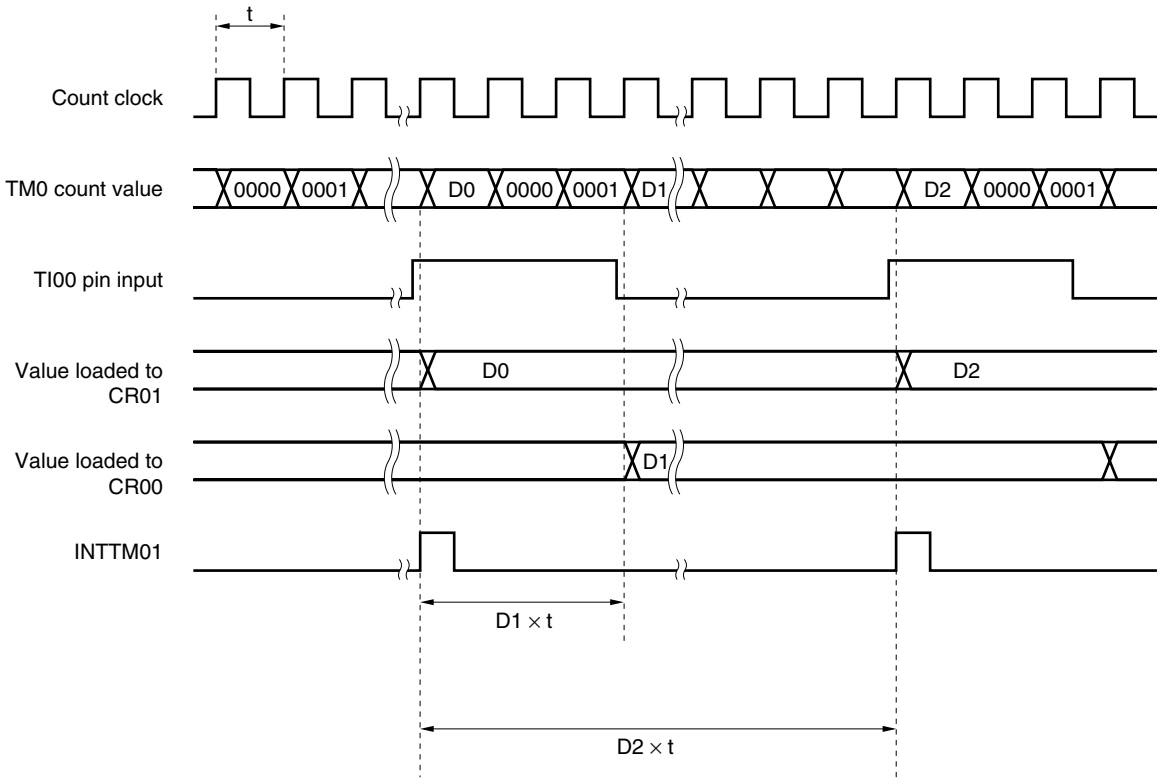



**(b) Capture/compare control register 0 (CRC0)**




**Remark** 0/1: When these bits are reset to 0 or set to 1, other functions can be used together with the pulse measurement function. For details, refer to **Figures 8-2** and **8-3**.

**Figure 8-19. Timing of Pulse Width Measurement by Restarting (with Rising Edge Specified)**




**Caution For simplification purposes, delay due to noise elimination is not taken into consideration in the capture operation by TI00 pin input and in the interrupt request generation timing in the above figure. For a more accurate picture, refer to Figure 8-14 CR01 Capture Operation with Rising Edge Specified.**

### 8.4.4 Operation as external event counter

The 16-bit timer/event counter can be used as an external event counter which counts the number of clock pulses input to the TI00/P35 pin from an external source by using 16-bit timer counter 0 (TM0).

Each time the valid edge specified by prescaler mode register 0 (PRM0) is input to the TI00/P35 pin, TM0 is incremented.

To perform a count operation using the TI00/P35 pin input clock, specify the TI00 valid edge with bits 0 and 1 of PRM0 (PRM00, PRM01).

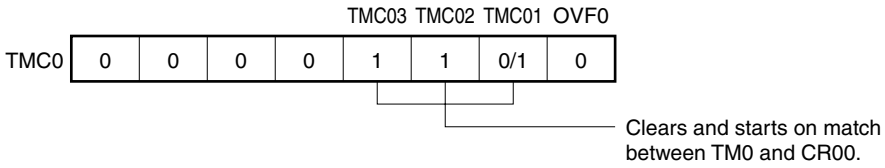
Set CR00 to a value other than 0000H (one-pulse count operation is not possible).

The edge of the TI00/P35 pin is specified by bits 4 and 5 (ES00 and ES01) of prescaler mode register 0 (PRM0). The rising, falling, or both the rising and falling edges can be specified.

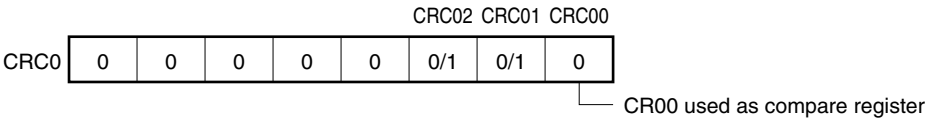
When using the TI00 pin input as the count clock, sampling for valid edge detection is locked by the main system clock ($f_{XX}$) and the capture operation is not performed until the valid level is detected twice. Therefore, noise with a short pulse width can be eliminated.

**Figure 8-20. Control Register Settings in External Event Counter Mode**

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




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




**Remark** 0/1: When these bits are reset to 0 or set to 1, other functions can be used together with the external event counter function. For details, refer to **Figures 8-2** and **8-3.**

**Figure 8-21. Configuration of External Event Counter**

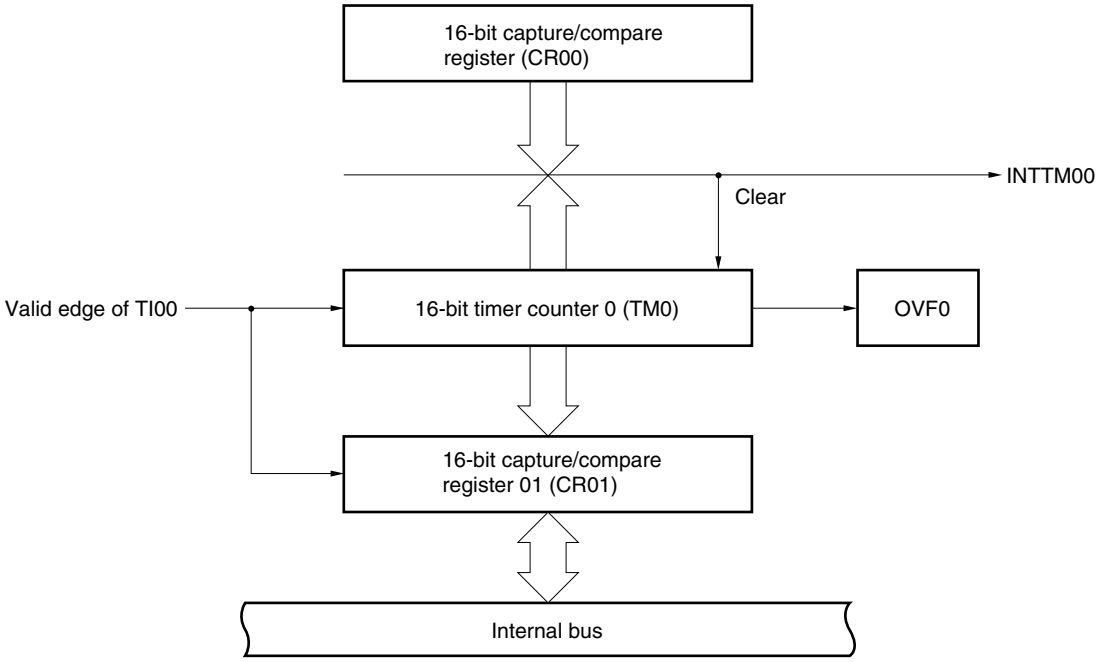
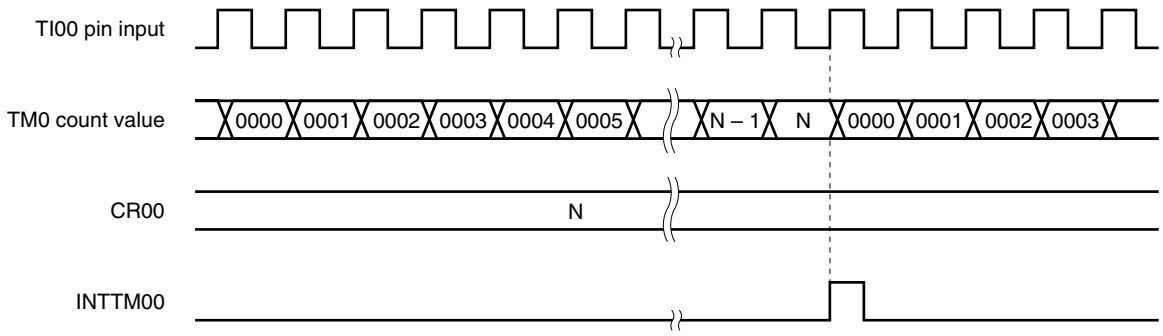


**Figure 8-22. Timing of External Event Counter Operation (with Rising Edge Specified)**




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

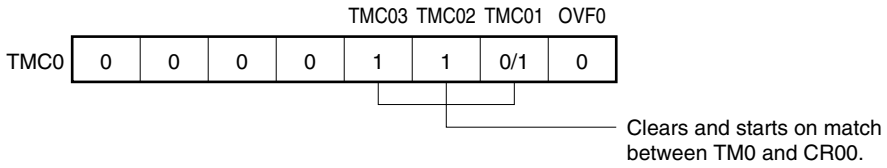
### 8.4.5 Operation to output square wave

The 16-bit timer/event counter outputs a square wave of any frequency at the interval specified by the count value preset to 16-bit capture/compare register 00 (CR00).

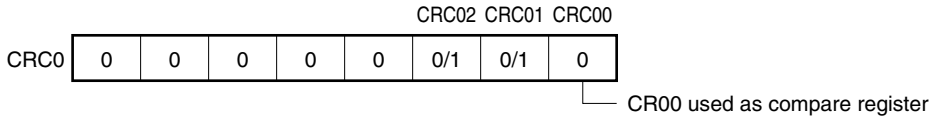
By setting bits 0 (TOE0) and 1 (TOC01) of 16-bit timer output control register 0 (TOC0) to 1, the output status of the TO0/P30 pin is inverted at the interval specified by the count value preset to CR00. In this way, a square wave of any frequency can be output.

**Figure 8-23. Control Register Settings in Square Wave Output Mode**

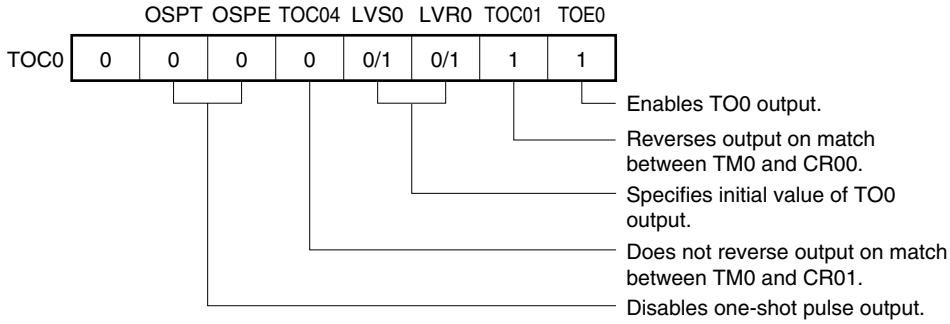
**(a) 16-bit timer mode control register 0 (TMC0)**




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

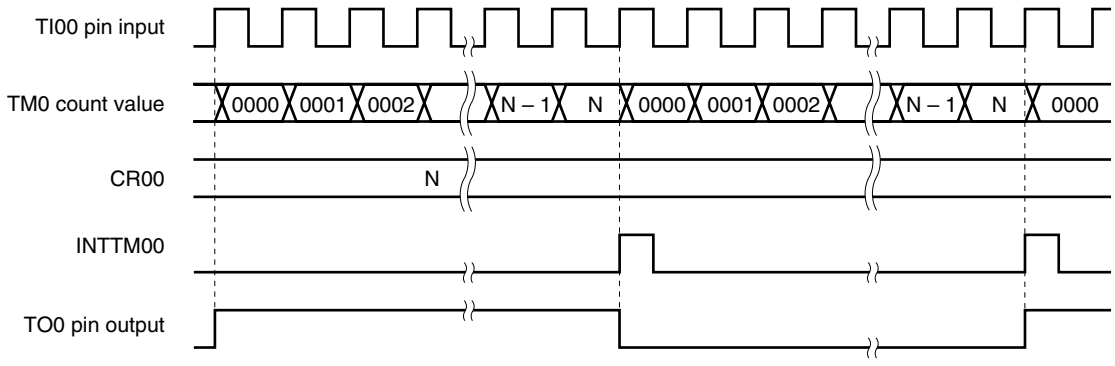



**(c) 16-bit timer output control register 0 (TOC0)**




**Remark** 0/1: When these bits are reset to 0 or set to 1, other functions can be used together with the square-wave output function. For details, refer to **Figures 8-2** to **8-4**.

**Figure 8-24. Timing of Square Wave Output Operation**



### 8.4.6 Operation to output one-shot pulse

The 16-bit timer/event counter can output a one-shot pulse in synchronization with a software trigger and an external trigger (TI00/P35 pin input).

**(1) One-shot pulse output with software trigger**

A one-shot pulse can be output from the TO0/P30 pin by setting 16-bit timer mode control register 0 (TMC0), capture/compare control register 0 (CRC0), and 16-bit timer output control register 0 (TOC0) as shown in Figure 8-25, and by setting bit 6 (OSPT) of TOC0 by software.

By setting OSPT to 1, the 16-bit timer/event counter is cleared and started, and its output is asserted at the count value set in advance to 16-bit capture/compare register 01 (CR01). After that, the output is deasserted at the count value set in advance to 16-bit capture/compare register 00 (CR00).

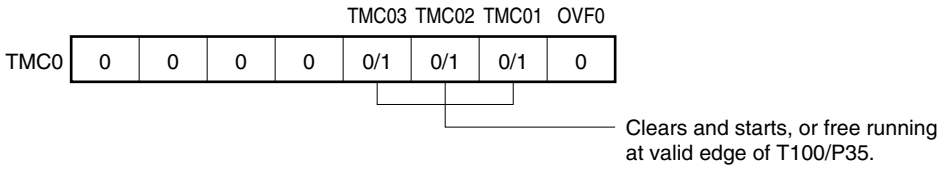
Even after the one-shot pulse has been output, TM0 continues its operation. To stop TM0, TMC0 must be reset to 00H.

**Cautions 1. Do not set OSPT to 1 while the one-shot pulse is being output. To output the one-shot pulse again, wait until INTTM00, which occurs on a match between TM0 and CR00, occurs.**

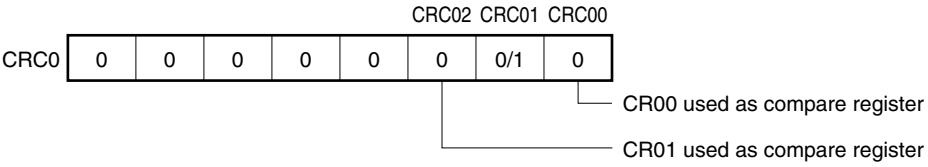
**2. The software trigger (bit 6 (OSPT) of 16-bit timer output control register 0 (TOC0) = 1) and the external trigger (TI00 input) are always valid in one-shot pulse output mode.**
**If the software trigger is used in one-shot pulse output mode, the TI00 pin cannot be used as a general-purpose port pin. Therefore, fix the TI00 pin to either high level or low level.**

**Figure 8-25. Control Register Settings for One-Shot Pulse Output by Software Trigger**

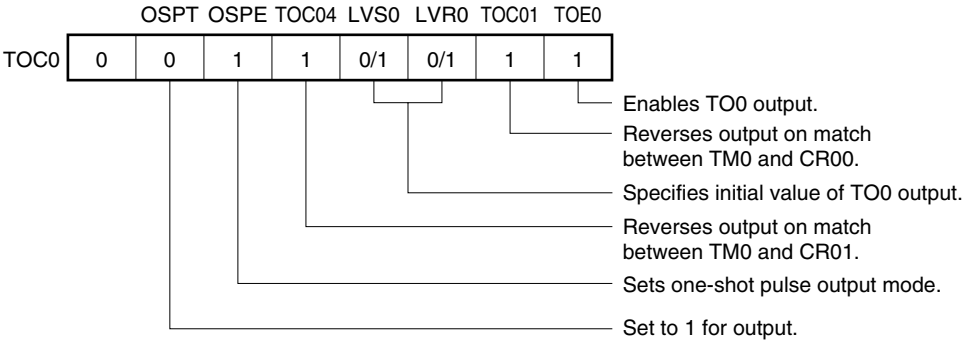
**(a) 16-bit timer mode control register 0 (TMC0)**




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



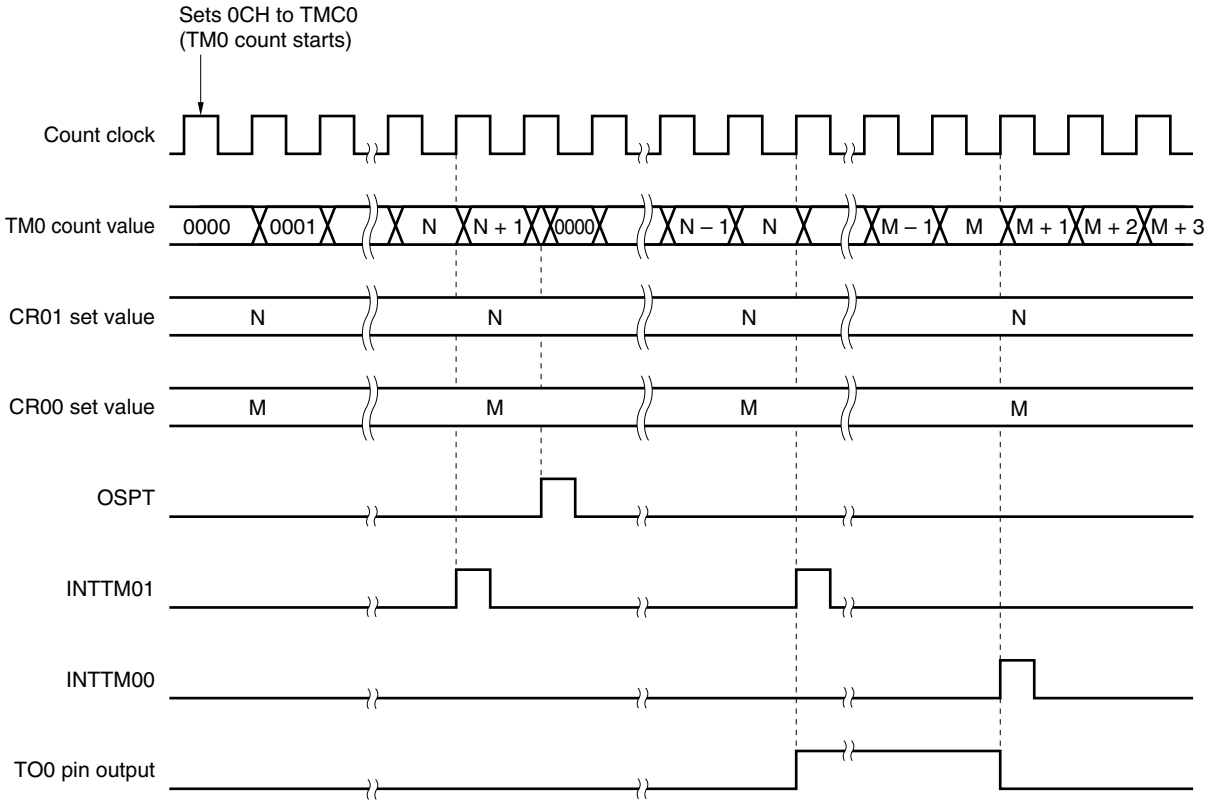

**(c) 16-bit timer output control register 0 (TOC0)**




**Caution Set CR00 and CR01 to a value in the following range.**
**0000H < CR01 < CR00 ≤ FFFFH**

**Remark** 0/1: When these bits are reset to 0 or set to 1, other functions can be used together with the one-shot pulse output function. For details, refer to **Figures 8-2** to **8-4**.

**Figure 8-26. Timing of One-Shot Pulse Output Operation by Software Trigger**




**Cautions 1. 16-bit timer counter 0 starts operating as soon as TMC02 and TMC03 are set to a value other than 0, 0 (operation stop mode).**

**2. The software trigger (bit 6 (OSPT) of 16-bit timer output control register 0 (TOC0) = 1) and the external trigger (TI00 input) are always valid in one-shot pulse output mode.**
**If the software trigger is used in one-shot pulse output mode, the TI00 pin cannot be used as a general-purpose port pin. Therefore, fix the TI00 pin to either high level or low level.**

**(2) One-shot pulse output with external trigger**

A one-shot pulse can be output from the TO0/P30 pin by setting 16-bit timer mode control register 0 (TMC0), capture/compare control register 0 (CRC0), and 16-bit timer output control register 0 (TOC0) as shown in Figure 8-27, and by using the valid edge of the TI00/P35 pin as an external trigger.

The valid edge of the TI00/P35 pin is specified by bits 4 and 5 (ES00 and ES01) of prescaler mode register 0 (PRM0). The rising, falling, or both the rising and falling edges can be specified.

When the valid edge of the TI00/P35 pin is detected, the 16-bit timer/event counter is cleared and started, and the output is asserted at the count value set in advance to 16-bit capture/compare register 01 (CR01).

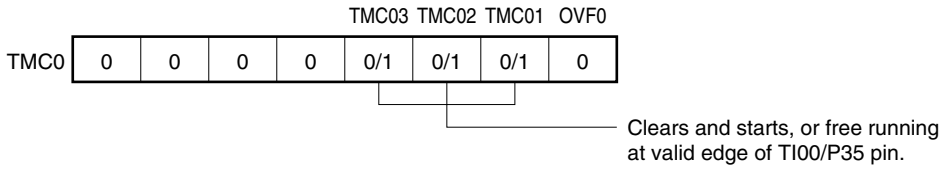
After that, the output is deasserted at the count value set in advance to 16-bit capture/compare register 00 (CR00).

**Cautions 1. If the external trigger is generated while the one-shot pulse is being output, the counter is cleared and restarted, and the one-shot pulse is output again.**

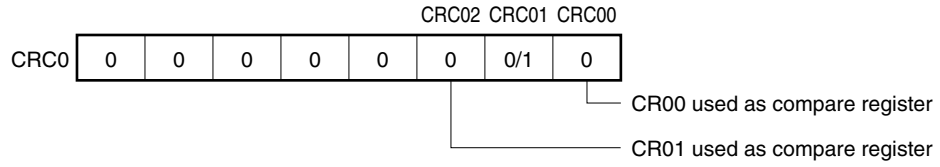
**2. The software trigger (bit 6 (OSPT) of 16-bit timer output control register 0 (TOC0) = 1) and the external trigger (TI00 input) are always valid in one-shot pulse output mode.**
**If the software trigger is used in one-shot pulse output mode, the TI00 pin cannot be used as a general-purpose port pin. Therefore, fix the TI00 pin to either high level or low level.**

**Figure 8-27. Control Register Settings for One-Shot Pulse Output by External Trigger**

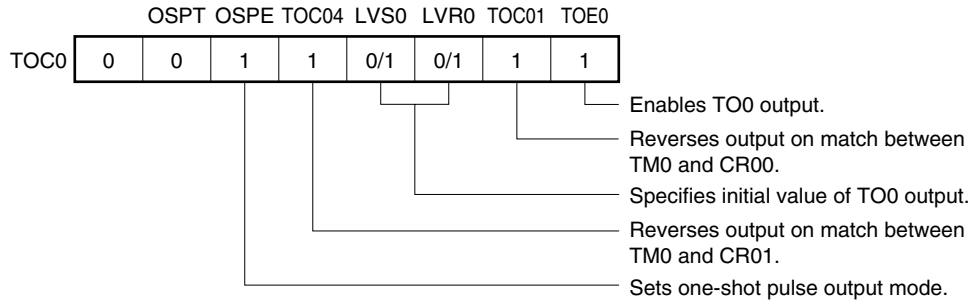
**(a) 16-bit timer mode control register 0 (TMC0)**




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



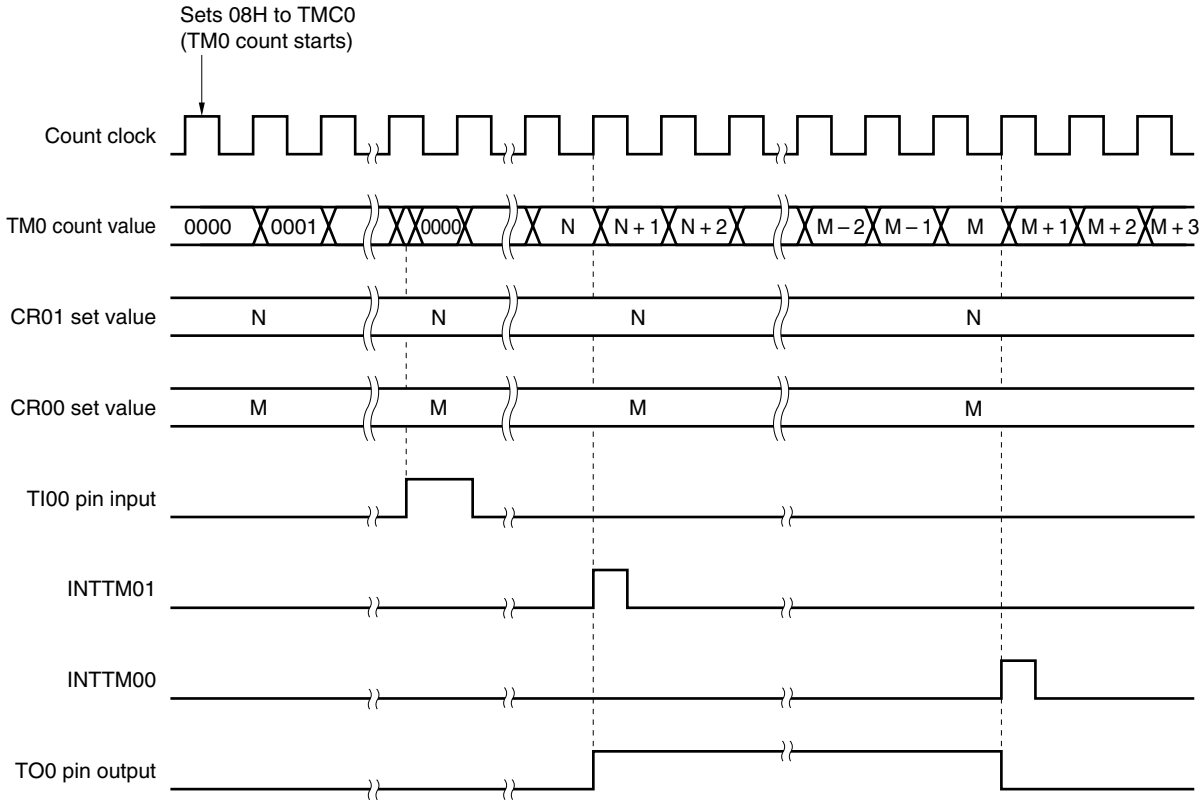

**(c) 16-bit timer output control register 0 (TOC0)**




**Caution Set CR00 and CR01 to a value in the following range.**
**0000H < CR01 < CR00 ≤ FFFFH**

**Remark** 0/1: When these bits are reset to 0 or set to 1, other functions can be used together with the one-shot pulse output function. For details, refer to **Figures 8-2** to **8-4**.

**Figure 8-28. Timing of One-Shot Pulse Output Operation by External Trigger (with Rising Edge Specified)**




**Cautions 1. 16-bit timer counter 0 starts operating as soon as TMC02 and TMC03 are set to a value other than 0, 0 (operation stop mode).**

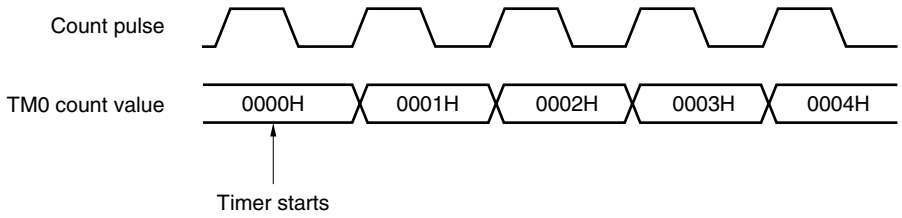
**2. The software trigger (bit 6 (OSPT) of 16-bit timer output control register 0 (TOC0) = 1) and the external trigger (TI00 input) are always valid in one-shot pulse output mode.**
**If the software trigger is used in one-shot pulse output mode, the TI00 pin cannot be used as a general-purpose port pin. Therefore, fix the TI00 pin to either high level or low level.**

## 8.5 Cautions

**(1) Error on starting timer**

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

**Figure 8-29. Start Timing of 16-Bit Timer Counter 0**




**(2) 16-bit compare register settings**

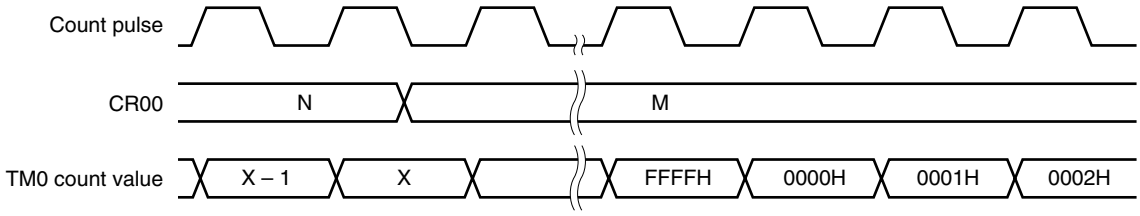
Set 16-bit capture/compare registers 00 and 01 (CR00, 01) to a value other than 0000H.

A one-pulse count operation is consequently not possible when using these registers as event counters.

**(3) Operation after changing compare register during timer count operation**

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

**Figure 8-30. Timing After Changing Compare Register During Timer Count Operation**

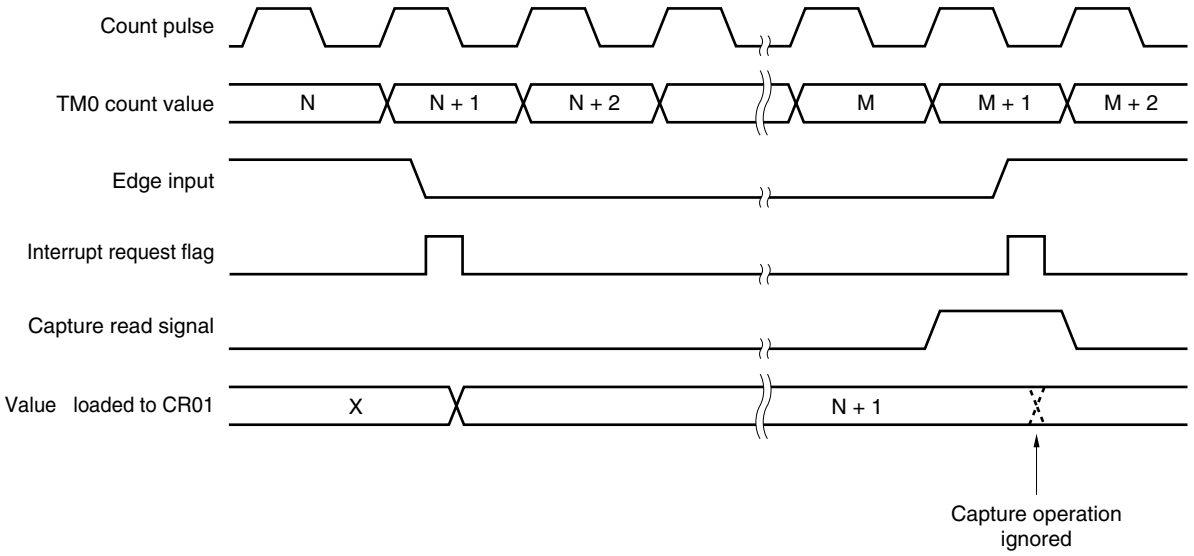



**Remark** N > X > M

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

If the valid edge is input to the TI00/P35 pin while 16-bit capture/compare register 01 (CR01) is being read, CR01 performs the capture operation, but this capture value is not guaranteed. However, the interrupt request flag (INTTM01) is set as a result of detection of the valid edge

**Figure 8-31. Data Hold Timing of Capture Register**



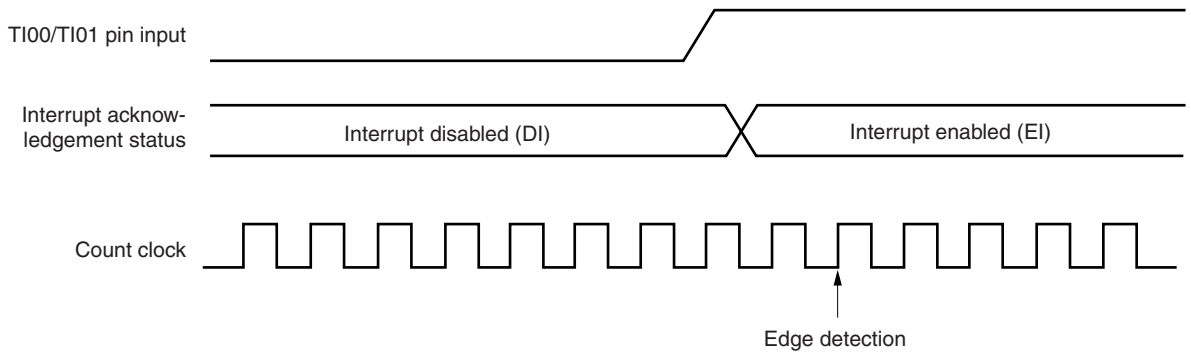

**(5) Setting valid edge**

To set the valid edge of the TI00/P35 pin, set bits 2 and 3 of 16-bit timer mode control register 0 (TMC0) to 0,0 as soon as the timer operation has been halted. The valid edge is specified with bits 4 and 5 of prescaler mode register 0 (ES00, ES01).

**(6) Cautions on edge detection**

**<1>** When the TI00/TI01 pin is high level immediately after system reset, it may be detected as a rising edge after the first 16-bit timer/event counter operation is enabled. Bear this in mind when pulling up, etc.

**<2>** Regardless of whether interrupt acknowledgement is disabled (DI) or enabled (EI), the edge of the external input signal is detected at the second clock after the signal is changed.



**(7) Trigger for one-shot pulse**

The software trigger (bit 6 (OSPT) of 16-bit timer output control register 0 (TOC0) = 1) and the external trigger (TI00 input) are always valid in one-shot pulse output mode.

If the software trigger is used in one-shot pulse output mode, the TI00 pin cannot be used as a general-purpose port pin. Therefore, fix the TI00 pin to either high level or low level.

**(8) Re-triggering one-shot pulse**

**(a) One-shot pulse output by software**

When a one-shot pulse is output, do not set OSPT to 1. Do not output the one-shot pulse again until INTTM00, which occurs on match between TM0 and CR00, occurs.

**(b) One-shot pulse output with external trigger**

If the external trigger is generated while the one-shot pulse is being output, the counter is cleared and restarted, and the one-shot pulse is output again.

**(9) Operation of OVF0 flag**

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

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

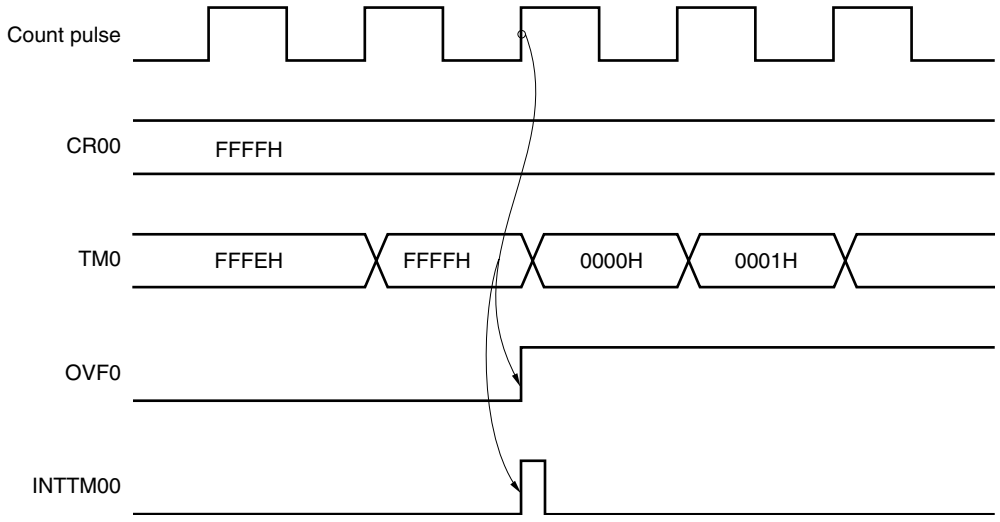
↓

Set CR00 to FFFFH.

↓

When TM0 counts up from FFFFH to 0000H

**Figure 8-32. Operation Timing of OVF0 Flag**



### (10) Conflicting operations

**<1>** Conflict between the read period of the 16-bit capture/compare registers (CR00 and CR01) and the capture trigger input (CR00 and CR01 are used as capture registers)
The capture trigger input has priority. The read data of CR00 and CR01 is undefined.

**<2>** Match timing conflict between the write the period of the 16-bit capture/compare registers (CR00 and CR01) and 16-bit timer counter 0 (TM0). (CR00 and CR01 are used as compare registers)
A match discrimination is not normally performed. Do not perform write to CR00 and CR01 around the match timing.

# CHAPTER 9 8-BIT TIMER/EVENT COUNTERS 1, 2

## 9.1 Functions

8-bit timer/event counters 1 and 2 (TM1, TM2) have the following two modes.

- Mode using 8-bit timer/event counters 1 and 2 (TM1, TM2) alone (discrete mode)
- Mode using 8-bit timer/event counters 1 and 2 connected in cascade (16-bit resolution: cascade connection mode)

These two modes are described next.

**(1) Mode using 8-bit timer/event counters 1 and 2 alone (discrete mode)**

The timer operates as an 8-bit timer/event counter with the following functions.

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

**(2) Mode using 8-bit timer/event counters 1 and 2 connected in cascade (16-bit resolution: cascade connection mode)**

The timer operates as a 16-bit timer/event counter connected in cascade with the following functions.

- Interval timer with 16-bit resolution
- External event counter with 16-bit resolution
- Square-wave output with 16-bit resolution

## 9.2 Configuration

8-bit timer/event counters 1 and 2 include the following hardware.

**Table 9-1. Configuration of 8-Bit Timer/Event Counters 1 and 2**

| Item | Configuration |
|---|---|
| Timer counter | 8-bit × 2 (TM1, TM2) |
| Register | 8-bit × 2 (CR10, CR20) |
| Timer output | 2 (TO1, TO2) |
| Control registers | 8-bit timer mode control register 1 (TMC1)<br>8-bit timer mode control register 2 (TMC2)<br>Prescaler mode register 1 (PRM1)<br>Prescaler mode register 2 (PRM2) |

**Figure 9-1. Block Diagram of 8-Bit Timer/Event Counters 1 and 2 (1/2)**

**(1) 8-bit timer/event counter 1**

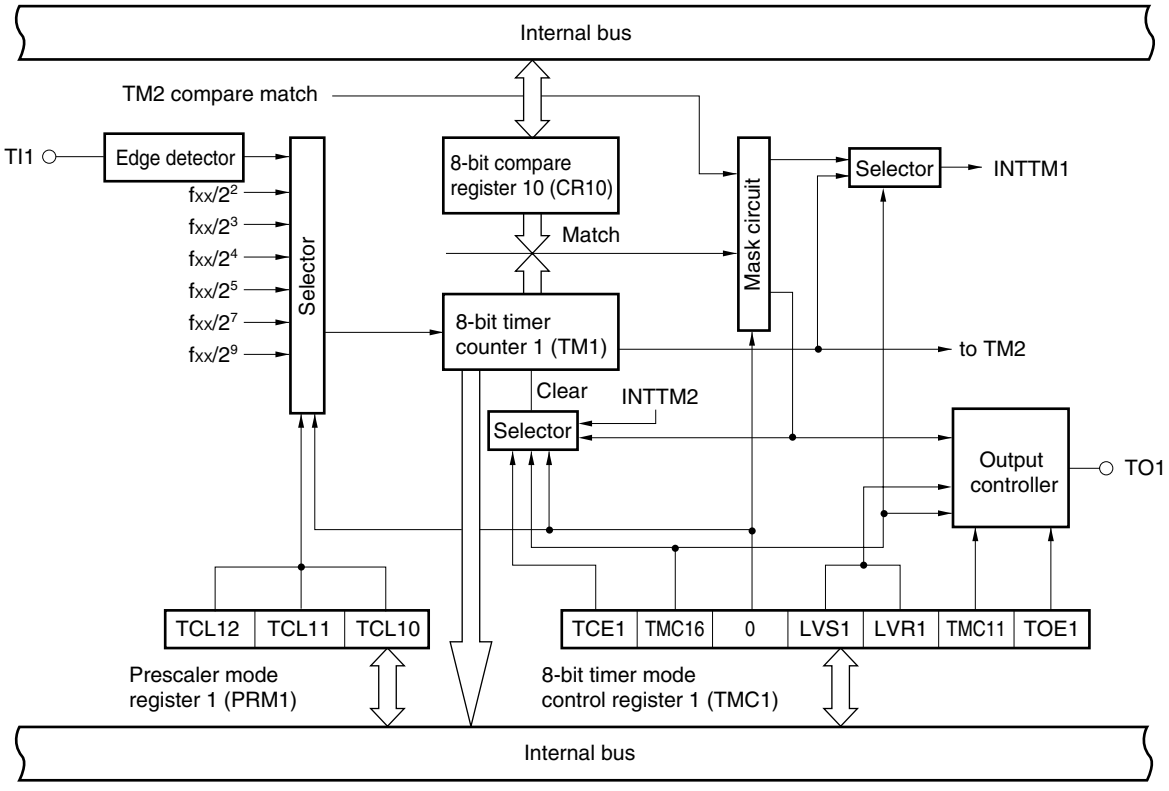
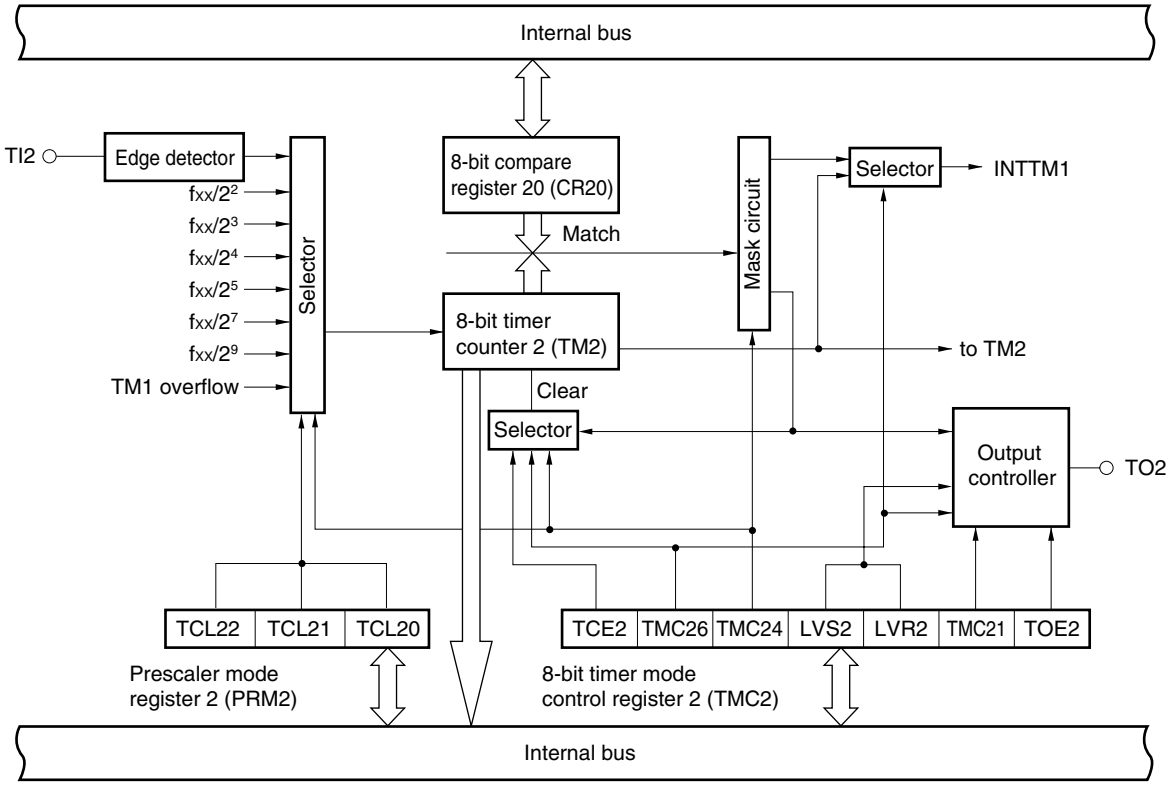


**Figure 9-1. Block Diagram of 8-Bit Timer/Event Counters 1 and 2 (2/2)**

**(2) 8-bit timer/event counter 2**



**(1) 8-bit timer counters 1 and 2 (TM1, TM2)**

TM1 and TM2 are 8-bit read-only registers that count the count pulses.

The counter is incremented in synchronization with the rising edge of the count clock. When the count is read out during operation, the count clock input temporarily stops and the count is read at that time. In the following cases, the count becomes 00H.

<1> $\overline{\text{RESET}}$ is input.
<2> TCEn is cleared.
<3> TMn and CRn0 match in the clear and start mode.

**Caution In cascade connection mode, the count becomes 00H by clearing both bit 7 (TCE1) of 8-bit timer mode control register 1 (TMC1) and bit 7 (TCE2) of 8-bit timer mode control register 2 (TMC2).**

**Remark** n = 1, 2

**(2) 8-bit compare registers 10 and 20 (CR10, CR20)**

The values set in CR10 and CR20 are compared to the count value in 8-bit timer register 1 (TM1) and 8-bit timer counter 2 (TM2), respectively. If the two values match, an interrupt request (INTTM1, INTTM2) is generated (except in the PWM mode).

The values of CR10 and CR20 can be set in the range of 00H to FFH, and can be written during counting.

**Caution Be sure to stop the timer before setting data in cascade connection mode. To stop the timer operation, clear both bit 7 of TMC1 (TCE1) and bit 7 of TMC2 (TCE2).**

## 9.3 Control Registers

The following four registers control 8-bit timer/event counters 1 and 2.

- 8-bit timer mode control registers 1 and 2 (TMC1, TMC2)
- Prescaler mode registers 1 and 2 (PRM1, PRM2)

**(1) 8-bit timer mode control registers 1 and 2 (TMC1, TMC2)**

TMC1 and TMC2 make the following six settings.

<1> Control of the counting for 8-bit timer counters 1 and 2 (TM1, TM2).
<2> Selection of the operation mode of 8-bit timer counters 1 and 2 (TM1, TM2).
<3> Selection of the discrete mode or cascade mode.
<4> Setting of the state of the timer output (only for TMC2).
<5> Control of the timer or selection of the active level in PWM (free-running) mode.
<6> Control of timer output.

TMC1 and TMC2 are set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets TMC1 and TMC2 to 00H.

Figures 9-2 and 9-3 show the TMC1 format and TMC2 format respectively.

**Figure 9-2. Format of 8-Bit Timer Mode Control Register 1 (TMC1)**

Address: 0FF54H After reset: 00H R/W

| Symbol | ⑦ | 6 | 5 | 4 | ③ | ② | 1 | ⓪ |
|---|---|---|---|---|---|---|---|---|
| TMC1 | TCE1 | TMC16 | 0 | 0 | LVS1 | LVR1 | TMC11 | TOE1 |

| TCE1 | TM1 count control |
|---|---|
| 0 | Counting is disabled (prescaler disabled) after the counter is cleared to 0. |
| 1 | Start counting |

| TMC16 | TM1 operation mode selection |
|---|---|
| 0 | Clear and start mode when TM1 and CR10 match. |
| 1 | PWM (free-running) mode |

| LVS1 | LVR1 | Timer output control by software |
|---|---|---|
| 0 | 0 | No change |
| 0 | 1 | Reset (to 0) |
| 1 | 0 | Set (to 1) |
| 1 | 1 | Setting prohibited |

| TMC11 | Other than PWM mode (TMC16 = 0) | PWM mode (TMC16 = 1) |
|---|---|---|
| | Timer output control | Active level selection |
| 0 | Disable inversion operation | Active high |
| 1 | Enable inversion operation | Active low |

| TOE1 | Timer output control |
|---|---|
| 0 | Output disabled (port mode) |
| 1 | Output enabled |

**Caution When selecting the TM1 operation mode using TMC16, stop the timer operation in advance.**

**Remarks 1.** In the PWM mode, the PWM output is set to the inactive level by TCE1 = 0.

**2.** If LVS1 and LVR1 are read after setting data, 0 is read.

**Figure 9-3. Format of 8-Bit Timer Mode Control Register 2 (TMC2)**

Address: 0FF55H After reset: 00H R/W

| Symbol | ⑦ | 6 | 5 | 4 | ③ | ② | 1 | ⓪ |
|---|---|---|---|---|---|---|---|---|
| TMC2 | TCE2 | TMC26 | 0 | TMC24 | LVS2 | LVR2 | TMC21 | TOE2 |

| TCE2 | TM2 count control |
|---|---|
| 0 | Counting is disabled (prescaler disabled) after the counter is cleared to 0. |
| 1 | Start counting |

| TMC26 | TM2 operation mode selection |
|---|---|
| 0 | Clear and start mode when TM2 and CR20 match |
| 1 | PWM (free-running) mode |

| TMC24 | Discrete mode or cascade connection mode selection |
|---|---|
| 0 | Discrete mode |
| 1 | Cascade connection mode (connection with TM1) |

| LVS2 | LVR2 | Timer output control by software |
|---|---|---|
| 0 | 0 | No change |
| 0 | 1 | Reset (to 0) |
| 1 | 0 | Set (to 1) |
| 1 | 1 | Setting prohibited |

| TMC21 | Other than PWM mode (TMC26 = 0) | PWM mode (TMC26 = 1) |
|---|---|---|
| | Timer output control | Active level selection |
| 0 | Disable inversion operation | Active high |
| 1 | Enable inversion operation | Active low |

| TOE2 | Timer output control |
|---|---|
| 0 | Output disabled (port mode) |
| 1 | Output enabled |

**Caution When selecting the TM2 operation mode using TMC26 or selecting discrete/cascade connection mode using TMC24, stop timer operation in advance. To stop the timer operation during cascade connection, clear both bit 7 (TCE1) of 8-bit timer mode control register 1 (TMC1) and bit 7 (TCE2) of TMC2.**

**Remarks 1.** In the PWM mode, the PWM output is set to the inactive level by TCE2 = 0.

**2.** If LVS2 and LVR2 are read after setting data, 0 is read.

**(2) Prescaler mode registers 1 and 2 (PRM1, PRM2)**

This register sets the count clock of 8-bit timer counters 1 and 2 (TM1, TM2) and the valid edge of the TI1 and TI2 inputs.

PRM1 and PRM2 are set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets PRM1 and PRM2 to 00H.

**Figure 9-4. Format of Prescaler Mode Register 1 (PRM1)**

Address: 0FF56H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PRM1 | 0 | 0 | 0 | 0 | 0 | TCL12 | TCL11 | TCL10 |

| TCL12 | TCL11 | TCL10 | Count clock selection |
|---|---|---|---|
| 0 | 0 | 0 | Falling edge of TI1 |
| 0 | 0 | 1 | Rising edge of TI1 |
| 0 | 1 | 0 | $f_{XX}/4$ (3.13 MHz) |
| 0 | 1 | 1 | $f_{XX}/8$ (1.56 MHz) |
| 1 | 0 | 0 | $f_{XX}/16$ (781 kHz) |
| 1 | 0 | 1 | $f_{XX}/32$ (391 kHz) |
| 1 | 1 | 0 | $f_{XX}/128$ (97.6 kHz) |
| 1 | 1 | 1 | $f_{XX}/512$ (24.4 kHz) |

**Cautions 1. If data different from that of PRM1 is written, stop the timer beforehand.**

**2. Be sure to set bits 3 to 7 of PRM1 to 0.**

**3. When specifying the valid edge of TI1 for the count clock, set the count clock to $f_{XX}/4$ or below.**

**Remark** Values in parentheses apply to operation at $f_{XX}$ = 12.5 MHz.

**Figure 9-5. Format of Prescaler Mode Register 2 (PRM2)**

Address: 0FF57H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PRM2 | 0 | 0 | 0 | 0 | 0 | TCL22 | TCL21 | TCL20 |

| TCL22 | TCL21 | TCL20 | Count clock selection |
|---|---|---|---|
| 0 | 0 | 0 | Falling edge of TI2 |
| 0 | 0 | 1 | Rising edge of TI2 |
| 0 | 1 | 0 | $f_{XX}/4$ (3.13 MHz) |
| 0 | 1 | 1 | $f_{XX}/8$ (1.56 MHz) |
| 1 | 0 | 0 | $f_{XX}/16$ (781 kHz) |
| 1 | 0 | 1 | $f_{XX}/32$ (391 kHz) |
| 1 | 1 | 0 | $f_{XX}/128$ (97.6 kHz) |
| 1 | 1 | 1 | $f_{XX}/512$ (24.4 kHz) |

**Cautions 1. If data different from that of PRM2 is written, stop the timer beforehand.**

**2. Be sure to set 0 to bits 3 to 7 of PRM2.**

**3. When specifying the valid edge of TI2 for the count clock, set the count clock to $f_{XX}/4$ or below.**

**Remark** Figures in parentheses apply to operation at $f_{XX}$ = 12.5 MHz.

## 9.4 Operation

### 9.4.1 Operation as interval timer (8-bit operation)

The timer operates as an interval timer that repeatedly generates interrupt requests at the interval of the count preset in 8-bit compare registers 10 and 20 (CR10, CR20).

If the count in 8-bit timer counters 1 and 2 (TM1, TM2) matches the value set in CR10 and CR20, the values of TM1 and TM2 are cleared to 0 and the count continues. At the same time, an interrupt request signal (INTTM1, INTTM2) is generated.

The TM1 and TM2 count clock can be selected with bits 0 to 2 (TCLn0 to TCLn2) of prescaler mode registers 1 and 2 (PRM1, PRM2).

**<Setting method>**

<1> Set each register.
- PRMn: Selects the count clock.
- CRn0: Compare value
- TMCn: Selects the clear and start mode when TMn and CRn0 match.
  (TMCn = 0000×××0B, × = Don't care)

<2> When TCEn = 1 is set, counting starts.

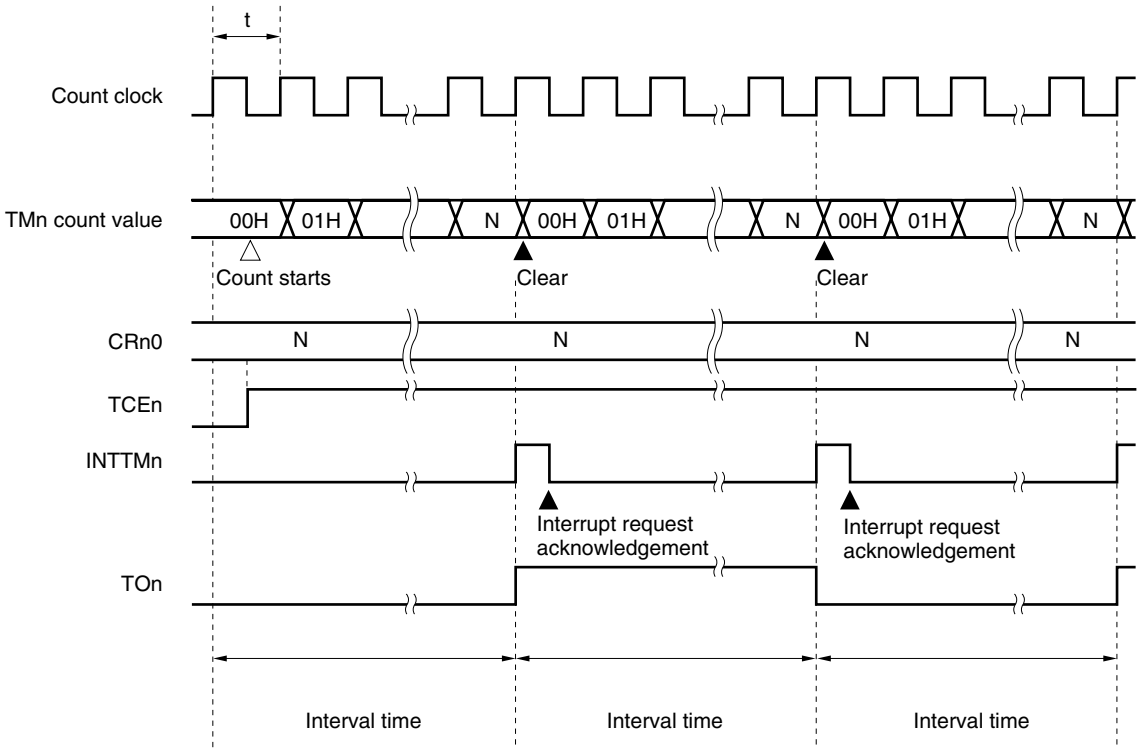
<3> When the values of TMn and CRn0 match, INTTMn is generated (TMn is cleared to 00H).

<4> Then, INTTMn is repeatedly generated at the same interval. When counting stops, set TCEn = 0.

**Remark** n = 1, 2

**Figure 9-6. Timing of Interval Timer Operation (1/3)**

**(a) Basic operation**

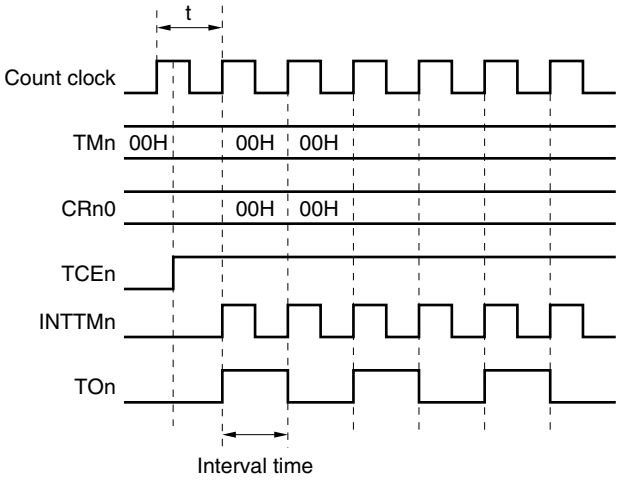



**Remarks 1.** Interval time = (N + 1) × t: N = 00H to FFH

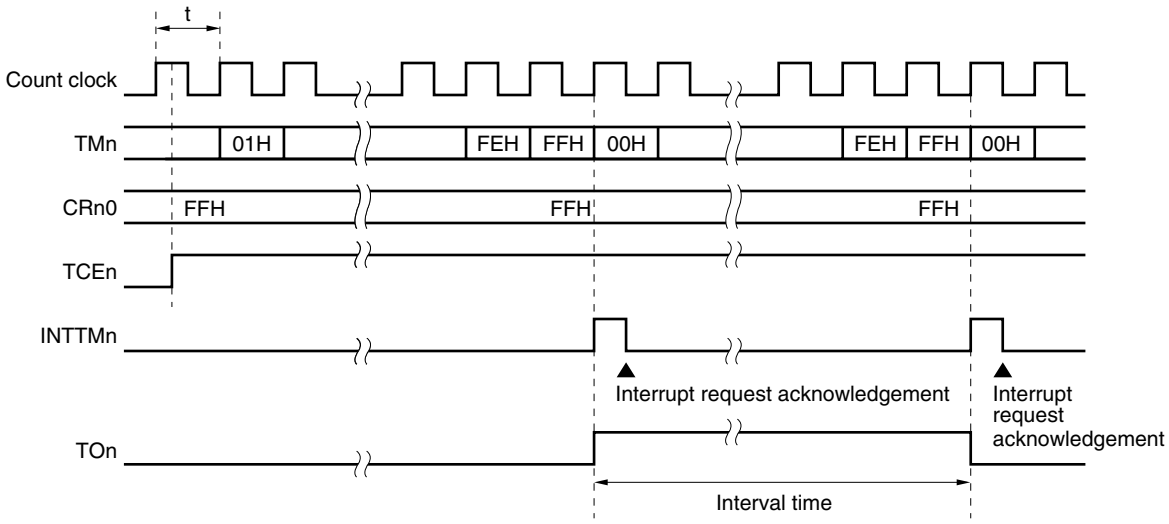
**2.** n = 1, 2

**Figure 9-6. Timing of Interval Timer Operation (2/3)**

**(b) When CRn0 = 00H**



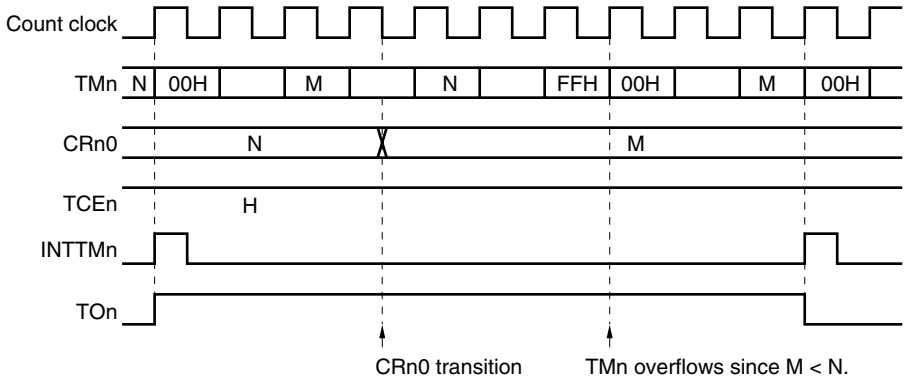

**(c) When CRn0 = FFH**



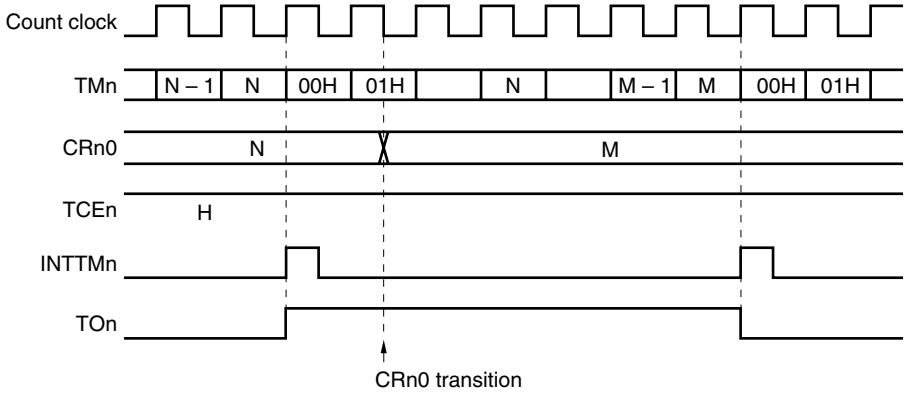

**Remark** n = 1, 2

**Figure 9-6. Timing of Interval Timer Operation (3/3)**

**(d) Operated by CRn0 transition (M < N)**




**(e) Operated by CRn0 transition (M > N)**




**Remark** n = 1, 2

### 9.4.2 Operation as external event counter

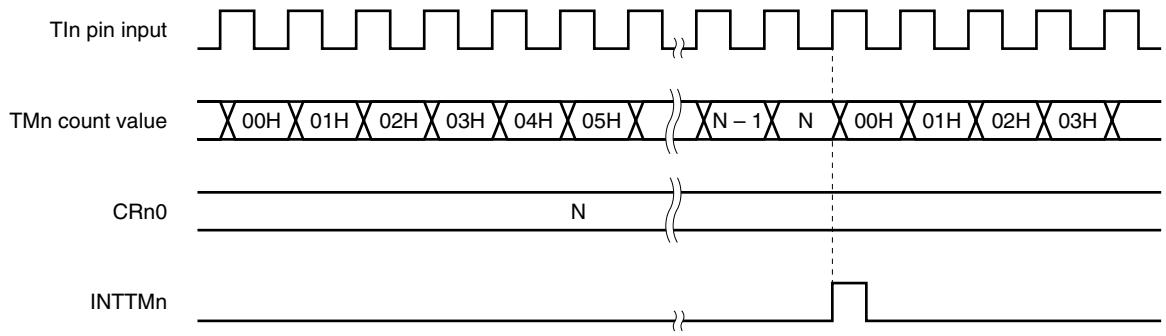
The external event counter counts the number of external clock pulses that are input to the TI1/P33 and TI1/P34 pins with 8-bit timer counters 1 and 2 (TM1, TM2).

Each time the valid edge specified by prescaler mode registers 1 and 2 (PRM1, PRM2) is input, TM1and TM2 are incremented. The edge setting is selected to be either the rising edge or falling edge.

If the count of TM1and TM2 matches the values of 8-bit compare registers 10 and 20 (CR10, CR20), TM1and TM2 are cleared to 0 and an interrupt request signal (INTTM1, INTTM2) is generated.

INTTM1 and INTTM2 are generated each time the values of the TM1 and TM2 match the values of CR10 and CR20.

**Figure 9-7. Timing of External Event Counter Operation (with Rising Edge Is Specified)**




**Remark** N = 00H to FFH
n = 1, 2

### 9.4.3 Operation to output square-wave (8-bit resolution)

A square wave with any frequency is output at the interval preset in 8-bit compare registers 10 and 20 (CR10, CR20).

By setting bit 0 of 8-bit timer mode control registers 1 and 2 (TMC1, TMC2) to 1, the output state of TO1 and TO2 is inverted with the count preset in CR510 and CR20 as the interval. Therefore, square wave output of any frequency (duty cycle = 50%) is possible.

**<Setting method>**

<1> Set the registers.

- Set the port latch, which also functions as a timer output pin and the port mode register to 0.
- PRMn: Selects the count clock.
- CRn0: Compare value
- TMCn: Clear and start mode when TMn and CRn0 match.

| LVSn | LVRn | Timer Output Control by Software |
|---|---|---|
| 1 | 0 | High-level output |
| 0 | 1 | Low-level output |

Inversion of timer output enabled
Timer output enabled → TOEn = 1

<2> When TCEn = 1 is set, the counter starts operating.

<3> If the values of TMn and CRn0 match, the timer output is inverted. Also, INTTMn is generated and TMn is cleared to 00H.

<4> Then, the timer output is inverted at the same interval to output a square wave from TOn.

**Remark** n = 1, 2

### 9.4.4 Operation to output 8-bit PWM

By setting bit 6 (TMC16, TMC26) of 8-bit timer mode control registers 1 and 2 (TMC1, TMC2) to 1, the timer operates as a PWM output.

Pulses with the duty cycle determined by the value set in 8-bit compare registers 10 and 20 (CR10, CR20) is output from TO1 and TO2.

Set the width of the active level of the PWM pulse in CR10 and CR20. The active level can be selected by bit 1 (TMC11, TMC12) of TMC1 and TMC2.

The count clock can be selected by bits 0 to 2 (TCLn0 to TCLn2) of prescaler mode registers 1 and 2 (PRM1, PRM2).

The PWM output can be enabled and disabled by bit 0 (TOE1, TOE2) of TMC1 and TMC2.

#### (1) Basic operation of the PWM output

**<Setting method>**

<1> Set the port latch, which also functions as a timer output pin and the port mode register to 0.
<2> Set the active level width in 8-bit compare register n (CRn0).
<3> Select the count clock in prescaler mode register n (PRMn).
<4> Set the active level in bit 1 (TMCn1) of TMCn.
<5> Set bit 0 of TMCn (TOEn) to 1 to enable timer output.
<6> If bit 7 (TCEn) of TMCn is set to 1, counting starts. When counting stops, set TCEn to 0.

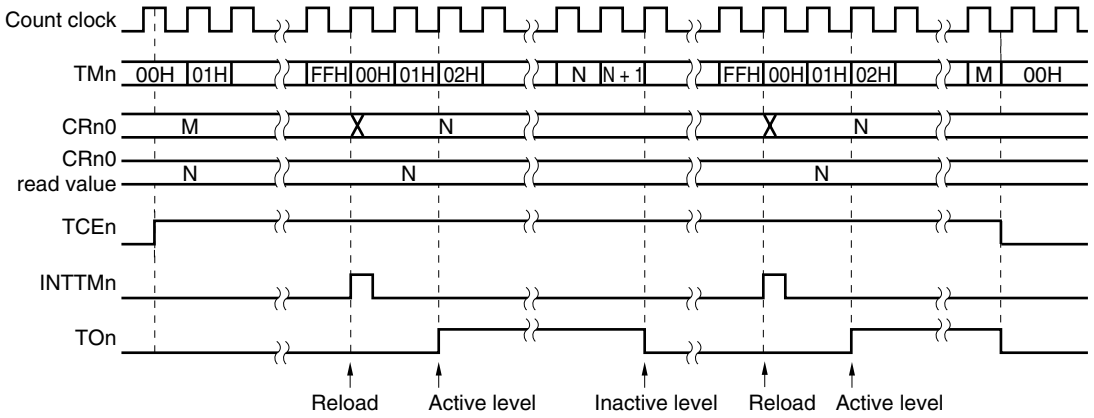
**<PWM output operation>**

<1> When counting starts, the PWM output (output from TOn) outputs the inactive level until an overflow occurs.
<2> When an overflow occurs, the active level is output. The active level is output until CRn0 and the count of 8-bit timer counter n (TMn) match.
<3> The PWM output after CRn and the count value match is the inactive level until an overflow occurs again.
<4> Steps <2> and <3> repeat until counting stops.
<5> If counting is stopped by TCEn = 0, the PWM output goes to the inactive level.

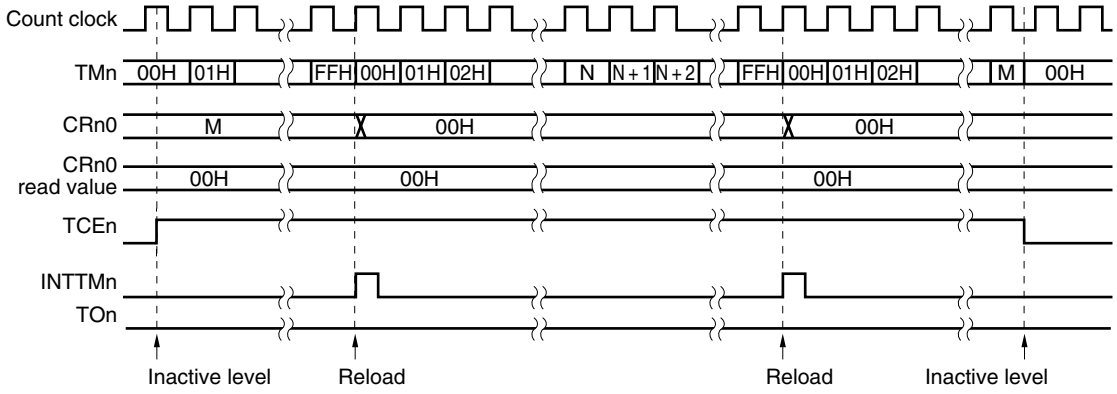
**Remark** n = 1, 2

**Figure 9-8. Timing of PWM Output**

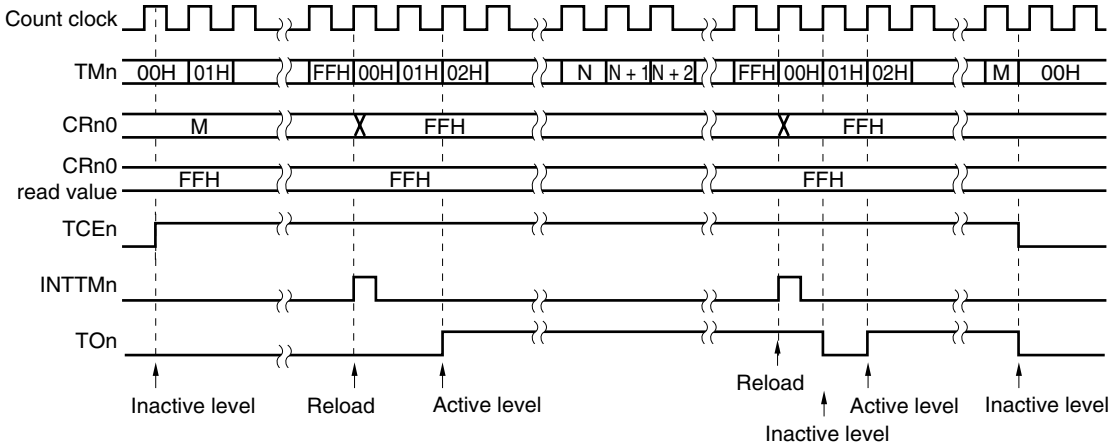
**(a) Basic operation (active level = H)**




**(b) When CRn0 = 0**



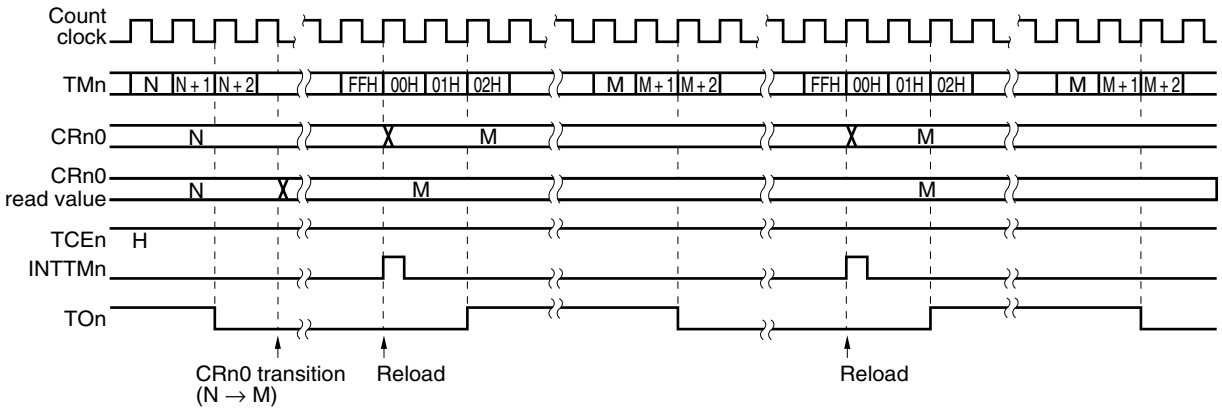

**(c) When CRn0 = FFH**



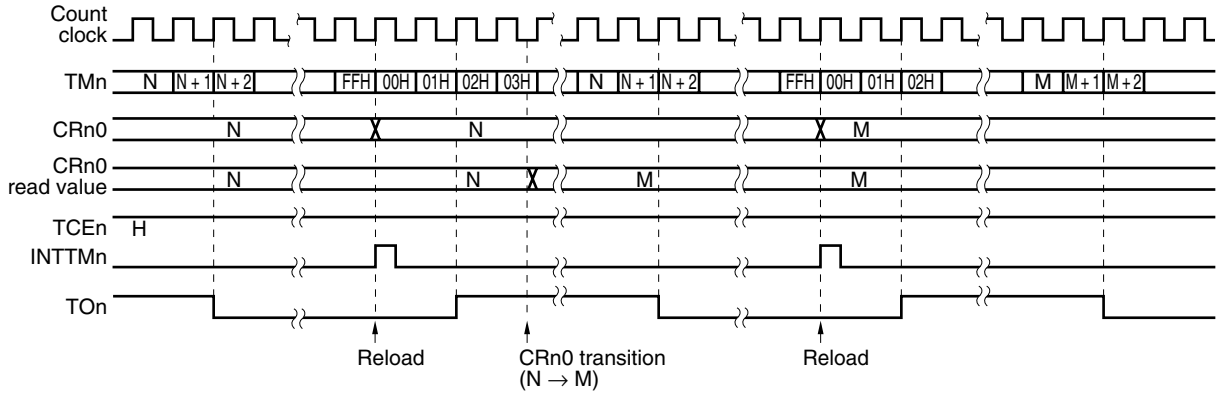

**Remark** n = 1, 2

**Figure 9-9. Timing of Operation Based on CRn0 Transitions**

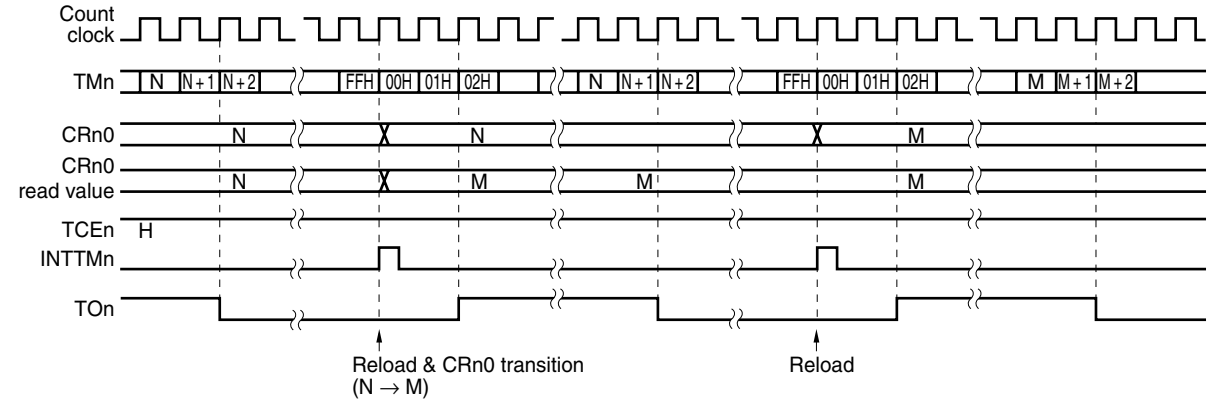
**(a) When the CRn0 value changes from N to M before TMn overflows**




**(b) When the CRn0 value changes from N to M after TMn overflows**




**(c) When the CRn0 value changes from N to M within two clocks (00H, 01H) immediately after TMn overflows**




**Remarks 1.** n = 1, 2

**2.** CRn0(M): Master side, CRn0(S): Slave side

### 9.4.5 Operation as interval timer (16-bit operation)

- **Cascade connection (16-bit timer) mode**

  By setting bit 4 (TMC24) of 8-bit timer mode control register 2 (TMC2) to 1, the timer enters the timer/counter mode with 16-bit resolution.

  With the count preset in 8-bit compare registers 10 and 20 (CR10, CR20) as the interval, the timer operates as an interval timer by repeatedly generating interrupt requests.

**<Setting method>**

<1> Set each register.

- PRM1: TM1 selects the count clock. TM2 connected in cascade is not used for setting.
- CRn0: Compare values (each compare value can be set from 00H to FFH).
- TMCn: Select the clear and start mode when TMn and CRn0 match.

  TM1 → TMC1 = 0000×××0B, ×: Don't care
  TM2 → TMC2 = 0001×××0B, ×: Don't care

<2> Setting TCE2 = 1 in TMC2 and finally setting TCE1 = 1 in TMC1 starts the count operation.

<3> If the values of TMn of all timers connected in cascade and CRn0 match, INTTM1 of TM1 is generated. (TM1 and TM2 are cleared to 00H.)

<4> INTTM1 are repeatedly generated at the same interval.

**Cautions 1. Always set the compare register (CR10, CR20) after stopping timer operation.**

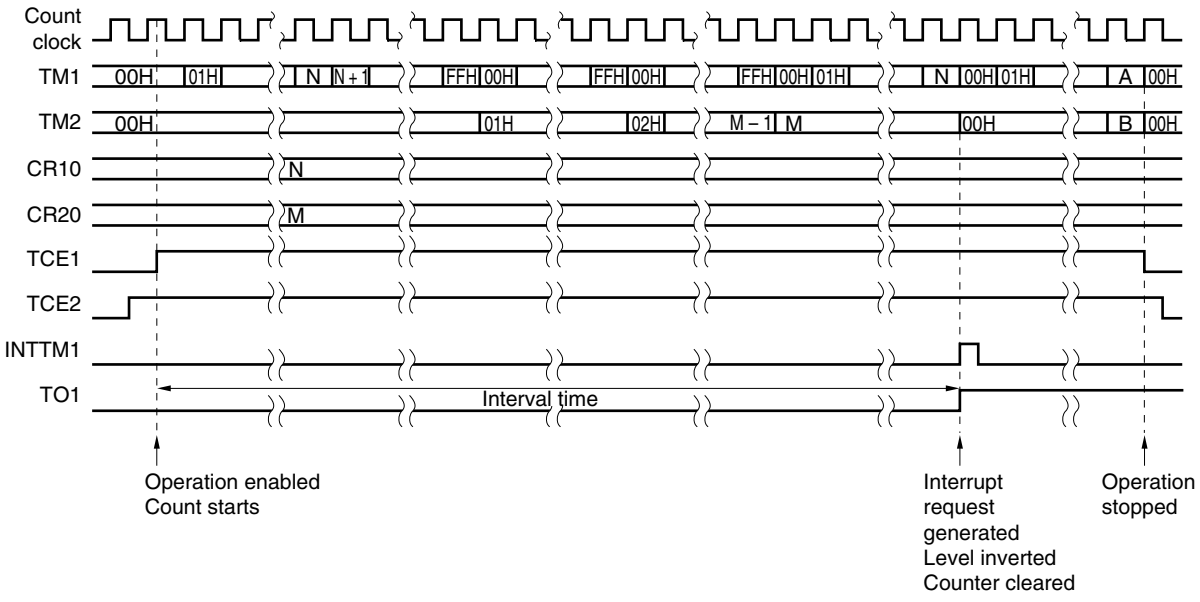
**2. If the TM2 count value matches CR20 even when used in a cascade connection, INTTM2 of TM2 is generated. Always mask the higher timer in order to disable interrupts.**

**3. Set TCE1 and TCE2 in TM2 first. Set TM1 last.**

**4. Restarting and stopping the count is possible by setting only 1 or 0 in TCE1 of TMC1 to start and stop operation. Note, however, that the TCE1 bit of TMC1 and TCE2 bit of TMC2 must be cleared when setting the compare register (CR10, CR20).**

Figure 9-10 shows a timing example of the cascade connection mode with 16-bit resolution.

**Figure 9-10. Cascade Connection Mode with 16-Bit Resolution**

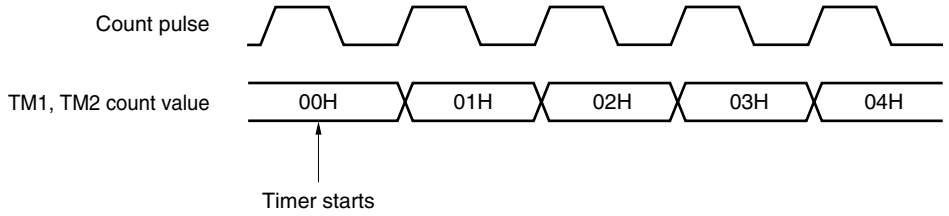


## 9.5 Cautions

**(1) Error when the timer starts**

An error of up to 1 clock occurs before the match signal is generated after the timer is started. This is because 8-bit timer counters 1 and 2 (TM1, TM2) are started asynchronously to the count pulse.

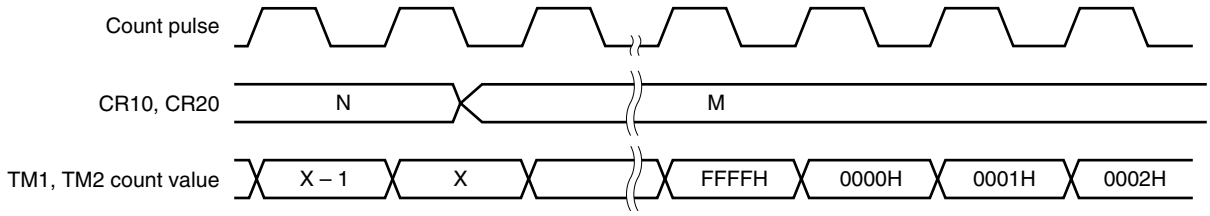
**Figure 9-11. Start Timing of 8-Bit Timer Counter**




**(2) Operation after the compare register is changed while the timer is counting**

If the value after 8-bit compare registers 10 and 20 (CR10, CR20) changes is less than the value of 8-bit timer counter 1 and 2 (TM1, TM2), counting continues, overflows, and counting starts again from 0. Consequently, when the value (M) after CR10 and CR20 change is less than the value (N) before the change, the timer must be restarted after CR10 and CR20 change.

**Figure 9-12. Timing After Compare Register Changes During Timer Counting**




**Caution Except when the TI1, TI2 input is selected, always set TCE1 = 0, TCE2 = 0 before setting the STOP mode.**

**Remark** N > X > M

**(3) TM1, TM2 read out during timer operation**

Since the count clock stops temporarily when TM1 and TM2 are read during operation, select a waveform with a high and low level that exceed 2 cycles of the CPU clock for the count clock.

When reading TM1 and TM2 in cascade connection mode, to avoid reading while the count is changing, take measures such as obtaining a count match by reading twice using software.

# CHAPTER 10 8-BIT TIMERS 5, 6

## 10.1 Functions

8-bit timers 5 and 6 (TM5, TM6) have the following two modes.

- Mode using 8-bit timers 5 and 6 (TM5, TM6) alone (discrete mode)
- Mode using 8-bit timers 5 and 6 connected in cascade (16-bit resolution: cascade connection mode)

These two modes are described next.

**(1) Mode using 8-bit timers 5 and 6 alone (discrete mode)**

The timer operates as an 8-bit timer with the following function.

- Interval timer

**(2) Mode using 8-bit timers 5 and 6 connected in cascade (16-bit resolution: cascade connection mode)**

The timer operates as a 16-bit timer connected in cascade with the following functions.

- Interval timer with 16-bit resolution

## 10.2 Configuration

8-bit timers 5 and 6 include the following hardware.

**Table 10-1. Configuration of 8-Bit Timers 5 and 6**

| Item | Configuration |
|---|---|
| Timer counter | 8-bit × 2 (TM5, TM6) |
| Register | 8-bit × 2 (CR50, CR60) |
| Control register | 8-bit timer mode control register 5 (TMC5)<br>8-bit timer mode control register 6 (TMC6)<br>Prescaler mode register 5 (PRM5)<br>Prescaler mode register 6 (PRM6) |

**Figure 10-1. Block Diagram of 8-Bit Timers 5 and 6 (1/2)**

**(1) 8-bit timer 5**

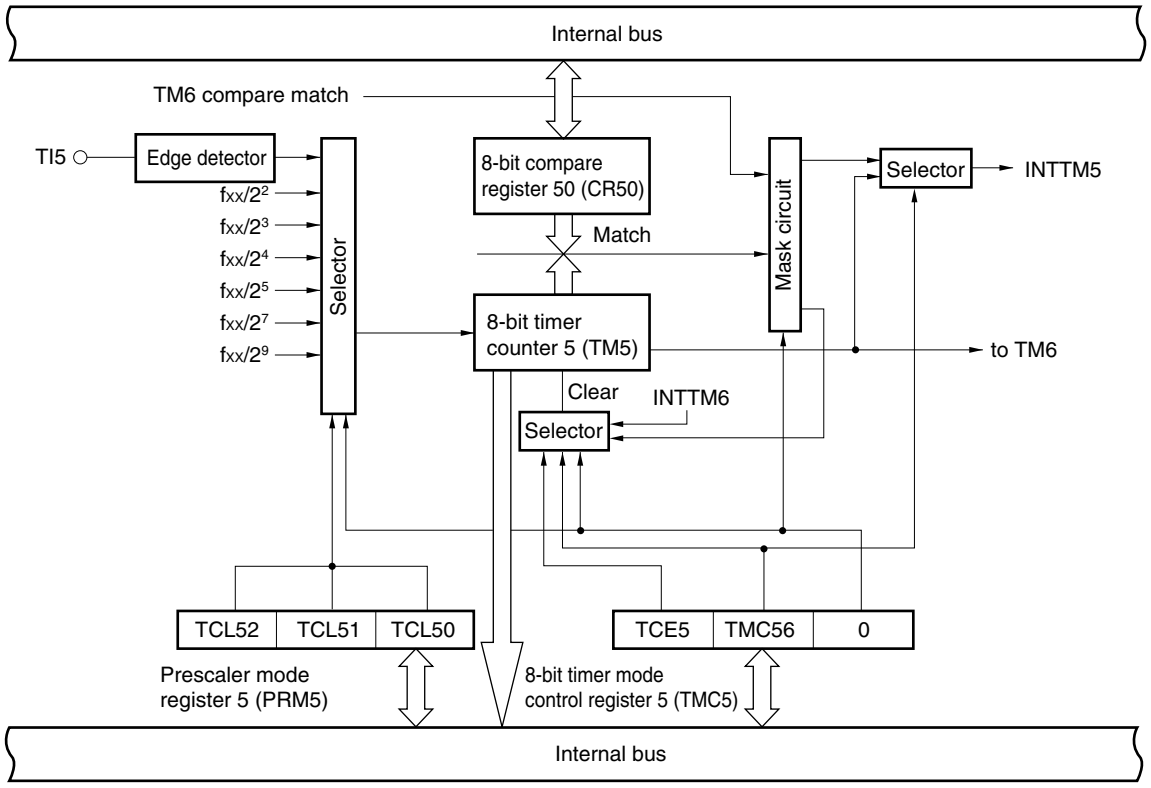
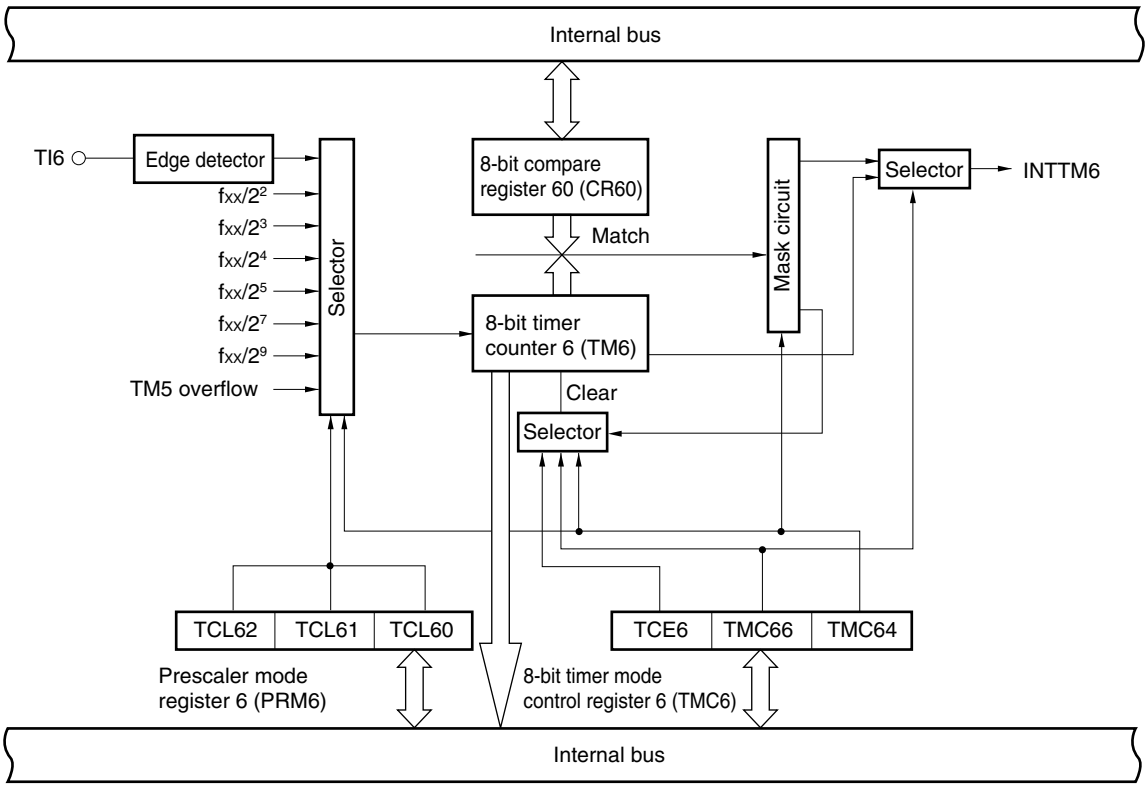


**Figure 10-1. Block Diagram of 8-Bit Timers 5 and 6 (2/2)**

**(2) 8-bit timer 6**



**(1) 8-bit timer counters 5 and 6 (TM5, TM6)**

TM5 and TM6 are 8-bit read-only registers that count the count pulses.

The counter is incremented in synchronization with the rising edge of the count clock. When the count is read out during operation, the count clock input temporarily stops and the count is read at that time. In the following cases, the count becomes 00H.

<1> $\overline{\text{RESET}}$ is input.
<2> TCEn is cleared.
<3> TMn and CRn0 match in the clear and start mode.

**Caution In cascade connection mode, the count becomes 00H by clearing bit 7 (TCE5) of 8-bit timer mode control register 5 (TMC5) and bit 7 (TCE6) of 8-bit timer mode control register 6 (TMC6).**

**Remark** n = 5, 6

**(2) 8-bit compare registers 50 and 60 (CR50, CR60)**

The value set in CR50 and CR60 are compared to the count value in 8-bit timer counter 5 (TM5) and 8-bit timer counter 6 (TM6), respectively. If the two values match, an interrupt request (INTTM5, INTTM6) is generated (except in the PWM mode).

The values of CR50 and CR60 can be set in the range of 00H to FFH, and can be written during counting.

**Caution Be sure to stop the timer operation before setting data in cascade connection mode. To stop the timer operation, clear both TCE5 of TMC5 and TCE6 of TMC6.**

## 10.3 Control Registers

The following four registers control 8-bit timers 5 and 6.

- 8-bit timer mode control registers 5 and 6 (TMC5, TMC6)
- Prescaler mode registers 5 and 6 (PRM5, PRM6)

**(1) 8-bit timer mode control registers 5 and 6 (TMC5, TMC6)**

TMC5 and TMC6 make the following three settings.

<1> Control of the counting for 8-bit timer counters 5 and 6 (TM5, TM6).
<2> Selection of the operation mode of 8-bit timer counters 5 and 6 (TM5, TM6).
<3> Selection of the discrete mode or cascade connection mode (TMC6 only).

TMC5 and TMC6 are set by a 1-bit or 8-bit memory manipulation instruction.
$\overline{\text{RESET}}$ input sets TMC5 and TMC6 to 00H.
Figures 10-2 and 10-3 show the TMC5 format and TMC6 format respectively.

**Figure 10-2. Format of 8-Bit Timer Mode Control Register 5 (TMC5)**

Address: 0FF68H After reset: 00H R/W

| Symbol | ⑦ | 6 | 5 | 4 | 3 | 2 | 1 | ⓪ |
|---|---|---|---|---|---|---|---|---|
| TMC5 | TCE5 | TMC56 | 0 | 0 | 0 | 0 | 0 | 0 |

| TCE5 | TM5 count control |
|---|---|
| 0 | Counting is disabled (prescaler disabled) after the counter is cleared to 0. |
| 1 | Start counting |

| TMC56 | TM5 operation mode selection |
|---|---|
| 0 | Clear and start mode when TM5 and CR50 match. |
| 1 | PWM (free running) mode |

**Caution When selecting the TM5 operation mode using TMC56, stop the timer operation in advance.**

**Remarks 1.** In the PWM mode, the PWM output is set to the inactive level by TCE5 = 0.
**2.** If LVS5 and LVR5 are read after setting data, 0 is read.

**Figure 10-3. Format of 8-Bit Timer Mode Control Register 6 (TMC6)**

Address: 0FF69H After reset: 00H R/W

| Symbol | ⑦ | 6 | 5 | 4 | ③ | ② | 1 | ⓪ |
|---|---|---|---|---|---|---|---|---|
| TMC6 | TCE6 | TMC66 | 0 | TMC64 | 0 | 0 | 0 | 0 |

| TCE6 | TM6 count control |
|---|---|
| 0 | Counting is disabled (prescaler disabled) after the counter is cleared to 0. |
| 1 | Start counting |

| TMC66 | TM6 operation mode selection |
|---|---|
| 0 | Clear and start mode when TM6 and CR60 match |
| 1 | PWM (free running) mode |

| TMC64 | Discrete mode or cascade connection mode selection |
|---|---|
| 0 | Discrete mode |
| 1 | Cascade connection mode (connection with TM5) |

**Caution When selecting the TM6 operation mode using TMC66 or selecting discrete/cascade connection mode using TMC64, stop the timer operation in advance. To stop the timer operation during cascade connection, clear both bit 7 (TCE5) of 8-bit timer mode control register 5 (TMC5) and bit 7 (TCE6) of TMC6.**

**(2) Prescaler mode registers 5 and 6 (PRM5, PRM6)**

This register sets the count clock of 8-bit timer counters 5 and 6 (TM5, TM6).

PRM5 and PRM6 are set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets PRM5 and PRM6 to 00H.

**Figure 10-4. Format of Prescaler Mode Register 5 (PRM5)**

Address: 0FF6CH After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PRM5 | 0 | 0 | 0 | 0 | 0 | TCL52 | TCL51 | TCL50 |

| TCL52 | TCL51 | TCL50 | Count clock selection |
|---|---|---|---|
| 0 | 1 | 0 | $f_{XX}/4$ (3.13 MHz) |
| 0 | 1 | 1 | $f_{XX}/8$ (1.56 MHz) |
| 1 | 0 | 0 | $f_{XX}/16$ (781 kHz) |
| 1 | 0 | 1 | $f_{XX}/32$ (391 kHz) |
| 1 | 1 | 0 | $f_{XX}/128$ (97.6 kHz) |
| 1 | 1 | 1 | $f_{XX}/512$ (24.4 kHz) |
| Other than above | | | Setting prohibited |

**Cautions 1. If data different from that of PRM5 is written, stop the timer beforehand.**

**2. Be sure to set bits 3 to 7 of PRM5 to 0.**

**Remark** Values in parentheses apply to operation at $f_{XX}$ = 12.5 MHz.

**Figure 10-5. Format of Prescaler Mode Register 6 (PRM6)**

Address: 0FF6DH After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PRM6 | 0 | 0 | 0 | 0 | 0 | TCL62 | TCL61 | TCL60 |

| TCL62 | TCL61 | TCL60 | Count clock selection |
|---|---|---|---|
| 0 | 1 | 0 | $f_{XX}$/4 (3.13 MHz) |
| 0 | 1 | 1 | $f_{XX}$/8 (1.56 MHz) |
| 1 | 0 | 0 | $f_{XX}$/16 (781 kHz) |
| 1 | 0 | 1 | $f_{XX}$/32 (391 kHz) |
| 1 | 1 | 0 | $f_{XX}$/128 (97.6 kHz) |
| 1 | 1 | 1 | $f_{XX}$/512 (24.4 kHz) |
| Other than above | | | Setting prohibited |

**Cautions 1. If data different from that of PRM6 is written, stop the timer beforehand.**

**2. Be sure to set bits 3 to 7 of PRM6 to 0.**

**Remark** Values in parentheses apply to operation at $f_{XX}$ = 12.5 MHz.

## 10.4 Operation

### 10.4.1 Operation as interval timer (8-bit operation)

The timer operates as an interval timer that repeatedly generates interrupt requests at the interval of the count preset in 8-bit compare registers 50 and 60 (CR50, CR60).

If the count in 8-bit timer counters 5 and 6 (TM5, TM6) matches the value set in CR50 and CR60, the values of TM5 and TM6 are cleared to 0 and the count continues. At the same time, an interrupt request signal (INTTM5, INTTM6) is generated.

The TM5 and TM6 count clock can be selected with bits 0 to 2 (TCLn0 to TCLn2) of prescaler mode registers 5 and 6 (PRM5, PRM6).

**<Setting method>**

<1> Set each register.

- PRMn: Selects the count clock.
- CRn0: Compare value
- TMCn: Selects the clear and start mode when TMn and CRn0 match.
  (TMCn = 0000×××0B, × = Don't care)

<2> When TCEn = 1 is set, counting starts.

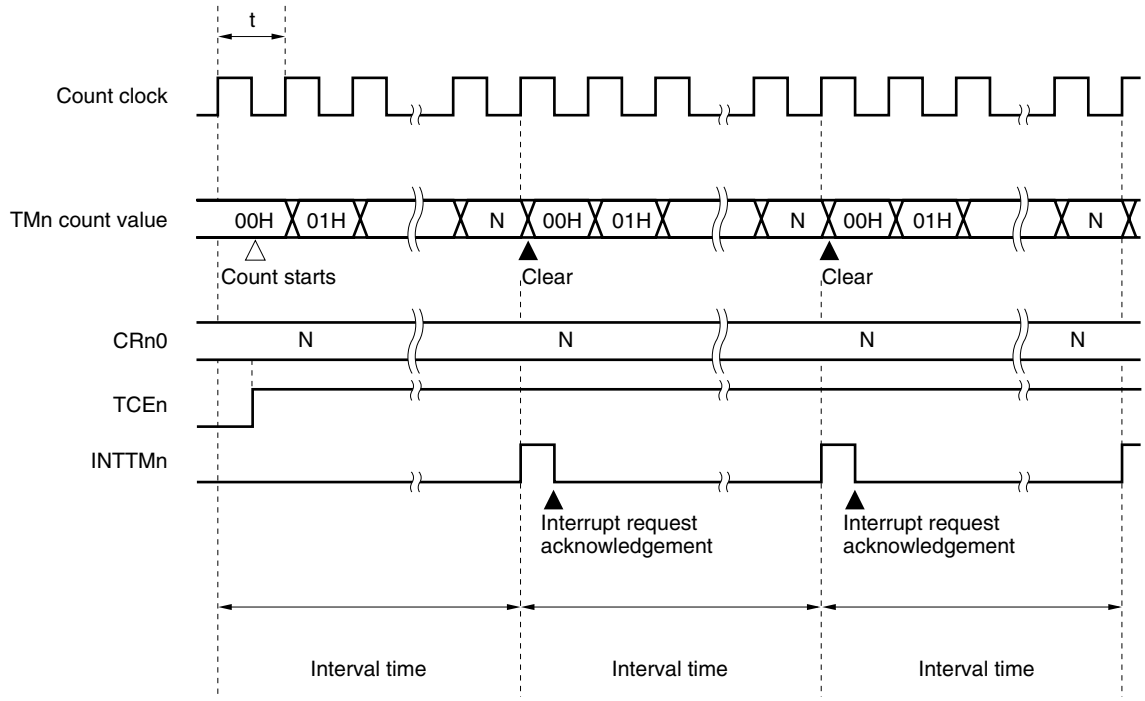
<3> When the values of TMn and CRn0 match, INTTMn is generated (TMn is cleared to 00H).

<4> Then, INTTMn is repeatedly generated at the same interval. When counting stops, set TCEn = 0.

**Remark** n = 5, 6

**Figure 10-6. Timing of Interval Timer Operation (1/3)**

**(a) Basic operation**

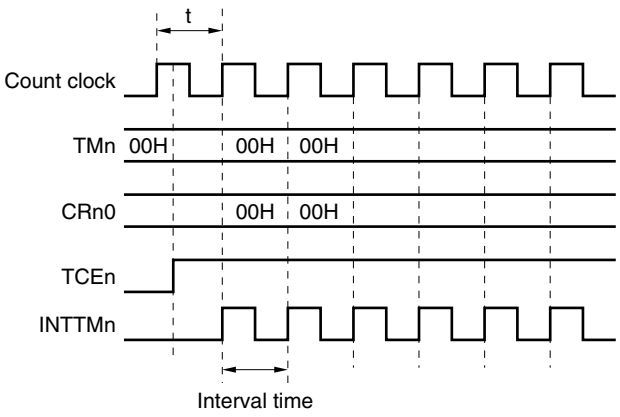



**Remarks 1.** Interval time = (N + 1) × t; N = 00H to FFH

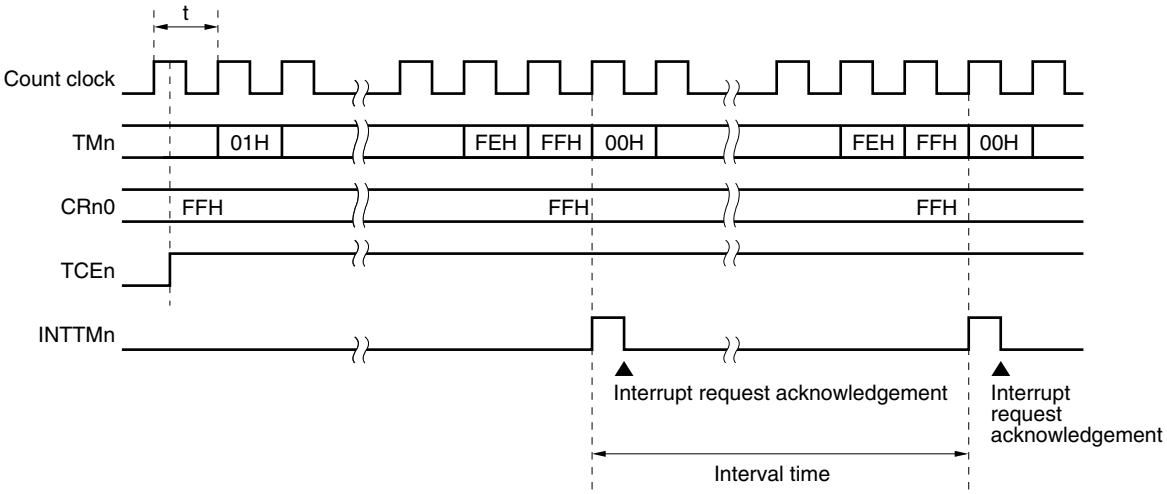
**2.** n = 5, 6

**Figure 10-6. Timing of Interval Timer Operation (2/3)**

**(b) When CRn0 = 00H**



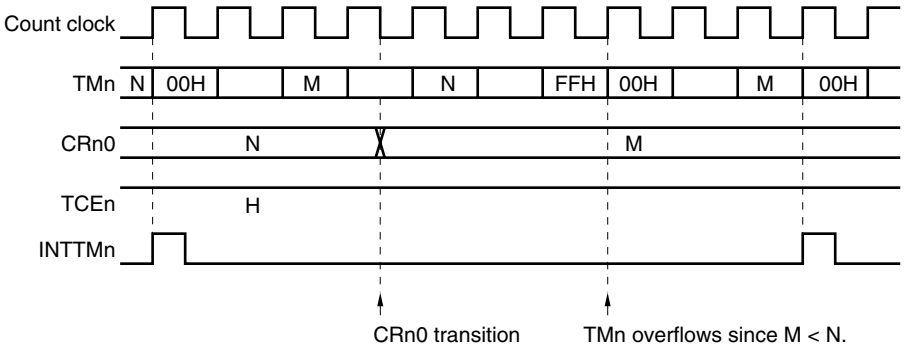

**(c) When CRn0 = FFH**



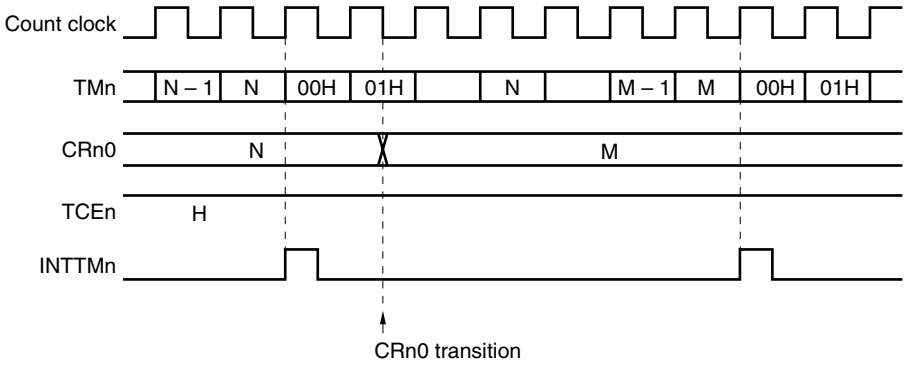

**Remark** n = 5, 6

**Figure 10-6. Timing of Interval Timer Operation (3/3)**

**(d) Operated by CRn0 transition (M < N)**



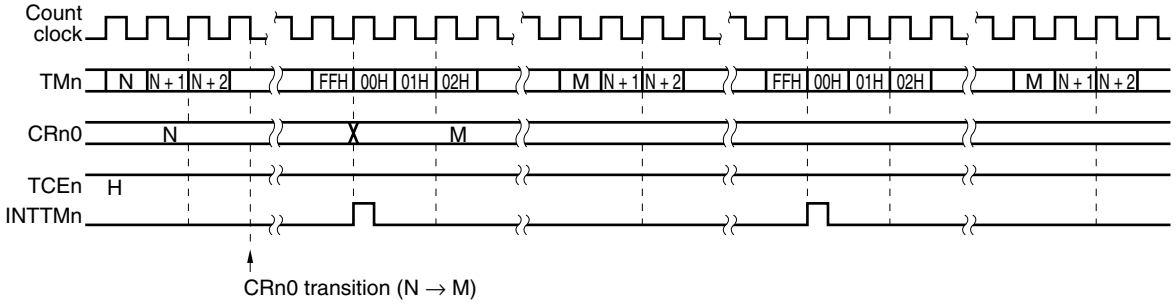

**(e) Operated by CRn0 transition (M > N)**



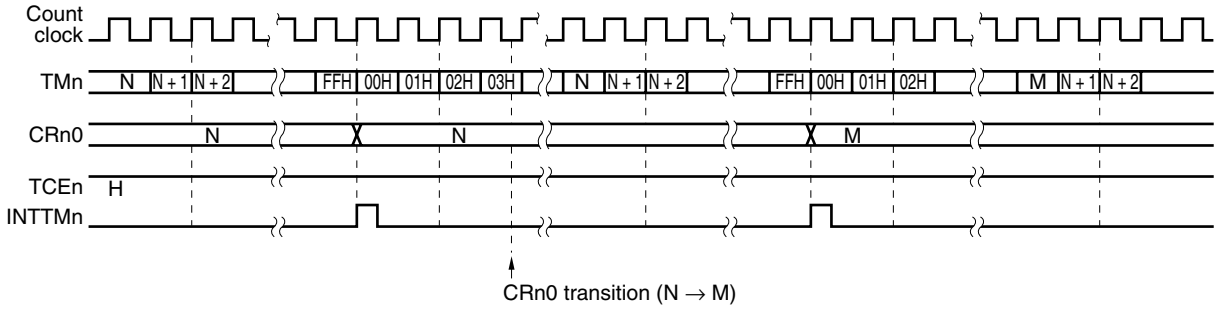

**Remark** n = 5, 6

**Figure 10-7. Timing of Operation Based on CRn0 Transitions**

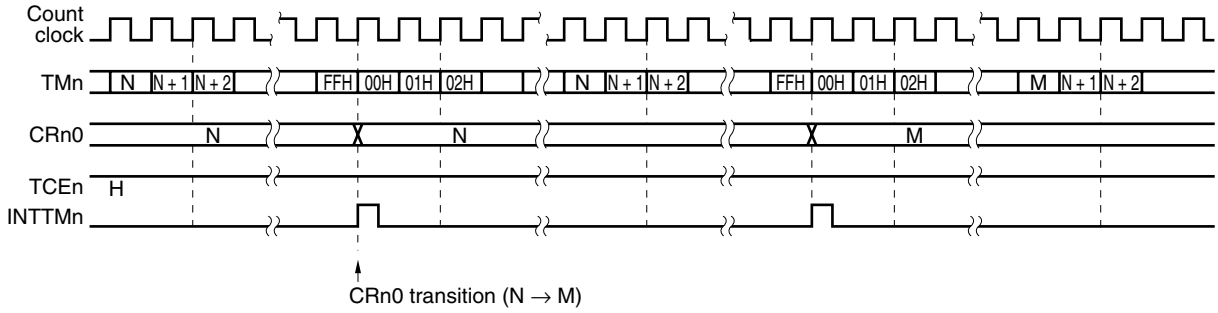
**(a) When the CRn0 value changes from N to M before TMn overflows**




**(b) When the CRn0 value changes from N to M after TMn overflows**




**(c) When the CRn0 value changes from N to M within two clocks (00H, 01H) immediately after TMn overflows**




**Remark** n = 5, 6

### 10.4.2 Operation as interval timer (16-bit operation)

- **Cascade connection (16-bit timer) mode**

  By setting bit 4 (TMC64) of 8-bit timer mode control register 6 (TMC6) to 1, the timer enters the timer mode with 16-bit resolution.

  With the count preset in 8-bit compare registers 50 and 60 (CR50, CR60) as the interval, the timer operates as an interval timer by repeatedly generating interrupt requests.

**<Setting method>**

<1> Set each register.

- PRM5: TM5 selects the count clock. TM6 connected in cascade is not used for setting.
- CRn0: Compare values (each compare value can be set from 00H to FFH).
- TMCn: Select the clear and start mode when TMn and CRn0 match.

  TM5 → TMC5 = 0000×××0B, ×: Don't care
  TM6 → TMC6 = 0001×××0B, ×: Don't care

<2> Setting TCE6 = 1 for TMC6 and setting TCE5 = 1 for 8-bit timer mode control register 5 (TMC5) starts the count operation.

<3> If the values of TMn of all timers connected in cascade and CRn0 match, INTTM5 of TM5 is generated. (TM5 and TM6 are cleared to 00H.)

<4> INTTM5 are repeatedly generated at the same interval.

**Cautions 1. Always set the compare register (CR50, CR60) after stopping timer operation.**

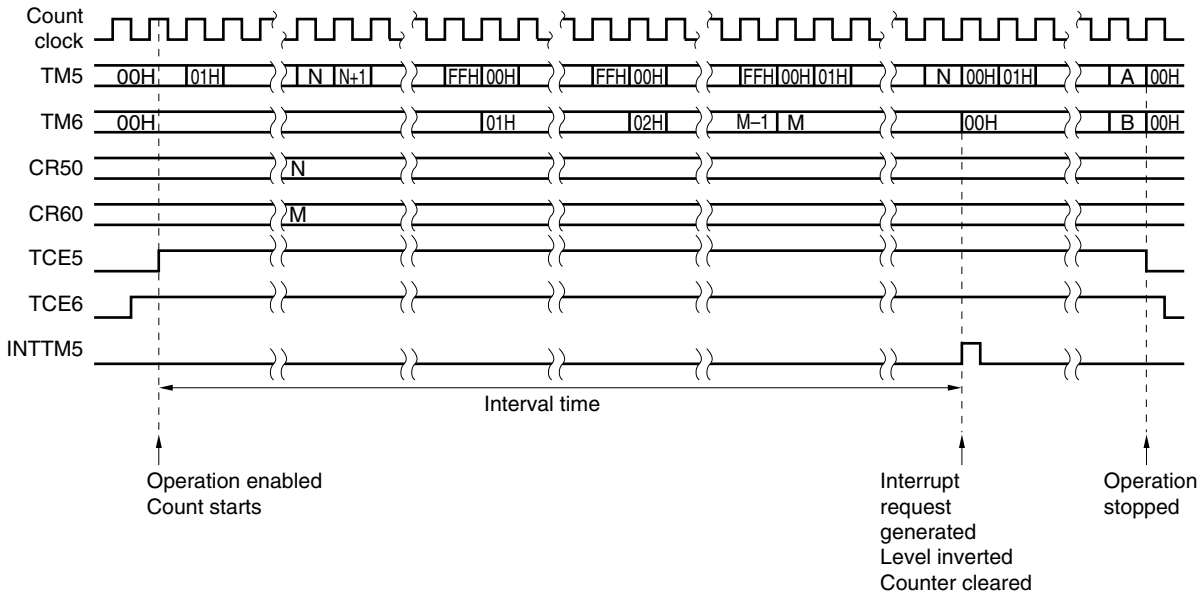
**2. If the TM6 count value matches CR60 even when used in a cascade connection, INTTM6 of TM6 is generated. Always mask TM6 in order to disable interrupts.**

**3. Set TCE5 and TCE6 in TM6 first. Set TM5 last.**

**4. Restarting and stopping the count is possible by setting 1 or 0 only in TCE5 of TMC5. Note, however, that TCE5 of TMC5 and bit 7 (TCE6) of TMC6 must be cleared when setting CR50 and CR60.**

Figure 10-8 shows a timing example of the cascade connection mode with 16-bit resolution.

**Figure 10-8. Cascade Connection Mode with 16-Bit Resolution**

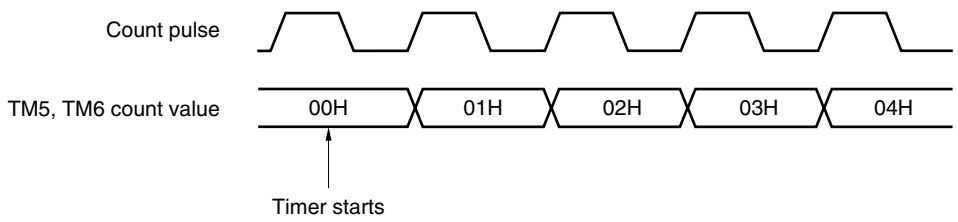


## 10.5 Cautions

**(1) Error when the timer starts**

An error of up to 1 clock occurs before the match signal is generated after the timer is started. This is because 8-bit timer counters 5 and 6 (TM5, TM6) are started asynchronously to the count pulse.

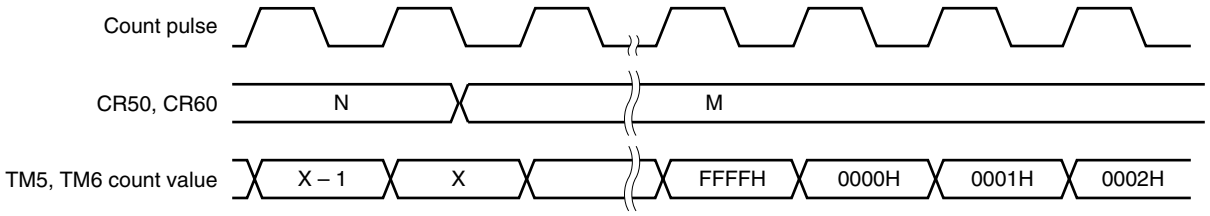
**Figure 10-9. Start Timing of 8-Bit Timer Counter**




**(2) Operation after the compare register is changed while the timer is counting**

If the value after 8-bit compare registers 50 and 60 (CR50, CR60) changes is less than the value of 8-bit timer counters 5 and 6 (TM5, TM6), counting continues, overflows, and counting starts again from 0. Consequently, when the value (M) after CR50 and CR60 change is less than the value (N) before the change, the timer must be restarted after CR50 and CR60 change.

**Figure 10-10. Timing After Compare Register Changes During Timer Counting**




**Caution Except when the TI5, TI6 input is selected, always set TCE5 = 0, TCE6 = 0 before setting the STOP mode.**

**Remark** N > X > M

**(3) TM5, TM6 read out during timer operation**

Since reading out TM5 and TM6 during operation occurs while the selected clock is temporarily stopped, select a high- or low-level waveform that is longer than the selected clock.

When reading TM5 and TM6 in cascade connection mode, to avoid reading while the count is changing, take measures such as obtaining a count match by reading twice using software.

# CHAPTER 11 WATCH TIMER

## 11.1 Function

The watch timer has the following functions.

- Watch timer
- Interval timer

The watch timer and interval timer functions can be used at the same time.

**(1) Watch timer**

The watch timer generates an interrupt request (INTWT) at time intervals of $2^{14}/f_W$ or $2^5/f_W$ by using the main system clock of 4.19 MHz or subsystem clock of 32.768 kHz.

**Caution A time interval of 0.5 second cannot be generated by the 12.5 MHz main system clock. Use the 32.768 kHz subsystem clock to generate the 0.5-second time interval.**

**Remark** $f_W$: Watch timer clock oscillation frequency ($f_{XX}/2^7$ or $f_{XT}$)
$f_{XX}$: Main system clock oscillation frequency
$f_{XT}$: Subsystem clock oscillation frequency

**(2) Interval timer**

The watch timer generates an interrupt request (INTTM3) at time intervals specified in advance.

**Table 11-1. Interval Time of Interval Timer**

| Interval Time | $f_{XX}$ = 12.5 MHz | $f_{XX}$ = 4.19 MHz | $f_{XT}$ = 32.768 kHz |
|---|---|---|---|
| $2^{11} \times 1/f_{XX}$ | 164 $\mu$s | 488 $\mu$s | 488 $\mu$s |
| $2^{12} \times 1/f_{XX}$ | 328 $\mu$s | 977 $\mu$s | 977 $\mu$s |
| $2^{13} \times 1/f_{XX}$ | 655 $\mu$s | 1.95 ms | 1.95 ms |
| $2^{14} \times 1/f_{XX}$ | 1.31 ms | 3.91 ms | 3.91 ms |
| $2^{15} \times 1/f_{XX}$ | 2.62ms | 7.81 ms | 7.81 ms |
| $2^{16} \times 1/f_{XX}$ | 5.24 ms | 15.6 ms | 15.6 ms |

**Remark** $f_{XX}$: Main system clock oscillation frequency
$f_{XT}$: Subsystem clock oscillation frequency

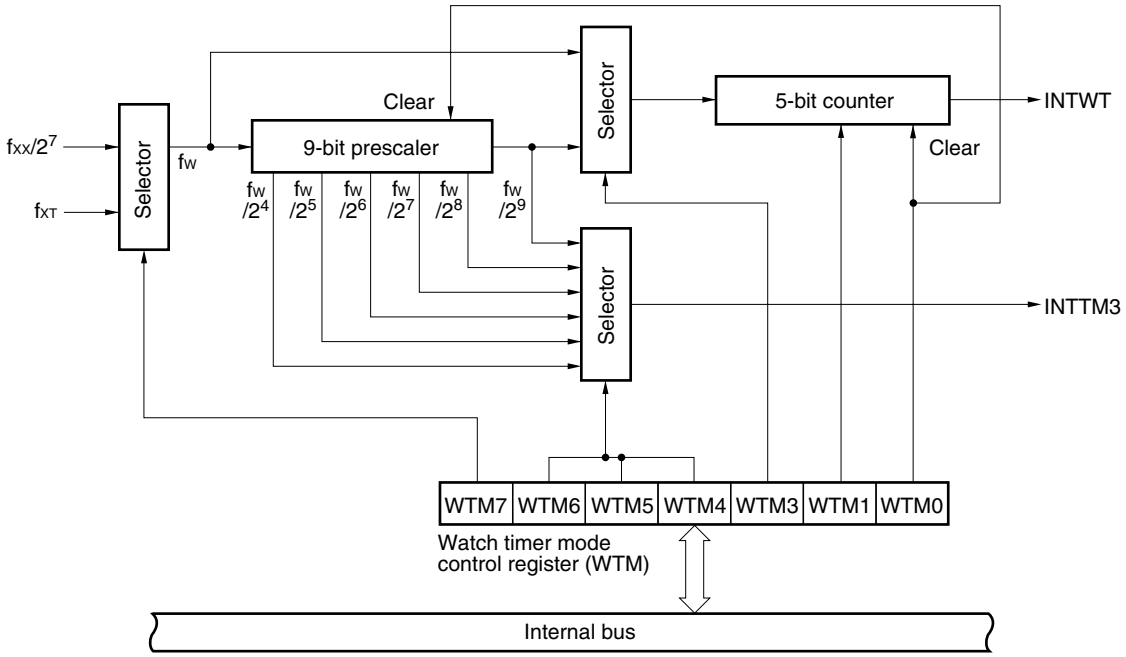
## 11.2 Configuration

The watch timer includes the following hardware.

**Table 11-2. Configuration of Watch Timer**

| Item | Configuration |
|---|---|
| Counter | 5 bits × 1 |
| Prescaler | 9 bits × 1 |
| Control register | Watch timer mode control register (WTM) |

**Figure 11-1. Block Diagram of Watch Timer**




**Remark** fXX: Main system clock oscillation frequency
fXT: Subsystem clock oscillation frequency
fW: Watch timer clock oscillation frequency (fXX/$2^7$ or fXT)

## 11.3 Watch Timer Control Register

- **Watch timer mode control register (WTM)**
  This register enables or disables the count clock and operation of the watch timer, sets the interval time of the prescaler, controls the operation of the 5-bit counter, and sets the set time of the watch flag.
  WTM is set by a 1-bit or 8-bit memory manipulation instruction.
  $\overline{\text{RESET}}$ input sets WTM to 00H.

**Figure 11-2. Format of Watch Timer Mode Control Register (WTM)**

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

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| WTM | WTM7 | WTM6 | WTM5 | WTM4 | WTM3 | 0 | WTM1 | WTM0 |

| WTM7 | Selects count clock of watch timer |
|---|---|
| 0 | Main system clock ($f_{XX}/2^7$) |
| 1 | Subsystem clock ($f_{XT}$) |

| WTM6 | WTM5 | WTM4 | Selects interval time of prescaler |
|---|---|---|---|
| 0 | 0 | 0 | $2^4/f_W$ (488 $\mu$s) |
| 0 | 0 | 1 | $2^5/f_W$ (977 $\mu$s) |
| 0 | 1 | 0 | $2^6/f_W$ (1.95 ms) |
| 0 | 1 | 1 | $2^7/f_W$ (3.91 ms) |
| 1 | 0 | 0 | $2^8/f_W$ (7.81 ms) |
| 1 | 0 | 1 | $2^9/f_W$ (15.6 ms) |
| Other than above | | | Setting prohibited |

| WTM3 | Selects set time of watch flag |
|---|---|
| 0 | $2^{14}/f_W$ (0.5 s) |
| 1 | $2^5/f_W$ (977 $\mu$s) |

| WTM1 | Controls operation of 5-bit counter |
|---|---|
| 0 | Clear after operation stop |
| 1 | Start |

| WTM0 | Controls operation of 5-bit counter |
|---|---|
| 0 | Operation stop (clear both prescaler and timer) |
| 1 | Operation enable |

**Cautions 1. Stop the timer operation before overwriting WTM.**

**2. Do not overwrite WTM when both the watch timer and interval timer are being used. If the timer is stopped to overwrite WTM, both the prescaler and timer are cleared, causing an error to occur for the watch timer interrupt (INTWT).**

**Remarks 1.** $f_W$: Watch timer clock oscillation frequency ($f_{XX}/2^7$ or $f_{XT}$)

$f_{XX}$: Main system clock oscillation frequency

$f_{XT}$: Subsystem clock oscillation frequency

**2.** Figures in parentheses apply to operation at $f_W$ = 32.768 kHz.

## 11.4 Operation

### 11.4.1 Operation as watch timer

The watch timer operates with time intervals of $2^{14}/f_W$ or $2^5/f_W$ with the main system clock (4.19 MHz) or subsystem clock (32.768 kHz).

The watch timer generates an interrupt request (INTWT) at fixed time intervals.

The count operation of the watch timer is started when bits 0 (WTM0) and 1 (WTM1) of the watch timer mode control register (WTM) are set to 1. When these bits are cleared to 0, the 5-bit counter is cleared, and the watch timer stops the count operation.

When the interval timer function is started at the same time, the watch timer can be started from 0 seconds by resetting WTM1 to 0. However, an error of up to $2^{14}/f_W$ or $2^5/f_W$ may occur when the watch timer overflows (INTWT).

**Caution A time interval of 0.5 second cannot be generated by the 12.5 MHz main system clock. Use the 32.768 kHz subsystem clock to generate the 0.5-second time interval**

### 11.4.2 Operation as interval timer

The watch timer can also be used as an interval timer that repeatedly generates an interrupt request (INTTM3) at intervals specified by a count value set in advance.

The interval time can be selected by bits 4 through 6 (WTM4 through WTM6) of the watch timer mode control register (WTM).

**Table 11-3. Interval Time of Interval Timer**

| WTM6 | WTM5 | WTM4 | Interval Time | $f_{XX}$ = 12.5 MHz | $f_{XX}$ = 4.19 MHz | $f_{XT}$ = 32.768 kHz |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | $2^4 \times 1/f_W$ | 164 $\mu$s | 488 $\mu$s | 488 $\mu$s |
| 0 | 0 | 1 | $2^5 \times 1/f_W$ | 328 $\mu$s | 977 $\mu$s | 977 $\mu$s |
| 0 | 1 | 0 | $2^6 \times 1/f_W$ | 655 $\mu$s | 1.95 ms | 1.95 ms |
| 0 | 1 | 1 | $2^7 \times 1/f_W$ | 1.31 ms | 3.91 ms | 3.91 ms |
| 1 | 0 | 0 | $2^8 \times 1/f_W$ | 2.62 ms | 7.81 ms | 7.81 ms |
| 1 | 0 | 1 | $2^9 \times 1/f_W$ | 5.24 ms | 15.6 ms | 15.6 ms |
| Other than above | | | Setting prohibited | | | |

**Cautions 1. Stop the timer operation before overwriting WTM.**

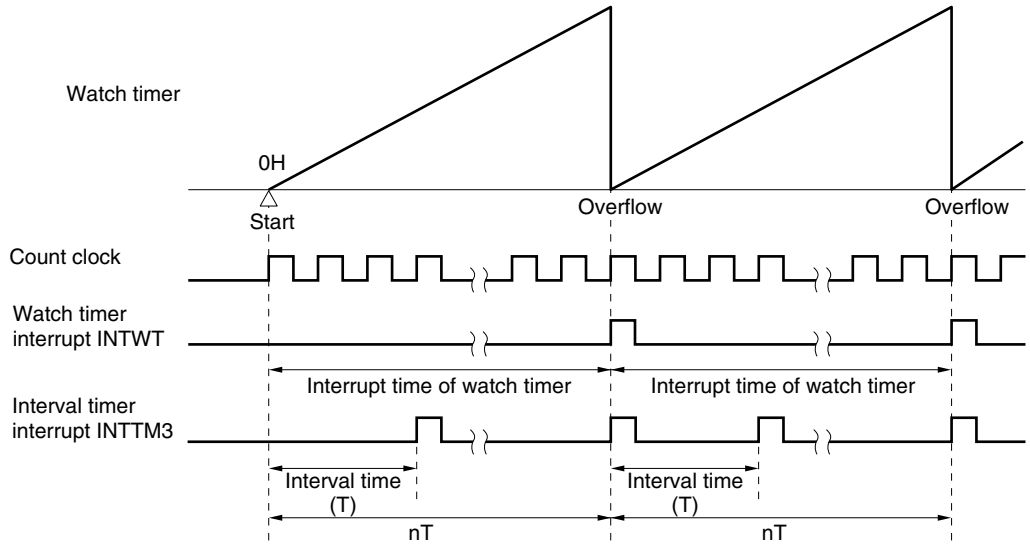
**2. Do not overwrite WTM when both the watch timer and interval timer are being used. If the timer is stopped to overwrite WTM, both the prescaler and timer are cleared, causing an error to occur for the watch timer interrupt (INTWT).**

**Remark** $f_W$: Watch timer clock oscillation frequency ($f_{XX}/2^7$ or $f_{XT}$)

$f_{XX}$: Main system clock frequency

$f_{XT}$: Subsystem clock oscillation frequency

**Figure 11-3. Operation Timing of Watch Timer/Interval Timer**




**Caution When enabling operation of the watch timer mode control register (WTM), watch timer, and 5-bit counter, the time until the first watch timer interrupt request (INTWT) is generated is not exactly the same time as set by bits 4 to 6 of WTM (WTM4 to WTM6). This is because the 5-bit counter starts counting 1 cycle after 9-bit prescaler output. Following the first INTWT generation, the INTWT signal is generated at the set interval time.**

**Remarks** n: Number of interval timer operations

# CHAPTER 12 WATCHDOG TIMER

The watchdog timer detects inadvertent program loops.

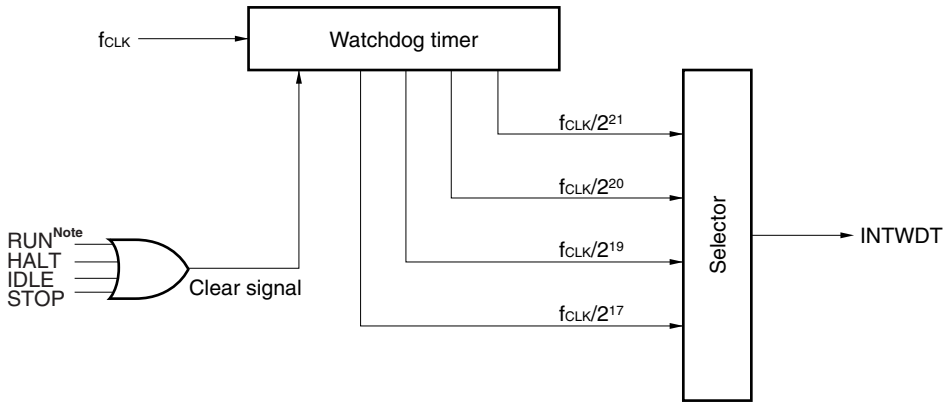
Program or system errors are detected by the generation of watchdog timer interrupts. Therefore, at each location in the program, the instruction that clears the watchdog timer (starts the count) within a constant time is input.

If the watchdog timer overflows without executing the instruction that clears the watchdog timer within the set period, a watchdog timer interrupt (INTWDT) is generated to signal a program error.

## 12.1 Configuration

Figure 12-1 is a block diagram of the watchdog timer.

**Figure 12-1. Block Diagram of Watchdog Timer**




**Note** Write 1 to bit 7 (RUN) of the watchdog timer mode register (WDM).

**Remark** $f_{CLK}$: Internal system clock ($f_{XX}$ to $f_{XX}/8$)

## 12.2 Control Register

- **Watchdog timer mode register (WDM)**

  The WDM is the 8-bit register that controls watchdog timer operation.

  To prevent the watchdog timer from erroneously clearing this register due to an inadvertent program loop, this register is only written by a special instruction. This special instruction has a special code format (4 bytes) in the MOV WDM #byte instruction. Writing takes place only when the third and fourth opcodes are mutual 1's complements. If the third and fourth opcodes 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 inadvertently loops (the correct special instruction is only generated when MOV WDM #byte is described in the NEC assembler RA78K4), make the program initialize the system.

  Other write instructions (MOV WDM, A; AND WDM, #byte instruction, SET1 WDM, etc.) are ignored and nothing happens. In other words, WDM is not written, and interrupts, such as operand error interrupts, are not generated.

  After a system reset ($\overline{\text{RESET}}$ input), when the watchdog timer starts (when the RUN bit is set to 1), 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 by an 8-bit data transfer instruction.

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

  Figure 12-2 shows the WDM format.

**Figure 12-2. Format of Watchdog Timer Mode Register (WDM)**

Address: 0FFC2H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| WDM | RUN | 0 | 0 | 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<br><NMI pin input interrupt request |
| 1 | Watchdog timer interrupt request<br>>NMI pin input interrupt request |

| WDT2 | WDT1 | Count clock | Overflow time [ms] ($f_{CLK}$ = 12.5 MHz) |
|---|---|---|---|
| 0 | 0 | $f_{CLK}/2^{17}$ | 10.5 |
| 0 | 1 | $f_{CLK}/2^{19}$ | 41.9 |
| 1 | 0 | $f_{CLK}/2^{20}$ | 83.9 |
| 1 | 1 | $f_{CLK}/2^{21}$ | 167.8 |

**Cautions 1. The watchdog timer mode register (WDM) can only by written by a special instruction (MOV WDM, #byte).**

**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. Once the RUN bit is set to 1, it cannot be reset to 0 by the software.**

**Remark** $f_{CLK}$: Internal system clock ($f_{XX}$ to $f_{XX}/8$)

$f_{XX}$: Main system clock oscillation frequency

## 12.3 Operations

### 12.3.1 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 before the overflow time elapses, the watchdog timer is cleared, and counting restarts.

### 12.3.2 Interrupt priority order

The watchdog timer interrupt (INTWDT) can be specified as either maskable or non-maskable according to the interrupt selection control register (SNMI) setting. When writing 0 to bit 1 (SWDT) of SNMI, the watchdog timer interrupt can be used as a non-maskable interrupt. In addition to INTWDT, non-maskable interrupts include the interrupt (NMI) from the NMI pin. By setting bit 4 of the watchdog timer mode register (WDM), the acknowledgment order when INTWDT and NMI are simultaneously generated can be set.

If acknowledging the NMI is given priority, even if INTWDT is generated in an NMI servicing program that is being executed, INTWDT is not acknowledged, but is acknowledged after the NMI servicing program ends.

## 12.4 Cautions

### 12.4.1 General cautions when using the watchdog timer

(1) The watchdog timer is one way to detect an inadvertent program loop, but not all the program loops can be detected. Therefore, in a device that demands particularly high reliability, the inadvertent program loop 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 inadvertent program loops in the following cases.

<1> When the watchdog timer is cleared in a timer interrupt servicing program
<2> When there are successive temporary stores of interrupt requests and macro services (see **22.9 When Interrupt Requests and Macro Service Are Temporarily Held Pending**)
<3> When an inadvertent program loop 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 an inadvertent program loop
<5> When the STOP mode and HALT mode or IDLE mode is the result of an inadvertent program loop
<6> When the watchdog timer also inadvertently loops when the CPU hangs up because of introduced noise

In cases <1>, <2>, and <3>, detection becomes possible by correcting the program.
In case <4>, the watchdog timer can be cleared only by the 4-byte special instruction. Similarly in <5>, if there is no 4-byte special instruction, the STOP mode and HALT mode or IDLE mode cannot be set. Since the result of the inadvertent program loop is to enter state <2>, three or more bytes of consecutive data must be a specific pattern (example, BT PSWL.bit, $$). Therefore, the results of <4>, <5>, and the inadvertent program loop are believed to very rarely enter state <2>.

### 12.4.2 Cautions about the μPD784225 Subseries watchdog timer

(1) The watchdog timer mode register (WDM) can only be written by a special instruction (MOV WDM, #byte).

(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) Once the RUN bit is set to 1, it cannot be reset to 0 by the software.

# CHAPTER 13 A/D CONVERTER

## 13.1 Functions

The A/D converter converts analog inputs into 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 (ADCR).

A/D conversion can be started by the following two methods.

**(1) Hardware start**

Conversion is started by trigger input (P03) (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 (ADM).

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.

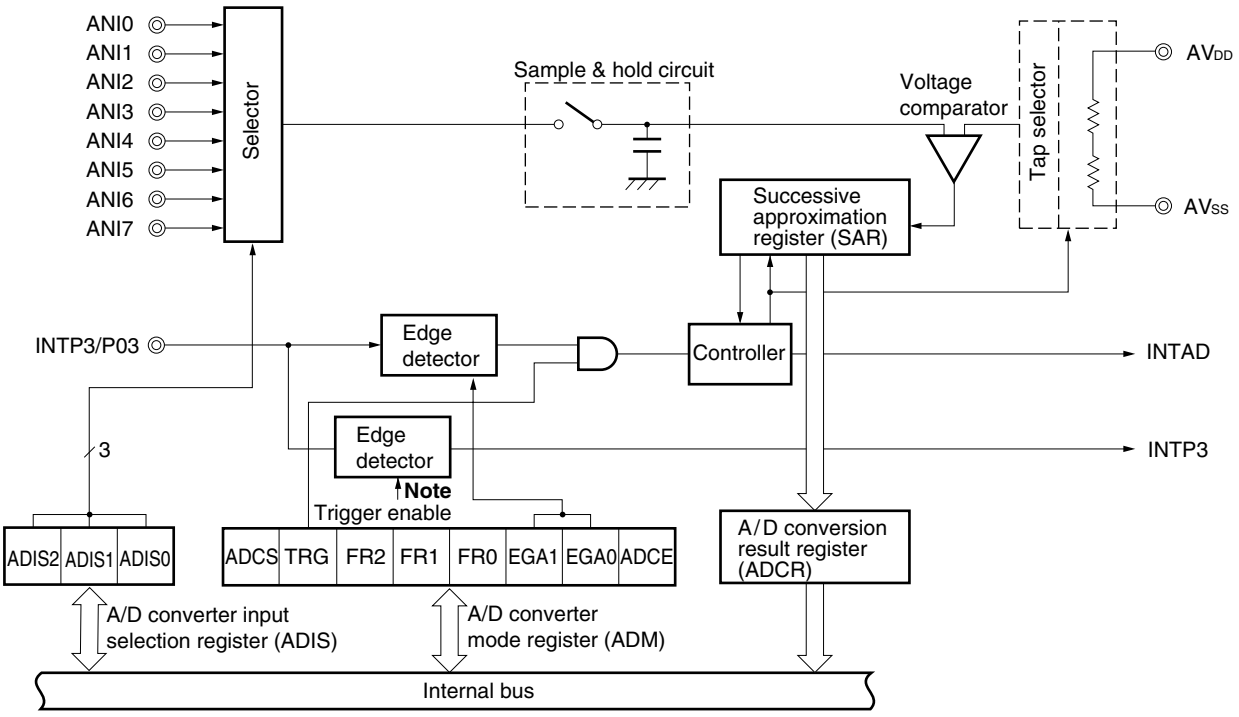
## 13.2 Configuration

The A/D converter includes the following hardware.

**Table 13-1. Configuration of A/D Converter**

| Item | Configuration |
|---|---|
| Analog input | 8 channels (ANI0 to ANI7) |
| Control registers | A/D converter mode register (ADM)<br>A/D converter input selection register (ADIS) |
| Registers | Successive approximation register (SAR)<br>A/D conversion result register (ADCR) |

**Figure 13-1. Block Diagram of A/D Converter**




**Note** Valid edge specified with bit 3 (EGP3, EGN3) of external interrupt rising edge/falling edge enable register 0 (EGP0, EGN0) (Refer to **Figure 21-1 Format of External Interrupt Rising Edge Enable Register 0 (EGP0) and External Interrupt Falling Edge Enable Register 0 (EGN0)**).

**(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 after the least significant bit (LSB) is saved (A/D conversion finished).

**(2) A/D conversion result register (ADCR)**

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 operation.

$\overline{\text{RESET}}$ input makes ADCR 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 saved 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 $AV_{DD}$ and $AV_{SS}$, generates the voltage that is compared with that of the 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 A/D converter input selection register (ADIS) can be used as input ports.

**Cautions 1. Use the ANI0 to ANI7 input voltages within the rated voltage range. Inputting a voltage equal to or greater than $AV_{DD}$, 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 pins (AN10 to AN17) function alternately as input port pins (P10 to P17). When performing an A/D conversion with any one of the AN10 to AN17 inputs selected, do not execute input instructions to port 1 during conversion, otherwise the conversion resolution may decrease. When a digital pulse is applied to a pin that adjoins the pin undergoing A/D conversion, the expected A/D conversion value may not be acquired due to coupling noise. Therefore do not apply a pulse to a pin that adjoins the pin undergoing A/D conversion.**

**(7) $AV_{SS}$ pin**

Ground pin of the A/D converter. Always use this pin at the same potential as the $V_{SS}$ pin, even when not using the A/D converter.

**(8) $AV_{DD}$ pin**

Analog power supply pin and reference voltage input pin of the A/D converter. Always use this pin at the same potential as the $V_{DD}$ pin, even when not using the A/D converter.

## 13.3 Control Registers

The A/D converter is controlled by the following two registers.

- A/D converter mode register (ADM)
- A/D converter input selection register (ADIS)

**(1) A/D converter mode register (ADM)**

Used to set the A/D conversion time of the analog input to be converted, start/stop of the conversion operation, and external triggers.

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

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

**Figure 13-2. Format of A/D Converter Mode Register (ADM)**

Address: 0FF80H After reset: 00H R/W

| Symbol | ⑦ | ⑥ | ⑤ | ④ | ③ | ② | ① | ⓪ |
|---|---|---|---|---|---|---|---|---|
| ADM | ADCS | TRG | FR2 | FR1 | FR0 | EGA1 | EGA0 | ADCE |

| ADCS | A/D conversion control |
|---|---|
| 0 | Conversion stop |
| 1 | Conversion enable |

| TRG | Software start/hardware start selection |
|---|---|
| 0 | Software start |
| 1 | Hardware start |

| FR2 | FR1 | FR0 | A/D conversion time selection | | |
|---|---|---|---|---|---|
| | | | Number of clocks | @fxx = 12.5 MHz | @fxx = 6.25 MHz |
| 0 | 0 | 0 | 144/fxx | Setting prohibited | 23.0 μs |
| 0 | 0 | 1 | 120/fxx | Setting prohibited | 19.2 μs |
| 0 | 1 | 0 | 96/fxx | Setting prohibited | 15.4 μs |
| 1 | 0 | 0 | 288/fxx | 23.0 μs | 46.1 μs |
| 1 | 0 | 1 | 240/fxx | 19.2 μs | 38.4 μs |
| 1 | 1 | 0 | 192/fxx | 15.4 μs | 30.7 μs |
| Other than above | | | – | Setting prohibited | |

| EGA1 | EGA0 | 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 |

| ADCE | Reference voltage circuit control |
|---|---|
| 0 | Circuit stopped[Note] |
| 1 | Circuit operating |

**Note** The reference voltage circuit operates when ADCS is 1

**Cautions 1. Set the A/D conversion time as follows.**

**When $V_{DD}$ = 2.7 V to 5.5 V: 14 μs or more**

★ **When $V_{DD}$ = 2.0 V to 2.7 V: 28 μs or more**

**When $V_{DD}$ = 1.9 V to 2.0 V: 48 μs or more (μPD78F4225, 78F4225Y)**

**When $V_{DD}$ = 1.8 V to 2.0 V: 48 μs or more (μPD784224, 784225, 784224Y, 784225Y)**

**2. When overwriting FR0 to FR2 to different data, temporarily halt the A/D conversion operations before continuing.**

**3. If ADCS is set after ADCE is set and the following time has elapsed, the first A/D conversion value can be used.**

**When $V_{DD}$ = 2.7 V to 5.5 V: 14 μs or more**

★ **When $V_{DD}$ = 2.0 V to 2.7 V: 28 μs or more**

**When $V_{DD}$ = 1.9 V to 2.0 V: 48 μs or more (μPD78F4225, 78F4225Y)**

**When $V_{DD}$ = 1.8 V to 2.0 V: 48 μs or more (μPD784224, 784225, 784224Y, 784225Y)**

**4. If ADCS is set when ADCE = 0, the first A/D conversion value is undefined.**

**Remark** fxx: Main system clock frequency (fx or fx/2)
fx: Main system clock oscillation frequency

### (2) A/D converter input selection register (ADIS)

Used to specify the input ports for analog signals to be A/D converted.

ADIS can be set by a 1-bit or 8-bit memory manipulation instruction.

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

**Figure 13-3. Format of A/D Converter Input Selection Register (ADIS)**

Address: 0FF81H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ADIS | 0 | 0 | 0 | 0 | 0 | ADIS2 | ADIS1 | ADIS0 |

| ADIS2 | ADIS1 | ADIS0 | 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 |

## 13.4 Operations

### 13.4.1 Basic operations of A/D converter

<1> Select one channel for A/D conversion with the A/D converter input selection register (ADIS).

<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 resistance string at (1/2)$AV_{DD}$.

<5> The difference in voltage between the series resistance string's voltage tap and analog input is compared with the voltage comparator. If the analog input is greater than (1/2)$AV_{DD}$, the setting for the MSB in SAR will remain the same. If it is smaller than (1/2)$AV_{DD}$, 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)$AV_{DD}$
- Bit 7 = 0: (1/4)$AV_{DD}$

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

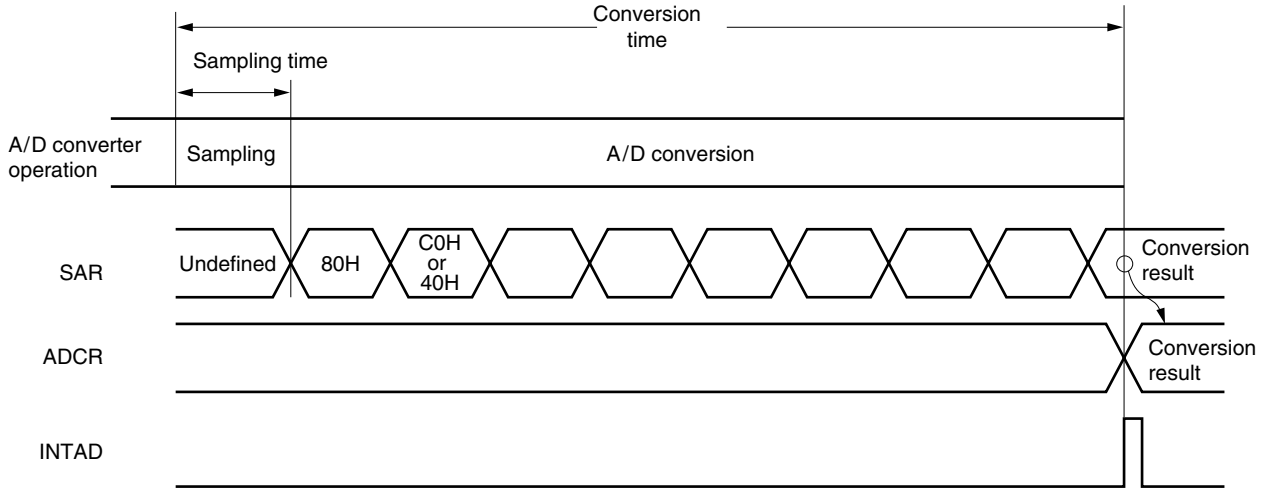
<7> Comparisons of this kind are repeated until bit 0 of SAR.

<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 and latched.

At the same time, it is possible to issue an A/D conversion end interrupt request (INTAD).

**Caution The value of the first A/D conversion is undefined if ADCS is set when bit 0 (ADCE) of the A/D converter mode register (ADM) is 0.**

**Figure 13-4. Basic Operations of A/D Converter**




A/D conversion is performed continuously until bit 7 (ADCS) of the A/D converter mode register (ADM) is reset 0 by software.

If a write operation to ADM or the A/D converter input selection register (ADIS) is performed during A/D conversion, the conversion operation is initialized and conversion starts from the beginning if the ADCS bit is set 1.

$\overline{\text{RESET}}$ input makes the A/D conversion result register (ADCR) undefined.

If bit 0 (ADCE) of the A/D converter mode register is not set to 1, the value of the first A/D conversion is undefined immediately after A/D conversion starts. Poll the A/D conversion end interrupt request (INTAD) and discard the first A/D conversion result.

### 13.4.2 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 the A/D conversion result register (ADCR)) is expressed by the following equation.

$$ADCR = INT\left(\frac{V_{IN}}{AV_{DD}} \times 256 + 0.5\right)$$

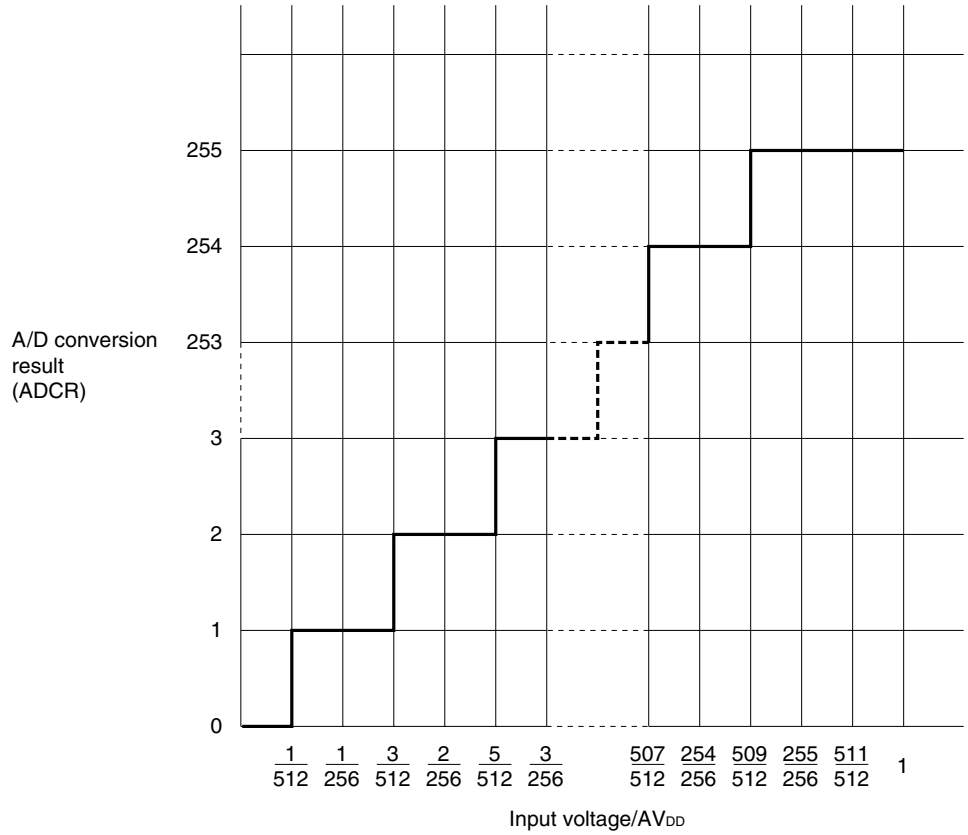
or

$$(ADCR - 0.5) \times \frac{AV_{DD}}{256} \leq V_{IN} < (ADCR + 0.5) \times \frac{AV_{DD}}{256}$$

**Remark** INT( ): Function returning the integer portion of the value in parentheses
$V_{IN}$: Analog input voltage
$AV_{DD}$: $AV_{DD}$ pin voltage
ADCR: A/D conversion result register (ADCR) value

Figure 13-5 shows the relationship between the analog input voltage and the A/D conversion result.

**Figure 13-5. Relationship Between Analog Input Voltage and A/D Conversion Result**



### 13.4.3 Operation modes of A/D converter

Select one channel for analog input from between ANI0 to ANI7 with the A/D converter input selection register (ADIS) and commence A/D conversion.

A/D conversion can be started in the following two ways.

- Hardware start: Conversion start by trigger input (P03)
- Software start: Conversion start by setting A/D converter mode register (ADM)

In addition to this, the result of A/D conversion will be stored in the A/D conversion result register (ADCR), 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 (ADCS) of the A/D converter mode register (ADM). When an external trigger signal (P03) is input, conversion of the voltage applied to the analog input pin set with ADIS begins.

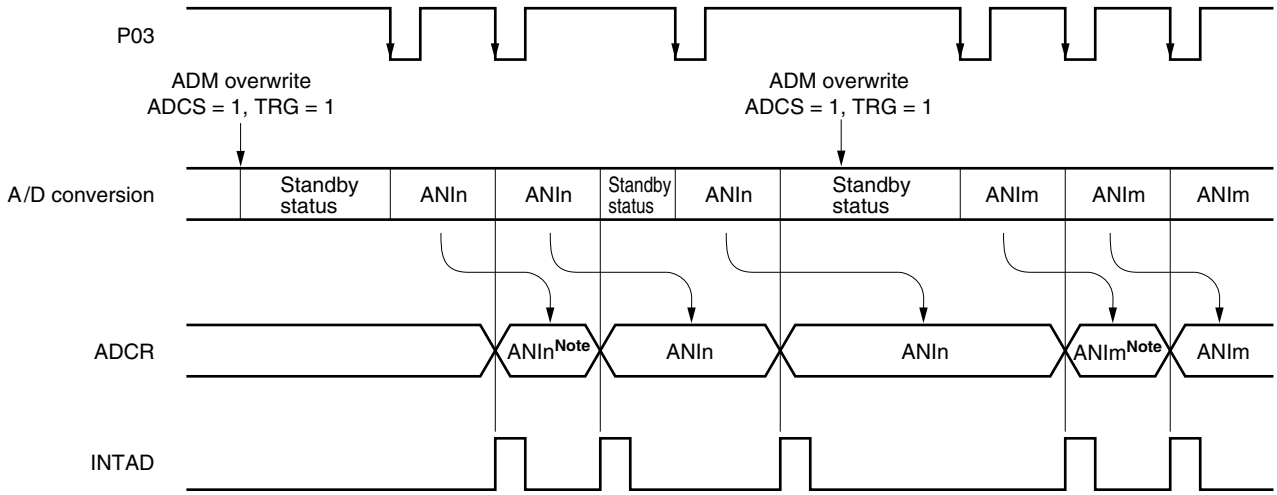
The result of conversion will be stored in the A/D conversion result register (ADCR) when the A/D conversion operation has finished, and an interrupt request signal (INTAD) will be issued. When the A/D conversion operation that was started completes the first A/D conversion, no other A/D conversion operation is started unless an external trigger signal is input.

The A/D conversion process will be suspended if ADCS is overwritten during A/D conversion, and it will enter a standby mode 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 standby mode.

If, during A/D conversion, data where ADCS is 0 is written to ADM, A/D conversion is immediately stopped.

**Caution When P03/INTP3 is used as the external trigger input (P03), specify the valid edge with bits 1 and 2 (EGA0 and EGA1) of the A/D converter mode register (ADM) and set the interrupt mask flag (PMK3) to 1.**

**Figure 13-6. A/D Conversion Operation by Hardware Start (When Falling Edge Is Specified)**




**Remark** n = 0, 1, ..... , 7
m = 0, 1, ..... , 7

**Note** If bit 0 (ADCE) of the A/D converter mode register is not set to 1, the value of the first A/D conversion is undefined immediately after A/D conversion starts. Poll the A/D conversion end interrupt request (INTAD) and discard the first A/D conversion result.

**(2) A/D conversion operation by software start**

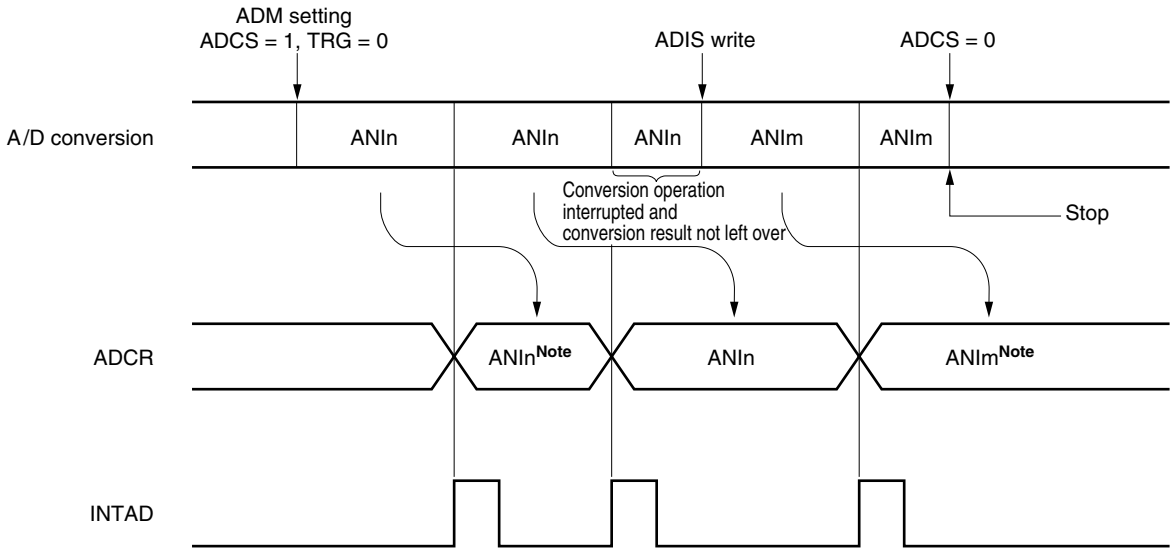
A/D conversion of the voltage applied to the analog input pin specified with the A/D converter input selected register (ADIS) is started by setting “0” to bit 6 (TRG) and “1” to bit 7 (ADCS) of the A/D converter mode register (ADM).

When A/D conversion ends, the conversion result is saved in the A/D conversion result register (ADCR), 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 ADM.

The A/D conversion process will be suspended if ADCS is overwritten during A/D conversion, and an A/D conversion operation for the newly selected analog input channel will be started.

If, during A/D conversion, data where ADCS is 0 is written to ADM, the A/D conversion operation is immediately stopped.

**Figure 13-7. A/D Conversion Operation by Software Start**




**Remark** n = 0, 1, ..... , 7
m = 0, 1, ..... , 7

**Note** If bit 0 (ADCE) of the A/D converter mode register is not set to 1, the value of the first A/D conversion is undefined immediately after A/D conversion starts. Poll the A/D conversion end interrupt request (INTAD) and discard the first A/D conversion result.

## 13.5 Reading A/D Converter Characteristics Table

Words used specifically for the A/D converter are defined below.

**(1) Resolution**

The lowest identifiable analog input voltage, or the ratio of the analog input voltage to one bit of a digital output, is known as 1LSB (least significant bit). The ratio to the full scale of 1LSB is expressed as %FSR (full-scale range).

In the case of 8-bit resolution:

$1LSB = 1/2^8 = 1/256$
$= 0.4\%FSR5$

The accuracy is unrelated to the resolution, and is determined by the overall error.

**(2) Overall error**

This indicates the maximum difference between the actual and theoretical measurement values.
Zero-scale error, full-scale error, integral linearity error, differential linearity error, and combinations of these errors are expressed as the overall error.
Note that the quantization error is not included in the overall error.

**(3) Quantization error**

When analog values are converted to digital values, an error of ±1/2 LSB inevitably occurs. In an A/D converter, because analog input voltages within a range of ± 1/2 LSB are converted into the same digital code, this quantization error cannot be avoided.
Note that the quantization error is not included in the overall error, zero-scale error, full-scale error, integral linearity error, and differential linearity error values shown in the characteristics table.

**Figure 13-8. Overall Error**

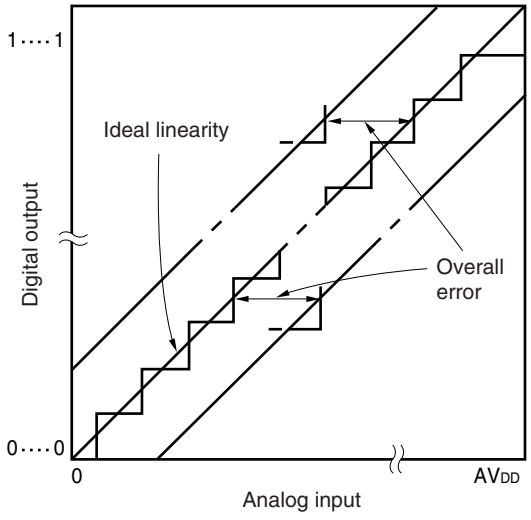
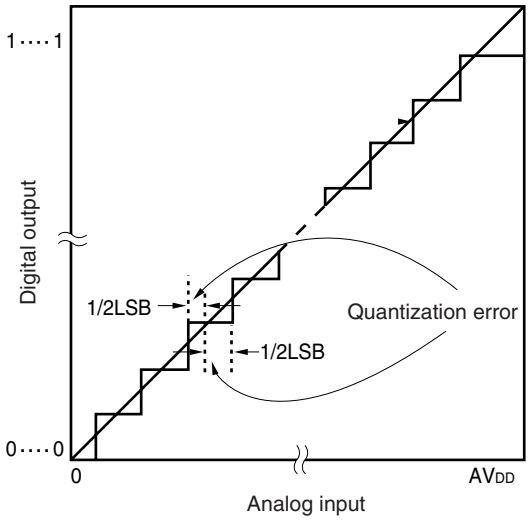



**Figure 13-9. Quantization Error**



**(4) Zero-scale error**

This expresses the difference between the actual and theoretical analog input voltage measurement values when the digital output changes from 0·····000 to 0·····001 (1/2LSB). If the actual value is higher than the theoretical value, the zero-scale error indicates the difference between the actual and theoretical analog input voltage measurement values when the digital output changes from 0·····001 to 0·····010 (3/2LSB).

**(5) Full-scale error**

This expresses the difference between the actual and theoretical analog input voltage measurement values when the digital output changes from 1·····110 to 1·····111 (full-scale – 3/2LSB).

**(6) Integral linearity error**

This expresses the extent to which the conversion characteristics differ from the theoretical linear relationship. The integral linearity error indicates the maximum error between the actual and theoretical measurement values when the zero-scale and full-scale errors are both 0.

**(7) Differential linearity error**

This expresses the difference between the actual and theoretical measurement values of the width output by a certain code when the width output by a theoretical code is 1LSB.

**Figure 13-10. Zero-Scale Error**

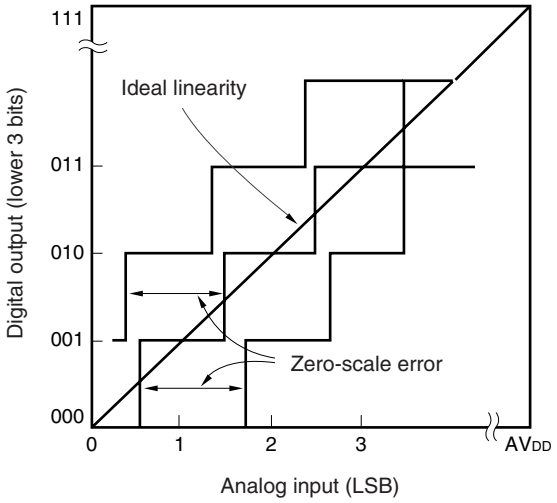



**Figure 13-11. Full-Scale Error**

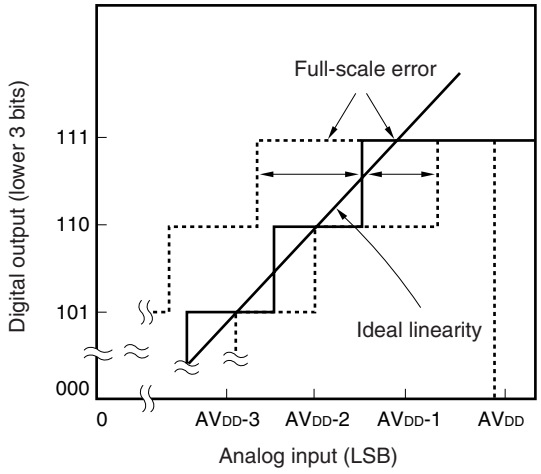


**Figure 13-12. Integral Linearity Error**

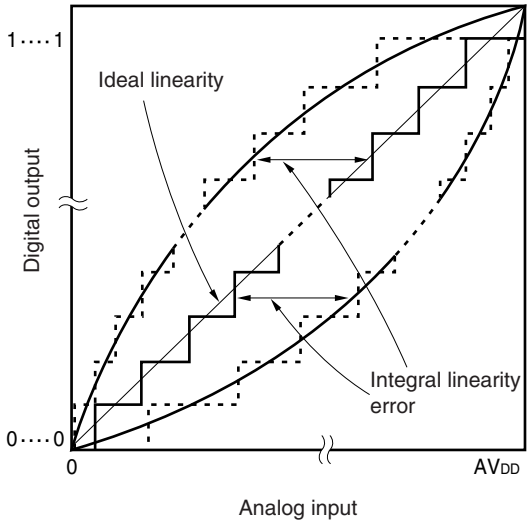
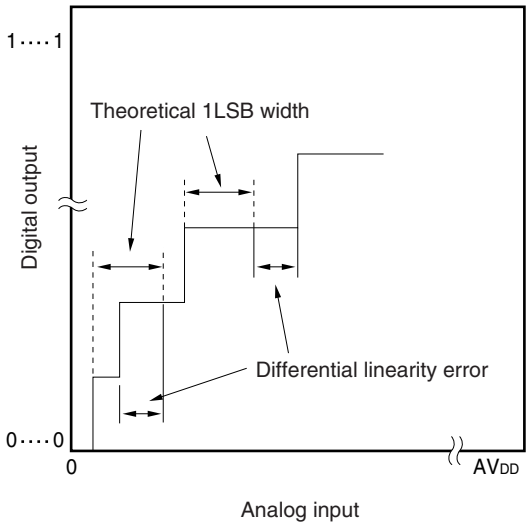



**Figure 13-13. Differential Linearity Error**

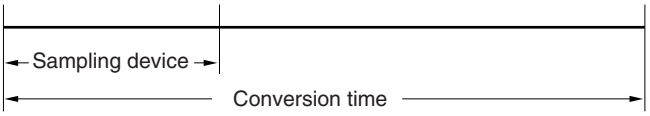



**(8) Conversion time**

This expresses the time from when the analog input voltage is applied to when the digital output is obtained. The sampling time is included in the conversion time value shown in the characteristic table.

**(9) Sampling time**

This is the time during which the analog switch is on to allow the analog voltage to be fetched in the sample & hold circuit.



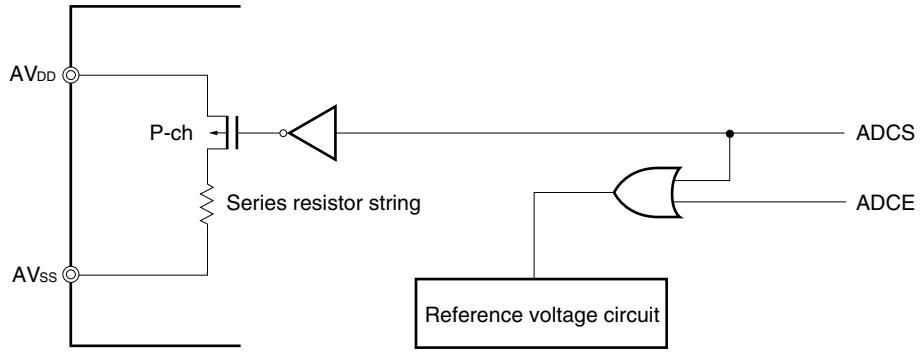
## 13.6 Cautions

**(1) Current consumption in standby mode**

The A/D converter operation is stopped in the standby mode. At this time, the current consumption can be reduced by setting bit 7 (ADCS) of the A/D converter mode register (ADM) to 0 or by stopping the reference voltage circuit (bit 0 of ADM (ADCE) = 0).

The method to reduce the current consumption in the standby mode is shown in Figure 13-14.

**Figure 13-14. Method to Reduce Current Consumption in Standby Mode**




**(2) ANI0 to ANI7 input range**

Use the ANI0 to ANI7 input voltages within the rated voltage range. Inputting a voltage equal to or greater than $AV_{DD}$, 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.

**(3) Conflicting operations**

**<1> Conflict between A/D conversion result register (ADCR) write and read of ADCR by instruction at conversion end**

The read operation to ADCR is prioritized. After the read operation, a new conversion result is written to ADCR.

**<2> Conflict between ADCR write and external trigger signal input at conversion end**

External trigger signals cannot be received during A/D conversion. Therefore, external trigger signals are not received during an ADCR write operation.

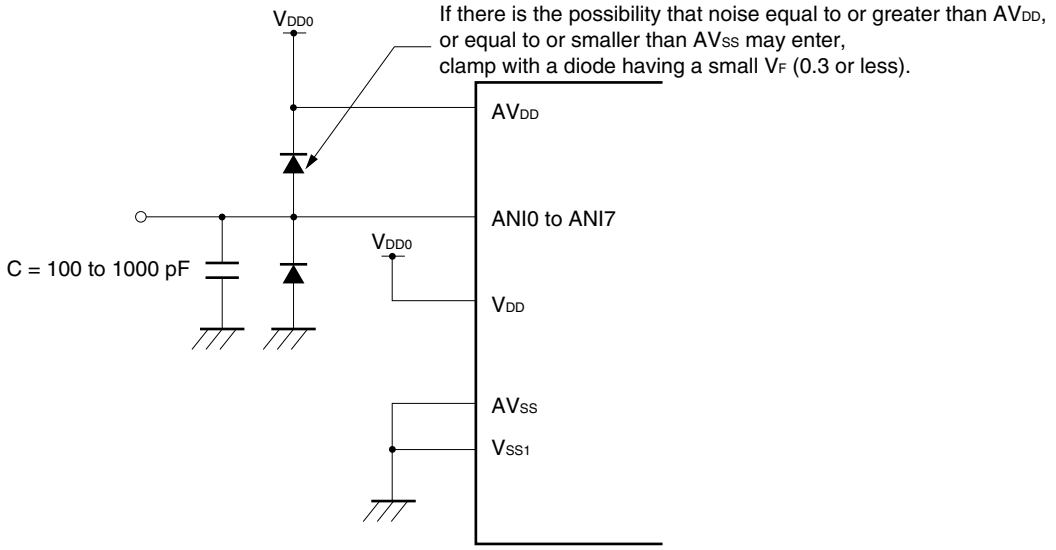
**<3> Conflict between ADCR write and A/D converter mode register (ADM) write, or between A/D converter input selection register (ADIS) write at conversion end**

The write operation to ADM or ADIS is prioritized. A write to ADCR is not performed. Moreover, no interrupt signal (INTAD) is issued at conversion end.

**(4) Anti-noise measures**

Attention must be paid to noise fed to $AV_{DD}$ 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 13-15.

**Figure 13-15. Handling of Analog Input Pin**




**(5) ANI0/P10 to ANI7/P17**

The analog input pins (ANI0 to ANI7) can also be used as input port pins (P10 to P17).

Do not execute an input command that corresponds with port 1 during conversion if any of the ANI0 to ANI7 pins have been selected for A/D conversion, as this would result in a lowered resolution.

Moreover, if a digital pulse is applied to other analog input pins during A/D conversion, the A/D conversion value will not be obtained as expected because of coupling noise. Therefore, do not apply a pulse to other analog input pins during A/D conversion.

**(6) Input impedance of $AV_{DD}$ pin**

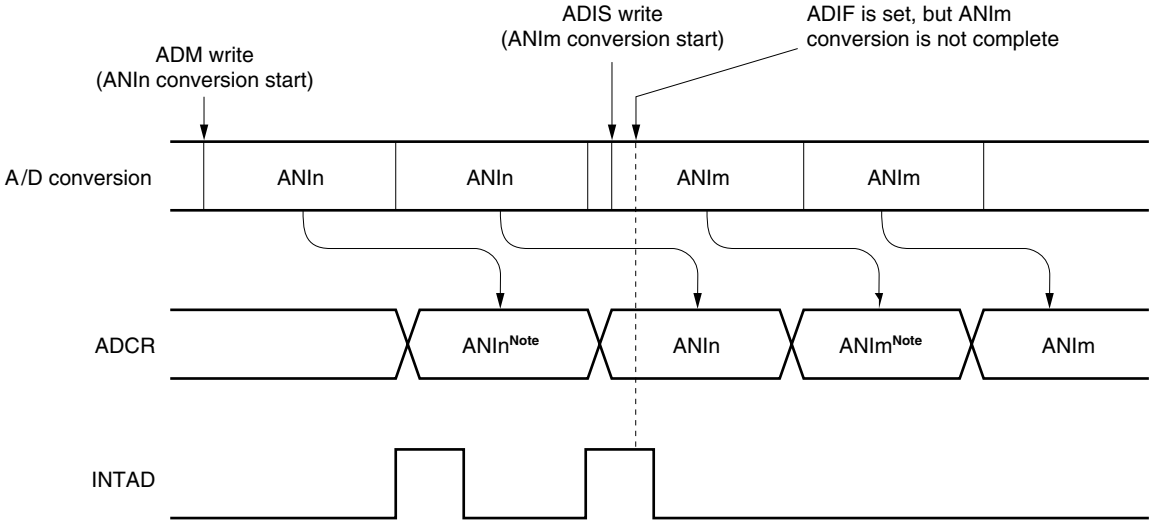
In the $\mu$PD784225, the $AV_{DD}$ pin can also be used as the reference voltage source and a series resistor string of approximately 46 kΩ is connected between the $AV_{DD}$ and $AV_{SS}$ pins.

Therefore, if the output impedance of the reference voltage source is high, connecting in series a series resistor string between the $AV_{DD}$ and $AV_{SS}$ pins will result in a large reference voltage error.

**(7) Interrupt request flag (ADIF)**

The interrupt request flag (ADIF) is not cleared even if the A/D converter input selection register (ADIS) is changed. Owing to this, there will be cases when the A/D conversion result and ADIF that correspond with the pre-amended analog input immediately prior to ADM overwriting will be set if the analog input pin is amended during A/D conversion. It must therefore be noted that the ADIF will be set regardless of whether the A/D conversion for the amended analog input has finished or not when ADIF is read immediately after ADIS has been overwritten. Moreover, if A/D conversion is stopped once and then resumed, clear ADIF before resuming conversion.

**Figure 13-16. A/D Conversion End Interrupt Request Generation Timing**




**Remark** n = 0, 1, ···, 7
m = 0, 1, ···, 7

**Note** If bit 0 (ADCE) of the A/D converter mode register is not set to 1, the value of the first A/D conversion is undefined immediately after A/D conversion starts. Take measures such as polling the A/D conversion end interrupt request (INTAD) and discarding the first A/D conversion result.

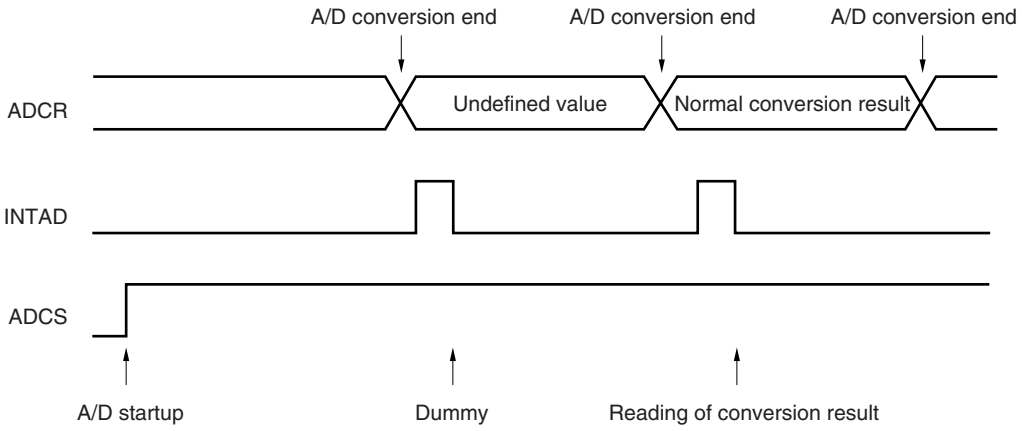
**(8) Bit 0 (ADCE) of A/D converter mode register (ADM)**

Setting ADCE to 1 allows the value of the first A/D conversion immediately after A/D conversion operation start to be used.

**(9) Conversion result immediately after A/D conversion is started**

If bit 7 (ADCS0) of the A/D converter mode register (ADM) is set to 1 without setting bit 0 (ADCE) to 1, the value of the first A/D conversion is undefined immediately after the A/D conversion operation starts. Take measures such as polling the A/D conversion end interrupt request (INTAD) and discarding the first conversion result.

**Figure 13-17. Conversion Results Immediately After A/D Conversion Is Started**



**(10) Reading A/D conversion result register (ADCR)**

If the conversion result register (ADCR) is read after stopping the A/D conversion operation, the conversion result may be undefined. Therefore, be sure to read ADCR before stopping operation of the A/D converter.

**(11) Timing that makes the A/D conversion result undefined**

If the timing of the end of A/D conversion and the timing of the stop of operation of the A/C converter conflict, the A/D conversion value may be undefined. Because of this, be sure to read the A/D conversion result while the A/D converter is in operation. Furthermore, when reading an A/D conversion result after the A/D converter operation has stopped, be sure to have done so by the time the next conversion result is complete.

The conversion result read timing is shown in Figures 13-18 and 13-19 below.

**Figure 13-18. Conversion Result Read Timing (When Conversion Result Is Undefined)**

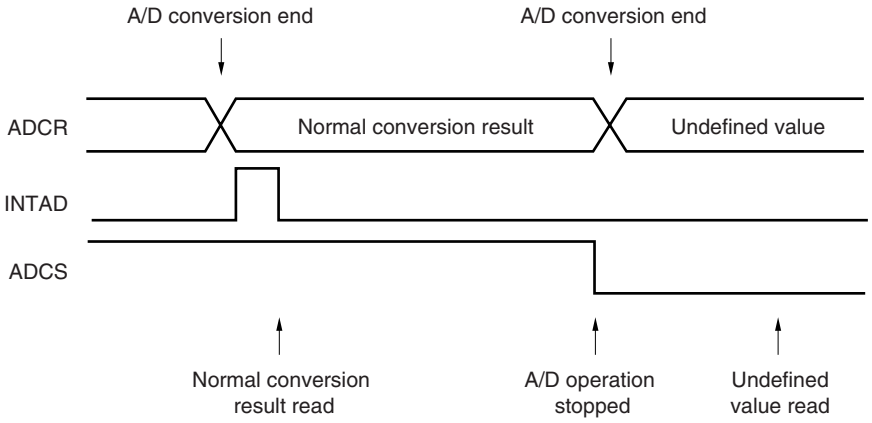
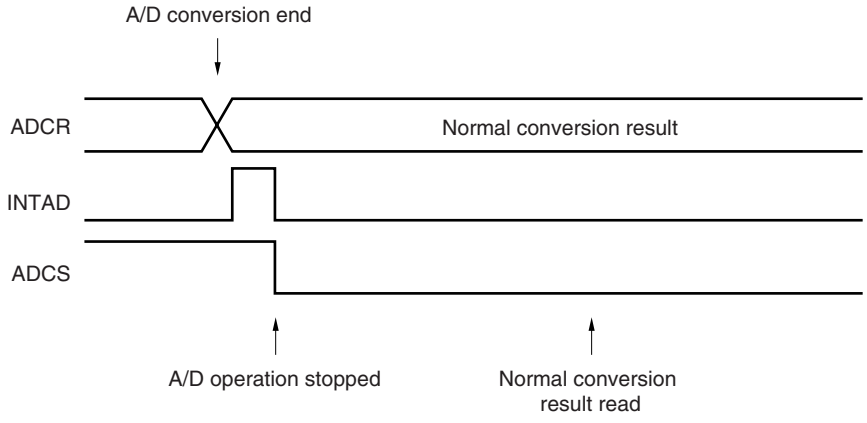



**Figure 13-19. Conversion Result Read Timing (When Conversion Result Is Normal)**



**(12) Cautions on board design**

In order to avoid negative effects from digital circuit noise on the board, analog circuits must be placed as far away as possible from digital circuits. It is particularly important to prevent analog and digital signal lines from crossing or coming into close proximity, as A/D conversion characteristics are vulnerable to degradation from the induction of noise or other such factors.

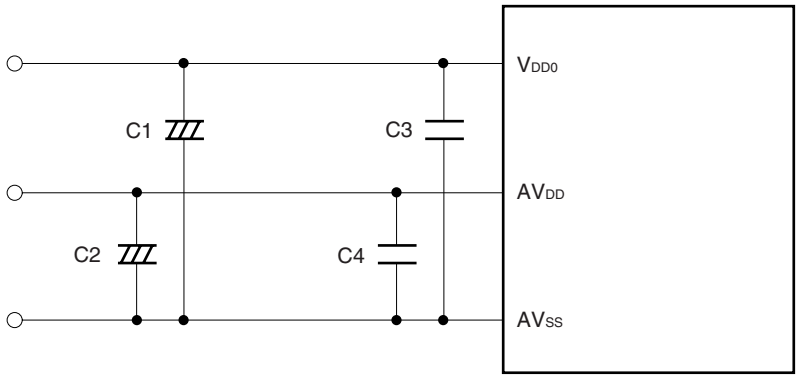
Connect $AV_{SS}$ and $V_{SS1}$ to a single stable location on the board.

**(13) $V_{DD0}$ and $AV_{DD}$ pins**

The $AV_{DD}$ pin functions as the analog circuit power supply pin and the A/D converter's reference voltage input pin, and also supplies power to the input circuits of ANI0/P10 to ANI7/P17.

Connect a capacitor between the $V_{DD0}$ and $AV_{DD}$ pins to minimize conversion error caused by noise. Note also that the voltage applied to the $V_{DD0}$ and $AV_{DD}$ pins following the resumption of A/D conversion after it was stopped may be unstable, causing a degradation in the accuracy of the A/D conversion. Be sure to connect a capacitor between the $V_{DD0}$ and $AV_{DD}$ pins in this case also. An example of capacitor connection is shown in Figure 13-20 below.

**Figure 13-20. Example of Capacitor Connection Between $V_{DD0}$ and $AV_{DD}$**




**Remark** C1, C2: 4.7 $\mu$F to 10 $\mu$F (reference values)
C3, C4: 0.01 $\mu$F to 0.1 $\mu$F (reference values)
Connect C3 and C4 as close to the pins as possible.

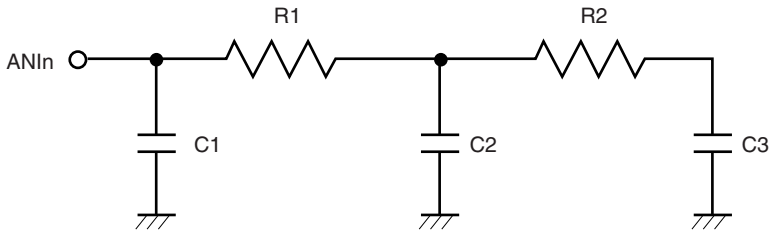
**(14) Internal equivalence circuit and allowable signal source impedance of ANI0 to ANI7**

In order to complete sampling within the sampling time and obtain a high enough A/D conversion accuracy, it is necessary to sufficiently reduce the impedance of the sensor and other signal sources. Figure 13-21 shows the internal equivalence circuit of the ANI0 to ANI7 pins in the microcontroller.

If the impedance of the signal source is high, it can be made to seem smaller by connecting a large capacitance to the ANI0 to ANI7 pins. A circuit example is shown in Figure 13-22. In this case, because a low pass filter is configured in the circuit, impedance will no longer be able to follow analog signals with large differential coefficients.

When converting high-speed analog signals or performing conversion in scan mode, be sure to insert a low-impedance buffer.

**Figure 13-21. Internal Equivalence Circuit of ANI0 to ANI7 Pins**



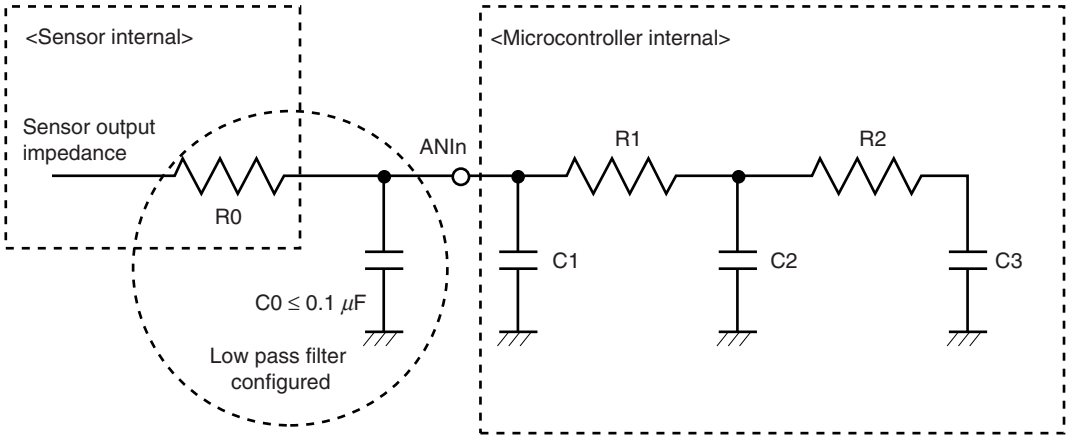

**Remark** n = 0 to 7

**Table 13-2. Resistance and Capacitance Values for Equivalence Circuits (Reference Values)**

| $V_{DD0}$ | R1 | R2 | C1 | C2 | C3 |
|---|---|---|---|---|---|
| 1.8 V | 75 kΩ | 30 kΩ | 3 pF | 4 pF | 2 pF |
| 2.7 V | 12 kΩ | 8 kΩ | 3 pF | 3 pF | 2 pF |
| 4.5 V | 3 kΩ | 2.7 kΩ | 3 pF | 1.4 pF | 2 pF |

**Caution The resistance and capacitance values in Table 13-2 cannot be guaranteed.**

**Figure 13-22. Example of Circuit When Signal Source Impedance Is High**




**Remark** n = 0 to 7

# CHAPTER 14 D/A CONVERTER

## 14.1 Function

The D/A converter converts the digital input into analog values and consists of two voltage output D/A converter channels with 8-bit resolution.

The conversion method is an R-2R resistor ladder.

Set DACE0 of D/A converter mode register 0 (DAM0) and DACE1 of D/A converter mode register 1 (DAM1) to start D/A conversion.

The D/A converter has the following two modes.

**(1) Normal mode**

After D/A conversion, the analog voltage is immediately output.

**(2) Real-time output mode**

After D/A conversion, the analog voltage is output synchronized with the output trigger.

Since a sine wave is created when this mode is used, MSK modems can be easily incorporated into cordless phones.

**Caution If only one channel of the D/A converter is used when $AV_{REF1} < V_{DD}$, make either of the following settings at pins that are not used for analog output.**

- **Set the port mode register (PM13X) to 1 (input mode) and connect to $V_{SS}$.**
- **Set the port mode register (PM13X) to 0 (output mode) and the output latch to 0, and output a low level.**

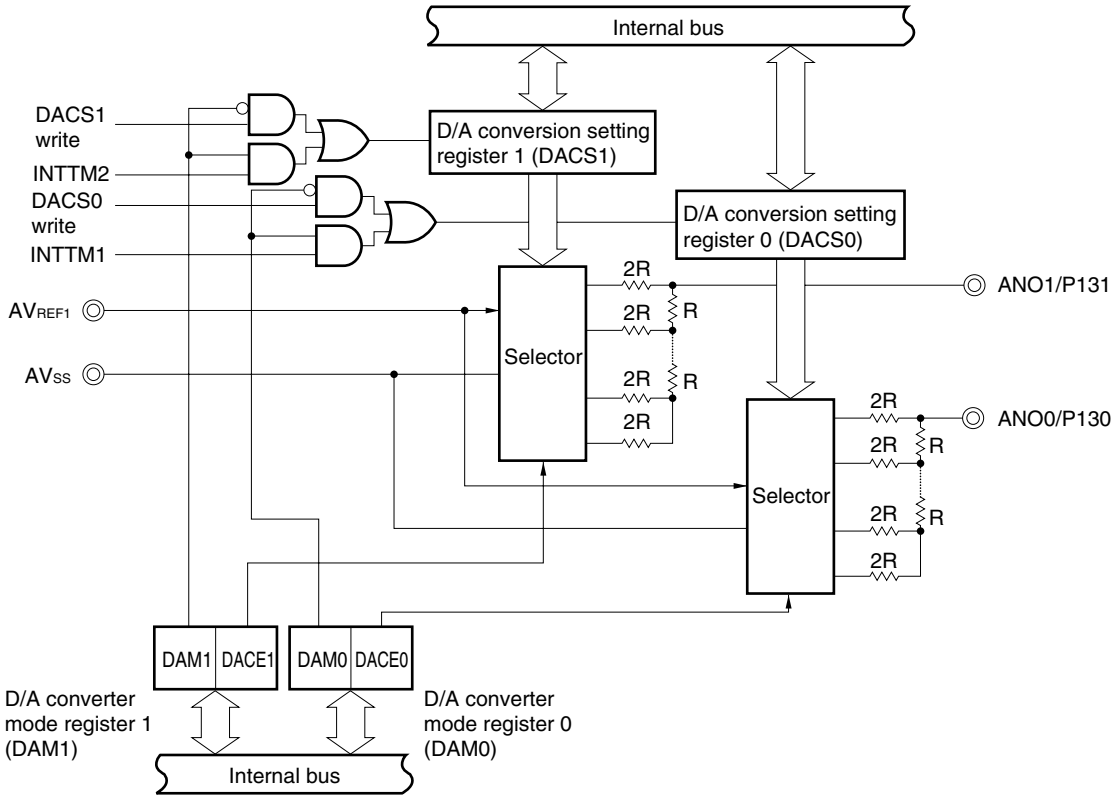
## 14.2 Configuration

The D/A converter includes the following hardware.

**Table 14-1. Configuration of D/A Converter**

| Item | Configuration |
|---|---|
| Registers | D/A conversion setting register 0 (DACS0)<br>D/A conversion setting register 1 (DACS1) |
| Control registers | D/A converter mode register 0 (DAM0)<br>D/A converter mode register 1 (DAM1) |

**Figure 14-1. Block Diagram of D/A Converter**




**(1) D/A conversion setting registers 0 and 1 (DACS0, DACS1)**

DACS0 and DACS1 set the analog voltages that are output to the ANO0 and ANO1 pins, respectively.

DACS0 and DACS1 are set by an 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets DACS0 and DACS1 to 00H.

The analog voltages output by the ANO0 and ANO1 pins are determined by the following equation.

$$\text{ANOn output voltage} = AV_{REF1} \times \frac{\text{DACSn}}{256}$$

n = 0, 1

**Cautions 1. In the real-time output mode, when the data set in DACS0 and DACS1 is read before the output trigger is generated, the set data is not read and the previous data is read.**

**2. In the real-time output mode, set the data of DACS0 and DACS1 until the next output trigger is generated after the output trigger is generated.**

## 14.3 Control Registers

- **D/A converter mode registers 0 and 1 (DAM0, DAM1)**

  The D/A converter is controlled by D/A converter mode registers 0 and 1 (DAM0, DAM1). These registers enable or stop the operation of the D/A converter.

  DAM0 and DAM1 are set by a 1-bit and 8-bit memory manipulation instruction.

  $\overline{\text{RESET}}$ input sets DAM0 and DAM1 to 00H.

**Figure 14-2. Format of D/A Converter Mode Registers 0 and 1 (DAM0, DAM1)**

Address: 0FF86H, 0FF87H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | ⓪ |
|---|---|---|---|---|---|---|---|---|
| DAMn | 0 | 0 | 0 | 0 | 0 | 0 | DAMn | DACEn |

| DAMn | D/A converter channel n operation mode |
|---|---|
| 0 | Normal mode |
| 1 | Real-time output mode |

| DACEn | D/A converter channel n control |
|---|---|
| 0 | Stop conversion |
| 1 | Enable conversion |

**Cautions 1. When the D/A converter is used, set the alternate-function port pins to the input mode and disconnect the pull-up resistors.**

**2. Always set bits 2 to 7 to 0.**

**3. The output when the D/A converter operation has stopped enters a high-impedance state.**

**4. The output triggers in the real-time output mode are INTTM1 for channel 0 and INTTM2 for channel 1.**

**Remark** n = 0, 1

## 14.4 Operation

<1> Select the operation mode of channel 0 using DAM0 of D/A converter mode register 0 (DAM0) and the operation mode of channel 1 using DAM1 of D/A converter mode register 1 (DAM1).

<2> Set the data that corresponds to the analog voltages that are output to pins ANO0/P130 and ANO1/P131 of D/A conversion setting registers 0 and 1 (DACS0, DACS1).

<3> Set DACE0 of DAM0 and DACE1 of DAM1 to start D/A conversion for channels 0 and 1, respectively.

<4> After D/A conversion in the normal mode, the analog voltages at pins ANO0/P130 and ANO1/P131 are immediately output. In the real-time output mode, the analog voltage is output synchronized with the output trigger.

<5> In the normal mode, the output analog voltages are maintained until new data is set in DACS0 and DACS1. In the real-time output mode, after new data is set in DACS0 and DACS1, it is held until the next output trigger is generated.

**Caution Set DACE0 and DACE1 after data has been set in DACS0 and DACS1.**

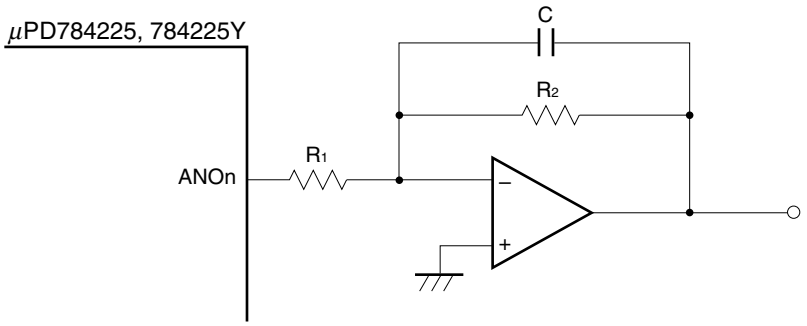
## 14.5 Cautions

**(1) Output impedance of the D/A converter**

Since the output impedance of the D/A converter is high, current cannot be taken from the ANOn pin (n = 0,1). If the input impedance of the load is low, insert a buffer amplifier between the load and the ANOn pin. In addition, use the shortest possible wire from the buffer amplifier or load (to increase the output impedance). If the wire is long, surround it with a ground pattern.

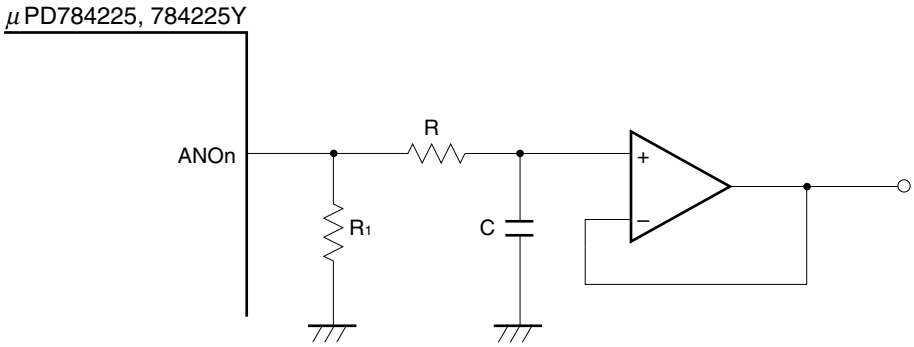
**Figure 14-3. Buffer Amplifier Insertion Example**

**(a) Inverting Amplifier**




- The input impedance of the buffer amplifier is $R_1$.

**(b) Voltage follower**




- The input impedance of the buffer amplifier is $R_1$.
- If there is no $R_1$ and $\overline{\text{RESET}}$ is low, the output is undefined.

**(2) Output voltage of the D/A converter**

Since the output voltage of the D/A converter changes in stages, use the signals output from the D/A converter after passing them through a low-pass filter.

**(3) $AV_{REF1}$ pin**

Handle pins not being used for analog output in either of the following ways when the D/A converter is only using one channel with $AV_{REF1} < V_{DD}$.

- Set the port mode register (PM13X) to 1 (input mode) and connect to $V_{SS0}$.
- Set the port mode register (PM13X) to 0 (output mode), set the output latch to 0, and output a low level.

# CHAPTER 15 SERIAL INTERFACE OVERVIEW

The μPD784225 Subseries has a serial interface with three independent channels. Therefore, communication outside and within the system can be performed simultaneously using all three channels.

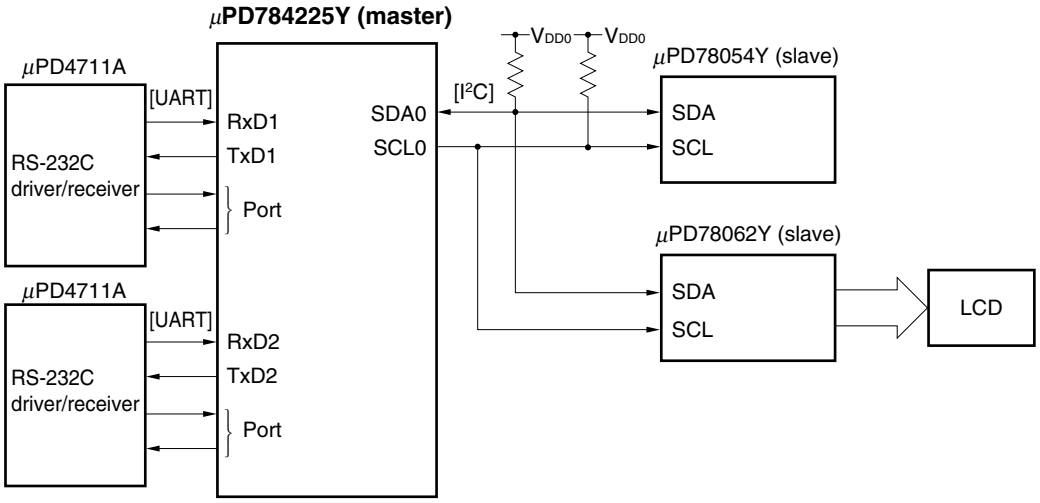
- Asynchronous serial interface (UART)/3-wire serial I/O (IOE) × 2 channels
  → See **CHAPER 16**.

- Clocked serial interface (CSI) × 1 channel
  - 3-wire serial I/O mode (MSB first)
    → See **CHAPTER 17**.

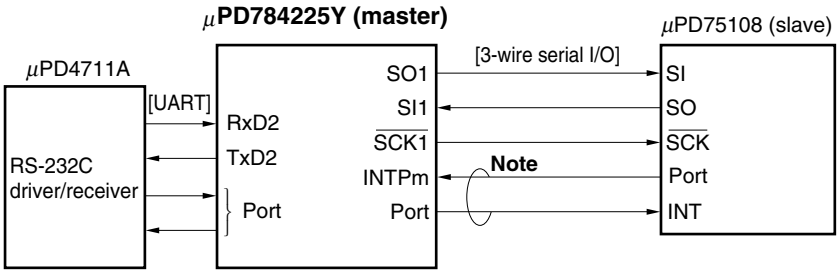
  - $I^2C$ bus mode (multimaster compatible) (only in the μPD784225Y Subseries)
    → See **CHAPTER 18**.

**Figure 15-1. Serial Interface Example**

**(a) UART + $I^2C$**




**(b) UART + 3-wired serial I/O**




**Note** Handshake lines

# CHAPTER 16 ASYNCHRONOUS SERIAL INTERFACE/3-WIRE SERIAL I/O

The μPD784225 provides two on-chip serial interface channels for which the asynchronous serial interface (UART) mode and the 3-wire serial I/O (IOE) mode can be selected.

These two serial interface channels have exactly the same functions.

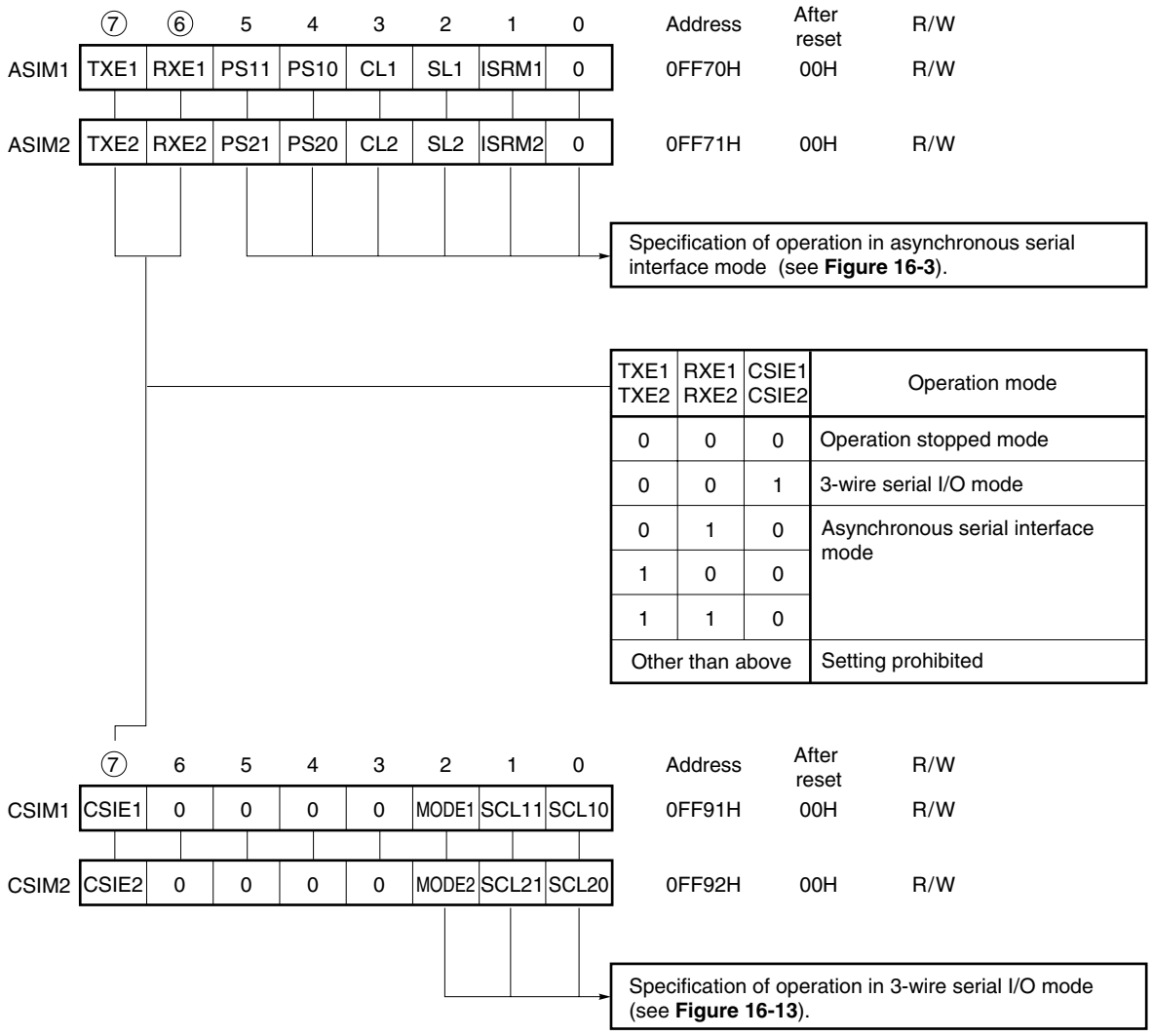
**Table 16-1. Differences in Names Between UART1/IOE1 and UART2/IOE2**

| Item | UART1/IOE1 | UART1/IOE2 |
|---|---|---|
| Pin name | P22/ASCK1/$\overline{SCK1}$, P20/RxD1/SI1, P21/TxD1/SO1 | P72/ASCK2/$\overline{SCK2}$, P70/RxD2/SI2, P71/TxD2/SO2 |
| Asynchronous serial interface mode register | ASIM1 | ASIM2 |
| Name of bits inside asynchronous serial interface mode register | TXE1, RXE1, PS11, PS10, CL1, SL1, ISRM1 | TXE2, RXE2, PS21, PS20, CL2, SL2, ISRM2 |
| Asynchronous serial interface status register | ASIS1 | ASIS2 |
| Name of bits inside asynchronous serial interface status register | PE1, FE1, OVE1 | PE2, FE2, OVE2 |
| Serial operation mode register | CSIM1 | CSIM2 |
| Name of bits inside serial operation mode register | CSIE1, MODE1, SCL11, SCL10 | CSIE2, MODE2, SCL21, SCL20 |
| Baud rate generator control register | BRGC1 | BRGC2 |
| Name of bits inside baud rate generator control register | TPS10 to TPS12, MDL10 to MDL13 | TPS20 to TPS22, MDL20 to MDL23 |
| Interrupt request name | INTSR1/INTCSI1, INTSER1, INTST1 | INTSR2/INTCSI2, INTSER2, INTST2 |
| Interrupt control register and name of bits used in this chapter | SRIC1, SERIC1, STIC1, SRIF1, SERIF1, STIF1 | SRIC2, SERIC2, STIC2, SRIF2, SERIF2, STIF2 |

## 16.1 Switching Asynchronous Serial Interface Mode and 3-Wire Serial I/O Mode

The asynchronous serial interface mode and the 3-wire serial I/O mode cannot be used at the same time. These modes can be switched by setting asynchronous serial interface mode registers 1 and 2 (ASIM1, ASIM2) and serial operation mode registers 1 and 2 (CSIM1, CSIM2), as shown in Figure 16-1 below.

**Figure 16-1. Switching Asynchronous Serial Interface Mode and 3-Wire Serial I/O Mode**



★ **Table 16-2. Serial Interface Operation Mode Settings**

**(1) Operation stopped mode**

| ASIMn | | CSIMn | | | PM20 | P20 | PM21 | P21 | PM22 | P22 | First | Shift | P20/RxD1/SI1 | P21/TxD1/SO1 | P22/ASCK1/$\overline{SCK1}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TXEn | RXEn | CSIEn | SCLn1 | SCLn0 | PM70 | P70 | PM71 | P71 | PM72 | P72 | Bit | Clock | P70/RxD2/SI2 Pin Function | P71/TxD2/SO2 Pin Function | P72/ASCK2/$\overline{SCK2}$ Pin Function |
| 0 | 0 | 0 | × | × | ×Note 1 | ×Note 1 | ×Note 1 | ×Note 1 | ×Note 1 | ×Note 1 | – | – | P20<br>P70 | P21<br>P71 | P22<br>P72 |
| Other than above | | | | | | | | | | | Setting prohibited | | | | |

**(2) Asynchronous serial interface mode**

| ASIMn | | CSIMn | | | PM20 | P20 | PM21 | P21 | PM22 | P22 | First | Shift | P20/RxD1/SI1 | P21/TxD1/SO1 | P22/ASCK1/$\overline{SCK1}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TXEn | RXEn | CSIEn | SCLn1 | SCLn0 | PM70 | P70 | PM71 | P71 | PM72 | P72 | Bit | Clock | P70/RxD2/SI2 Pin Function | P71/TxD2/SO2 Pin Function | P72/ASCK2/$\overline{SCK2}$ Pin Function |
| 1 | 0 | 0 | × | × | ×Note 1 | ×Note 1 | 0Note 2 | 0 | 1 | × | LSB | External clock | P20<br>P70 | TxDn<br>(CMOS output) | ASCKn input |
| | | | | | | | | | ×Note 1 | ×Note 1 | | Internal clock | | | P22<br>P72 |
| 0 | 1 | | | | 1 | × | ×Note 1 | ×Note 1 | 1 | × | | External clock | RxDn | P21<br>P71 | ASCKn input |
| | | | | | | | | | ×Note 1 | ×Note 1 | | Internal clock | | | P22<br>P72 |
| 1 | 1 | | | | | | 0Note 2 | 0 | 1 | × | | External clock | | TxDn<br>(CMOS output) | ASCKn input |
| | | | | | | | | | ×Note 1 | ×Note 1 | | Internal clock | | | P22<br>P72 |
| Other than above | | | | | | | | | | | Setting prohibited | | | | |

**(3) 3-wire serial I/O mode**

| ASIMn | | CSIMn | | | PM20 | P20 | PM21 | P21 | PM22 | P22 | First | Shift | P20/RxD1/SI1 | P21/TxD1/SO1 | P22/ASCK1/$\overline{SCK1}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TXEn | RXEn | CSIEn | SCLn1 | SCLn0 | PM70 | P70 | PM71 | P71 | PM72 | P72 | Bit | Clock | P70/RxD2/SI2 Pin Function | P71/TxD2/SO2 Pin Function | P72/ASCK2/$\overline{SCK2}$ Pin Function |
| 0 | 0 | 1 | 0 | 0 | 1Note 3 | ×Note 3 | 0 | 0 | 1 | × | MSB | External clock | SInNote 3 | SOn<br>(CMOS output) | $\overline{SCKn}$ input |
| | | | Note 4 | Note 4 | | | | | 0 | 0 | | Internal clock | | | $\overline{SCKn}$ output |
| Other than above | | | | | | | | | | | Setting prohibited | | | | |

**Notes 1.** These pins can be used for port functions.

**2.** Refer to **16.3.2 Asynchronous serial interface (UART) mode (2) Communication operation (c) Transmission**.

**3.** When only transmission is used, these pins can be used as P20 and P70 (CMOS I/O).

**4.** Refer to serial operation mode registers 1 and 2 (CSIM1, CSIM2).

**Remark** ×: Don't care

n = 1, 2

## 16.2 Asynchronous Serial Interface Mode

The asynchronous serial interface (UART: Universal Asynchronous Receiver Transmitter) offers the following two modes.

**(1) Operation stopped 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 the 1-byte data that follows the start bit, and supports full-duplex transmission.

A UART-dedicated baud rate generator is provided on-chip, enabling transmission at any baud rate within a broad range. The baud rate can also be defined by dividing the input clock to the ASCK pin.

The MIDI standard baud rate (31.25 Kbps) can be used by utilizing the UART-dedicated baud rate generator.

### 16.2.1 Configuration

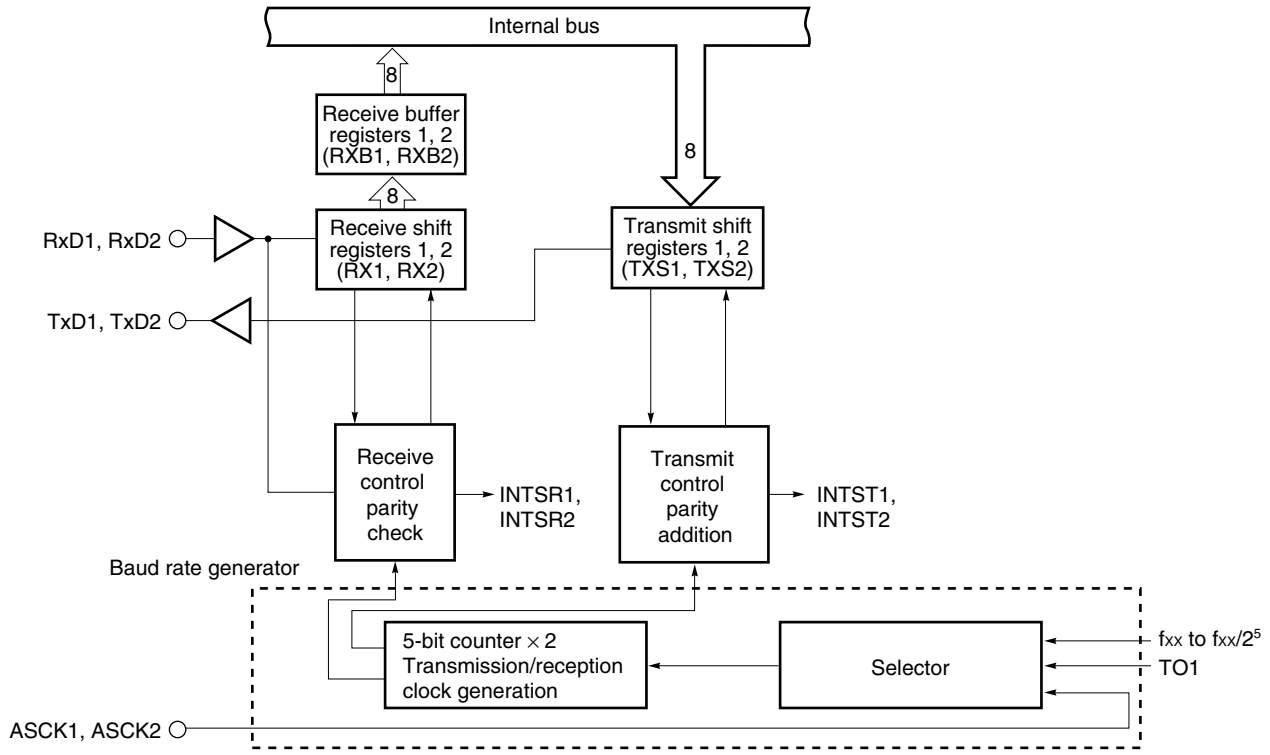
The asynchronous serial interface includes the following hardware.

Figure 16-2 shows the block diagram of the asynchronous serial interface.

**Table 16-3. Configuration of Asynchronous Serial Interface**

| Item | Configuration |
|---|---|
| Registers | Transmit shift registers 1, 2 (TXS1, TXS2)<br>Receive shift registers 1, 2 (RX1, RX2)<br>Receive buffer registers 1, 2 (RXB1, RXB2) |
| Control registers | Asynchronous serial interface mode registers 1, 2 (ASIM1, ASIM2)<br>Asynchronous serial interface status registers 1, 2 (ASIS1, ASIS2)<br>Baud rate generator control registers 1, 2 (BRGC1, BRGC2) |

**Figure 16-2. Block Diagram in Asynchronous Serial Interface Mode**



**(1) Transmit shift registers 1, 2 (TXS1, TXS2)**

These registers are used to set transmit data. Data written to TXS1 and TXS2 is sent as serial data.

If a data length of 7 bits is specified, bits 0 to 6 of the data written to TXS1 and TXS2 are transferred as transmit data. Transmission is started by writing data to TXS1 and TXS2.

TX1 and TX2 can be written with an 8-bit memory manipulation instruction, but cannot be read.

$\overline{\text{RESET}}$ input sets TXS1 and TXS2 to FFH.

**Caution Do not write to TXS1 and TXS2 during transmission.**
**TXS1, TXS2 and receive buffer registers 1, 2 (RXB1, RXB2) are allocated to the same address. Therefore, attempting to read TXS1 and TXS2 will result in reading the values of RXB1 and RXB2.**

**(2) Receive shift registers 1, 2 (RX1, RX2)**

These registers are used to convert serial data input to the RxD1 and RxD2 pins to parallel data. Receive data is transferred to receive buffer registers 1 and 2 (RXB1, RSB2) one byte at a time as it is received.

RX1 and RX2 cannot be directly manipulated by program.

**(3) Receive buffer registers 1, 2 (RXB1, RXB2)**

These registers are used to hold receive data. Each time one byte of data is received, new receive data is transferred from receive shift registers 1 and 2 (RX1, RX2)

If a data length of 7 bits is specified, receive data is transferred to bits 0 to 6 of RXB1 and RXB2, and the MSB of RXB1 and RXB2 always becomes 0.

RXB1 and RXB2 can be read by an 8-bit memory manipulation instruction, but cannot be written.

$\overline{\text{RESET}}$ input sets RXB1 and RXB2 to FFH.

**Caution RXB1, RXB2 and transmit shift registers 1, 2 (TXB1, TXB2) are allocated to the same address. Therefore, attempting to read RXB1 and RXB2 will result in reading the values of TXB1 and TXB2.**

**(4) Transmission controller**

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 registers 1 and 2 (TXS1, TXS2), according to contents set to asynchronous serial interface mode registers 1 and 2 (ASIM1, ASIM2).

**(5) Reception controller**

This circuit controls reception according to the contents set to asynchronous serial interface mode registers 1 and 2 (ASIM1, ASIM2). 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 asynchronous serial interface status registers 1 and 2 (ASIS1, ASIS2).

### 16.2.2 Control registers

The asynchronous serial interface is controlled by the following six registers.

- Asynchronous serial interface mode registers 1, 2 (ASIM1, ASIM2)
- Asynchronous serial interface status registers 1, 2 (ASIS1, ASIS2)
- Baud rate generator control registers 1, 2 (BRGC1, BRGC2)

#### (1) Asynchronous serial interface mode registers 1, 2 (ASIM1, ASIM2)

ASIM1 and ASIM2 are 8-bit registers that control serial transfer using the asynchronous serial interface.

ASIM1 and ASIM2 are set by a 1-bit or 8-bit memory manipulation operation.

$\overline{\text{RESET}}$ input sets ASIM1 and ASIM2 to 00H.

**Figure 16-3. Format of Asynchronous Serial Interface Mode Registers 1 and 2 (ASIM1, ASIM2)**

Address: 0FF70H, 0FF71H After reset: 00H R/W

| Symbol | ⑦ | ⑥ | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ASIMn | TXEn | RXEn | PSn1 | PSn0 | CLn | SLn | ISRMn | 0[Note] |

| TXEn | RXEn | Operation mode | RxD1/P20, RxD2/P70 pin function | TxD1/P21, TxD2/P71 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 |

| PSn1 | PSn0 | Parity bit specification |
|---|---|---|
| 0 | 0 | No parity |
| 0 | 1 | Always add 0 parity during transmission<br>Do not perform parity check during reception (parity error not generated) |
| 1 | 0 | Odd parity |
| 1 | 1 | Even parity |

| CLn | Transmit data character length specification |
|---|---|
| 0 | 7 bits |
| 1 | 8 bits |

| SLn | Transmit data stop bit length specification |
|---|---|
| 0 | 1 bit |
| 1 | 2 bits |

| ISRMn | 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** Be sure to write “0” to bit 0.

**Caution Switch across to the operational mode after stopping serial sending and receiving operations.**

**Remark** n = 1, 2

**(2) Asynchronous serial interface status registers 1 and 2 (ASIS1, ASIS2)**

ASIS1 and ASIS2 are registers used display the type of error when a receive error occurs.

ASIS1 and ASIS2 can be read by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets ASIS1 and ASIS2 to 00H.

**Figure 16-4. Format of Asynchronous Serial Interface Status Registers 1 and 2 (ASIS1, ASIS2)**

Address: 0FF72H, 0FF73H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | ② | ① | ⓪ |
|---|---|---|---|---|---|---|---|---|
| ASISn | 0 | 0 | 0 | 0 | 0 | PEn | FEn | OVEn |

| PEn | Parity error flag |
|---|---|
| 0 | Parity error not generated |
| 1 | Parity error generated<br>(when parity of transmit data does not match) |

| FEn | Framing error flag |
|---|---|
| 0 | Framing error not generated |
| 1 | Framing error generated**Note 1**<br>(when stop bit(s) is not detected) |

| OVEn | Overrun error flag |
|---|---|
| 0 | Overrun error not generated |
| 1 | Overrun error generated**Note 2**<br>(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 asynchronous serial interface mode register n (ASIMn), stop bit detection during reception is only 1 bit.

**2.** Be sure to read receive buffer register n (RXBn) when an overrun error occurs.
An overrun error is generated each time data is received until RXBn is read.

**Remark** n = 1, 2

**(3) Baud rate generator control registers 1 and 2 (BRGC1, BRGC2)**

BRGC1 and BRGC2 are registers used to set the serial clock of the asynchronous serial interface.

BRGC1 and BRGC2 are set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets BRGC1 and BRGC2 to 00H.

**Figure 16-5. Format of Baud Rate Generator Control Registers 1 and 2 (BRGC1, BRGC2)**

Address: 0FF76H, 0FF77H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| BRGCn | 0 | TPSn2 | TPSn1 | TPSn0 | MDLn3 | MDLn2 | MDLn1 | MDLn0 |

| TPSn2 | TPSn1 | TPSn0 | 5-bit counter source clock selection | m |
|---|---|---|---|---|
| 0 | 0 | 0 | External clock (ASCKn) | 0 |
| 0 | 0 | 1 | $f_{XX}$ (12.5 MHz) | 0 |
| 0 | 1 | 0 | $f_{XX}/2$ (6.25 MHz) | 1 |
| 0 | 1 | 1 | $f_{XX}/4$ (3.13 MHz) | 2 |
| 1 | 0 | 0 | $f_{XX}/8$ (1.56 MHz) | 3 |
| 1 | 0 | 1 | $f_{XX}/16$ (781 kHz) | 4 |
| 1 | 1 | 0 | $f_{XX}/32$ (391 kHz) | 5 |
| 1 | 1 | 1 | TO1 (TM1 output) | 0 |

| MDLn3 | MDLn2 | MDLn1 | MDLn0 | 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 | – |

**Cautions 1. If a write operation to BRGC1 and BRGC2 is performed during communication, the baud rate generator output will become garbled and normal communication will not be achieved. Consequently, do not write to BRGC1 or BRGC2 during communication.**

**2. Refer to CHAPTER 29 ELECTRICAL SPECIFICATIONS for details of the high-/low-level width of ASCKn when selecting the external clock (ASCKn) for the source clock of the 5-bit counter.**

**Cautions 3. Set the 8-bit timer mode control register 1 (TMC1) as follows when selecting TO1 for the source clock of the 5-bit counter.**
**TMC16 = 0, LVS1 = 0, LVR1 = 0, TMC11 = 1**
**Moreover, set TOE1 to 0 when TO1 is not output externally and TOE1 to 1 when TO1 is output externally.**

**Remarks 1.** n = 1, 2

**2.** Figures in parentheses apply to operation at $f_{XX}$ = 12.5 MHz.

**3.** $f_{SCK}$: Source clock of 5-bit counter

**4.** m: Value set in TPSn0 to TPSn2 ($0 \leq m \leq 5$)

**5.** k: Value set in MDLn0 to MDLn3 ($0 \leq k \leq 14$)

## 16.3 Operation

The asynchronous serial interface has the following two operation modes.

- Operation stop mode
- Asynchronous serial interface (UART) mode

### 16.3.1 Operation stopped mode

Serial transfer cannot be performed in the operation stopped mode, resulting in reduced power consumption. Moreover, in the operation stopped mode, pins can be used as regular ports.

**(1) Register setting**

The operation stopped mode is set by asynchronous serial interface mode registers 1 and 2 (ASIM1, ASIM2). ASIM1 and ASIM2 are set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets ASIM1 and ASIM2 to 00H.

Address: 0FF70H, 0FF71H After reset: 00H R/W

| Symbol | ⑦ | ⑥ | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ASIMn | TXEn | RXEn | PSn1 | PSn0 | CLn | SLn | ISRMn | 0**Note** |

| TXEn | RXEn | Operation mode | RxD1/P20, RxD2/P70 pin function | TxD1/P21, TxD2/P71 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** Be sure to write “0” to bit 0.

**Caution Switch the operation mode after stopping serial transmission and reception.**

**Remark** n = 1, 2

### 16.3.2 Asynchronous serial interface (UART) mode

This mode is used to transmit and receive the 1-byte data following the start bit, and supports full-duplex operation.

A UART-dedicated baud rate generator is provided on-chip, enabling communication at any baud rate within a broad range.

The MIDI standard baud rate (31.25 Kbps) can be used by utilizing the UART-dedicated baud rate generator.

#### (1) Register setting

The UART mode is set with asynchronous serial interface mode registers 1 and 2 (ASIM1, ASIM2), asynchronous serial interface status registers 1 and 2 (ASIS1, ASIS2), and baud rate generator control registers 1 and 2 (BRGC1, BRGC2).

**(a) Asynchronous serial interface mode registers 1 and 2 (ASIM1, ASIM2)**

ASIM1 and ASIM2 can be set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets ASIM1 and ASIM2 to 00H.

Address: 0FF70H, 0FF71H After reset: 00H R/W

| Symbol | ⑦ | ⑥ | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ASIMn | TXEn | RXEn | PSn1 | PSn0 | CLn | SLn | ISRMn | 0**Note** |

| TXEn | RXEn | Operation mode | RxD1/P20, RxD2/P70 pin function | TxD1/P21, TxD2/P71 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 |

| PSn1 | PSn0 | Parity bit specification |
|---|---|---|
| 0 | 0 | No parity |
| 0 | 1 | Always add 0 parity during transmission<br>Do not perform parity check during reception (parity error not generated) |
| 1 | 0 | Odd parity |
| 1 | 1 | Even parity |

| CLn | Character length specification |
|---|---|
| 0 | 7 bits |
| 1 | 8 bits |

| SLn | Transmit data stop bit length specification |
|---|---|
| 0 | 1 bit |
| 1 | 2 bits |

| ISRMn | 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** Be sure to write “0” to bit 0.

**Caution Switch the operation mode after stopping serial transmission and reception.**

**Remark** n = 1, 2

**(b) Asynchronous serial interface status registers 1 and 2 (ASIS1, ASIS2)**

ASIS1 and ASIS2 can be read by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets ASIS1 and ASIS 2 to 00H.

Address: 0FF72H, 0FF73H After reset: 00H R

| Symbol | 7 | 6 | 5 | 4 | 3 | ② | ① | ⓪ |
|---|---|---|---|---|---|---|---|---|
| ASISn | 0 | 0 | 0 | 0 | 0 | PEn | FEn | OVEn |

| PEn | Parity error flag |
|---|---|
| 0 | Parity error not generated |
| 1 | Parity error generated<br>(when parity of transmit data does not match) |

| FEn | Framing error flag |
|---|---|
| 0 | Framing error not generated |
| 1 | Framing error generated[Note 1]<br>(when stop bit(s) is not detected) |

| OVEn | Overrun error flag |
|---|---|
| 0 | Overrun error not generated |
| 1 | Overrun error generated[Note 2]<br>(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 asynchronous serial interface mode register n (ASIMn), stop bit detection during reception is only 1 bit.

**2.** Be sure to read receive buffer register n (RXBn) when an overrun error occurs. An overrun error is generated each time data is received until RXBn is read.

**Remark** n = 1, 2

**(c) Baud rate generator control registers 1 and 2 (BRGC1, BRGC2)**

BRGC1 and BRGC2 are set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets BRGC1 and BRGC2 to 00H.

Address: 0FF76H, 0FF77H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| BRGCn | 0 | TPSn2 | TPSn1 | TPSn0 | MDLn3 | MDLn2 | MDLn1 | MDLn0 |

| TPSn2 | TPSn1 | TPSn0 | 5-bit counter source clock selection | m |
|---|---|---|---|---|
| 0 | 0 | 0 | External clock (ASCKn) | 0 |
| 0 | 0 | 1 | $f_{XX}$ (12.5 MHz) | 0 |
| 0 | 1 | 0 | $f_{XX}/2$ (6.25 MHz) | 1 |
| 0 | 1 | 1 | $f_{XX}/4$ (3.13 MHz) | 2 |
| 1 | 0 | 0 | $f_{XX}/8$ (1.56 MHz) | 3 |
| 1 | 0 | 1 | $f_{XX}/16$ (781 kHz) | 4 |
| 1 | 1 | 0 | $f_{XX}/32$ (391 kHz) | 5 |
| 1 | 1 | 1 | TO1 (TM1 output) | 0 |

| MDLn3 | MDLn2 | MDLn1 | MDLn0 | 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 | – |

**Cautions 1. If a write operation to BRGC1 and BRGC2 is performed during communication, the baud rate generator output will become garbled and normal communication will not be achieved. Consequently, do not write to BRGC1 or BRGC2 during communication.**

**2. Refer to CHAPTER 29 ELECTRICAL SPECIFICATIONS for details of the high-/low-level width of ASCKn when selecting the external clock (ASCKn) for the source clock of the 5-bit counter.**

**Cautions 3. Set the 8-bit timer mode control register 1 (TMC1) as follows when selecting TO1 for the source clock of the 5-bit counter.**
**TMC16 = 0, LVS1 = 0, LVR1 = 0, TMC11 = 1**
**Moreover, set TOE1 to 0 when TO1 is not output externally and TOE1 to 1 when TO1 is output externally.**

**Remarks 1.** n = 1, 2
**2.** Figures in parentheses apply to operation at $f_{XX}$ = 12.5 MHz.
**3.** $f_{SCK}$: Source clock of 5-bit counter
**4.** m: Value set in TPSn0 to TPSn2 ($0 \leq m \leq 5$)
**5.** k: Value set in MDLn0 to MDLn3 ($0 \leq k \leq 14$)

The transmit/receive clock for the baud rate to be generated is the signal obtained by dividing the 5-bit counter source clock.

- **Generation of transmit/receive clock for baud rate**
  The baud rate is obtained from the following equation.

$$[\text{Baud rate}] = \frac{T}{2^{m+1} \times (k + 16)} \text{ [Hz]}$$

T: 5-bit counter source clock
- When using a divided main system clock: Main system clock ($f_{XX}$)
- When an external clock (ASCKn) is selected: Output frequency of ASCKn
- When the timer 1 output (TO1) is selected: Output frequency of TO1

m: Value set in TPSn0 to TPSn2 ($0 \leq m \leq 5$)
k: Value set in MDLn0 to MDLn3 ($0 \leq k \leq 14$)

- **Baud rate capacity error range**

  The baud rate capacity range depends on the number of bits per frame and the counter division ratio [1/(16 + k)].

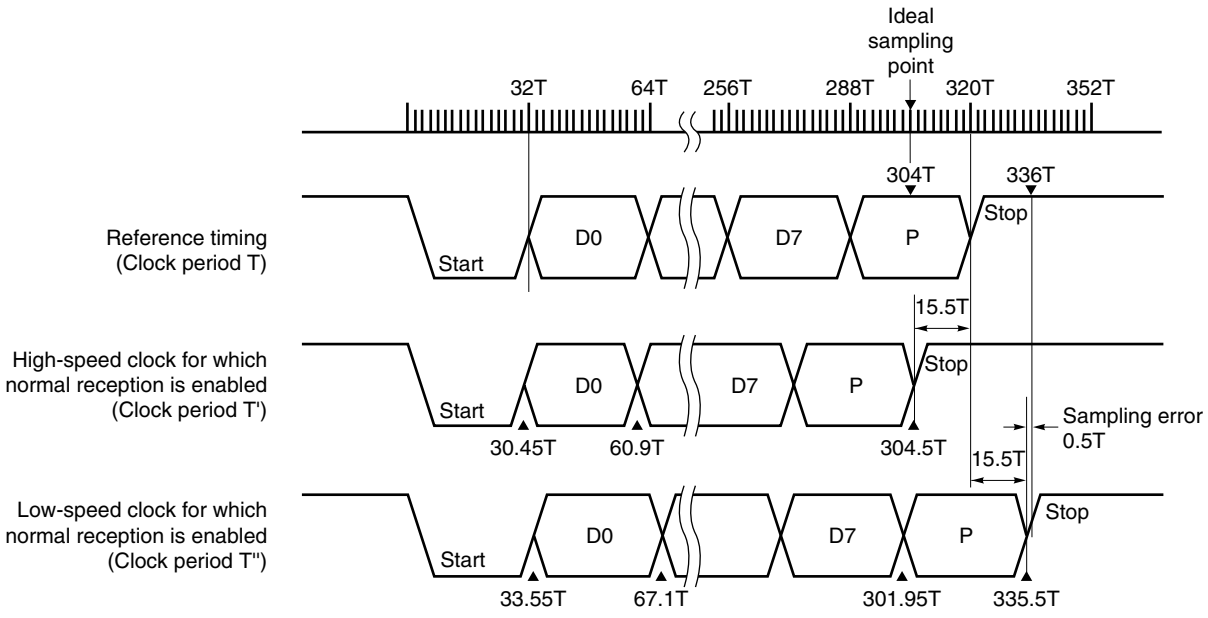
  Table 16-3 shows the relationship between the main system clock and the baud rate, Table 16-6 shows a baud rate allowable error example.

**Table 16-4. Relationship Between Main System Clock and Baud Rate**

| Baud Rate (bps) | $f_{XX}$ = 12.5 MHz | | $f_{XX}$ = 6.25 MHz | | $f_{XX}$ = 3.00 MHz | |
|---|---|---|---|---|---|---|
| | BRGC Value | Error (%) | BRGC Value | Error (%) | BRGC Value | Error (%) |
| 2400 | — | — | — | — | 64H | 2.34 |
| 4800 | — | — | 64H | 1.73 | 54H | 2.34 |
| 9600 | 64H | 1.73 | 54H | 1.73 | 44H | 2.34 |
| 19200 | 54H | 1.73 | 44H | 1.73 | 34H | 2.34 |
| 31250 | 49H | 0.00 | 39H | 0.00 | 28H | 0.00 |
| 38400 | 44H | 1.73 | 34H | 1.73 | 24H | 2.34 |
| 76800 | 34H | 1.73 | 24H | 1.73 | 14H | 2.34 |
| 150K | 24H | 1.73 | 14H | 1.73 | — | — |
| 300K | 14H | 1.73 | — | — | — | — |

**Remark** When TM1 output is used, 150 to 38400 bps is supported (during operation at $f_{XX}$ = 12.5 MHz)

**Figure 16-6. Baud Rate Allowable Error Considering Sampling Errors (When k = 0)**




**Remark** T: 5-bit counter source clock period

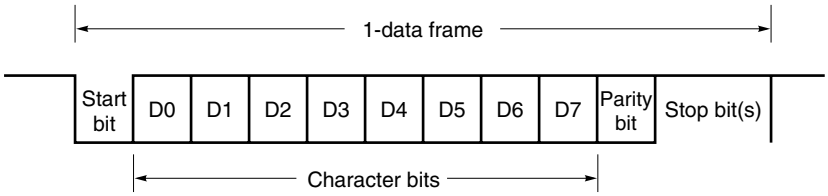
$$\text{Baud rate allowable error (k = 0)} \quad \frac{\pm 15.5}{320} \times 100 = 4.8438\ (\%)$$

### (2) Communication operation

#### (a) Data format

The format for transmitting and receiving data is shown in Figure 16-7.

**Figure 16-7. Format of Asynchronous Serial Interface Transmit/Receive Data**




Each data frame is composed of 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 data frames, selection of the parity, and selection of the stop bit length, are performed with asynchronous serial interface mode register n (ASIMn).

If 7 bits has been selected as the number of character bits, only the lower 7 bits (bits 0 to 6) are valid. In the case of transmission, the most significant bit (bit 7) is ignored. In the case of reception, the most significant bit (bit 7) always becomes “0”.

The setting of the serial transfer rate is performed with the ASIMn and baud rate generator control register n (BRGCn).

If a serial data reception error occurs, it is possible to determine the contents of the reception error by reading the status of asynchronous serial interface status register n (ASISn).

**Remark** n = 1, 2

**(b) Parity types and operations**

Parity bits serve to detect bit errors in transmit data. Normally, the parity bit used on the transmit side and the receive side are of the same type. In the case of even parity and odd parity, it is possible to detect "1" bit (odd number) errors. In the case of 0 parity and no parity, errors cannot be detected.

**(i) Even parity**

- **During transmission**

  Makes the number of "1"s in transmit data that includes the parity bit even. The value of the parity bit changes as follows.

  If the number of "1" bits in transmit data is odd: 1
  if the number of "1" bits in transmit data is even: 0

- **During reception**

  The number of "1" bits in receive data that includes the parity bit is counted, and if it is odd, a parity error occurs.

**(ii) Odd parity**

- **During transmission**

  Odd parity is the reverse of even parity. It makes the number of "1"s in transmit data that includes the parity bit odd. The value of the parity bit changes as follows.

  If the number of "1" bits in transmit data is odd: 0
  if the number of "1" bits in transmit data is even: 1

- **During reception**

  The number of "1" bits in receive data is counted, and if it is even, a parity error occurs.

**(iii) 0 Parity**

During transmission, makes the parity bit "0", regardless of the transmit data.

A parity bit check is not performed during reception. Therefore, no parity error occurs, regardless of whether the parity bit value is "0" or "1".

**(iv) No parity**

No parity is appended to transmit data.

Transmit data is received assuming that it has no parity bit. No parity error can occur because there is no parity bit.

**(c) Transmission**

Transmission is begun by writing transmit data to transmission shift register n (TXSn). The start bit, parity bit, and stop bit(s) are automatically added.

The contents of transmit shift register n (TXSn) are shifted out upon transmission start, and when transmit shift register n (TXSn) becomes empty, a transmit interrupt (INTSTn) 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 or PM71 = 1), and write 0 to the port latch.**

**<2> Set bit 7 (TXEn) of asynchronous serial interface mode register n (ASIMn) to 1 to enable UART transmission (output a high level from the TXDn pin).**

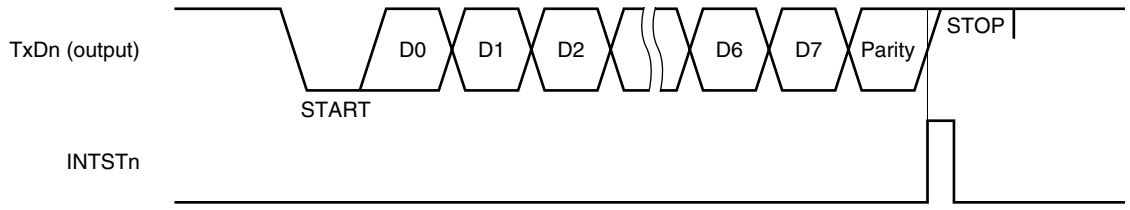
**<3> Set the port to the output mode (PM21 = 0 or PM71 = 0).**

**<4> Write transmit data to TXSn, and start transmission.**

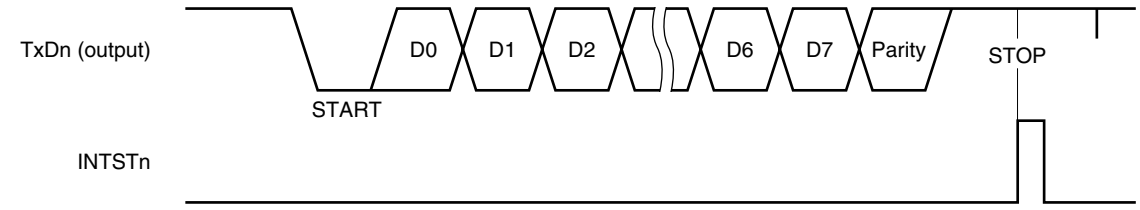
**If the port is set to the output mode first, 0 will be output from the pins, which may cause malfunction.**

**Remark** n = 1, 2

**Figure 16-8. Asynchronous Serial Interface Transmit Completion Interrupt Request Timing**




**(a) Stop bit length: 1**




**(b) Stop bit length: 2**

**Caution Do not write to asynchronous serial interface mode register n (ASIMn) during transmission. If you write to the ASIMn 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 (INTSTn) or the interrupt request flag (STIFn) set by INTSTn.**

**Remark** n = 1, 2

**(d) Reception**

When the RXEn bit of asynchronous serial interface mode register n (ASIMn) is set to 1, reception is enabled and sampling of the RxDn pin input is performed.

Sampling of the RxDn pin input is performed by the serial clock set by baud rate generator control register n (BRGCn).

The 5-bit counter for the baud rate generator will begin counting when the RxDn 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 RxDn 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. When, following the start bit, character data, the parity bit, and one stop bit are detected, reception of one frame of data is completed.

When reception of one frame of data is completed, the receive data in the shift register is transferred to receive shift register n (RXBn), and a receive completion interrupt (INTSRn) is generated.

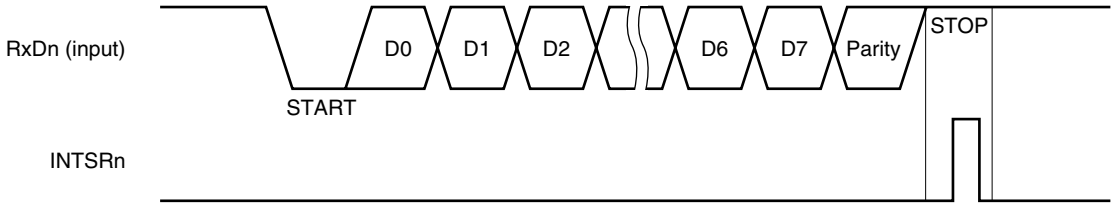
Also, even if an error occurs, the receiving data for which the error occurred is transferred to RXBn. If an error occurs, when bit 1 (ISRMn) of ASIMn is cleared (0), INTSRn is generated. (refer to **Figure 16-10**).

When bit ISRMn is set (1), INTSRn is not generated.

When bit RXEn is reset to 0 during a receive operation, the receive operation is immediately stopped. At this time, the contents of RXBn and ASISn remain unchanged, and INTSRn and INTSERn are not generated.

**Remark** n = 1, 2

**Figure 16-9. Asynchronous Serial Interface Receive Completion Interrupt Request Timing**




**Caution Even when a receive error occurs, be sure to read receive buffer register n (RXBn). If RXBn is not read, an overrun error will occur during reception of the next data, and the reception error status will continue indefinitely.**

**Remark** n = 1, 2

**(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 asynchronous serial interface status register n (ASISn), the receive error interrupt request (INTSERn) is generated. The receive error interrupt is generated before a receive completion interrupt request (INTSRn). Receive error causes are shown in Table 16-4.

What type of error has occurred during reception can be detected by reading the contents of asynchronous serial interface status register n (ASISn) during processing of the receive error interrupt (INTSERn) (refer to **Table 16-5** and **Figure 16-10**).

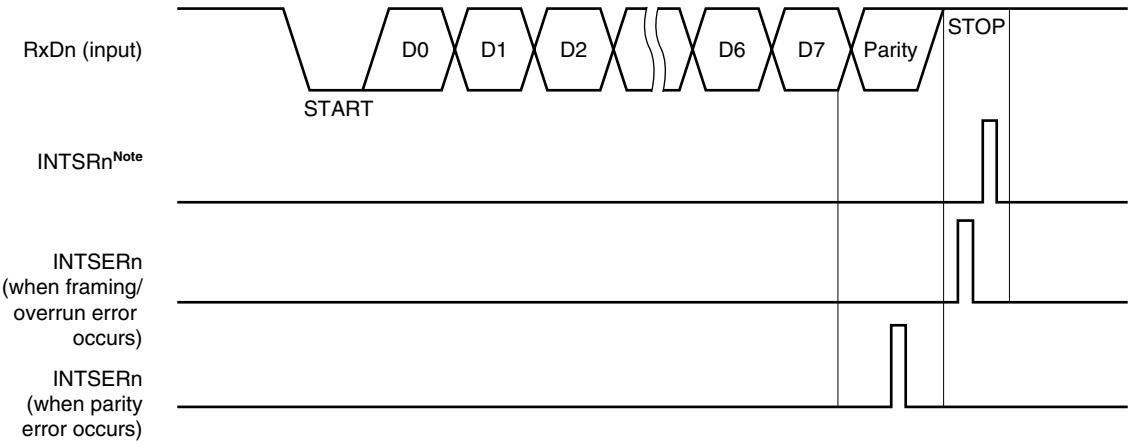
The contents of ASISn are reset to 0 either when receive buffer register n (RXBn) is read or when the next data is received (if the next data has an error, this error flag is set).

**Remark** n = 1, 2

**Table 16-5. Receive Error Causes**

| Receive Error | Cause | ASISn |
|---|---|---|
| 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 16-10. Receive Error Timing**




**Note** INTSRn will not be triggered if an error occurs when the ISRMn bit has been set (1).

**Cautions 1. The contents of ASISn are reset to 0 either when receive buffer register n (RXBn) is read or when the next data is received. To find out the contents of the error, be sure to read ASIS before reading RXBn.**

**2. Be sure to read receive buffer register n (RXBn) even when a receive error occurs. If RXBn is not read, an overrun error will occur at reception of the next data, and the receive error status will continue indefinitely.**

**Remark** n = 1, 2

### 16.3.3 Standby mode operation

**(1) HALT mode operation**

The serial transfer operation is normally performed.

**(2) STOP mode or IDLE mode operation**

**(a) When internal clock is selected as serial clock**

Asynchronous serial interface mode register n (ASIMn), transmit shift register n (TXSn), receive shift register n (RXn), and receive buffer register n (RXBn) stop operation holding the value immediately before the clock stops.

If the clock stops (STOP mode) during transmission, the TxDn 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.

**Remark** n = 1, 2

**(b) When external clock is selected as serial clock**

Serial transmission is performed normally. However, interrupt requests are held pending without being acknowledged. Interrupt requests are acknowledged after the STOP mode or IDLE mode has been released through NMI input, INTP0 to INTP5 input, or INTWT.

## 16.4 3-Wire Serial I/O Mode

This mode is used to perform 8-bit data transfer with the serial clock ($\overline{SCK1}$, $\overline{SCK2}$), serial output (SO1, SO2), and serial input (SI1, SI2) lines.

The 3-wire serial I/O mode supports simultaneous transmit/receive operations, thereby reducing the data transfer processing time.

The start bit of 8-bit data for serial transfer is fixed as the MSB.

The 3-wire serial I/O mode is effective when connecting peripheral I/O or a display controller with an on-chip clocked serial interface.

### 16.4.1 Configuration

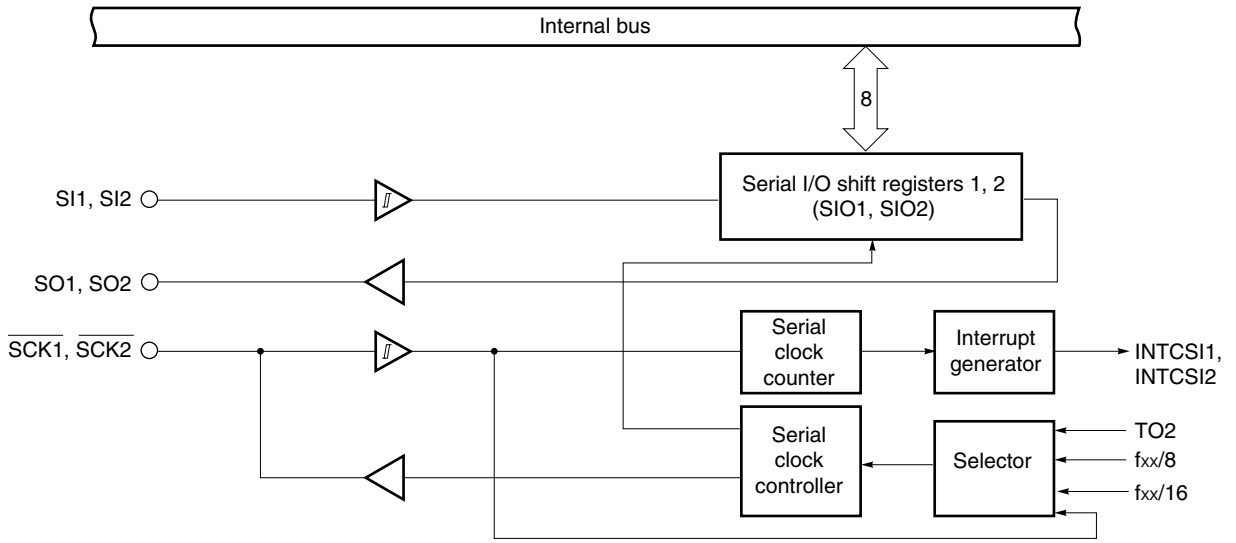
The 3-wire serial I/O mode includes the following hardware.

Figure 16-11 shows the block diagram for the 3-wire serial I/O mode.

**Table 16-6. 3-Wire Serial I/O Configuration**

| Item | Configuration |
|---|---|
| Registers | Serial I/O shift registers 1, 2 (SIO1, SIO2) |
| Control registers | Serial operation mode registers 1, 2 (CSIM1, CSIM2) |

**Figure 16-11. Block Diagram in 3-Wire Serial I/O Mode**




- **Serial I/O shift registers 1 and 2 (SIO1, SIO2)**

  These are 8-bit registers that perform parallel-serial conversion, and serial transmission/reception (shift operation) in synchronization with the serial clock.

  SIOn is set by an 8-bit memory manipulation instruction.

  When bit 7 (CSIEn) of serial operation mode register n (CSIMn) is 1, serial operation can be started by writing/reading data to/from SIOn.

  During transmission, data written to SIOn is output to the serial output pin (SOn).

  During reception, data is read into SIOn from the serial input pin (SIn).

  $\overline{\text{RESET}}$ input sets SIO1 and SIO2 to 00H.

**Caution During a transfer operation, do not access SIOn other than access as a transfer start trigger (read and write are prohibited when MODEn = 0 and MODEn = 1, respectively).**

**Remark** n = 1, 2

### 16.4.2 Control registers

- **Serial operation mode registers 1 and 2 (CSIM1, CSIM2)**

  CSIM1 and CSIM2 are used to set the serial clock, operation mode, and operation enable/disable in the 3-wire serial I/O mode.

  CSIM1 and SCIM2 can be set by a 1-bit or 8-bit memory manipulation instruction.

  $\overline{\text{RESET}}$ input sets CSIM1 and CSIM2 to 00H.

**Figure 16-12. Format of Serial Operation Mode Registers 1 and 2 (CSIM1, CSIM2)**

Address: 0FF91H, 0FF92H After reset: 00H R/W

| Symbol | ⑦ | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CSIMn | CSIEn | 0 | 0 | 0 | 0 | MODEn | SCLn1 | SCLn0 |

| CSIEn | SIOn operation enable/disable setting | | |
|---|---|---|---|
| | Shift register operation | Serial counter | Port |
| 0 | Operation disabled | Clear | Port function[Note] |
| 1 | Operation enabled | Counter operation enabled | Serial function + port function |

| MODEn | Transfer operation mode flag | | |
|---|---|---|---|
| | Operation mode | Transfer start trigger | SOn output |
| 0 | Transmit/receive mode | SIOn write | Normal output |
| 1 | Receive only mode | SIOn read | Fix to low level |

| SCLn1 | SCLn0 | Clock selection |
|---|---|---|
| 0 | 0 | External clock to $\overline{\text{SCKn}}$ |
| 0 | 1 | 8-bit timer counter 2 (TM2) output (TO2) |
| 1 | 0 | $f_{XX}$/8 (1.56 MHz) |
| 1 | 1 | $f_{XX}$/16 (781 kHz) |

**Notes 1.** When CSIEn = 0 (SIOn operation stop status), pins connected to SIn, SOn and $\overline{\text{SCKn}}$ can be used as ports.

**2.** Set the external clock and TO2 to $f_{XX}$/8 or below when selecting the external clock ($\overline{\text{SCKn}}$) and TM2 output (TO2) for the clock.

**Remarks 1.** n = 1, 2

**2.** Figures in parentheses apply to operation at $f_{XX}$ = 12.5 MHz.

### 16.4.3 Operation

The following two types of 3-wire serial I/O operation modes are available.

- Operation stopped mode
- 3-wire serial I/O mode

#### (1) Operation stopped mode

Serial transfer is not possible in the operation stopped mode, which reduces power consumption. Moreover, in operation stopped mode, pins can be used as normal I/O ports.

##### (a) Register setting

The operation stop mode is set by serial operation registers 1 and 2 (CSIM1, CSIM2).

CSIM1 and CSIM2 can be set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets CSIM1 and CSIM2 to 00H.

**Figure 16-13. Format of Serial Operation Mode Registers 1 and 2 (CSIM1, CSIM2)**

Address: 0FF91H, 0FF92H After reset: 00H R/W

| Symbol | ⑦ | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CSIMn | CSIEn | 0 | 0 | 0 | 0 | MODEn | SCLn1 | SCLn0 |

| CSIEn | SIOn operation enable/disable setting | | |
|---|---|---|---|
| | Shift register operation | Serial counter | Port |
| 0 | Operation disabled | Clear | Port function[Note] |
| 1 | Operation enabled | Counter operation enabled | Serial function + port function |

**Note** When CSIEn = 0 (SIOn operation stop status), pins connected to SIn, SOn and $\overline{\text{SCKn}}$ can be used as ports.

**Remark** n = 1, 2

**(2) 3-wire serial I/O mode**

The 3-wire serial I/O mode is effective when connecting peripheral I/O or a display controller with an on-chip clocked serial interface.

This mode is used to perform communication with the serial clock ($\overline{SCK1}$, $\overline{SCK2}$), serial output (SO1, SO2), and serial input (SI1, SI2) lines.

**(a) Register setting**

The 3-wire serial I/O mode is set by serial operation mode registers 1 and 2 (CSIM1, CSIM2).

CSIM1 and CSIM2 can be set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{RESET}$ input sets CSIM1 and CSIM2 to 00H.

**Figure 16-14. Format of Serial Operation Mode Registers 1 and 2 (CSIM1, CSIM2)**

Address: 0FF91H, 0FF92H After reset: 00H R/W

| Symbol | ⑦ | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CSIMn | CSIEn | 0 | 0 | 0 | 0 | MODEn | SCLn1 | SCLn0 |

| CSIEn | SIOn operation enable/disable setting | | |
|---|---|---|---|
| | Shift register operation | Serial counter | Port |
| 0 | Operation disabled | Clear | Port function[Note] |
| 1 | Operation enabled | Counter operation enabled | Serial function + port function |

| MODEn | Transfer operation mode flag | | |
|---|---|---|---|
| | Operation mode | Transfer start trigger | SOn output |
| 0 | Transmit/receive mode | SIOn write | Normal output |
| 1 | Receive only mode | SIOn read | Fix to low level |

| SCLn1 | SCLn0 | Clock selection |
|---|---|---|
| 0 | 0 | External clock to $\overline{SCKn}$ |
| 0 | 1 | 8-bit timer counter 2 (TM2) output (TO2) |
| 1 | 0 | $f_{XX}$/8 (1.56 MHz) |
| 1 | 1 | $f_{XX}$/16 (781 kHz) |

**Notes 1.** When CSIEn = 0 (SIOn operation stop status), pins connected to SIn, SOn and $\overline{SCKn}$ can be used as ports.

**2.** Set the external clock and TO2 to $f_{XX}$/8 or below when selecting the external clock ($\overline{SCKn}$) and TM2 output (TO2) for the clock.

**Remarks 1.** n = 1, 2

**2.** Figures in parentheses apply to operation at $f_{XX}$ = 12.5 MHz.

**(b) Communication operation**

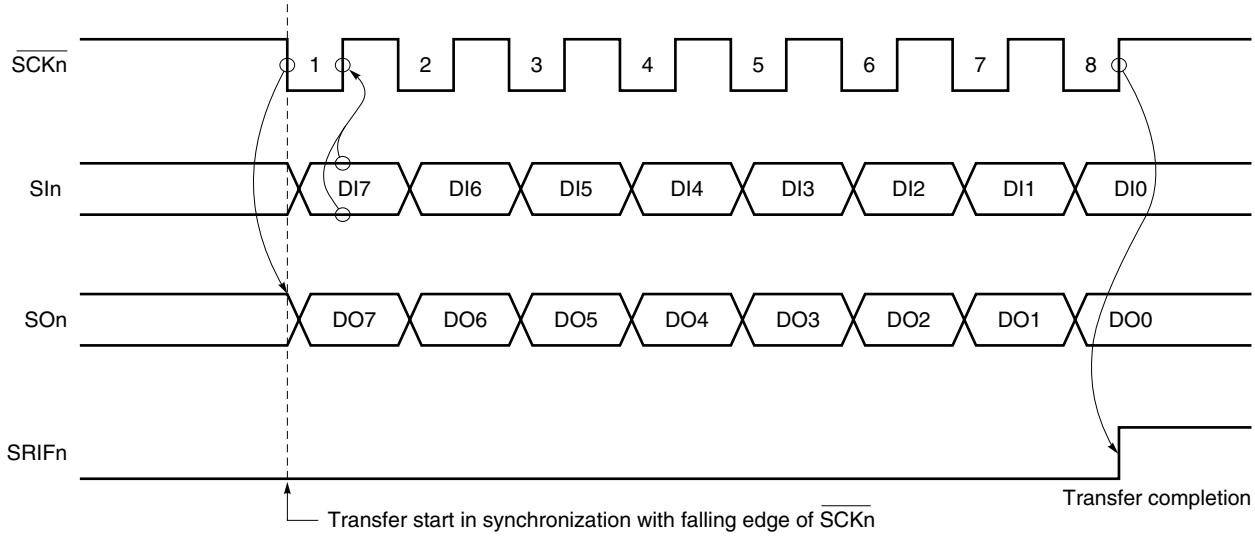
The 3-wire serial I/O mode performs data transfer in 8-byte units. Data is transmitted and received one byte at a time in synchronization with the serial clock.

The shift operation of the serial I/O shift register n (SIOn) is performed in synchronization with the falling edge of the serial clock ($\overline{SCKn}$). Transmit data is held in the SOn latch, and is output from the SOn pin. Receive data input to the SIn pin is latched to SIOn at the rising edge of the $\overline{SCKn}$ signal.

SIOn operation is automatically stopped when 8-bit transfer ends, and an interrupt request flag (SRIFn) is set.

**Remark** n = 1, 2

**Figure 16-15. 3-Wire Serial I/O Mode Timing**




**Remark** n = 1, 2

**(c) Transfer start**

Serial transfer is started by setting transmit data to (or reading) serial I/O shift register n (SIOn) when the following two conditions are satisfied.

- SIOn operation control bit (CSIEn) = 1
- Following 8-bit serial transfer, the internal serial clock is stopped, or $\overline{SCKn}$ is high level

- Transmit/receive mode
  When CSIEn = 1 and MODEn = 0, and transfer is started when SIOn is written
- Receive-only mode
  When CSIEn = 1 and MODEn = 1, and transfer is started when SIOn is read.

**Caution After data is written to SIOn, transfer will not start even if CSIEn is set to "1".**

Serial transfer automatically stops at the end of 8-bit transfer, and the interrupt request flag (SRIFn) is set.

**Remark** n = 1, 2

# CHAPTER 17 3-WIRE SERIAL I/O MODE

## 17.1 Function

This mode transfers 8-bit data by using the three lines of the serial clock ($\overline{SCK0}$), the serial output (SO0), and the serial input (SI0).

Since the 3-wire serial I/O mode can perform simultaneous transmission and reception, the data transfer processing time becomes shorter.

The start bit of the 8-bit data to be serially transferred is fixed to the MSB.

The 3-wire serial I/O mode is effective when connecting peripheral I/O or a display controller with an internal clocked serial interface.

## 17.2 Configuration

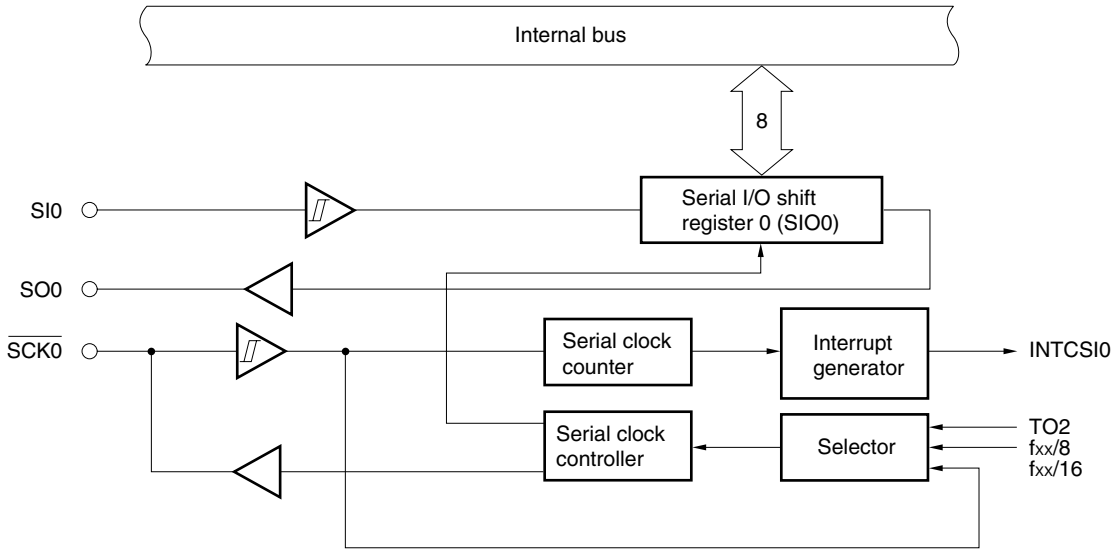
The 3-wire serial I/O mode includes the following hardware.

Figure 17-1 is a block diagram of the clocked serial interface (CSI) in the 3-wire serial I/O mode.

**Table 17-1. 3-Wire Serial I/O Configuration**

| Item | Configuration |
|---|---|
| Register | Serial I/O shift register 0 (SIO0) |
| Control register | Serial operation mode register 0 (CSIM0) |

**Figure 17-1. Block Diagram of Clocked Serial Interface (in 3-Wire Serial I/O Mode)**




- **Serial I/O shift register 0 (SIO0)**

  This 8-bit shift register performs parallel to serial conversion and serial communication (shift operations) synchronized with the serial clock.

  SIO0 is set by an 8-bit memory manipulation instruction.

  When bit 7 (CSIE0) in serial operation mode register 0 (CSIM0) is 1, 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 (SO0).

  When receiving, data is read from the serial input (SI0) 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.)**

## 17.3 Control Registers

- **Serial operation mode register 0 (CSIM0)**

The CSIM0 register sets the serial clock and operation 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 17-2. Format of Serial Operation Mode Register 0 (CSIM0)**

Address: 0FF90H After reset: 00H R/W

| Symbol | ⑦ | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CSIM0 | CSIE0 | 0 | 0 | 0 | 0 | MODE0 | SCL01 | SCL00 |

| CSIE0 | SIO0 operation enable/disable setting | | |
|---|---|---|---|
| | Shift register operation | Serial counter | Port |
| 0 | Operation disabled | Clear | Port function[Note] |
| 1 | Operation enabled | Count enabled | Serial function + Port function |

| MODE0 | Transfer operation mode flag | | |
|---|---|---|---|
| | Operation mode | Transfer start trigger | SO0 output |
| 0 | Transmit/receive communication mode | SIO0 write | Normal output |
| 1 | Receive only mode | SIO0 read | Fixed low |

| SCL01 | SCL00 | Clock selection |
|---|---|---|
| 0 | 0 | External clock to $\overline{\text{SCK0}}$ |
| 0 | 1 | 8-bit timer counter 2 (TM2) output (TO2) |
| 1 | 0 | $f_{XX}/8$ (1.56 MHz) |
| 1 | 1 | $f_{XX}/16$ (781 kHz) |

**Note** If CSIE0 = 0 (SIO0 operation stopped state), the pins connected to SI0, SO0 and $\overline{\text{SCK0}}$ can function as ports.

**Cautions 1. Set 8-bit timer mode control register 2 (TMC2) as follows when selecting 8-bit timer counter 2 (TM2) output as the clock.**
**TMC26 = 0, TMC24 = 0, LVS2 = 0, LVR2 = 0, TMC21 = 1**
**Moreover, set TOE2 to 0 when TO2 is not output externally and TOE2 to 1 when TO2 is output externally.**

**2. Set the external clock and TO2 to $f_{XX}/8$ or below when selecting the external clock ($\overline{\text{SCKn}}$) and TM2 output (TO2) for the clock.**

**Remark** Figures in parentheses apply to operation at $f_{XX}$ = 12.5 MHz.

★ **Table 17-2. Serial Interface Operation Mode Settings**

**(1) Operation stopped mode**

| CSIM0 | | | PM25 | P25 | PM26 | P26 | PM27 | P27 | First Bit | Shift Clock | P25/SI0/SDA0 Pin Function | P26/SO0 Pin Function | P27/$\overline{SCK0}$/SCL0 Pin Function |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CSIE0 | SCL01 | SCL00 | | | | | | | | | | | |
| 0 | × | × | ×Note 1 | ×Note 1 | ×Note 1 | ×Note 1 | ×Note 1 | ×Note 1 | – | – | P25 | P26 | P27 |
| Other than above | | | | | | | | | Setting prohibited | | | | |

**(2) 3-wire serial I/O mode**

| CSIM0 | | | PM25 | P25 | PM26 | P26 | PM27 | P27 | First Bit | Shift Clock | P25/SI0/SDA0 Pin Function | P26/SO0 Pin Function | P27/$\overline{SCK0}$/SCL0 Pin Function |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CSIE0 | SCL01 | SCL00 | | | | | | | | | | | |
| 1 | 0 | 0 | 1Note 2 | ×Note 2 | 0 | 0 | 1 | × | MSB | External clock | SIONote 2 | SO0 (CMOS output) | $\overline{SCK0}$ input |
| | Note 3 | Note 3 | | | | | 0 | 0 | | Internal clock | | | $\overline{SCK0}$ output |
| Other than above | | | | | | | | | Setting prohibited | | | | |

**Notes 1.** These pins can be used for port functions.

**2.** When only transmission is used, this pin can be used as P25 (CMOS I/O).

**3.** Refer to serial operation mode register 0 (CSIM0).

**Remark** ×: Don't care

## 17.4 Operation

3-wire serial I/O has the following two operation 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, the power consumption can be decreased.

In the operation stopped mode, the pins can be used as ordinary I/O port pins.

#### (a) Register settings

The operation stopped mode is set by serial operation 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 17-3. Format of Serial Operation Mode Register 0 (CSIM0)**

Address: 0FF90H After reset: 00H R/W

| Symbol | ⑦ | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CSIM0 | CSIE0 | 0 | 0 | 0 | 0 | MODE0 | SCL01 | SCL00 |

| CSIE0 | SIO0 operation enable/disable setting | | |
|---|---|---|---|
| | Shift register operation | Serial counter | Port |
| 0 | Operation disabled | Clear | Port function[Note] |
| 1 | Operation enabled | Operation count enabled | Serial function + port function |

**Note** If CSIE0 = 0 (SIO0 operation stopped state), the pins connected to SI0, SO0 and $\overline{\text{SCK0}}$ can function as ports.

### (2) 3-wire serial I/O mode

The 3-wire serial I/O mode is effective when connecting peripheral I/O or a display controller with an internal clocked serial interface.

Communication is over three lines, the serial clock ($\overline{SCK0}$), serial output (SO0), and serial input (SI0).

#### (a) Register setting

The 3-wire serial I/O mode is set by in serial operation 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 17-4. Format of Serial Operation Mode Register 0 (CSIM0)**

Address: 0FF90H After reset: 00H R/W

| Symbol | (7) | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CSIM0 | CSIE0 | 0 | 0 | 0 | 0 | MODE0 | SCL01 | SCL00 |

| CSIE0 | SIO0 operation enable/disable setting | | |
|---|---|---|---|
| | Shift register operation | Serial counter | Port |
| 0 | Operation disabled | Clear | Port function[Note] |
| 1 | Operation enabled | Operation count enabled | Serial function + port function |

| MODE0 | Transfer operation mode flag | | |
|---|---|---|---|
| | Operation mode | Transfer start trigger | SO0 output |
| 0 | Transmit/receive communication mode | SIO0 write | Normal output |
| 1 | Receive only mode | SIO0 read | Low level fixed |

| SCL01 | SCL00 | Clock selection |
|---|---|---|
| 0 | 0 | External clock to $\overline{SCK0}$ |
| 0 | 1 | 8-bit timer counter 2 (TM2) output (TO2) |
| 1 | 0 | $f_{XX}$/8 (1.56 MHz) |
| 1 | 1 | $f_{XX}$/16 (781 kHz) |

**Note** If CSIE0 = 0 (SIO0 operation stopped state), the pins connected to SI0, SO0 and $\overline{SCK0}$ can function as ports.

**Cautions 1. Set 8-bit timer mode control register 2 (TMC2) as follows when selecting 8-bit timer counter 2 (TM2) output as the clock.**
**TMC26 = 0, TMC24 = 0, LVS2 = 0, LVR2 = 0, TMC21 = 1**
**Moreover, set TOE2 to 0 when TO2 is not output externally and TOE2 to 1 when TO2 is output externally.**

**2. Set the external clock and TO2 to $f_{XX}$/8 or below when selecting the external clock ($\overline{SCK0}$) and TM2 output (TO2) for the clock.**

**Remark** Figures in parentheses apply to operation at $f_{XX}$ = 12.5 MHz.

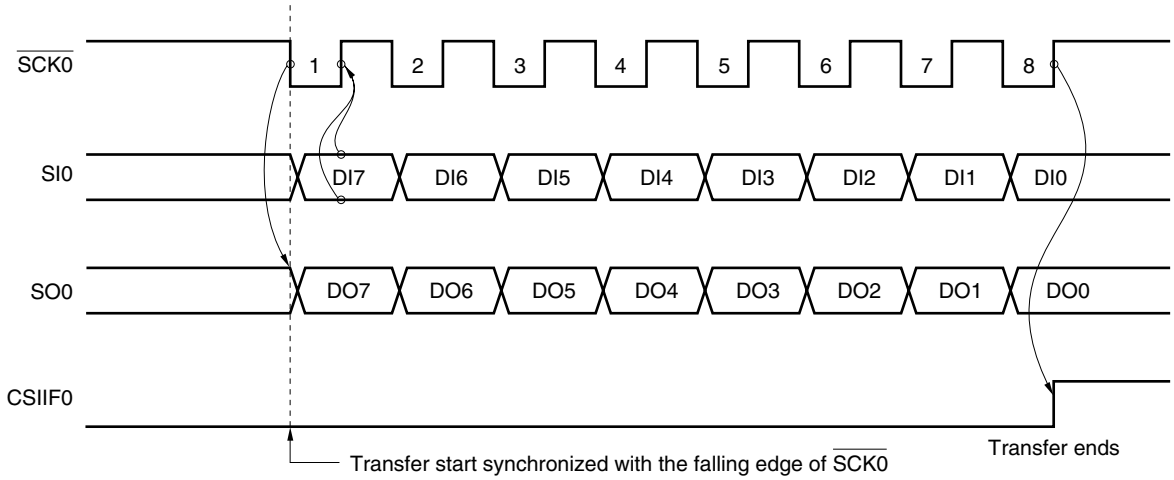
**(b) Communication operation**

The 3-wire serial I/O mode transmits and receives data in 8-bit units. Data is transmitted and received with each bit synchronized with the serial clock.

The shifting of serial I/O shift register 0 (SIO0) is synchronized with the falling edge of the serial clock ($\overline{SCK0}$). The transmitted data is held in the latch and output from the SO0 pin. At the rising edge of $\overline{SCK0}$, the received data that was input to the SI0 pin is latched to SIO0.

At the end of the 8-bit transfer, the SIO0 operation automatically stops and the interrupt request flag (CSIIF0) is set.

**Figure 17-5. 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 (SIO0).

- Control bit (CSIE0) = 1 during SIO0 operation
- After an 8-bit serial transfer, the internal serial clock enters the stopped state or $\overline{SCK0}$ is high.

- 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 (CSIIF0) is set.

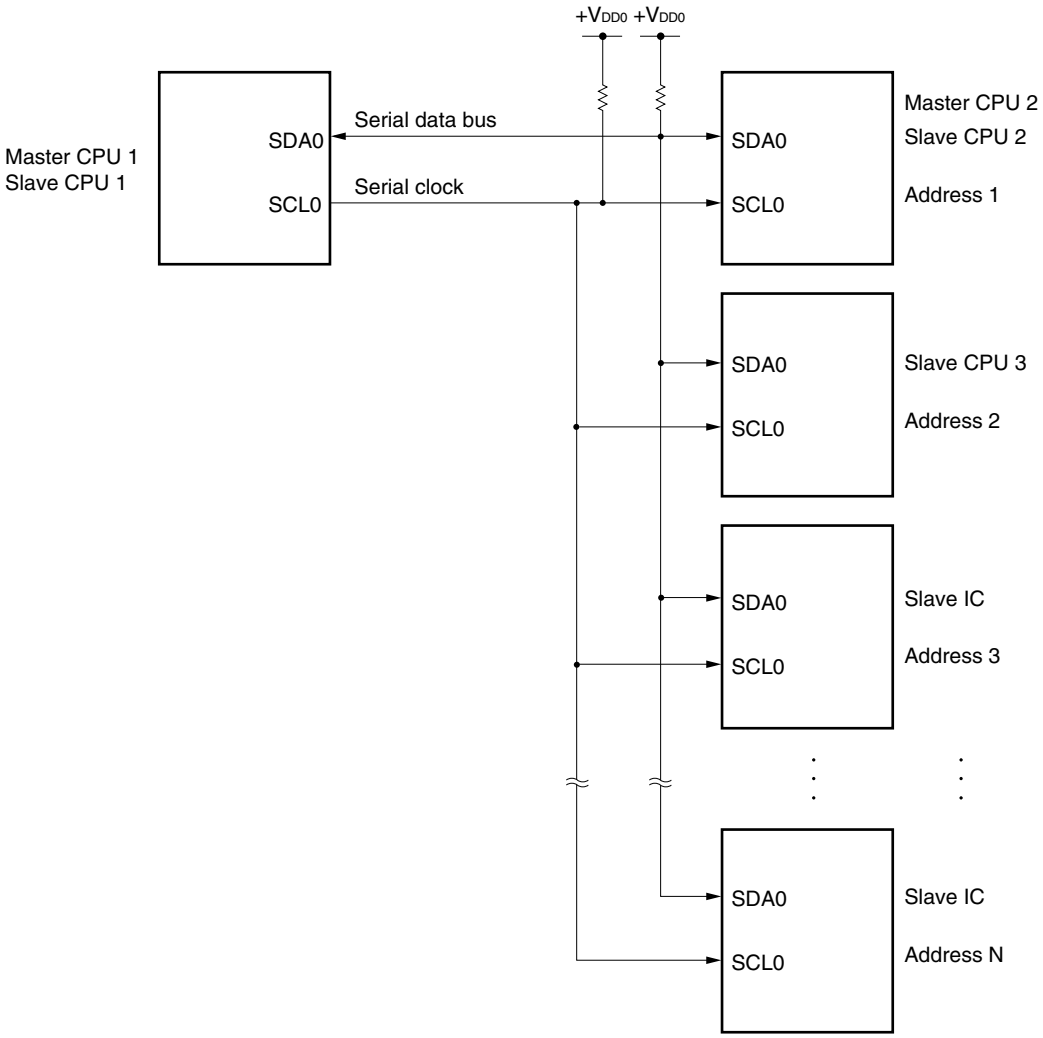
# CHAPTER 18 $I^2C$ BUS MODE ($\mu$PD784225Y SUBSERIES ONLY)

## 18.1 Function Overview

- **$I^2C$ (Inter IC) bus mode (supporting multi master)**
  This interface communicates with devices that conform to the $I^2C$ bus format.
  Eight bit data transfers with multiple devices are performed by the two lines of the serial clock (SCL0) and the serial data bus (SDA0).
  In the $I^2C$ bus mode, the master can output the start condition, data, and stop condition on the serial data bus to the slaves.
  The slaves automatically detect the received data by hardware. The $I^2C$ bus control portion of the application program can be simplified by using this function.
  Since SCL0 and SDA0 become open-drain outputs in the $I^2C$ bus mode, pull-up resistors are required on the serial clock line and serial data bus line.

  **Cautions 1. If the power to the $\mu$PD784225Y is disconnected while $\mu$PD784225Y functions are not used, $I^2C$ communication may no longer be possible. Even when not being used, do not disconnect the power to the $\mu$PD784225Y.**
  **2. If the $I^2C$ bus mode is used, set the SCL0/P27 and SDA0/P25 pins to N-channel open-drains by setting the port function control register (PF2).**

**Figure 18-1. Serial Bus Configuration Example in I²C Bus Mode**




## 18.2 Configuration

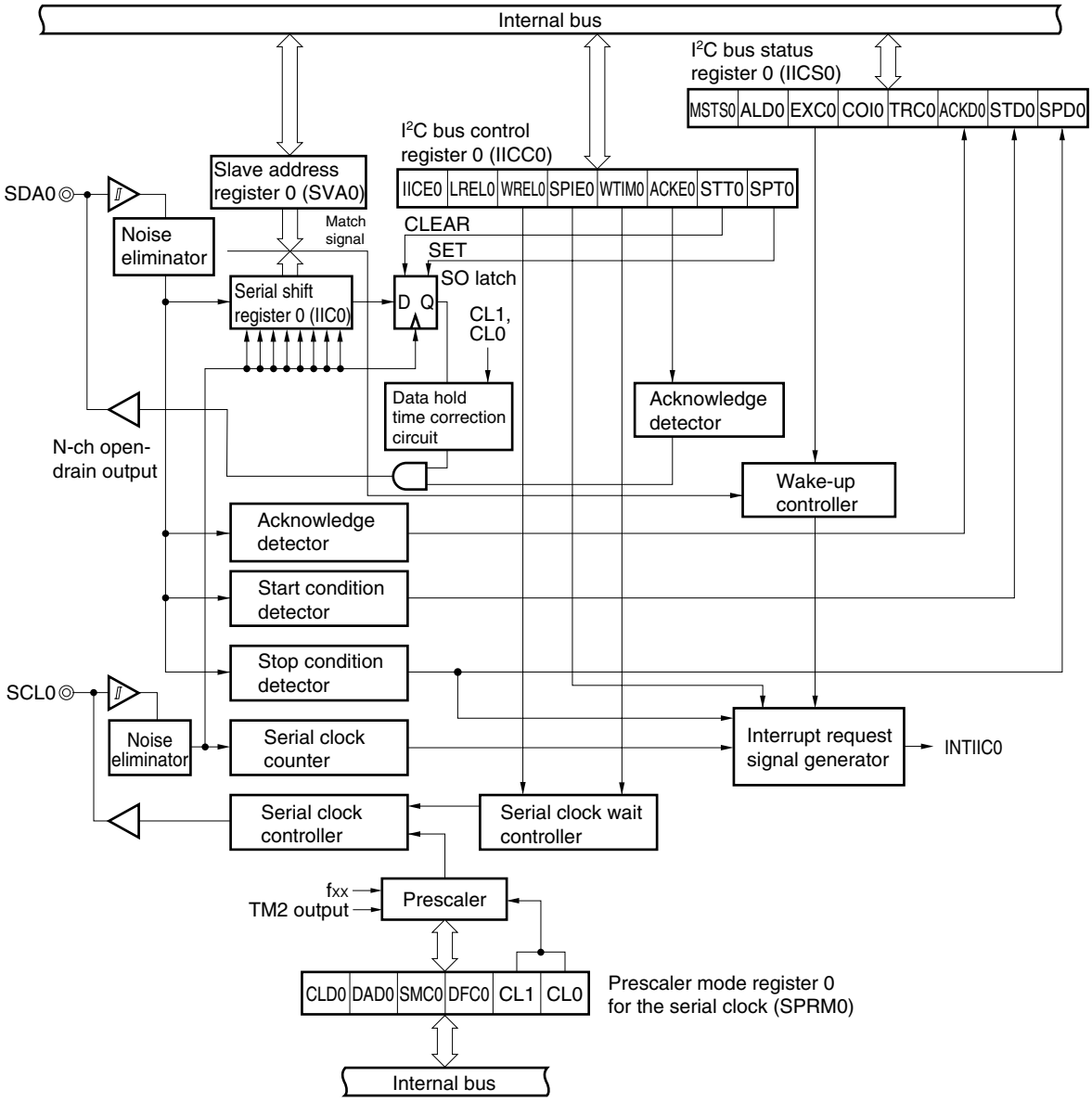
The clocked serial interface in the I²C bus mode includes the following hardware.

Figure 18-2 is a block diagram of clocked serial interface (IIC0) in the I²C bus mode.

**Table 18-1. I²C Bus Mode Configuration**

| Item | Configuration |
|---|---|
| Registers | Serial shift register 0 (IIC0)<br>Slave address register 0 (SVA0) |
| Control registers | I²C bus control register 0 (IICC0)<br>I²C bus status register 0 (IICS0)<br>Prescaler mode register 0 for serial clock (SPRM0) |

**Figure 18-2. Block Diagram of Clocked Serial Interface (I²C Bus Mode)**



**(1) Serial shift register 0 (IIC0)**

The IIC0 register converts 8-bit serial data into 8-bit parallel data and 8-bit parallel data into 8-bit serial data. IIC0 is used in both transmission and reception.

The actual transmission and reception are controlled by writing and reading IIC0.

IIC0 is set by an 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets IIC0 to 00H.

**(2) Slave address register 0 (SVA0)**

When used as a slave, this register sets a slave address.

SVA0 is set by an 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets SVA0 to 00H.

**(3) SO latch**

The SO latch holds the output level of the SDA0 pin.

**(4) Wake-up controller**

This circuit generates an interrupt request when the address set in slave address register 0 (SVA0) and the receive address match, or when an extended code is received.

**(5) Clock selector**

This selects the sampling clock that is used.

**(6) Serial clock counter**

The serial clock that is output or input during transmission or reception is counted to check 8-bit data communication.

**(7) Interrupt request signal generator**

This circuit controls the generation of the interrupt request signal (INTIIC0).

The I²C interrupt request is generated by the following two triggers.

- Eighth or ninth clock of the serial clock (set by the WTIM0 bit**Note**)
- Interrupt request is generated by detecting the stop condition (set by the SPIE0 bit**Note**)

**Note** WTIM0 bit: Bit 3 in I²C bus control register 0 (IICC0)
SPIE0 bit: Bit 4 in I²C bus control register 0 (IICC0)

**(8) Serial clock controller**

In the master mode, the clock output to pin SCL0 is generated by the sampling clock.

**(9) Serial clock wait controller**

This circuit controls the wait timing.

**(10) Acknowledge output circuit, stop condition detector, start condition detector, acknowledge detector**

These circuits output and detect the control signals.

**(11) Data hold time correction circuit**

This circuit generates the hold time of the data to the falling edge of the serial clock.

## 18.3 Control Registers

The I²C bus mode is controlled by the following three registers.

- I²C bus control register 0 (IICC0)
- I²C bus status register 0 (IICS0)
- Prescaler mode register 0 for serial clock (SPRM0)

The following registers are also used.

- Serial shift register 0 (IIC0)
- Slave address register 0 (SVA0)

**(1) I²C bus control register 0 (IICC0)**

The IICC0 register enables and disables the I²C bus mode, sets the wait timing, and sets other I²C bus mode operations.

IICC0 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets IICC0 to 00H.

**Figure 18-3. Format of I²C Bus Control Register 0 (IICC0) (1/4)**

Address: 0FFB0H After reset: 00H R/W

| Symbol | ⑦ | ⑥ | ⑤ | ④ | ③ | ② | ① | ⓪ |
|---|---|---|---|---|---|---|---|---|
| IICC0 | IICE0 | LREL0 | WREL0 | SPIE0 | WTIM0 | ACKE0 | STT0 | SPT0 |

| IICE0 | I²C operation enable |
|---|---|
| 0 | Operation disabled. Presets the I²C bus status register (IICS0). Stops internal operation. SCL0 and SDA0 lines output low level. |
| 1 | Enables operation. |

| • Clear condition (IICE0 = 0) | • Set condition (IICE0 = 1) |
|---|---|
| • Cleared by an instruction<br>• When $\overline{\text{RESET}}$ is input | • Set by an instruction |

| LREL0 | Release communication |
|---|---|
| 0 | Normal operation |
| 1 | Releases microcontroller from the current communication and sets it in the wait state. Automatically clears after execution.<br>The extended code that is unrelated to the base is used during reception.<br>The SCL0 and SDA0 lines are put in the high impedance state.<br>The following flags are cleared.<br>• STD0 • ACKD0 • TRC0 • COI0 • EXC0 • MSTS0 • STT0 • SPT0 |

Until the following communication participation conditions are satisfied, the wait state that released the microcontroller from communication is entered.
- Start as the master after detecting the stop condition.
- Address match or extended code reception after the start condition

| Clear condition (LREL0 = 0)[Note] | Set condition (LREL0 = 1) |
|---|---|
| • Automatically cleared after execution.<br>• When $\overline{\text{RESET}}$ is input | • Set by an instruction |

| WREL0 | Wait release |
|---|---|
| 0 | The wait is not released. |
| 1 | The wait is released. After the wait is released, it is automatically cleared. |

| Clear condition (WREL0 = 0)[Note] | Set condition (WREL0 = 1) |
|---|---|
| • Automatically cleared after execution.<br>• When $\overline{\text{RESET}}$ is input | • Set by an instruction |

| SPIE0 | Enable/disable generation of interrupt request by stop condition detection |
|---|---|
| 0 | Disable |
| 1 | Enable |

| Clear condition (SPIE0 = 0)[Note] | Set condition (SPIE0 = 1) |
|---|---|
| • Cleared by an instruction<br>• When $\overline{\text{RESET}}$ is input | • Set by an instruction |

**Note** This flag signal becomes invalid by setting IICE0 to 0.

**Figure 18-3. Format of I²C Bus Control Register 0 (IICC0) (2/4)**

| WTIM0 | Control of wait and interrupt request generation |
|---|---|
| 0 | Interrupt request generated at the falling edge of the eighth clock<br>For the master: After the eighth clock is output, wait with the clock output low.<br>For the slave: After the eighth clock is input, the master waits with the clock low. |
| 1 | Interrupt request generated at the falling edge of the ninth clock<br>For the master: After the ninth clock is output, wait with the clock output low.<br>For the slave: After the ninth clock is input, the master waits with the clock low. |
| This bit setting becomes invalid during an address transfer, and becomes valid after the transfer ends. In the master, a wait is inserted at the falling edge of the ninth clock in an address transfer. The slave that received the base address inserts a wait at the falling edge of the ninth clock after the acknowledge is generated. The slave that received the extended code inserts the waits at the falling edge of the eighth clock. | |
| Clear condition (WTIM0 = 0)**Note** | Set condition (WTIM0 = 1) |
| • Cleared by an instruction<br>• When $\overline{\text{RESET}}$ is input | • Set by an instruction |

| ACKE0 | Acknowledge control |
|---|---|
| 0 | Acknowledge is disabled. |
| 1 | Acknowledge is enabled. The SDA0 line during the ninth clock period goes low. However, the control is invalid during an address transfer. When EXC0 = 1, the control is valid. |
| Clear condition (ACKE0 = 0)**Note** | Set condition (ACKE0 = 1) |
| • Cleared by an instruction<br>• When $\overline{\text{RESET}}$ is input | Set by an instruction |

**Note** This flag signal becomes invalid by setting IICE0 to 0.

**Figure 18-3. Format of I²C Bus Control Register 0 (IICC0) (3/4)**

| STT0 | Start condition trigger |
|---|---|
| 0 | The start condition is not generated. |
| 1 | • When the bus is released (stop condition):<br>The start condition is generated (started as the master). The SDA0 line is changed from high to low, and the start condition is generated. Then, the standard time is guaranteed, and SCL0 goes low.<br>• When not participating with the bus:<br>The trigger functions as the start condition reserved flag. When set, the start condition is automatically generated after the bus is released.<br>• Wait status (when master)<br>The wait status is canceled and the restart condition is generated. |

Cautions on set timing

- Master reception: Setting is prohibited during transfer. STT0 can be set only during the wait period after ACKE0 = 0 is set and the fact that reception is completed is passed to the slave.
- Master transmission: During the ACK0 acknowledge period, the start condition may not be normally generated. Set STT0 during the wait period.
- Setting synchronized to SPT0 is prohibited.
- Resetting between setting STT0 and the generation of the clear condition is prohibited.

| Clear condition (STT0 = 0) | Set condition (STT0 = 1) |
|---|---|
| • Cleared by an instruction<br>• IICE0 = 0<br>• LREL0 = 1<br>• When arbitration failed<br>• Clear after generating the start condition in the master.<br>• When $\overline{\text{RESET}}$ is input | • Set by an instruction |

**Figure 18-3. Format of I²C Bus Control Register 0 (IICC0) (4/4)**

| SPT0 | Stop condition trigger |
|---|---|
| 0 | The stop condition is not generated. |
| 1 | The stop condition is generated (ends the transfer as the master).<br>After the SDA0 line goes low, the SCL0 line goes high, or wait until SCL0 goes high. Then, the standard time is guaranteed; the SDA0 line is changed from low to high; and the stop condition is generated. |

Cautions on set timing

- Master reception: Setting is prohibited during transfer.
  SPT0 can be set only during the wait period after ACK0=0 is set and the fact that reception is completed is passed to the slave.
- Master transmission: During the ACK0 acknowledge period, the start condition may not be normally generated. Set SPT0 during the wait period.
- Setting synchronized to STT0 is prohibited.
- Resetting between setting SPT0 and the generation of the clear condition is prohibited.
- Set SPT0 only by the master[Note].
- When WTIM0 = 0 is set, be aware that if SPT0 is set during the wait period after the eighth clock is output, the stop condition is generated during the high level of the ninth clock after the wait is released.
  When the ninth clock must be output, set WTIM0 = 0 → 1 during the wait period after the eighth clock is output, and set SPT0 during the wait period after the ninth clock is output.

| Clear condition (SPT0 = 0) | Set condition (SPT0 = 1) |
|---|---|
| • Cleared by an instruction<br>• IICE0 = 0<br>• LREL0 = 0<br>• When arbitration failed<br>• Automatically clear after the stop condition is detected<br>• When $\overline{\text{RESET}}$ is input | • Set by an instruction |

**Note** Set SPT0 only by the master. However, SPT0 must be set once and the stop condition generated to operate the master by the time the first stop condition is detected after operation is enabled. For details, refer to **18.5.15 Additional warnings**.

**Cautions 1. When bit 3 (TRC0) = 1 in I²C bus status register 0 (IICS0), after WREL0 is set at the ninth clock and the wait is released, TRC0 is cleared, and the SDA0 line becomes high impedance.**

**2. SPT0 and STT0 are 0 when read after data has been set.**

**Remark** STD0: Bit 1 in I²C bus status register 0 (IICS0)
ACKD0: Bit 2 in I²C bus status register 0 (IICS0)
TRC0: Bit 3 in I²C bus status register 0 (IICS0)
COI0: Bit 4 in I²C bus status register 0 (IICS0)
EXC0: Bit 5 in I²C bus status register 0 (IICS0)
MSTS0: Bit 7 in I²C bus status register 0 (IICS0)

### (2) I²C bus status register 0 (IICS0)

The IICS0 register displays the status of the I²C bus.

IICS0 is set by a 1-bit or 8-bit memory manipulation instruction. IICS0 can only be read.

$\overline{\text{RESET}}$ input sets IICS0 to 00H.

**Figure 18-4. Format of I²C Bus Status Register 0 (IICS0) (1/3)**

Address: 0FFB6H After reset: 00H R

| Symbol | ⑦ | ⑥ | ⑤ | ④ | ③ | ② | ① | ⓪ |
|---|---|---|---|---|---|---|---|---|
| IICS0 | MSTS0 | ALD0 | EXC0 | COI0 | TRC0 | ACKD0 | STD0 | SPD0 |

| MSTS0 | Master state |
|---|---|
| 0 | Slave state or communication wait state |
| 1 | Master communication state |

| Clear condition (MSTS0 = 0) | Set condition (MSTS0 = 1) |
|---|---|
| • When the stop condition is detected<br>• When ALD0 = 1<br>• Cleared by LREL0 = 1<br>• When $\overline{\text{IICE0}}$ = 1 → 0<br>• When $\overline{\text{RESET}}$ is input | • When start condition is generated |

| ALD0 | Arbitration failed detection |
|---|---|
| 0 | No arbitration state or arbitration win state |
| 1 | Arbitration failed state. MSTS0 is cleared. |

| Clear condition (ALD0 = 0) | Set condition (ALD0 = 1) |
|---|---|
| • Automatically cleared after IICS0 is read[Note]<br>• When $\overline{\text{IICE0}}$ = 1 → 0<br>• When $\overline{\text{RESET}}$ is input | • When arbitration failed |

| EXC0 | Extended code reception detection |
|---|---|
| 0 | The extended code is not received. |
| 1 | The extended code is received. |

| Clear condition (EXC0 = 0) | Set condition (EXC0 = 1) |
|---|---|
| • When the start condition is detected<br>• When the stop condition is detected<br>• Cleared by LREL0 = 1<br>• When $\overline{\text{IICE0}}$ = 1 → 0<br>• When $\overline{\text{RESET}}$ is input | • When the most significant four bits of the received address data are 0000 or 1111 (set by the rising edge of the eighth clock) |

**Note** This is cleared when a bit manipulation instruction is executed for other bits in IICS0.

**Figure 18-4. Format of I²C Bus Status Register 0 (IICS0) (2/3)**

| COI0 | Address match detection |
|---|---|
| 0 | The address does not match. |
| 1 | The address matches. |

| Clear condition (COI0 = 0) | Set condition (COI0 = 1) |
|---|---|
| • During start condition detection<br>• During stop condition detection<br>• Cleared by LREL0 = 1<br>• When IICE0 = 1 → 0<br>• When $\overline{\text{RESET}}$ is input | • When the received address matches the base address (SVA0) (set at the rising edge of the eighth clock) |

| TRC0 | Transmission/reception state detection |
|---|---|
| 0 | Reception state (not the transmission state). The SDA0 line has high impedance. |
| 1 | Transmission state. The value in the SO latch can be output to the SDA0 line (valid after the falling edge of the ninth clock of the first byte) |

| Clear condition (TRC0 = 0) | Set condition (TRC0 = 1) |
|---|---|
| • When the stop condition is detected<br>• Cleared by LREL0 = 1<br>• When IICE0 = 1 → 0<br>• Cleared by WREL0 = 1**Note**<br>• When ALD0 = 0 → 1<br>• When $\overline{\text{RESET}}$ is input<br>In the master<br>• When 1 is output to the first byte LSB (transfer direction specification bit)<br>In the slave<br>• When the start condition is detected<br>When not participating in the communication | In the master<br>• When the start condition is generated<br>In the slave<br>• When 1 is input to the LSB of the first byte (transfer direction specification bit) |

**Note** If a wait is cancelled by setting bit 5 (WREL0) of I²C bus control register 0 (IICC0) at the ninth clock while bit 3 (TRC0) of I²C bus status register 0 (IICS0) is 1, TRC0 is cleared and the SDA0 line becomes high impedance.

**Figure 18-4. Format of I²C Bus Status Register 0 (IICS0) (3/3)**

| ACKD0 | Acknowledge detection |
|---|---|
| 0 | The acknowledge is not detected. |
| 1 | The acknowledge is detected. |

| Clear condition (ACKD0 = 0) | Set condition (ACKD0 = 1) |
|---|---|
| • When the stop condition is detected<br>• At the rising edge of the first clock in the next byte<br>• Cleared by LREL0 = 1<br>• When IICE0 = 1 → 0<br>• When $\overline{\text{RESET}}$ is input | When the SDA0 line is low at the rising edge of the ninth clock of SCL0 |

| STD0 | Start condition detection |
|---|---|
| 0 | The start condition is not detected. |
| 1 | The start condition is detected. This indicates the address transfer period. |

| Clear condition (STD0 = 0) | Set condition (STD0 = 1) |
|---|---|
| • When the stop condition is detected<br>• At the rising edge of the first clock of the next byte after transferring the address<br>• Cleared by LREL0 = 1<br>• When IICE0 = 1 → 0<br>• When $\overline{\text{RESET}}$ is input | • When the start condition is detected |

| SPD0 | Stop condition detection |
|---|---|
| 0 | The stop condition is not detected. |
| 1 | The stop condition is detected. Communication is ended by the master, and the bus is released. |

| Clear condition (SPD0 = 0) | Set condition (SPD0 = 1) |
|---|---|
| • After the bit is set, at the rising edge of the first clock in the address transfer byte after detecting the start condition<br>• When IICE0 = 1 → 0<br>• When $\overline{\text{RESET}}$ is input | • When the stop condition is detected |

**Remark** LREL0: Bit 6 of I²C bus control register 0 (IICC0)
IICE0: Bit 7 of I²C bus control register 0 (IICC0)

**(3) Prescaler mode register 0 for serial clock (SPRM0)**

The SPRM0 register sets the transfer clock of the I²C bus.

SPRM0 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets SPRM0 to 00H.

**Figure 18-5. Format of Prescaler Mode Register 0 for Serial Clock (SPRM0) (1/2)**

Address: 0FFB2H After reset: 00H R/W[Note]

| Symbol | 7 | 6 | ⑤ | ④ | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| SPRM0 | 0 | 0 | CLD | DAD | SMC | DFC | CL1 | CL0 |

| CLD | SCL0 line level detection (valid only when IICE0 = 1) |
|---|---|
| 0 | Detects a low SCL0 line. |
| 1 | Detects a high SCL0 line. |

| Clear condition (CLD = 0) | Set condition (CLD = 1) |
|---|---|
| • When the SCL0 line is low<br>• When $\overline{\text{IICE0}}$ = 0<br>• When $\overline{\text{RESET}}$ is input | • When the SCL0 line is high |

| DAD | SDA0 line level detection (valid only when IICE0 = 1) |
|---|---|
| 0 | Detects a low SDA0 line. |
| 1 | Detects a high SDA0 line. |

| Clear condition (DAD = 0) | Set condition (DAD = 1) |
|---|---|
| • When the SDA0 line is low<br>• When $\overline{\text{IICE0}}$ = 0<br>• When $\overline{\text{RESET}}$ is input | • When the SDA0 line is high |

**Note** Bits 4 and 5 are read only.

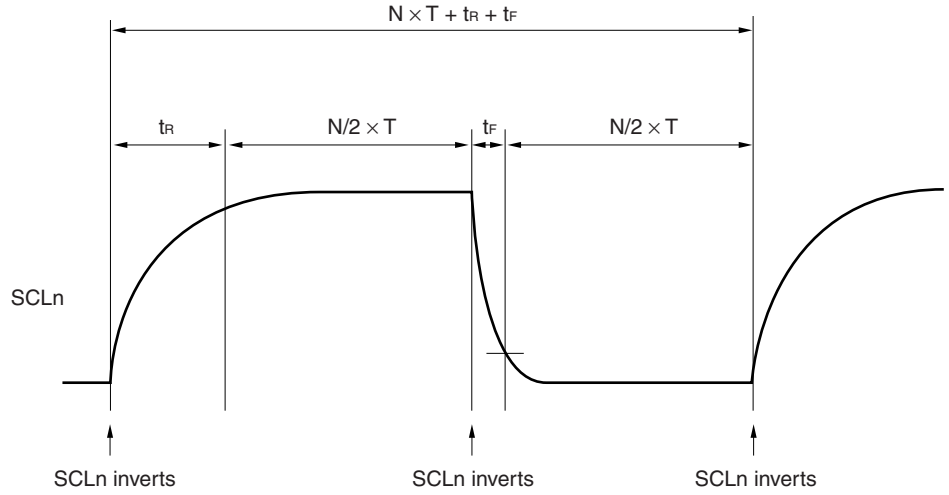
**Figure 18-5. Format of Prescaler Mode Register 0 for Serial Clock (SPRM0) (2/2)**

| SMC[Note 1] | DFC[Note 2] | CL1 | CL0 | Transfer clock | fxx setting allowable range |
|---|---|---|---|---|---|
| 0 | 1/0 | 0 | 0 | fxx/44 | 2 to 4.19 MHz |
| 0 | 1/0 | 0 | 1 | fxx/86 | 4.19 to 8.38 MHz |
| 0 | 1/0 | 1 | 0 | fxx/172 | 8.38 to 12.5 MHz |
| 0 | 1/0 | 1 | 1 | TM2 output/66 | |
| 1 | 1/0 | 0 | 1/0 | fxx/24 | 4 to 8.38 MHz |
| 1 | 1/0 | 1 | 0 | fxx/48 | 8 to 12.5 MHz |
| 1 | 1/0 | 1 | 1 | TM2 output/18 | |

**Notes 1.** SMC: Bit to change operation mode
0: Operates in normal mode
1: Operates in high-speed mode
**2.** DFC: Bit to control digital filter operation
0: Digital filter off
1: Digital filter on

**Cautions 1. Rewrite the SPRM0 after clearing the IICE0.**
**2. Set the transfer clock as follows:**
**When SMC = 0: 100 kHz or below**
**When SMC = 1: 400 kHz or below**

**Remarks 1.** IICE0: Bit 7 of I²C bus control register 0 (IICC0)
**2.** The transfer clock does not change due to the ON/OFF setting of bit 2 (DFC) in high-speed mode.
**3.** IIC clock: Clock frequency when fxx/N is selected




IIC clock frequency: $f_{SCL} = 1/(N \times T + tR + tF)$
T = 1/fxx, tR: SCLn rise time, tF: SCLn fall time
**Example** When fxx = 12.5 MHz, N = 172, $t_R$ = 200 ns, $t_F$ = 50 ns
IIC clock frequency: 1/(172 × 80 ns + 200 ns + 50 ns) ≅ 71.4 kHz

**(4) Serial shift register 0 (IIC0)**

This register performs serial communication (shift operation) synchronized with the serial clock.

Although this register can be read and written in 1-bit and 8-bit units, do not write data to IIC0 during a data transfer.

Address: 0FFB8H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| IIC0 | | | | | | | | |

**(5) Slave address register 0 (SVA0)**

This register stores the slave address of the I²C bus.

It can be read and written in 8-bit units, but bit 0 is fixed to 0.

Address: 0FFB4H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| SVA0 | | | | | | | | 0 |

## 18.4 I²C Bus Mode Function

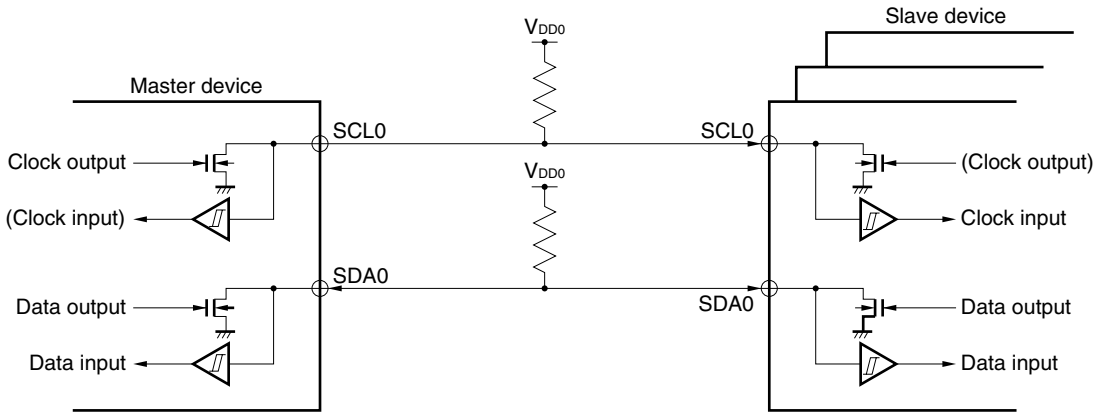
### 18.4.1 Pin configuration

The serial clock pin (SCL0) and the serial data bus pin (SDA0) have the following configurations.

(1) SCL0 ····· I/O pin for the serial clock
The outputs to both the master and slave are N-ch open drains. The input is a Schmitt input.
(2) SDA0 ····· Shared I/O pin for serial data
The outputs to both the master and slave are N-ch open drains. The input is a Schmitt input.

Since the outputs of the serial clock line and serial data bus line are N-ch open drains, external pull-up resistors are required.

**Figure 18-6. Pin Configuration**

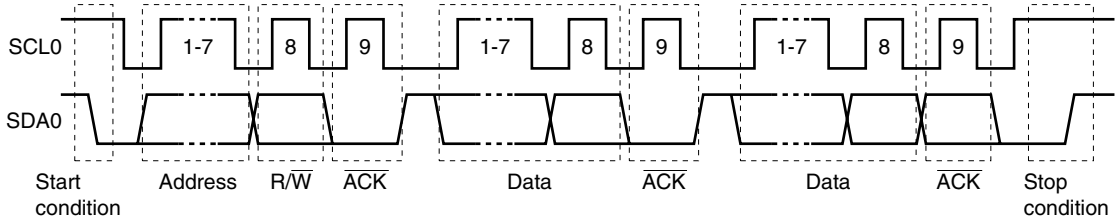


## 18.5 I²C Bus Definitions and Control Method

Next, the serial data communication formats of the I²C bus and the meanings of the signals used are described.

Figure 18-7 shows the transfer timing of the start condition, data, and stop condition that are output on the serial data bus of the I²C bus.

**Figure 18-7. Serial Data Transfer Timing of I²C Bus**




The master outputs the start condition, slave address, and stop condition.

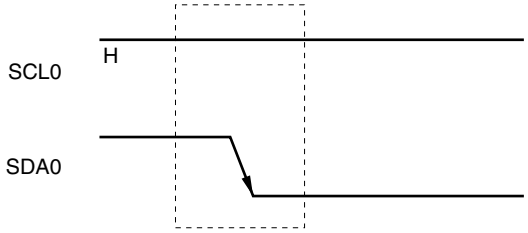
The acknowledge signal ($\overline{ACK}$) can be output by either the master or slave. (Normally, this is output on the side receiving 8-bit data.)

The serial clock (SCL0) continues to be output by the master. However, the slave can extend the SCL0 low-level period and insert waits.

### 18.5.1 Start condition

When the SCL0 pin is high, the start condition is the SDA0 pin changing from high to low. The start conditions for the SCL0 and SDA0 pins are the signals output when the master starts the serial transfer to the slave. The slave has hardware that detects the start condition.

**Figure 18-8. Start Condition**




The start condition is output when bit 1 (STT0) of I²C bus control register 0 (IICC0) is set to 1 in the stop condition detection state (SPD0: when bit 0 = 1 in I²C bus status register 0 (IICS0)). In addition, when the start condition is detected, bit 1 (STD0) in IICS0 is set to 1.

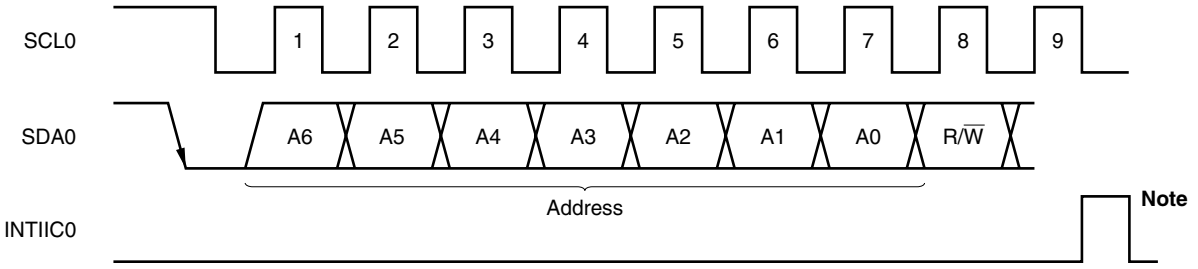
### 18.5.2 Address

The 7-bit data following the start condition defines the address.

The address is 7-bit data that is output so that the master selects a specific slave from the multiple slaves connected to the bus line. Therefore, the slaves on the bus line must have different addresses.

The slave detects this condition by hardware, and determines whether the 7-bit data matches slave address register 0 (SVA0). After the slave was selected when the 7-bit data matched the SVA0 value, communication with the master continues until the master sends a start or stop condition.

**Figure 18-9. Address**




**Note** When the base address or extended code is received during slave operation, INTIIC0 is not generated.

The address is output by writing the slave address and the transfer direction described in **18.5.3 Transfer direction specification** to serial shift register 0 (IIC0) as 8-bit data. In addition, the received address is written to IIC0.

The slave address is allocated to the higher seven bits of IIC0.

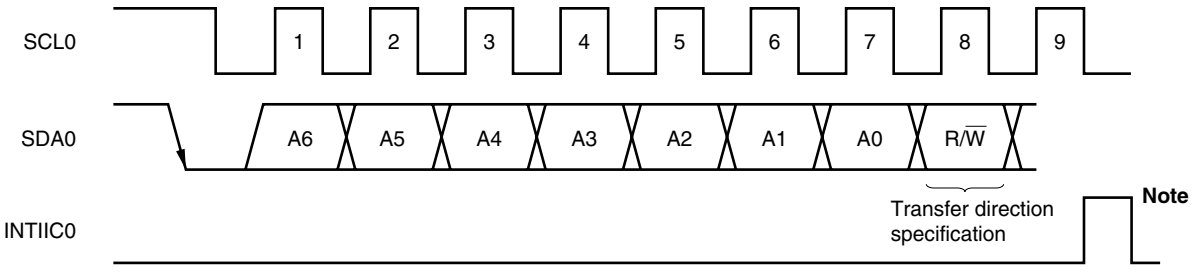
### 18.5.3 Transfer direction specification

Since the master specifies the transfer direction after the 7-bit address, 1-bit data is transmitted.

A transfer direction specification bit of 0 indicates that the master transmits the data to the slave.

A transfer direction specification bit of 1 indicates that the master receives the data from the slave.

**Figure 18-10. Transfer Direction Specification**




**Note** When the base address or extended code is received during slave operation, INTIIC0 is not generated.

### 18.5.4 Acknowledge signal ($\overline{ACK}$)

The acknowledge signal verifies the reception of the serial data on the transmitting and receiving sides.

The receiving side returns the acknowledge signal each time 8-bit data is received. Usually, after transmitting 8-bit data, the transmitting side receives an acknowledge signal. However, if the master receives, the acknowledge signal is not output when the last data is received. After an 8-bit transmission, the transmitting side detects whether the receiving side returned an acknowledge signal. If an acknowledge signal is returned, the next processing is performed assuming that reception was correctly performed. Since reception has not been performed correctly if the acknowledge signal is not returned from the slave, the master outputs the stop condition to stop transmission.

If an acknowledge signal is not returned, the following two causes are considered.

<1> The reception is not correct.
<2> The last data has been received.

When the receiving side sets the SDA0 line low at the ninth clock, the acknowledge signal becomes active (normal reception response).

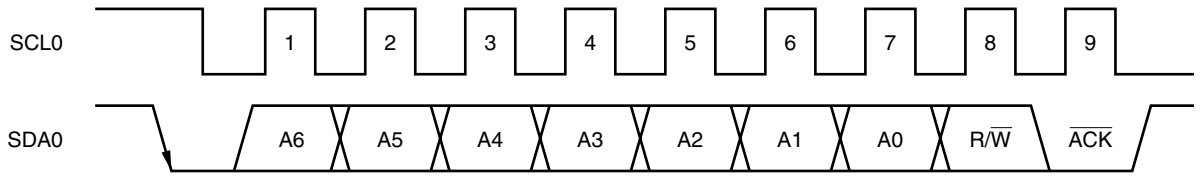
If bit 2 (ACKE0) = 1 in I²C bus control register 0 (IICC0), the acknowledge signal automatic generation enable state is entered.

Bit 3 (TRC0) in I²C bus status register 0 (IICS0) is set by the data in the eighth bit following the 7-bit address information. However, set ACKE0 = 1 in the reception state when TRC0 bit is 0.

When the slave is receiving (TRC0 = 0), the slave side receives multiple bytes and the next data is not required, when ACKE0 = 0, the master side cannot start the next transfer.

Similarly, if the next data is not needed when the master is receiving (TRC0 = 0), set ACKE0 = 0 so that the $\overline{ACK}$ signal is not generated when you want to output a restart or stop condition. This prevents the output of MSB data in the data on the SDA0 line when the slave is transmitting (transmission stopped).

**Figure 18-11. Acknowledge Signal**




When receiving the base address, the automatic output of the acknowledge is synchronized with the falling edge of the eighth clock of SCL0 regardless of the ACKE0 value. When receiving at an address other than the base address, the acknowledge signal is not output.

The output method of the acknowledge signal when receiving data is as follows based on the wait timing.

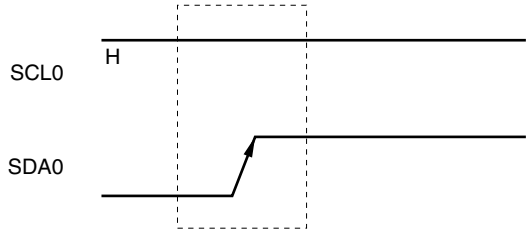
- When 8 clock waits are selected: The acknowledge signal is synchronized with the falling edge of the eighth clock of SCL output by setting ACKE0 = 1 before the wait is released.
- When 9 clock waits are selected: By setting ACKE0 = 1 beforehand, the acknowledge signal is synchronized with the falling edge of the eighth clock of SCL0 and is automatically output.

### 18.5.5 Stop condition

When the SCL0 pin is high and the SDA0 pin changes from low to high, a stop condition occurs.

The stop condition is the signal output by the master to the slave when serial transfer ends. The slave has hardware that detects the stop condition.

**Figure 18-12. Stop Condition**




The stop condition is generated when bit 0 (SPT0) of I²C bus control register 0 (IICC0) is set to 1. And when the stop condition is detected, if bit 0 (SPD0) in I²C bus status register 0 (IICS0) is set to 1 and bit 4 (SPIE0) of IICC0 is also set to 1, INTIIC0 is generated.

### 18.5.6 Wait signal ($\overline{\text{WAIT}}$)

The wait signal notifies the other side that the master or slave is being prepared (wait state) for data communication.

The wait state is reported to the other side by setting the SCL0 pin low. When both the master and the slave are released from the wait state, the next transfer can start.

**Figure 18-13. Wait Signal (1/2)**

**(1) The master has a 9-clock wait, and the slave has an 8-clock wait (Master: Transmitting, Slave: Receiving, ACKE0 = 1)**

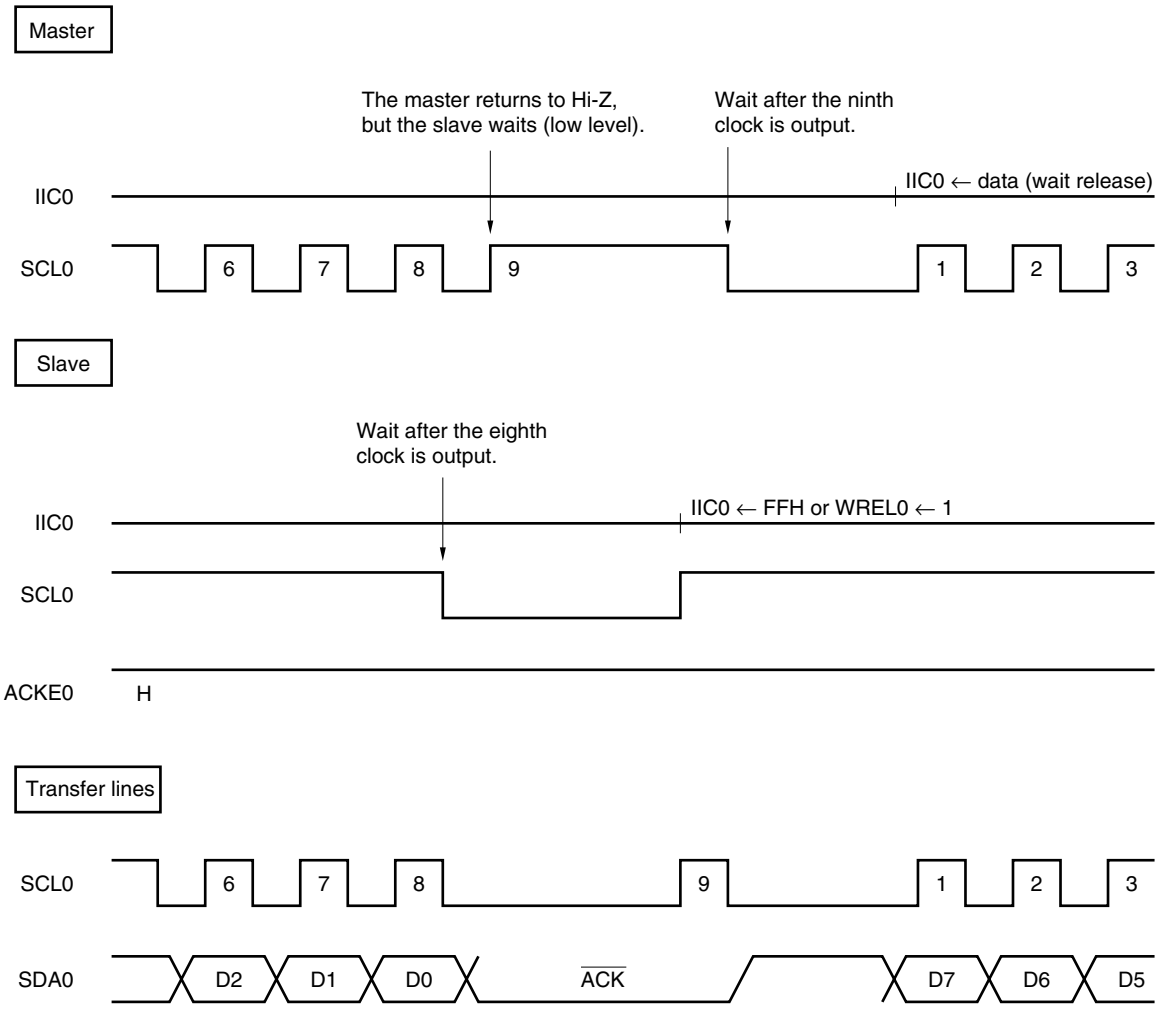
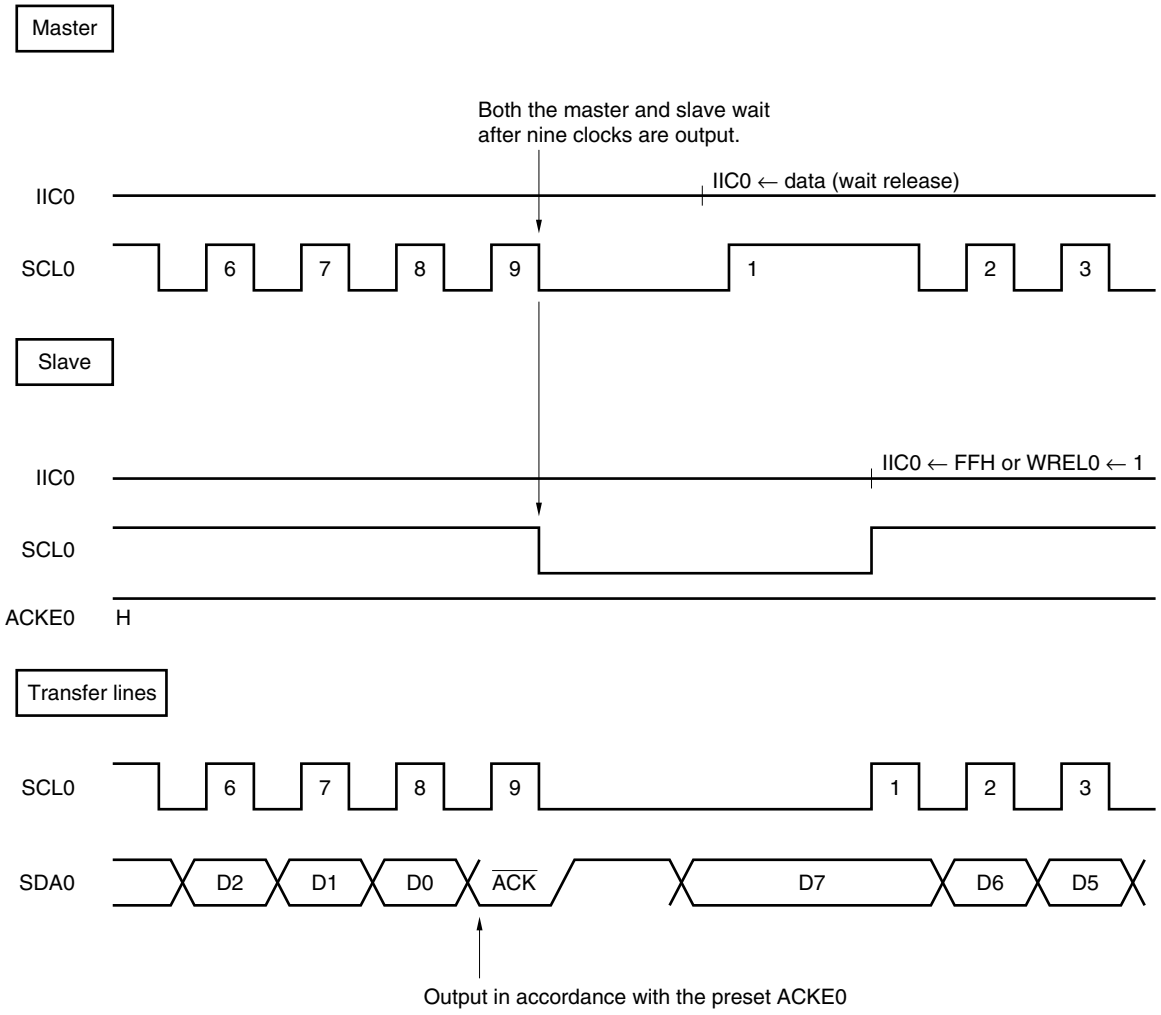


**Figure 18-13. Wait Signal (2/2)**

**(2) Both the master and slave have 9-clock waits**
**(Master: Transmitting, Slave: Receiving, ACKE0 = 1)**




**Remark** ACKE0: Bit 2 in I²C bus control register 0 (IICC0)
WREL0: Bit 5 in I²C bus control register 0 (IICC0)

A wait is automatically generated by setting bit 3 (WTIM0) of I²C bus control register 0 (IICC0).

Normally, when bit 5 (WREL0) = 1 in IICC0 or FFH is written to serial shift register 0 (IIC0), the receiving side releases the wait; when data is written to IIC0, the transmitting side releases the wait.

In the master, the wait can be released by the following methods.

- IICC0 bit 1 (STT0) = 1
- IICC0 bit 0 (SPT0) = 1

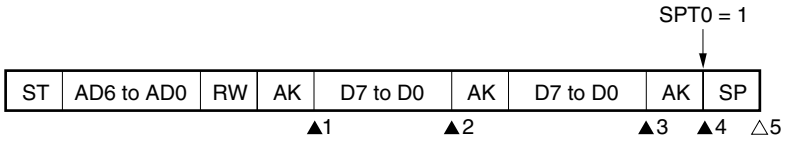
### 18.5.7 I²C interrupt request (INTIIC0)

This section describes the values of I²C bus status register 0 (IICS0) at the INTIIC0 interrupt request generation timing and the INTIIC0 interrupt request timing.

#### (1) Master operation

##### (a) Start - Address - Data - Data - Stop (normal communication)

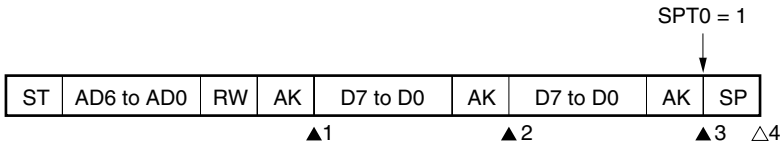
**<1> When WTIM0 = 0**




▲1: IICS0 = 10×××110B
▲2: IICS0 = 10×××000B
▲3: IICS0 = 10×××000B (WTIM0 = 1)
▲4: IICS0 = 10××××00B
△5: IICS0 = 00000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

**<2> When WTIM0 = 1**

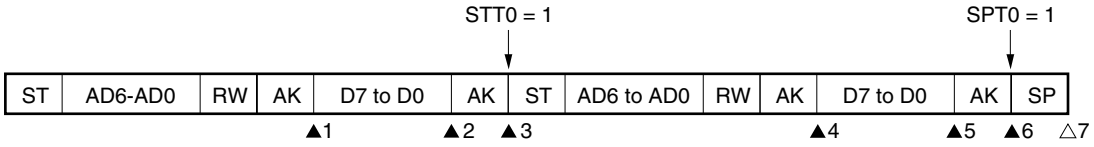



▲1: IICS0 = 10×××110B
▲2: IICS0 = 10×××100B
▲3: IICS0 = 10××××00B
△4: IICS0 = 00000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

### (b) Start - Address - Data - Start - Address - Data - Stop (Restart)

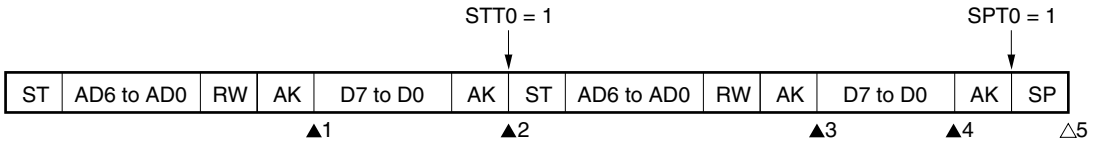
#### <1> When WTIM0 = 0




▲1: IICS0 = 10×××110B
▲2: IICS0 = 10×××000B (WTIM0 = 1)
▲3: IICS0 = 10××××00B (WTIM0 = 0)
▲4: IICS0 = 10×××110B (WTIM0 = 0)
▲5: IICS0 = 10×××000B (WTIM0 = 1)
▲6: IICS0 = 10××××00B
△7: IICS0 = 00000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

#### <2> When WTIM0 = 1

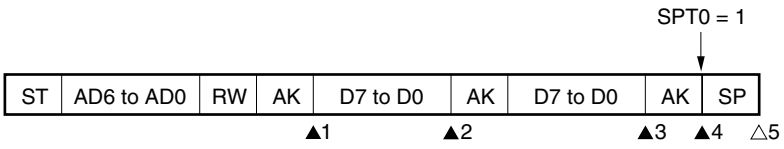



▲1: IICS0 = 10×××110B
▲2: IICS0 = 10××××00B
▲3: IICS0 = 10×××110B
▲4: IICS0 = 10××××00B
△5: IICS0 = 00000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

### (c) Start - Code - Data - Data - Stop (Extended code transmission)

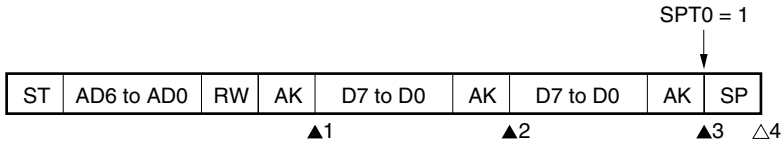
#### <1> When WTIM0 = 0




▲1: IICS0 = 1010×110B
▲2: IICS0 = 1010×000B
▲3: IICS0 = 1010×000B (WTIM0 = 1)
▲4: IICS0 = 1010××00B
△5: IICS0 = 00000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

#### <2> When WTIM0 = 1



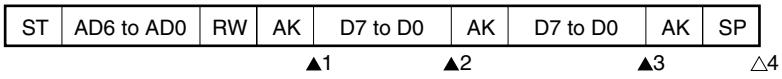

▲1: IICS0 = 1010×110B
▲2: IICS0 = 1010×100B
▲3: IICS0 = 1010××00B
△4: IICS0 = 00000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

### (2) Slave operation (when receiving slave address data (SVA0 match))

#### (a) Start - Address - Data - Data - Stop

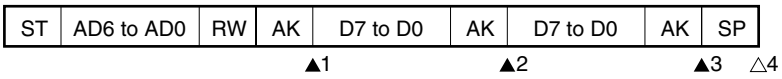
##### <1> When WTIM0 = 0




▲1: IICS0 = 0001×110B
▲2: IICS0 = 0001×000B
▲3: IICS0 = 0001×000B
△4: IICS0 = 00000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

##### <2> When WTIM0 = 1

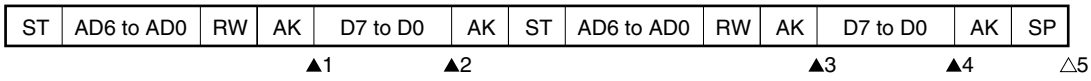



▲1: IICS0 = 0001×110B
▲2: IICS0 = 0001×100B
▲3: IICS0 = 0001××00B
△4: IICS0 = 00000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

### (b) Start - Address - Data - Start - Address - Data - Stop

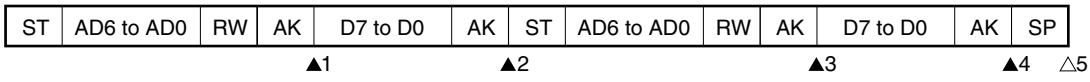
#### <1> When WTIM0 = 0 (SVA0 match after restart)




▲1: IICS0 = 0001×110B
▲2: IICS0 = 0001×000B
▲3: IICS0 = 0001×110B
▲4: IICS0 = 0001×000B
△5: IICS0 = 00000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

#### <2> When WTIM0 = 1 (SVA0 match after restart)

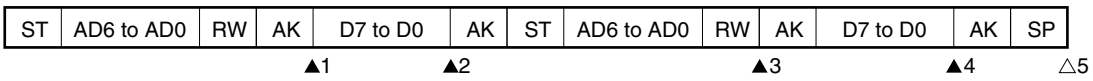



▲1: IICS0 = 0001×110B
▲2: IICS0 = 0001××00B
▲3: IICS0 = 0001×110B
▲4: IICS0 = 0001××00B
△5: IICS0 = 00000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

### (c) Start - Address - Data - Start - Code - Data - Stop

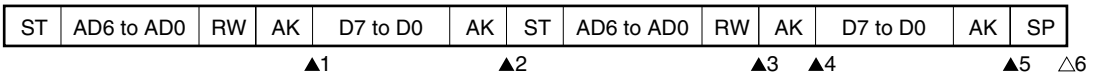
#### <1> When WTIM0 = 0 (extended code received after restart)




▲1: IICS0 = 0001×110B
▲2: IICS0 = 0001×000B
▲3: IICS0 = 0010×010B
▲4: IICS0 = 0010×000B
△5: IICS0 = 00000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

#### <2> When WTIM0 = 1 (extended code received after restart)

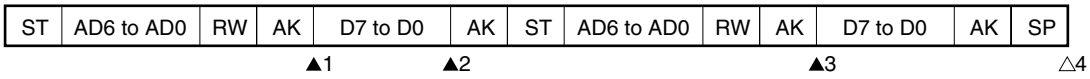



▲1: IICS0 = 0001×110B
▲2: IICS0 = 0001××00B
▲3: IICS0 = 0010×010B
▲4: IICS0 = 0010×110B
▲5: IICS0 = 0010××00B
△6: IICS0 = 00000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

### (d) Start - Address - Data - Start - Address - Data - Stop

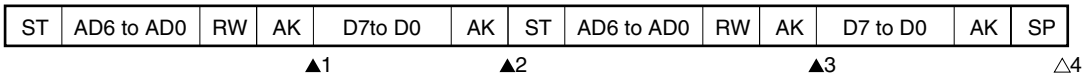
#### <1> When WTIM0 = 0 (no address match after restart (not extended code))




▲1: IICS0 = 0001×110B
▲2: IICS0 = 0001×000B
▲3: IICS0 = 00000×10B
△4: IICS0 = 00000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

#### <2> When WTIM0 = 1 (no address match after restart (not extended code))



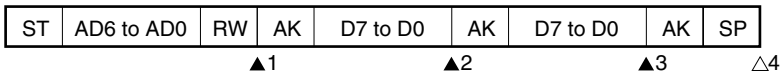

▲1: IICS0 = 0001×110B
▲2: IICS0 = 0001××00B
▲3: IICS0 = 00000×10B
△4: IICS0 = 00000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

### (3) Slave operation (when receiving the extended code)

#### (a) Start - Code - Data - Data - Stop

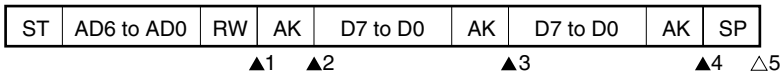
##### <1> When WTIM0 = 0




▲1: IICS0 = 0010×010B
▲2: IICS0 = 0010×000B
▲3: IICS0 = 0010×000B
△4: IICS0 = 00000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

##### <2> When WTIM0 = 1

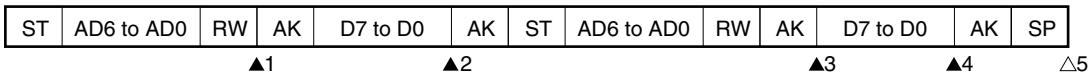



▲1: IICS0 = 0010×010B
▲2: IICS0 = 0010×110B
▲3: IICS0 = 0010×100B
▲4: IICS0 = 0010××00B
△5: IICS0 = 00000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

### (b) Start - Code - Data - Start - Address - Data - Stop

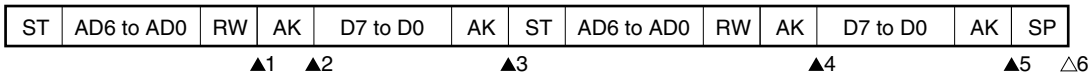
#### <1> When WTIM0 = 0 (SVA0 match after restart)




▲1: IICS0 = 0010×010B
▲2: IICS0 = 0010×000B
▲3: IICS0 = 0001×110B
▲4: IICS0 = 0001×000B
△5: IICS0 = 00000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

#### <2> When WTIM0 = 1 (SVA0 match after restart)

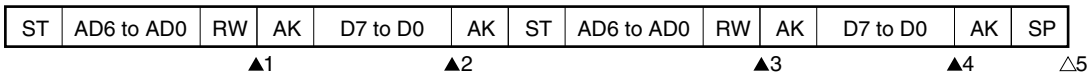



▲1: IICS0 = 0010×010B
▲2: IICS0 = 0010×110B
▲3: IICS0 = 0010××00B
▲4: IICS0 = 0001×110B
▲5: IICS0 = 0001××00B
△6: IICS0 = 00000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

### (c) Start - Code - Data - Start - Code - Data - Stop

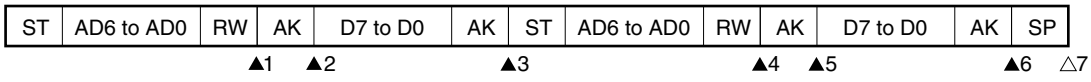
#### <1> When WTIM0 = 0 (extended code received after restart)




▲1: IICS0 = 0010×010B
▲2: IICS0 = 0010×000B
▲3: IICS0 = 0010×010B
▲4: IICS0 = 0010×000B
△5: IICS0 = 00000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

#### <2> When WTIM0 = 1 (extended code received after restart)

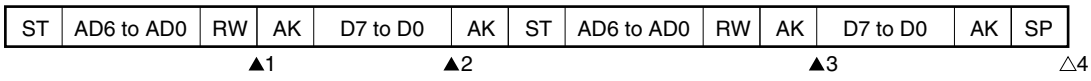



▲1: IICS0 = 0010×010B
▲2: IICS0 = 0010×110B
▲3: IICS0 = 0010××00B
▲4: IICS0 = 0010×010B
▲5: IICS0 = 0010×110B
▲6: IICS0 = 0010××00B
△7: IICS0 = 00000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

**(d) Start - Code - Data - Start - Address - Data - Stop**

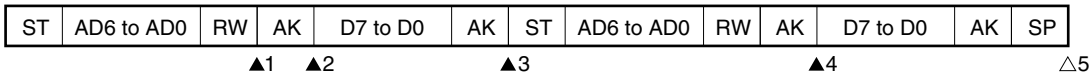
**<1> When WTIM0 = 0 (no address match after restart (not an extended code))**




▲1: IICS0 = 0010×010B
▲2: IICS0 = 0010×000B
▲3: IICS0 = 00000×10B
△4: IICS0 = 00000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

**<2> When WTIM0 = 1 (no address match after restart (not an extended code))**

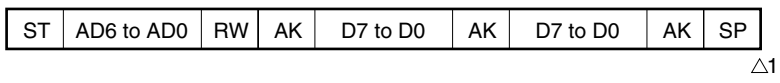



▲1: IICS0 = 0010×010B
▲2: IICS0 = 0010×110B
▲3: IICS0 = 0010××00B
▲4: IICS0 = 00000×10B
△5: IICS0 = 00000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

**(4) Not participating in communication**

**(a) Start - Code - Data - Data - Stop**



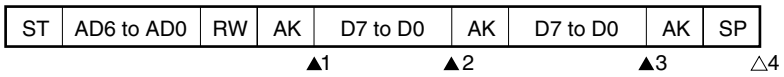

△1: IICS0 = 00000001B

**Remarks** △: Generated only when SPIE0 = 1

### (5) Arbitration failed operation (operates as the slave after arbitration fails)

#### (a) When arbitration failed during the transfer of slave address data

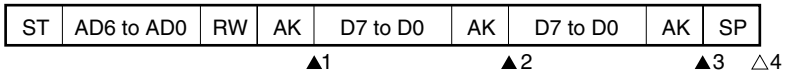
##### <1> When WTIM0 = 0




▲1: IICS0 = 0101×110B (Example: Read ALD0 during interrupt servicing.)
▲2: IICS0 = 0001×000B
▲3: IICS0 = 0001×000B
△4: IICS0 = 00000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

##### <2> When WTIM0 = 1

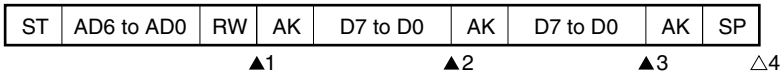



▲1: IICS0 = 0101×110B (Example: Read ALD0 during interrupt servicing.)
▲2: IICS0 = 0001×100B
▲3: IICS0 = 0001××00B
△4: IICS0 = 00000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

**(b) When arbitration failed while transmitting an extended code**

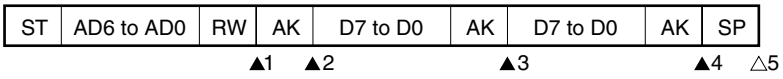
**<1> When WTIM0 = 0**




▲1: IICS0 = 0110×010B (Example: Read ALD0 during interrupt servicing.)
▲2: IICS0 = 0010×000B
▲3: IICS0 = 0010×000B
△4: IICS0 = 00000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

**<2> When WTIM0 = 1**

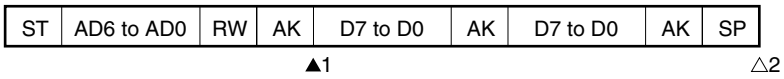



▲1: IICS0 = 0110×010B (Example: Read ALD0 during interrupt servicing.)
▲2: IICS0 = 0010×110B
▲3: IICS0 = 0010×100B
▲4: IICS0 = 0010××00B
△5: IICS0 = 00000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

### (6) Arbitration failed operation (no participation after arbitration failed)

#### (a) When arbitration failed while transmitting slave address data



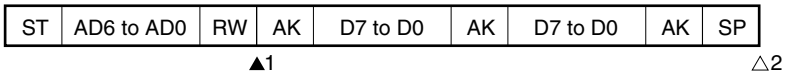

▲1: IICS0 = 01000110B (Example: Read ALD0 during interrupt servicing.)

△2: IICS0 = 00000001B

**Remarks** ▲: Always generated.

△: Generated only when SPIE0 = 1

#### (b) When arbitration failed while transmitting an extended code




▲1: IICS0 = 0110×010B (Example: Read ALD0 during interrupt servicing.)

Set LREL0 = 1 from the software.

△2: IICS0 = 00000001B

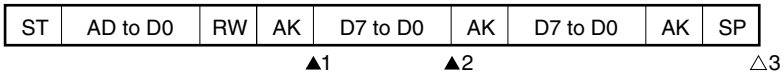
**Remarks** ▲: Always generated.

△: Generated only when SPIE0 = 1

×: Don't care

### (c) When arbitration failed during a data transfer

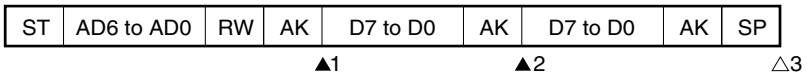
#### <1> When WTIM0 = 0




▲1: IICS0 = 10001110B
▲2: IICS0 = 01000000B (Example: Read ALD0 during interrupt servicing.)
△3: IICS0 = 00000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1

#### <2> When WTIM0 = 1

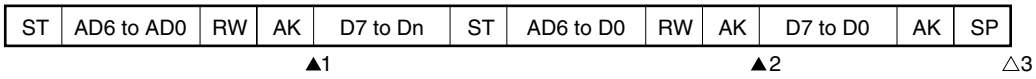



▲1: IICS0 = 10001110B
▲2: IICS0 = 01000100B (Example: Read ALD0 during interrupt servicing.)
△3: IICS0 = 00000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1

**(d) When failed in the restart condition during a data transfer**

**<1> Not an extended code (Example: SVA0 does not match)**




▲1: IICS0 = 1000×110B

▲2: IICS0 = 01000110B (Example: Read ALD0 during interrupt servicing.)

△3: IICS0 = 00000001B

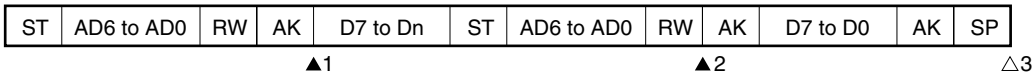
**Remarks** ▲: Always generated.

△: Generated only when SPIE0 = 1

×: Don't care

Dn = D6 to D0

**<2> Extended code**




▲1: IICS0 = 1000×110B

▲2: IICS0 = 0110×010B (Example: Read ALD0 during interrupt servicing.)

IICC0:LREL0 = 1 set by software.

△3: IICS0 = 00000001B

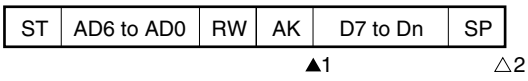
**Remarks** ▲: Always generated.

△: Generated only when SPIE0 = 1

×: Don't care

Dn = D6 to D0

**(e) When failed in the stop condition during a data transfer**



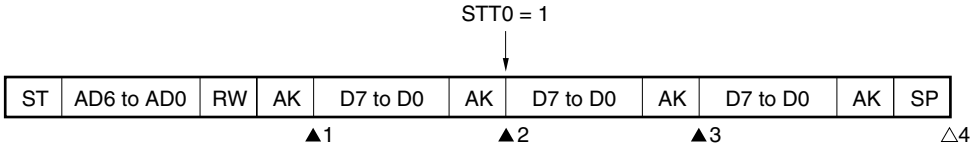

▲1: IICS0 = 1000×110B

△2: IICS0 = 01000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care
Dn = D6 to D0

**(f) When arbitration failed at a low data level and the restart condition was about to be generated**

**WTIM0 = 1**




▲1: IICS0 = 1000×110B

▲2: IICS0 = 1000××00B

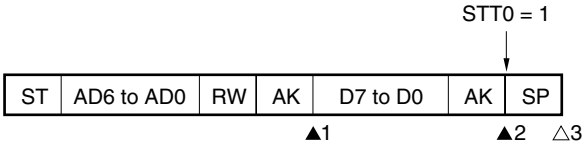
▲3: IICS0 = 01000100B (Example: Read ALD0 during interrupt servicing.)

△4: IICS0 = 00000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

**(g) When arbitration failed in a stop condition and the restart condition was about to be generated**

**WTIM0 = 1**




▲1: IICS0 = 1000×110B

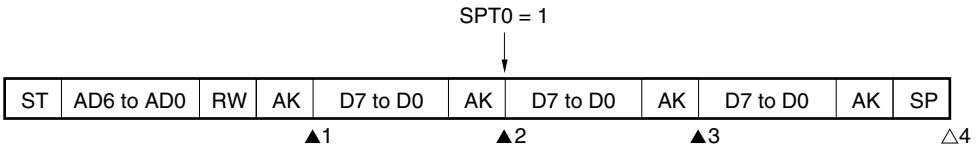
▲2: IICS0 = 1000××00B

△3: IICS0 = 01000001B

**Remarks** ▲: Always generated.
△: Generated only when SPIE0 = 1
×: Don't care

**(h) When arbitration failed in the low data level and the stop condition was about to be generated**

**WTIM0 = 1**




▲1: IICS0 = 1000×110B

▲2: IICS0 = 1000××00B

▲3: IICS0 = 01000000B (Example: Read ALD0 during interrupt servicing.)

△4: IICS0 = 00000001B

**Remarks** ▲: Always generated.

△: Generated only when SPIE0 = 1

×: Don't care

### 18.5.8 Interrupt request (INTIIC0) generation timing and wait control

By setting the WTIM0 bit in I²C bus control register 0 (IICC0), INTIIC0 is generated at the timing shown in Table 18-2 and wait control is performed.

**Table 18-2. INTIIC0 Generation Timing and Wait Control**

| WTIM0 | During Slave Operation | | | During Master Operation | | |
|---|---|---|---|---|---|---|
| | Address | Data Reception | Data Transmission | Address | Data Reception | Data Transmission |
| 0 | 9 Notes 1, 2 | 8 Note 2 | 8 Note 2 | 9 | 8 | 8 |
| 1 | 9 Notes 1, 2 | 9 Note 2 | 9 Note 2 | 9 | 9 | 9 |

**Notes 1.** The INTIIC0 signal and wait of the slave are generated on the falling edge of the ninth clock only when the address set in slave address register 0 (SVA0) matches.
In this case, $\overline{ACK}$ is output regardless of the ACKE0 setting. The slave that received the extended code generates INTIIC0 at the falling edge of the eighth clock.

**2.** When the address that received SVA0 does not match, INTIIC0 and wait are not generated.

**Remark** The numbers in the table indicate the number of clocks of the serial clock. In addition, the interrupt request and wait control are both synchronized with the falling edge of the serial clock.

#### (1) When transmitting and receiving an address

- When the slave is operating: The interrupt and wait timing are determined regardless of the WTIM0 bit.
- When the master is operating: The interrupt and wait timing are generated by the falling edge of the ninth clock regardless of the WTIM0 bit.

#### (2) When receiving data

- When the master and slave are operating: The interrupt and wait timing are set by the WTIM0 bit.

#### (3) When transmitting data

- When the master and slave are operating: The interrupt and wait timing are set by the WTIM0 bit.

#### (4) Releasing a wait

The following four methods release a wait.

- WREL0 = 1 in I²C bus control register 0 (IICC0)
- Writing to serial shift register 0 (IIC0)
- Setting the start condition (STT0 = 1 in IICC0)
- Setting the stop condition (SPT0 = 1 in IICC0)

When an eight-clock wait is selected (WTIM0 = 0), the output level of $\overline{ACK}$ must be determined before releasing the wait.

#### (5) Stop condition detection

INTIIC0 is generated when the stop condition is detected.

### 18.5.9 Address match detection

In the I²C bus mode, the master can select a specific slave device by transmitting the slave address.

Address matching can be detected automatically by the hardware. When the base address is set in slave address register 0 (SVA0), if the slave address transmitted from the master matches the address set in SVA0, or if the extended code is received, an INTIIC0 interrupt request occurs.

### 18.5.10 Error detection

In the I²C bus mode, since the state of the serial bus (SDA0) during transmission is input to serial shift register 0 (IIC0) of the transmitting device, transmission errors can be detected by comparing the IIC0 data before and after the transmission. In this case, if the two data differ, it can be judged that a transmission error occurred.

### 18.5.11 Extended codes

(1) If the most significant four bits of the receiving address are “0000” or “1111”, an extended code is received and the extended code received flag (EXC0) is set. The interrupt request (INTIIC0) is generated at the falling edge of the eighth clock. The local address stored in slave address register 0 (SVA0) is not affected.

(2) In 10-bit address transfers, the following occurs when “111110XX” is set in SVA0 and “111110XX0” is transferred from the master. However, INTIIC0 is generated at the falling edge of the eighth clock.

- Higher 4 bits of data match: EXC0 = 1**Note**
- 7-bit data match: COI0 = 1**Note**

**Note** EXC0: Bit 5 of I²C bus status register 0 (IICS0)
COI0: Bit 4 of I²C bus status register 0 (IICS0)

(3) Since the processing after an interrupt request is generated differs depending on the data that follows the extended code, this processing is performed by software.
For example, when operation as a slave is not desired after an extended code is received, enter the next communication wait state by setting bit 6 (LREL0) = 1 of I²C bus control register 0 (IICC0).

**Table 18-3. Definitions of Extended Code Bits**

| Slave Address | R/W Bit | Description |
|---|---|---|
| 0000 000 | 0 | General call address |
| 0000 000 | 1 | Start byte |
| 0000 001 | × | CBUS address |
| 0000 010 | × | Address reserved in the different bus format |
| 1111 0×× | × | 10-bit slave address setting |

### 18.5.12 Arbitration

When multiple masters simultaneously output start conditions (when STT0 = 1 occurs before STD0 = 1**Note**), the master communicates while the clock is adjusted until the data differ. This operation is called arbitration.

A master that failed arbitration sets the arbitration failed flag (ALD0) of I²C bus status register 0 (IICS0) at the timing of the failed arbitration. The SCL0 and SDA0 lines enter the high impedance state, and the bus is released.

Failed arbitration is detected when ALD0 = 1 by software at the timing of the interrupt request generated next (eighth or ninth clock, stop condition detection, etc.).

At the timing for generating the interrupt request, refer to **18.5.7 I²C interrupt request (INTIIC0)**.

**Note** STD0: Bit 1 in I²C bus status register 0 (IICS0)
STT0: Bit 1 in I²C bus control register 0 (IICC0)

**Figure 18-14. Example of Arbitration Timing**

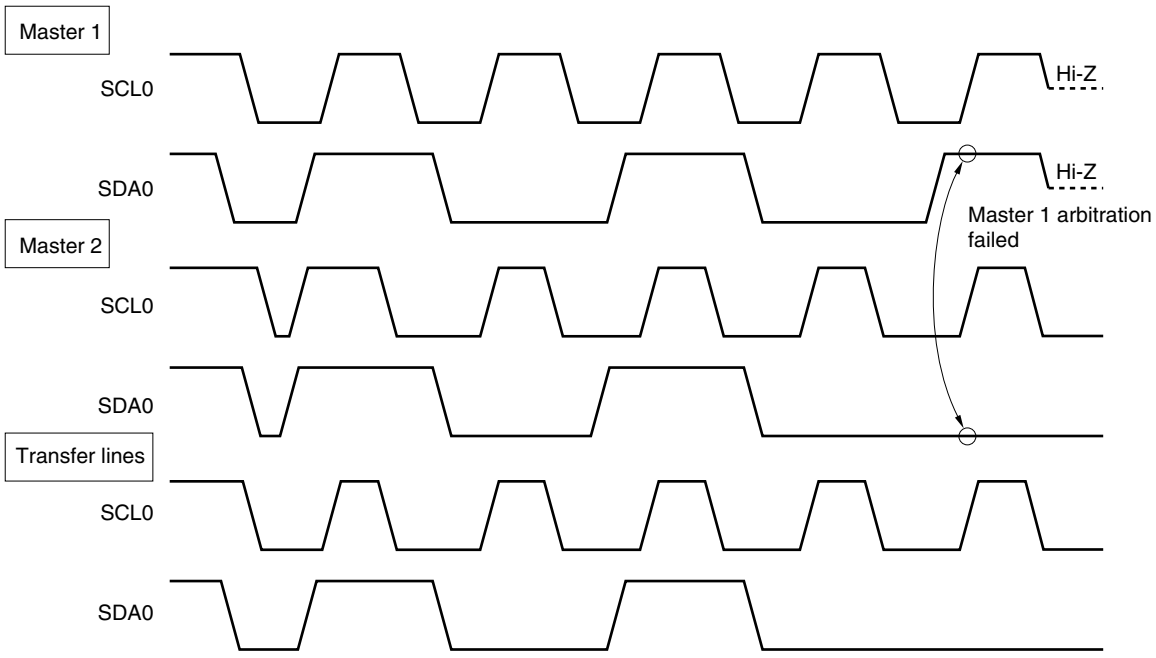



**Table 18-4. Arbitration Generation States and Interrupt Request Generation Timing**

| Arbitration Generation State | Interrupt Request Generation Timing |
|---|---|
| During address transmission | Falling edge of clock 8 or 9 after byte transfer[Note 1] |
| Read/write information after address transmission | |
| During extended code transmission | |
| Read/write information after extended code transmission | |
| During data transmission | |
| During $\overline{ACK}$ transfer period after data transmission | |
| Restart condition detection during data transfer | |
| Stop condition detection during data transfer | When stop condition is output (SPIE0 = 1)[Note 2] |
| Data is low when the restart condition is about to be output. | Falling edge of clock 8 or 9 after byte transfer[Note 1] |
| The restart condition should be output, but the stop condition is detected. | Stop condition is output (SPIE0 = 1)[Note 2] |
| Data is low when the stop condition is about to be output. | Falling edge of clock 8 or 9 after byte transfer[Note 1] |
| SCL0 is low when the restart condition is about to be output. | |

**Notes 1.** If WTIM0 = 1 (bit 3 = 1 in I²C bus control register 0 (IICC0)), an interrupt request is generated at the timing of the falling edge of the ninth clock. If WTIM0 = 0 and the slave address of the extended code is received, an interrupt request is generated at the timing of the falling edge of the eighth clock.

**2.** When arbitration is possible, use the master to set SPIE0 = 1.

**Remark** SPIE0: Bit 5 in I²C bus control register 0 (IICC0)

### 18.5.13 Wake-up function

This is a slave function of the I²C bus and generates an interrupt request (INTIIC0) when the base address and extended code are received.

When the address does not match, an unused interrupt request is not generated, and efficient processing is possible.

When the start condition is detected, the wake-up standby function is entered. Since the master can become a slave in an arbitration failure (when a start condition was output), the wake-up standby function is entered while the address is transmitted.

However, when the stop condition is detected, the generation of interrupt requests is enabled or disabled based on the setting of bit 5 (SPIE0) in I²C bus control register 0 (IICC0) unrelated to the wake-up function.

### 18.5.14 Communication reservation

When you want the master to communicate after being in a non-participation state on the bus, the start condition can be transmitted when the bus is released by reserving communication. The following two states are included when the bus does not participate.

- When there was no arbitration in the master and the slave
- When the extended code is received and operation is not as a slave (bus released when $\overline{ACK}$ is not returned, and bit 6 (LREL0) = 1 in the I²C bus control register (IICC0))

When bit 1 (STT0) of IICC0 is set in the not participating state in the bus, after the bus is released (after stop condition detection), the start condition is automatically generated, and the wait state is entered. When the bus release is detected (stop condition detection), the address transfer starts as the master by the write operation of serial shift register 0 (IIC0). In this case, set bit 4 (SPIE0) in IICC0.

When STT0 is set, whether it operates as a start condition or for communication reservation is determined by the bus state.

- When the bus is released ····· Start condition generation
- When the bus is not released (standby state) ····· Communication reservation

The method that detects the operation of STT0 sets STT0 and verifies the STT0 bit again after the wait time elapses.

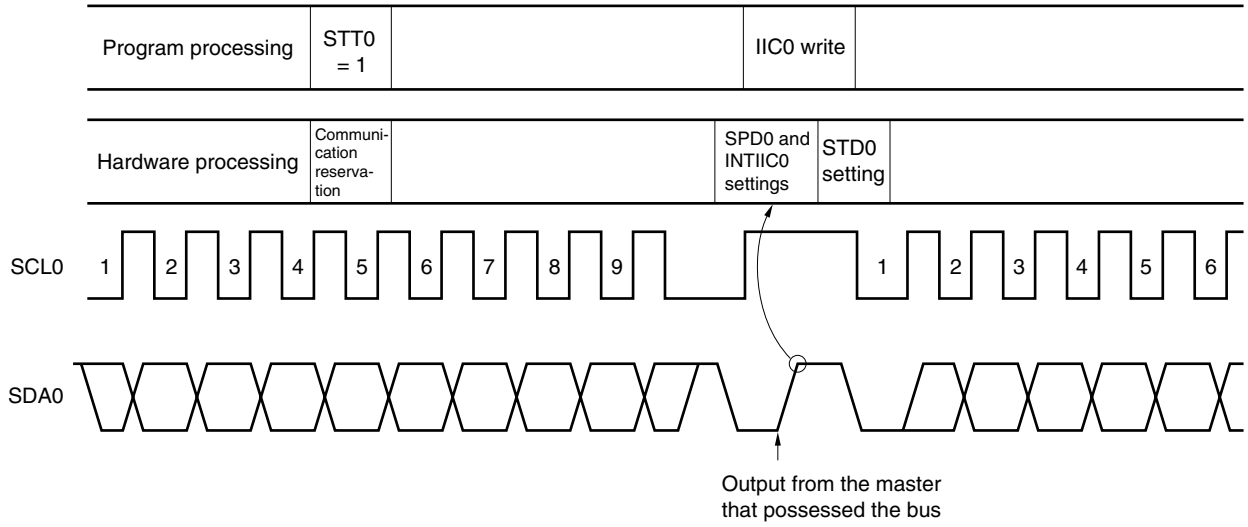
Use the software to save the wait time, which is listed in Table 18-5. The wait time can be set by bits 3, 1, and 0 (SMC, CL1, CL0) of prescaler mode register 0 for serial clock (SPRM0).

**Table 18-5. Wait Times**

| SMC | CL1 | CL0 | Wait Time |
|---|---|---|---|
| 0 | 0 | 0 | 26 clocks × 1/fxx |
| 0 | 0 | 1 | 46 clocks × 1/fxx |
| 0 | 1 | 0 | 92 clocks × 1/fxx |
| 0 | 1 | 1 | 37 clocks × 1/TM2 output |
| 1 | 0 | 0 | 16 clocks × 1/fxx |
| 1 | 0 | 1 | |
| 1 | 1 | 0 | 32 clocks × 1/fxx |
| 1 | 1 | 1 | 13 clocks × 1/TM2 output |

Figure 18-15 shows the timing of communication reservation.

**Figure 18-15. Timing of Communication Reservation**




IIC0: Serial shift register
STT0: Bit 1 in I²C bus control register 0 (IICC0)
STD0: Bit 1 in I²C bus status register 0 (IICS0)
SPD0: Bit 0 in I²C bus status register 0 (IICS0)

The communication reservation is accepted at the following timing. After bit 1 (STD0) = 1 in I²C bus status register 0 (IICS0), the communication is reserved by bit 1 (STT0) = 1 in I²C bus control register 0 (IICC0) until the stop condition is detected.

**Figure 18-16. Communication Reservation Acceptance Timing**

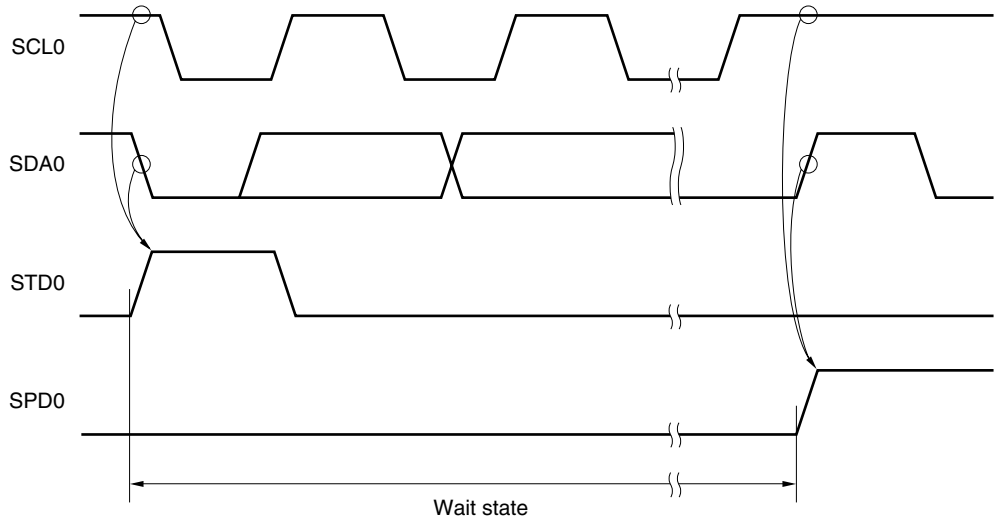
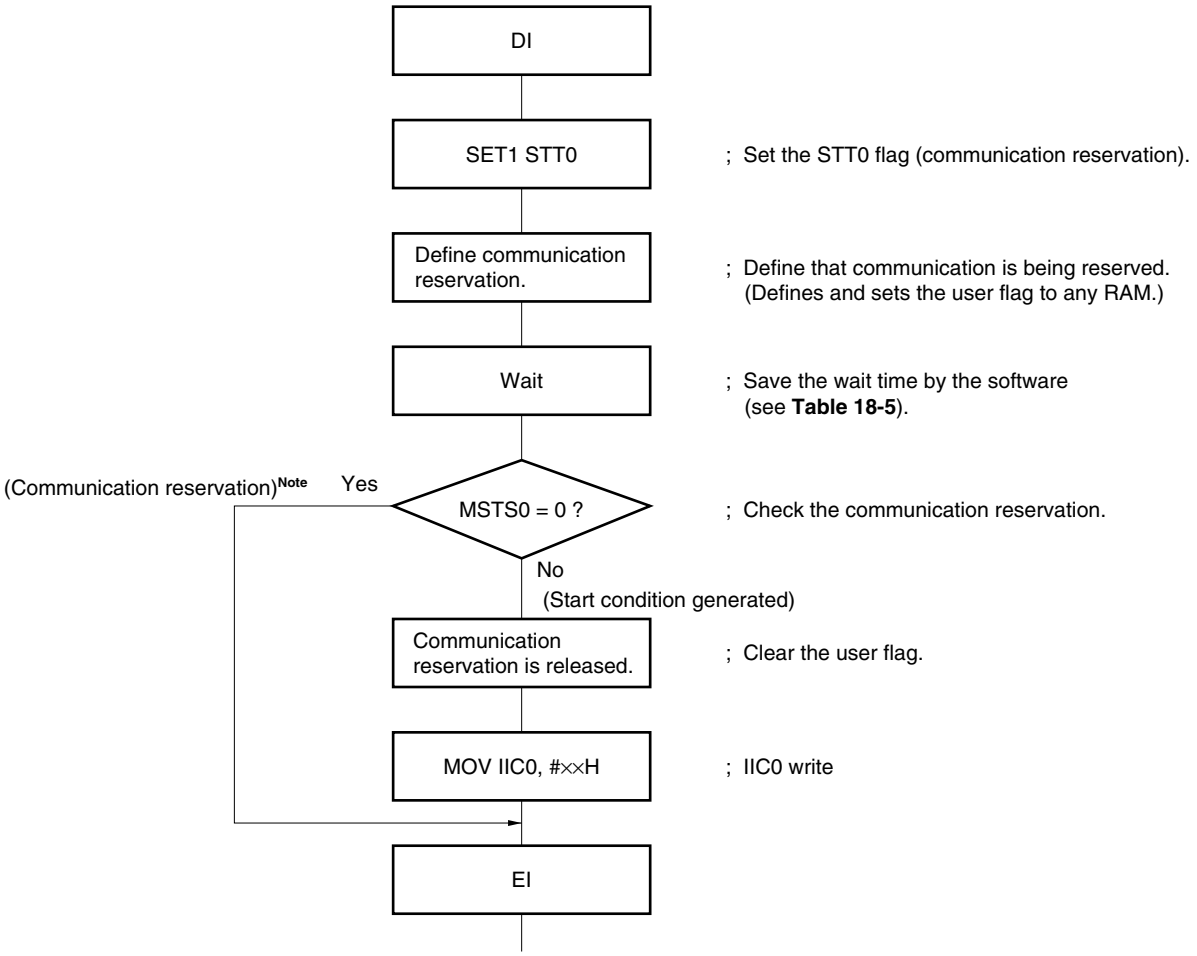


Figure 18-17 shows the communication reservation procedure.

**Figure 18-17. Communication Reservation Procedure**




**Note** When the communication is being reserved, serial shift register 0 (IIC0) is written by the stop condition interrupt.

**Remark** STT0: Bit 1 in I²C bus control register 0 (IICC0)
MSTS0: Bit 7 in I²C bus status register 0 (IICS0)
IIC0: Serial shift register 0

### 18.5.15 Additional cautions

After a reset, when the master is communicating from the state where the stop condition is not detected (bus is not released), perform master communication after the stop condition is first generated and the bus is released.

Multiple masters cannot communicate in the state where the bus is not released (the stop condition is not detected).

The following procedure generates the stop condition.

<1> Set prescaler mode register 0 (SPRM0) for the serial clock.

<2> Set bit 7 (IICE0) in I²C bus control register 0 (IICC0).

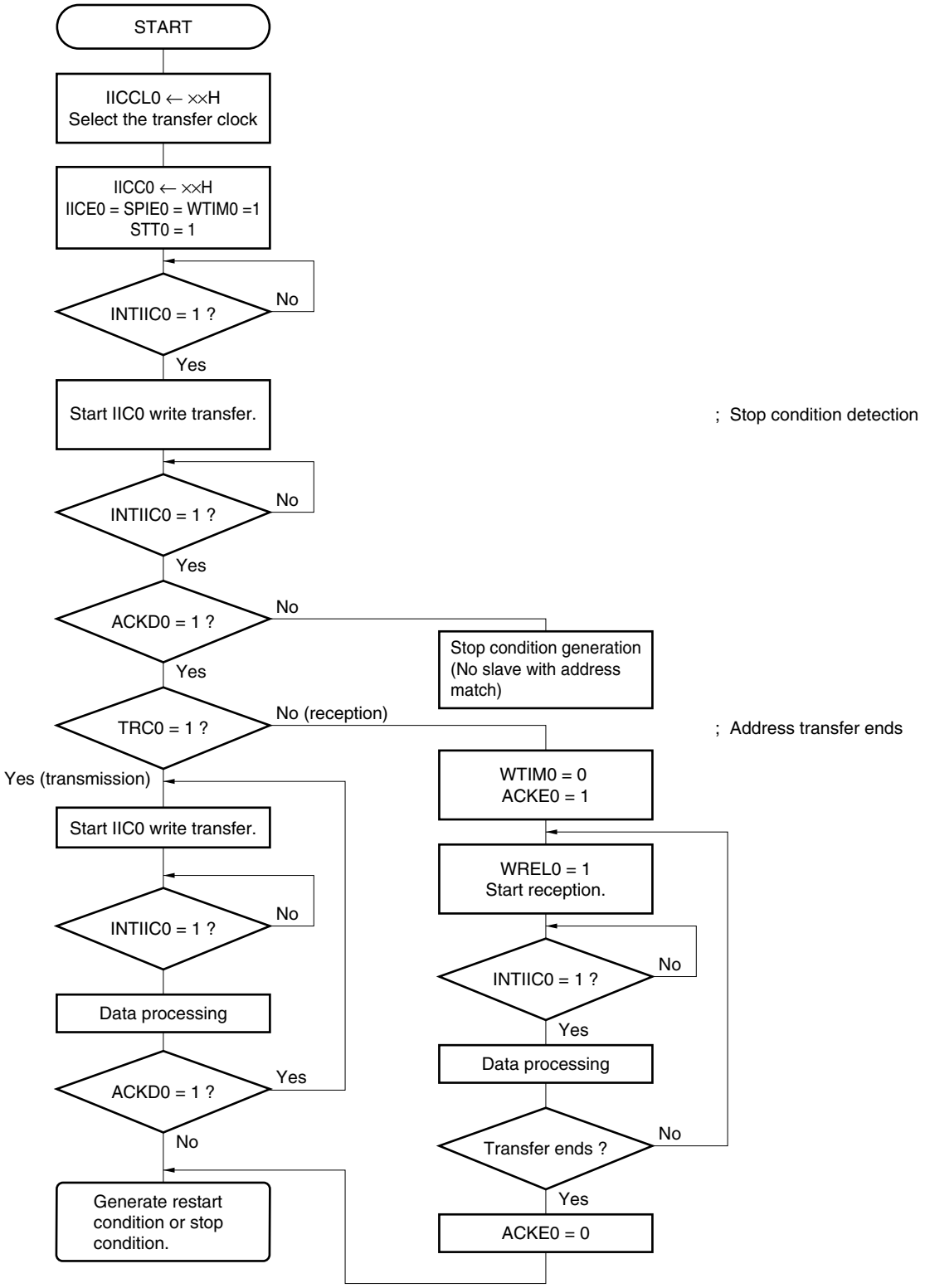
<3> Set bit 0 of IICC0.

### 18.5.16 Communication operation

#### (1) Master operation

The following example shows the master operation procedure.

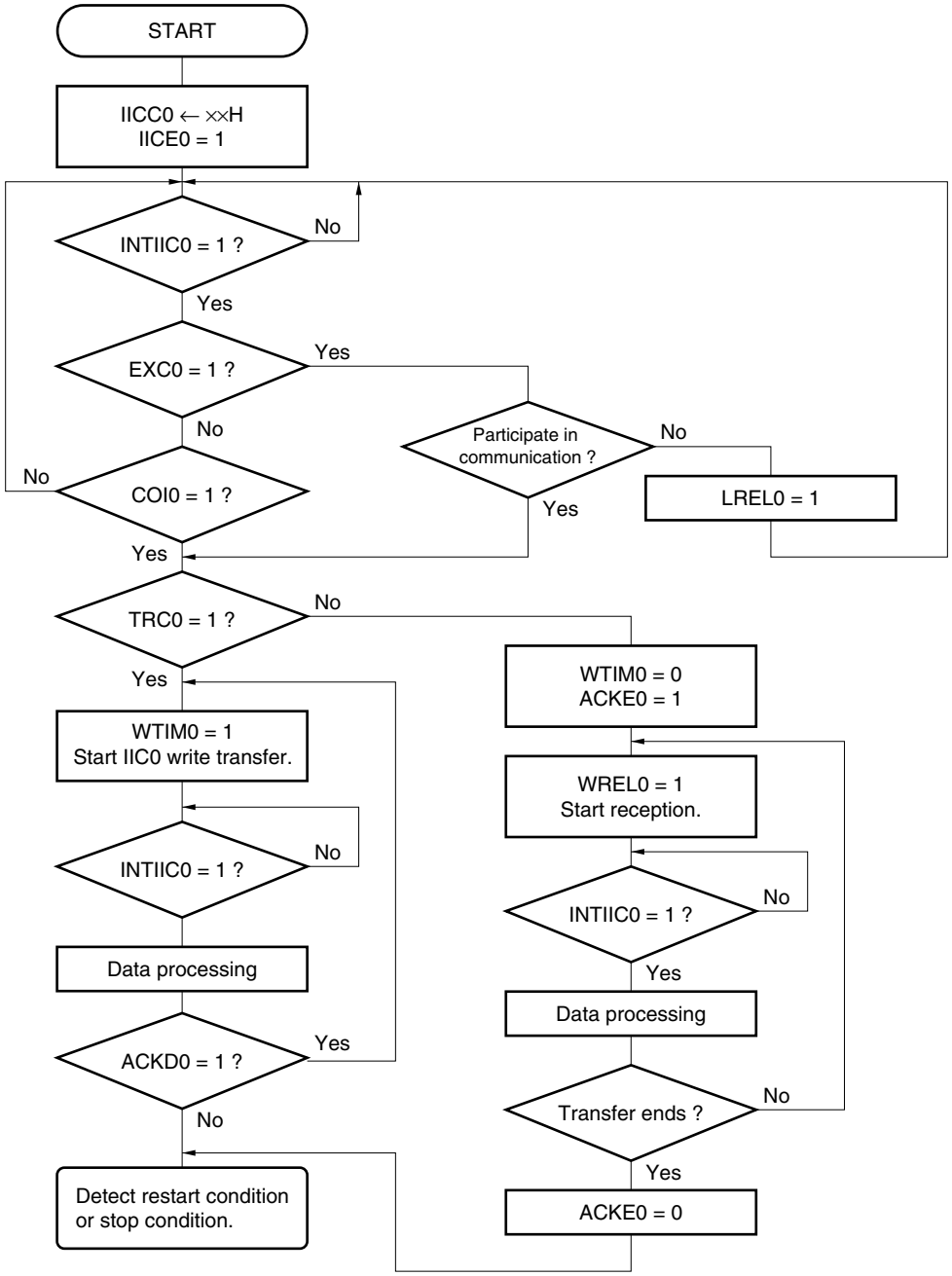
**Figure 18-18. Master Operation Procedure**



### (2) Slave operation

The following example is the slave operating procedure.

**Figure 18-19. Slave Operating Procedure**



## 18.6 Timing Charts

In the I²C bus mode, the master outputs an address on the serial bus and selects one of the slave devices from multiple slave devices as the communication target.

The master transmits the TRC0 bit, bit 3 of I²C bus status register 0 (IICS0), that indicates the transfer direction of the data after the slave address and starts serial communication with the slave.

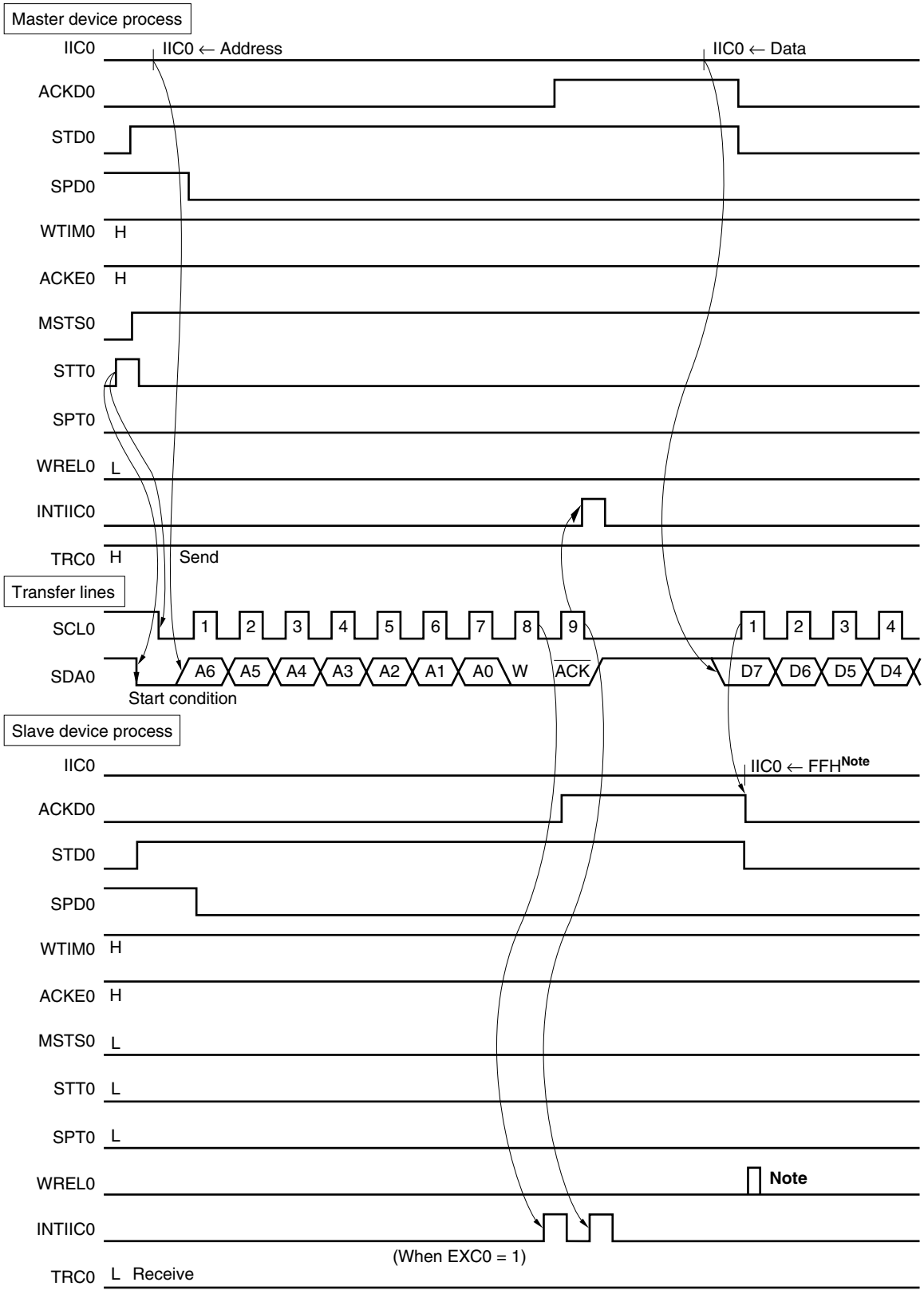
Figures 18-20 and 18-21 are the timing charts for data communication.

Shifting of serial shift register 0 (IIC0) is synchronized with the falling edge of the serial clock (SCL0). The transmission data is transferred to the SO0 latch and output from the SDA0 pin with the MSB first.

The data input at the SDA0 pin is received by IIC0 at the rising edge of SCL0.

**Figure 18-20. Master → Slave Communication Example (When Master and Slave Select 9-Clock Wait) (1/3)**

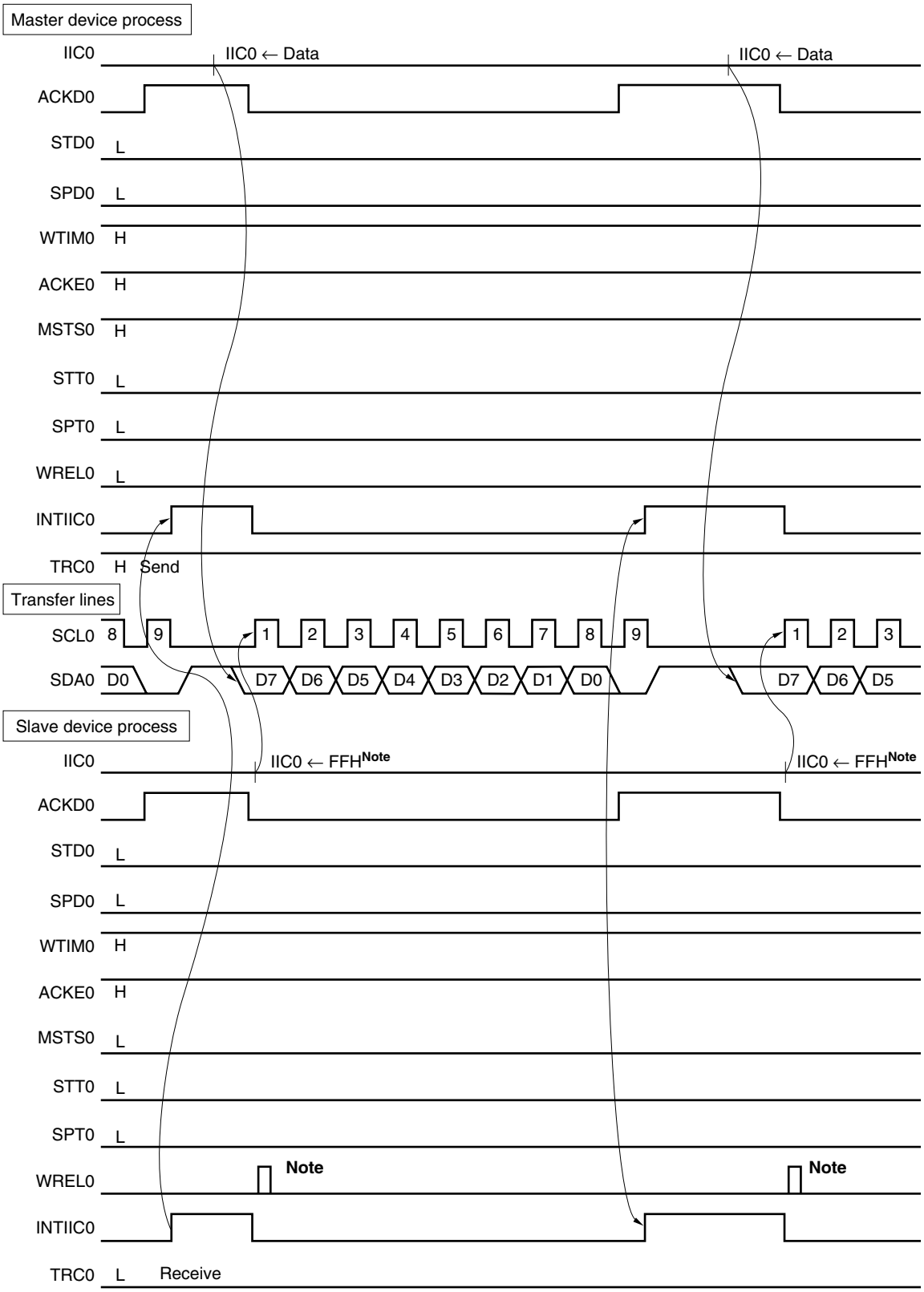
**(1) Start condition - Address**




**Note** Release the slave wait by either IIC0 ← FFH or setting WREL0.

**Figure 18-20. Master → Slave Communication Example (When Master and Slave Select 9-Clock Wait) (2/3)**

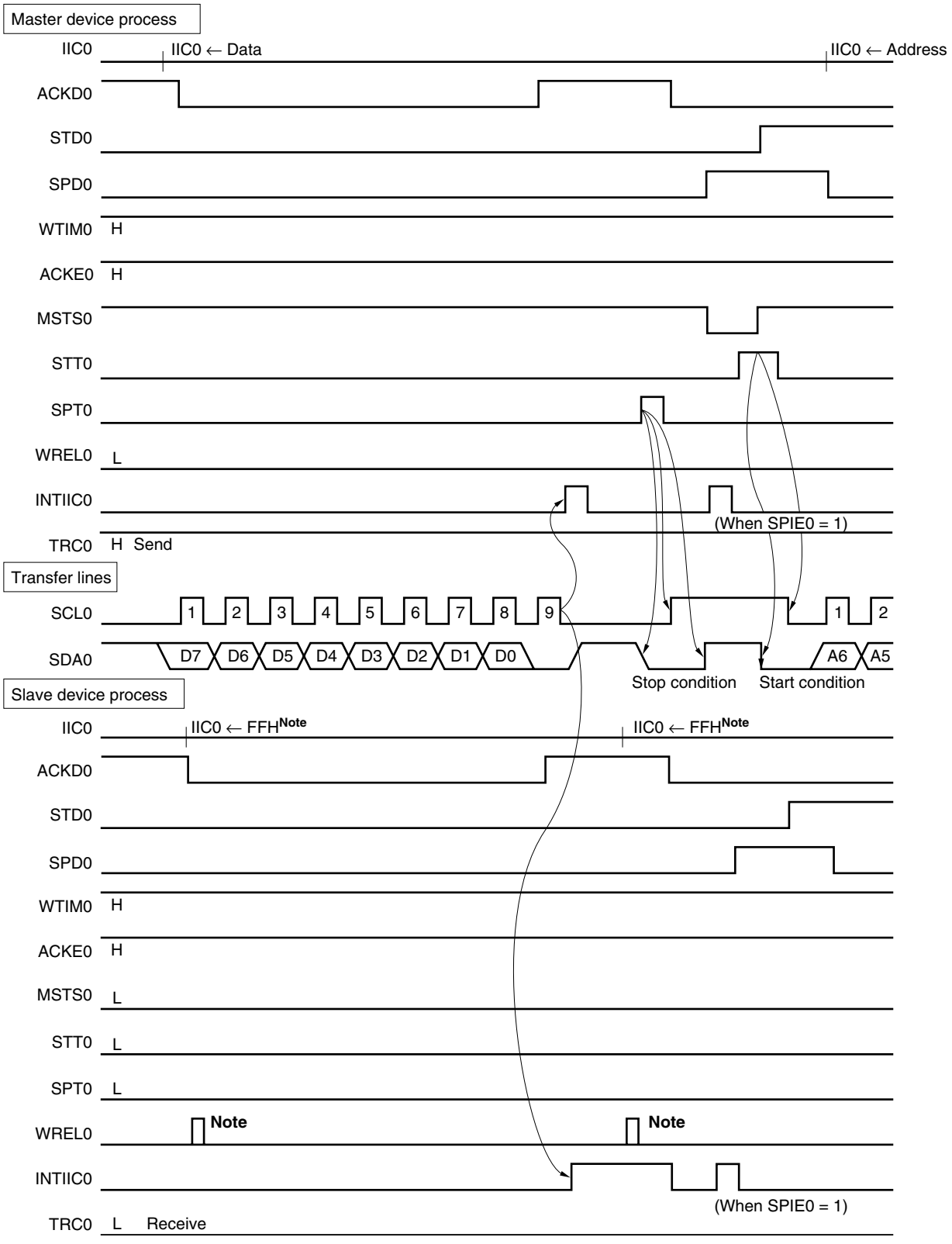
**(2) Data**




**Note** Release the slave wait by either IIC0 ← FFH or setting WREL0.

**Figure 18-20. Master → Slave Communication Example (When Master and Slave Select 9-Clock Wait) (3/3)**

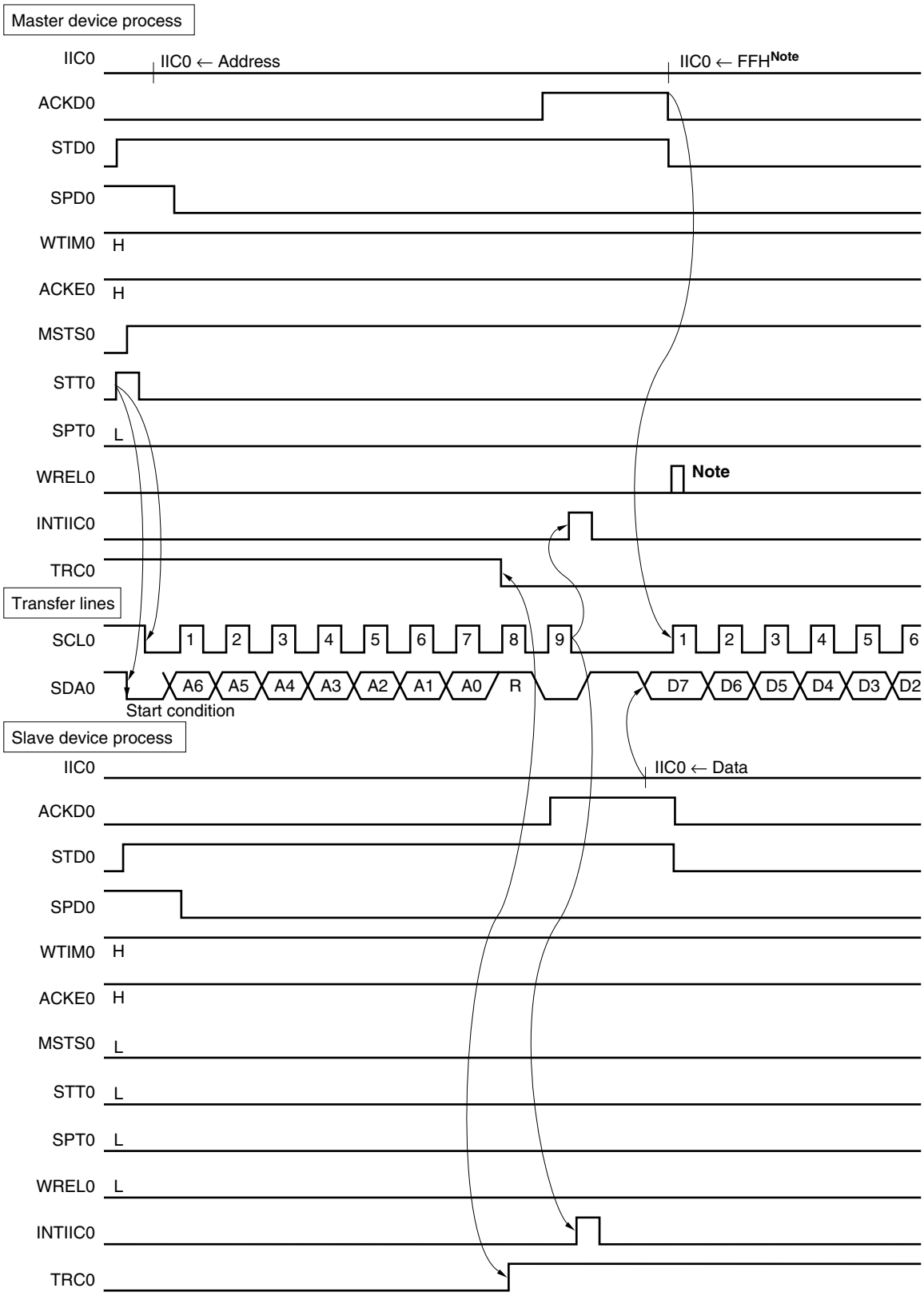
**(3) Stop condition**




**Note** Release the slave wait by either IIC0 ← FFH or setting WREL0.

**Figure 18-21. Slave → Master Communication Example (When Master and Slave Select 9-Clock Wait) (1/3)**

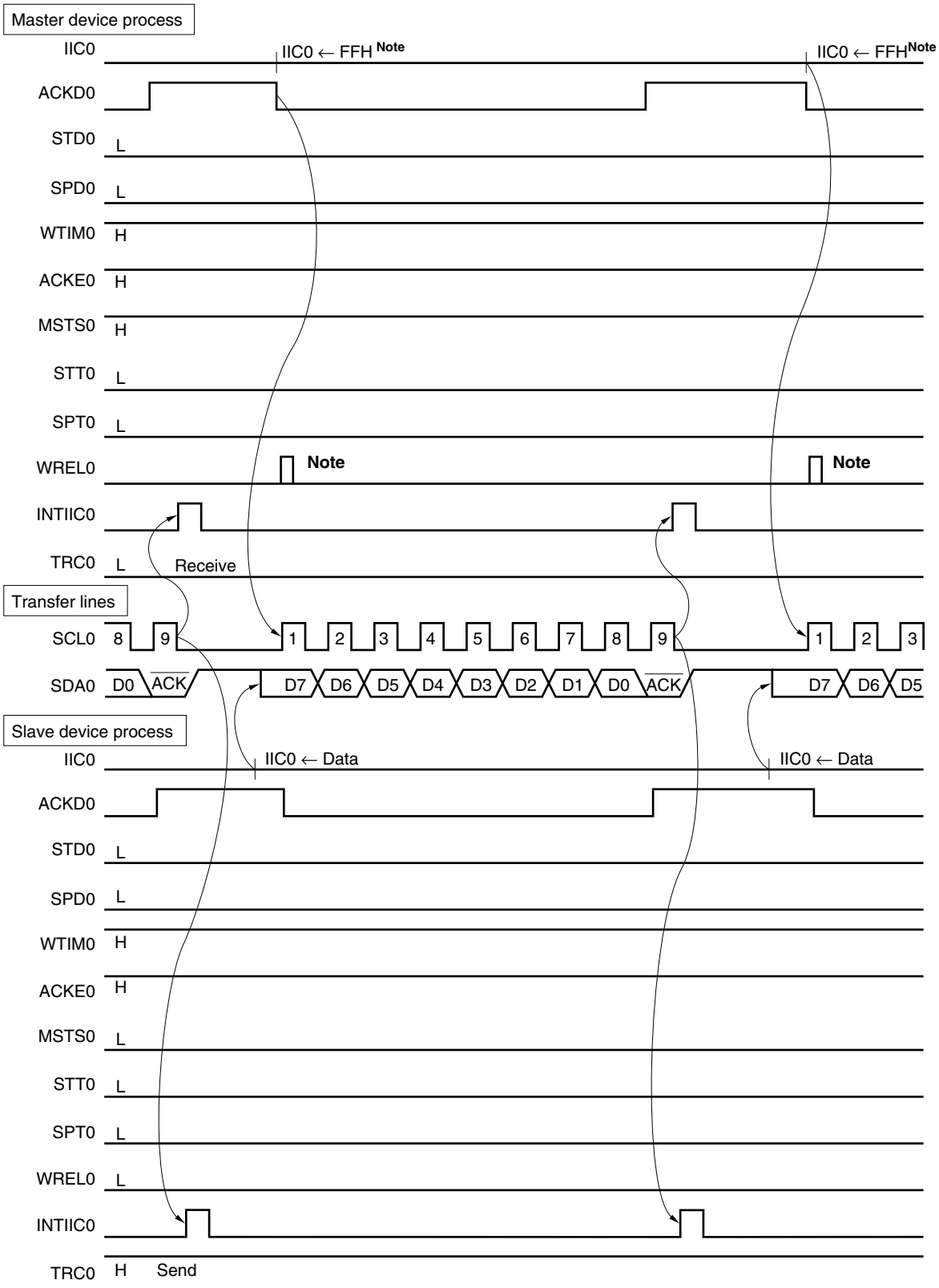
**(1) Start condition - Address**




**Note** Release the slave wait by either IIC0 ← FFH or setting WREL0.

**Figure 18-21. Slave → Master Communication Example (When Master and Slave Select 9-Clock Wait) (2/3)**

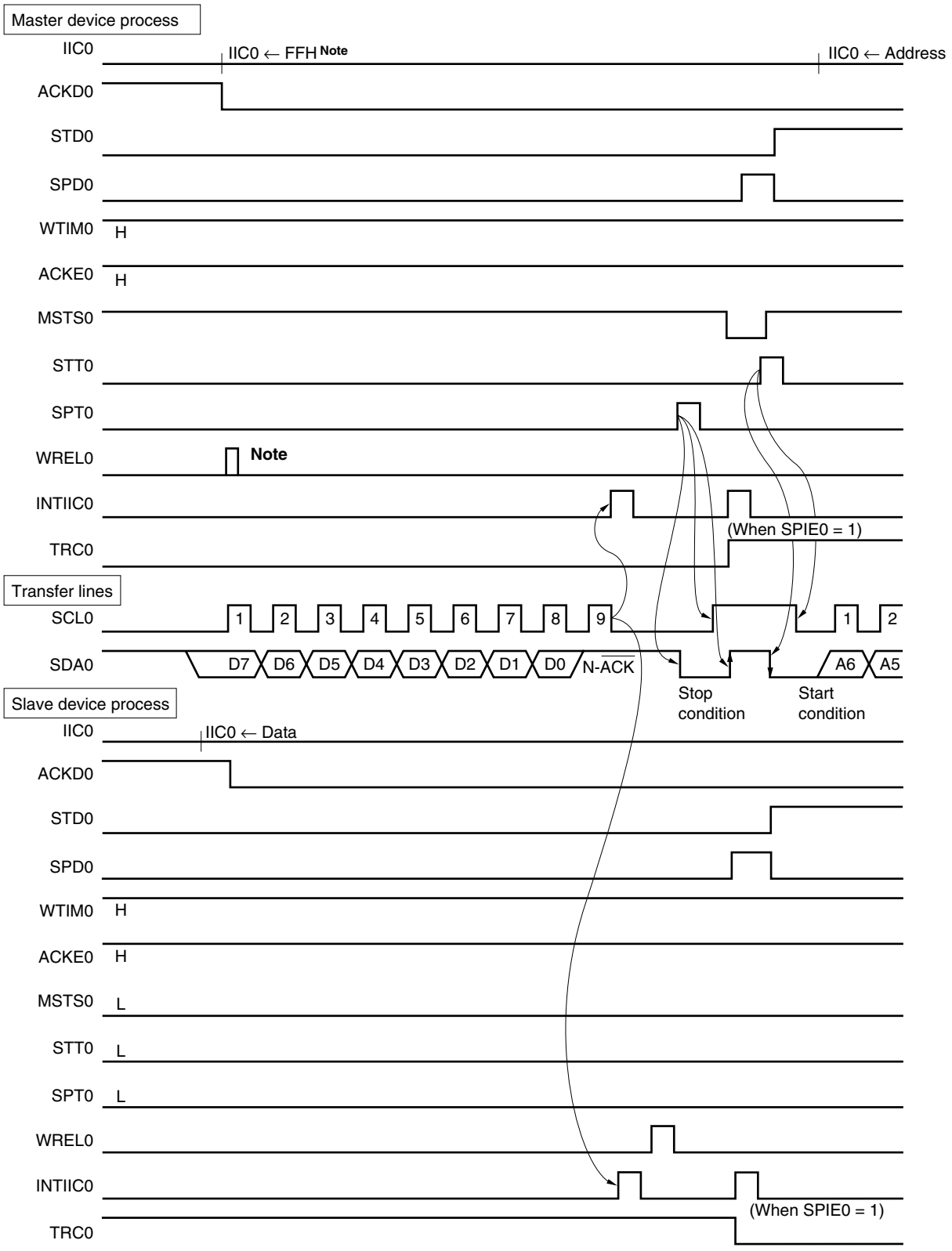
**(2) Data**




**Note** Release the slave wait by either IIC0 ← FFH or setting WREL0.

**Figure 18-21. Slave → Master Communication Example (When Master and Slave Select 9-Clock Wait) (3/3)**

**(3) Stop condition**




**Note** Release the slave wait by either IIC0 ← FFH or setting WREL0.

# CHAPTER 19 CLOCK OUTPUT FUNCTION

## 19.1 Functions

The clock output function is used to output the clock supplied to a peripheral LSI or carrier output during remote transmission. The clock selected by the clock output control register (CKS) is output from the PCL/P23 pin.

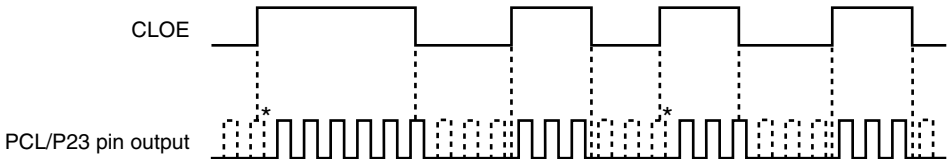
To output the clock pulse, follow the procedure described below.

<1> Select the output frequency of the clock pulse (while clock pulse output is disabled) with bits 0 to 3 (CCS0 to CCS3).
<2> Set the P23 output latch to 0.
<3> Set bit 3 (PM23) of the port mode register (PM2) to 0 (to set the output mode).
<4> Set bit 4 (CLOE) of CKS to 1.

**Caution If the output latch of P23 is set to 1, clock output cannot be used.**

**Remark** The clock output function is designed so that pulses with a narrow width are not output when clock output enable/disabled is switched (See "*" in **Figure 19-1**).

**Figure 19-1. Remote Control Output Application Example**



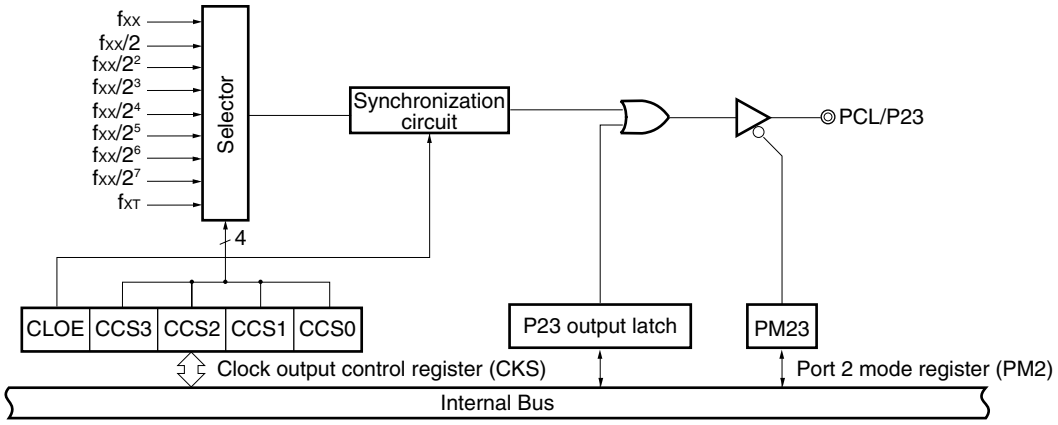
## 19.2 Configuration

The clock output function includes the following hardware.

**Table 19-1. Configuration of Clock Output Function**

| Item | Configuration |
|---|---|
| Control registers | Clock output control register (CKS)<br>Port 2 mode register (PM2) |

**Figure 19-2. Block Diagram of Clock Output Function**




## 19.3 Control Registers

The following two registers are used to control the clock output function.

- Clock output control register (CKS)
- Port 2 mode register (PM2)

**(1) Clock output control register (CKS)**

This register sets the PCL output clock.

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

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

**Remark** CKS provides a function for setting the buzzer output clock in addition to setting the PCL output clock.

**Figure 19-3. Format of Clock Output Control Register (CKS)**

Address: 0FF40H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CKS | BZOE | BCS1 | BCS0 | CLOE | CCS3 | CCS2 | CCS1 | CCS0 |

| | |
|---|---|
| BZOE | Buzzer output control (Refer to **Figure 20-2**) |

| | | |
|---|---|---|
| BCS1 | BCS0 | Buzzer output frequency selection (Refer to **Figure 20-2**) |

| CLOE | Clock output control |
|---|---|
| 0 | Clock output stop |
| 1 | Clock output start |

| CCS3 | CCS2 | CCS1 | CCS0 | Clock output frequency selection |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | $f_{XX}$ (12.5 MHz) |
| 0 | 0 | 0 | 1 | $f_{XX}/2$ (6.25 MHz) |
| 0 | 0 | 1 | 0 | $f_{XX}/4$ (3.13 MHz) |
| 0 | 0 | 1 | 1 | $f_{XX}/8$ (1.56 MHz) |
| 0 | 1 | 0 | 0 | $f_{XX}/16$ (781 kHz) |
| 0 | 1 | 0 | 1 | $f_{XX}/32$ (391 kHz) |
| 0 | 1 | 1 | 0 | $f_{XX}/64$ (195 kHz) |
| 0 | 1 | 1 | 1 | $f_{XX}/128$ (97.7 kHz) |
| 1 | 0 | 0 | 0 | $f_{XT}$ (32.768 kHz) |
| Other than above | | | | Setting prohibited |

**Remarks 1.** $f_{XX}$: Main system clock frequency ($f_X$ or $f_X/2$)

**2.** $f_X$: Main system clock oscillation frequency

**3.** $f_{XT}$: Subsystem clock oscillation frequency

**4.** Figures in parentheses apply to operation at $f_{XX}$ = 12.5 MHz or $f_{XT}$ = 32.768 kHz.

**(2) Port 2 mode register (PM2)**

This register sets input/output for port 2 in 1-bit units.

When using the P23/PCL pin for clock output, set the output latch of PM23 and P23 to 0.

PM2 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets PM2 to FFH.

**Figure 19-4. Format of Port 2 Mode Register (PM2)**

Address: 0FF22H After reset: FFH R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PM2 | PM27 | PM26 | PM25 | PM24 | PM23 | PM22 | PM21 | PM20 |

| PM2n | P2n pin I/O mode selection (n = 0 to 7) |
|---|---|
| 0 | Output mode (output buffer on) |
| 1 | Input mode (output buffer off) |

# CHAPTER 20 BUZZER OUTPUT FUNCTIONS

## 20.1 Function

This function outputs a square wave at frequencies of 1.5 kHz, 3.1 kHz, 6.1 kHz, and 12.2 kHz. The buzzer frequency selected by the clock output control register (CKS) is output from the BUZ/P24 pin.

The following procedure outputs the buzzer frequency.

<1> Select the buzzer output frequency by using bits 5 to 7 (BCS0, BCS1, BZOE) of CKS. (In a state where the buzzer is prohibited from sounding.)
<2> Set the P24 output latch to 0.
<3> Set bit 4 (PM24) of the port 2 mode register (PM2) to 0 (to set the output mode).

**Caution When the output latch of P24 is set to 1, the buzzer output cannot be used.**

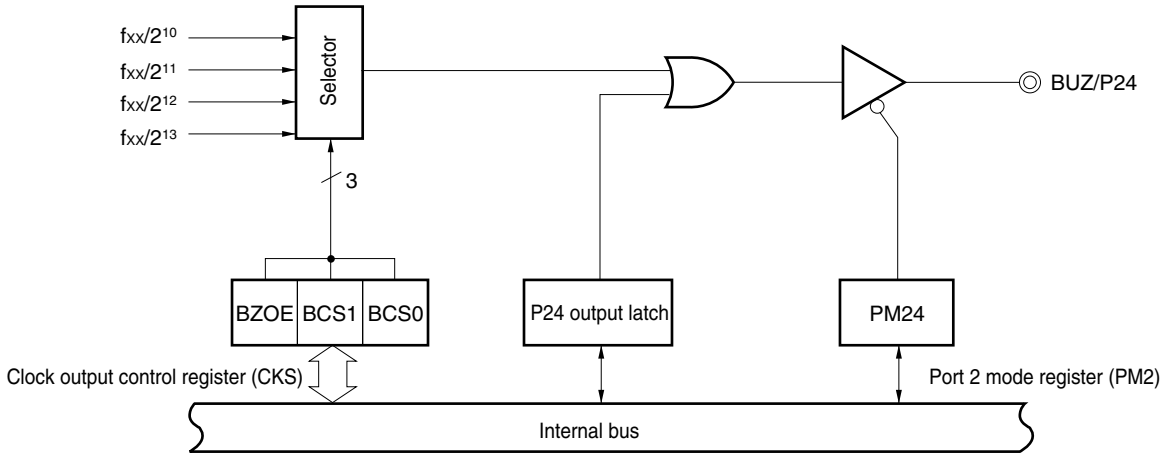
## 20.2 Configuration

The buzzer output function includes the following hardware.

**Table 20-1. Configuration of Buzzer Output Function**

| Item | Configuration |
|---|---|
| Control register | Clock output control register (CKS)<br>Port 2 mode register (PM2) |

**Figure 20-1. Block Diagram of Buzzer Output Function**



## 20.3 Control Registers

The buzzer output function is controlled by the following two registers.

- Clock output control register (CKS)
- Port 2 mode register (PM2)

### (1) Clock output control register (CKS)

This register sets the frequency of the buzzer output.

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

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

**Remark** CKS provides a function for setting the clock for PCL output in addition to setting the buzzer output frequency.

**Figure 20-2. Format of Clock Output Control Register (CKS)**

Address: 0FF40H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CKS | BZOE | BCS1 | BCS0 | CLOE | CCS3 | CCS2 | CCS1 | CCS0 |

| BZOE | Buzzer output buzzer |
|---|---|
| 0 | Stop buzzer output |
| 1 | Start buzzer output |

| BCS1 | BCS0 | Buzzer output frequency selection |
|---|---|---|
| 0 | 0 | $f_{XX}/2^{10}$ (12.2 kHz) |
| 0 | 1 | $f_{XX}/2^{11}$ (6.1 kHz) |
| 1 | 0 | $f_{XX}/2^{12}$ (3.1 kHz) |
| 1 | 1 | $f_{XX}/2^{13}$ (1.5 kHz) |

| CLOE | Clock output control (Refer to **Figure 19-3**) |
|---|---|

| CCS3 | CCS2 | CCS1 | CCS0 | Clock output frequency selection (Refer to **Figure 19-3**) |
|---|---|---|---|---|

**Remarks 1.** $f_{XX}$: Main system clock frequency ($f_X$ or $f_X/2$)

**2.** $f_X$: Main system clock oscillation frequency

**3.** Figures in parentheses apply to operation with $f_{XX}$ = 12.5 MHz.

**(2) Port 2 mode register (PM2)**

This register sets port 2 input/output in 1-bit units.

When the P24/BUZ pin is used as the buzzer output function, set the output latches of PM24 and P24 to 0.

PM2 is set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets PM2 to FFH.

**Figure 20-3. Format of Port 2 Mode Register (PM2)**

Address: 0FF22H After reset: FFH R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PM2 | PM27 | PM26 | PM25 | PM24 | PM23 | PM22 | PM21 | PM20 |

| PM2n | P2n pin I/O mode selection (n = 0 to 7) |
|---|---|
| 0 | Output mode (output buffer on) |
| 1 | Input mode (output buffer off) |

# CHAPTER 21 EDGE DETECTION FUNCTION

The P00 to P05 pins have an edge detection function that can be programmed to detect the rising edge or falling edge and send the detected edge to on-chip hardware components.

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 0 (EGP0), external interrupt falling edge enable register 0 (EGN0)**
  The EGP0 and EGN0 registers specify the valid edge to be detected by the P00 to P05 pins. They can be read/written by an 8-bit or 1-bit manipulation instruction.
  $\overline{\text{RESET}}$ input sets EGP0 and EGN0 to 00H.

**Figure 21-1. Format of External Interrupt Rising Edge Enable Register 0 (EGP0) and External Interrupt Falling Edge Enable Register 0 (EGN0)**

Address: 0FFA0H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| EGP0 | 0 | 0 | EGP5 | EGP4 | EGP3 | EGP2 | EGP1 | EGP0 |

Address: 0FFA2H After reset: 00H R/W

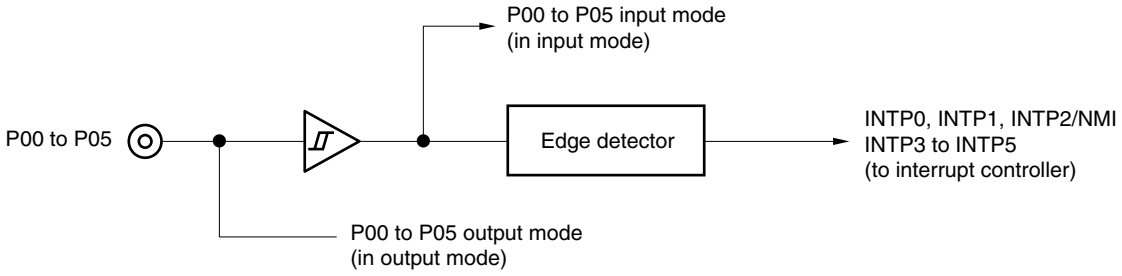
| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| EGN0 | 0 | 0 | EGN5 | EGN4 | EGN3 | EGN2 | EGN1 | EGN0 |

| EGPn | EGNn | INTPn pin valid edge (n = 0 to 5) |
|---|---|---|
| 0 | 0 | Interrupt disabled |
| 0 | 1 | Falling edge |
| 1 | 0 | Rising edge |
| 1 | 1 | Both rising and falling edges |

## 21.2 Edge Detection of P00 to P05 Pins

The P00 to P05 pins do not incorporate an analog delay-based noise eliminator. Therefore, a valid edge is input to the pins and edge detection is performed (acknowledged) immediately after passing through the hysteresis-type input buffer.

**Figure 21-2. Block Diagram of P00 to P05 Pins**



# CHAPTER 22 INTERRUPT FUNCTIONS

The μPD784225 is provided with three interrupt request service modes (refer to **Table 22-1**). These three service modes can be set as required in the program. However interrupt servicing 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 & restoring from stack | Executed by branching to service program at address[Note] specified by vector table |
| Context switching | | Saving to & restoring 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 preset service such as data transfer 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 μPD784225 has the 28 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 | INTWDTM (watchdog timer overflow) | Watchdog timer | WDTIC | Possible | Possible | 0FE06H | 0006H |
| | 1 | INTP0 (pin input edge detection) | Edge detection | PIC0 | | | 0FE08H | 0008H |
| | 2 | INTP1 (pin input edge detection) | | PIC1 | | | 0FE0AH | 000AH |
| | 3 | INTP2 (pin input edge detection) | | PIC2 | | | 0FE0CH | 000CH |
| | 4 | INTP3 (pin input edge detection) | | PIC3 | | | 0FE0EH | 000EH |
| | 5 | INTP4 (pin input edge detection) | | PIC4 | | | 0FE10H | 0010H |
| | 6 | INTP5 (pin input edge detection) | | PIC5 | | | 0FE12H | 0012H |
| | 7 | INTIIC0 (CSI0 $I^2C$ bus transfer end)[Note] | Clocked serial interface | CSIIC0 | | | 0FE16H | 0016H |
| | | INTCSI0 (CSI0 3-wire transfer end) | | | | | | |
| | 8 | INTSER1 (ASI1 UART reception error) | Asynchronous serial interface/ clocked serial interface 1 | SERIC1 | | | 0FE18H | 0018H |
| | 9 | INTSR1 (ASI1 UART reception end) | | SRIC1 | | | 0FE1AH | 001AH |
| | | INTCSI1 (CSI1 3-wire transfer end) | | | | | | |
| | 10 | INTST1 (ASI1 UART transmission end) | | STIC1 | | | 0FE1CH | 001CH |
| | 11 | INTSER2 (ASI2 UART reception error) | Asynchronous serial interface/ clocked serial interface 2 | SERIC2 | | | 0FE1EH | 001EH |
| | 12 | INTSR2 (ASI2 UART reception end) | | SRIC2 | | | 0FE20H | 0020H |
| | | INTCSI2 (CSI2 3-wire transfer end) | | | | | | |
| | 13 | INTST2 (ASI2 UART transmission end) | | STIC2 | | | 0FE22H | 0022H |
| | 14 | INTTM3 (reference time interval signal from watch timer) | Watch timer | TMIC3 | | | 0FE24H | 0024H |
| | 15 | INTTM00 (match signal generation of 16-bit timer counter 0 and capture/ compare register 00 (CR00)) | Timer/ counter | TMIC00 | | | 0FE26H | 0026H |
| | 16 | INTTM01 (match signal generation of 16-bit timer counter 0 and capture/ compare register 01 (CR01)) | | TMIC01 | | | 0FE28H | 0028H |
| | 17 | INTTM1 (match signal generation of 8-bit timer counter 1) | Timer/ counter 1 | TMIC1 | | | 0FE2AH | 002AH |
| | 18 | INTTM2 (match signal generation of 8-bit timer counter 2) | Timer/ counter 2 | TMIC2 | | | 0FE2CH | 002CH |
| | 19 | INTAD (A/D converter conversion end) | A/D converter | ADIC | | | 0FE2EH | 002EH |
| | 20 | INTTM5 (match signal generation of 8-bit timer counter 5) | Timer/ counter 5 | TMIC5 | | | 0FE30H | 0030H |
| | 21 | INTTM6 (match signal generation of 8-bit timer counter 6) | Timer/ counter 6 | TMIC6 | | | 0FE32H | 0032H |
| | 22 | INTWT (watch timer overflow) | Watch timer | WTIC | | | 0FE38H | 0038H |

**Note** $\mu$PD784225Y Subseries only

**Remarks 1.** The default priority is a fixed number and indicates the order of priority when interrupt requests specified as having the same priority are generated simultaneously.

**2.** ASI: Asynchronous serial interface
CSI: Clocked serial interface

**3.** The watchdog timer has two interrupt sources, a non-maskable interrupt (INTWDT) and a maskable interrupt (INTWDTM), either (but not both) of which can be selected.

### 22.1.1 Software interrupts

Interrupts by software consist of the BRK instruction, which generates a vectored interrupt, and the BRKCS instruction, which performs context switching.

Software interrupts are acknowledged even in the interrupt disabled state, and are not subject to priority control.

### 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 have a higher priority than any other interrupt.

**Note** Except during execution of the service program for the same non-maskable interrupt, or 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 servicing (though some interrupts cannot use macro servicing: 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, multiservicing 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 Servicing Modes

There are three μPD784225 interrupt servicing modes, as follows.

- Vectored interrupt servicing
- Macro servicing
- Context switching

### 22.2.1 Vectored interrupt servicing

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 servicing

When an interrupt is acknowledged, CPU execution is temporarily suspended and data transfer is performed by hardware. Since macro servicing 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 preset vector address in the register bank, and at the same time the current program counter (PC) and program status word (PSW) are saved in the register bank (refer to **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-purpose registers, the program counter (PC), program status word (PSW), and stack pointer (SP).

## 22.3 Interrupt Servicing Control Registers

μPD784225 interrupt servicing is controlled for each interrupt request by various control registers that specify interrupt servicing. The interrupt control registers are listed in Table 22-3.

**Table 22-3. Control Registers**

| Register Name | Symbol | Function |
|---|---|---|
| Interrupt control registers | WDTIC, PIC0, PIC1, PIC2, PIC3, PIC4, PIC5, CSIIC0, SERIC1, SRIC1, STIC1, SERIC2, SRIC2, STIC2, TMIC3, TMIC00, TMIC01, TMIC1, TMIC2, ADIC, TMIC5, TMIC6, WTIC | Record the generation of interrupt requests, control masking, specify vectored interrupt servicing or macro service processing, enable or disable the context switching function, and specify priority. |
| Interrupt mask registers | MK0 (MK0L, MK0H)<br>MK1 (MK1L, MK1H) | Control masking of maskable interrupt requests. Associated with the mask control flag in the 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 as lowest level (level 3). |
| Interrupt selection control register | SNMI | Selects whether to use input signal from the P02 pin and the interrupt signal from the watchdog timer as a maskable interrupt or NMI. |
| Watchdog timer mode register | WDM | Specifies the priority of interrupts from the NMI pin input and overflow of the watchdog timer. |
| Program status word | PSW | Enables or disables acknowledging maskable interrupts. |

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 | Priority Specification Flag | Context Switching Enable Flag |
| 0 | INTWDTM | WDTIC | WDTIF | WDTMK | WDTISM | WDTPR0<br>WDTPR1 | WDTCSE |
| 1 | INTP0 | PIC0 | PIF0 | PMK0 | PISM0 | PPR00<br>PPR01 | PCSE0 |
| 2 | INTP1 | PIC1 | PIF1 | PMK1 | PISM1 | PPR10<br>PPR11 | PCSE1 |
| 3 | INTP2 | PIC2 | PIF2 | PMK2 | PISM2 | PPR20<br>PPR21 | PCSE2 |
| 4 | INTP3 | PIC3 | PIF3 | PMK3 | PISM3 | PPR30<br>PPR31 | PCSE3 |
| 5 | INTP4 | PIC4 | PIF4 | PMK4 | PISM4 | PPR40<br>PPR41 | PCSE4 |
| 6 | INTP5 | PIC5 | PIF5 | PMK5 | PISM5 | PPR50<br>PPR51 | PCSE5 |
| 7 | INTIIC0<br>INTCSI0 | CSIIC0 | CSIIF0 | CSIMK0 | CSIISM0 | CSIPR00<br>CSIPR01 | CSICSE0 |
| 8 | INTSER1 | SERIC1 | SERIF1 | SERMK1 | SERISM1 | SERPR10<br>SERPR11 | SERCSE1 |
| 9 | INTSR1<br>INTCSI1 | SRIC1 | SRIF1 | SRMK1 | SRISM1 | SRPR10<br>SRPR11 | SRCSE1 |
| 10 | INTST1 | STIC1 | STIF1 | STMK1 | STISM1 | STPR10<br>STPR11 | STCSE1 |
| 11 | INTSER2 | SERIC2 | SERIF2 | SERMK2 | SERISM2 | SERPR20<br>SERPR21 | SERCSE2 |
| 12 | INTSR2<br>INTCSI2 | SRIC2 | SRIF2 | SRMK2 | SRISM2 | SRPR20<br>SRPR21 | SRCSE2 |
| 13 | INTST2 | STIC2 | STIF2 | STMK2 | STISM2 | STPR20<br>STPR21 | STCSE2 |
| 14 | INTTM3 | TMIC3 | TMIF3 | TMMK3 | TMISM3 | TMPR30<br>TMPR31 | TMCSE3 |
| 15 | INTTM00 | TMIC00 | TMIF00 | TMMK00 | TMISM00 | TMPR000<br>TMPR001 | TMCSE00 |
| 16 | INTTM01 | TMIC01 | TMIF01 | TMMK01 | TMISM01 | TMPR010<br>TMPR011 | TMCSE01 |
| 17 | INTTM1 | TMIC1 | TMIF1 | TMMK1 | TMISM1 | TMPR10<br>TMPR11 | TMCSE1 |
| 18 | INTTM2 | TMIC2 | TMIF2 | TMMK2 | TMISM2 | TMPR20<br>TMPR21 | TMCSE2 |
| 19 | INTAD | ADIC | ADIF | ADMK | ADISM | ADPR00<br>ADPR01 | ADCSE |
| 20 | INTTM5 | TMIC5 | TMIF5 | TMMK5 | TMISM5 | TMPR50<br>TMPR51 | TMCSE5 |
| 21 | INTTM6 | TMIC6 | TMIF6 | TMMK6 | TMISM6 | TMPR60<br>TMPR61 | TMCSE6 |
| 22 | INTWT | WTIC | WTIF | WTMK | WTISM | WTPR0<br>WTPR1 | WTCSE |

### 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 of individual interrupt sources for the 23 maskable interrupts. Up to 4 priority levels can be specified, and a number of interrupt sources can be specified at the same level. Among maskable interrupt sources, level 0 is the highest priority.

If multiple interrupt requests are generated simultaneously among interrupt source of the same priority level, they are acknowledged in default priority order.

These flags can be manipulated bit-wise by software.

$\overline{\text{RESET}}$ input sets all bits to 1.

**(2) Context switching enable flag (××CSE)**

The context switching enable flag specifies that a maskable interrupt request is to be serviced by context switching.

In context switching, the register bank specified beforehand is selected by hardware, a branch is made to a vector address stored beforehand in the register bank, and at the same time the current contents of the program counter (PC) and program status word (PSW) are saved in the register bank.

Context switching is suitable for real-time processing, since execution of interrupt servicing can be started faster than with normal vectored interrupt servicing.

This flag can be manipulated bit-wise by software.

$\overline{\text{RESET}}$ input sets all bits to 0.

**(3) Macro service enable flag (××ISM)**

The macro service enable flag specifies whether an interrupt request corresponding to that flag is to be handled as a vectored interrupt or by context switching, or by macro servicing.

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 servicing/ context switching servicing).

This flag can be manipulated bit-wise by software.

$\overline{\text{RESET}}$ input sets all bits to 0.

**(4) Interrupt mask flag (××MK)**

An interrupt mask flag specifies enabling/disabling of vectored interrupt servicing and macro service processing for the interrupt request corresponding to that flag.

The interrupt mask contents are not changed by the start of interrupt servicing, 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 software.

$\overline{\text{RESET}}$ input sets all bits to 1.

**(5) Interrupt request flag (××IF)**

An interrupt request flag is set (1) by generation of the interrupt request that corresponds to that flag. When the interrupt is acknowledged, the flag is automatically cleared (0) by hardware.

This flag can be manipulated by software.

$\overline{\text{RESET}}$ input sets all bits to 0.

**Figure 22-1. Interrupt Control Register (xxICn) (1/3)**

Address: 0FFE0H to 0FFE6H, 0FFE8H After reset: 43H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| WDTIC | WDTIF | WDTMK | WDTISM | WDCSE | 0 | 0 | WDTPR1 | WDTPR0 |
| PIC0 | PIF0 | PMK0 | PISM0 | PCSE0 | 0 | 0 | PPR01 | PPR00 |
| PIC1 | PIF1 | PMK1 | PISM1 | PCSE1 | 0 | 0 | PPR11 | PPR10 |
| PIC2 | PIF2 | PMK2 | PISM2 | PCSE2 | 0 | 0 | PPR21 | PPR20 |
| 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 |
| CSIIC0 | CSIIF0 | CSIMK0 | CSIISM0 | CSICSE0 | 0 | 0 | CSIPR01 | CSIPR00 |

| xxIFn | Interrupt request generation |
|---|---|
| 0 | No interrupt request (interrupt signal is not generated) |
| 1 | Interrupt request (interrupt signal is generated) |

| xxMKn | Interrupt servicing enable/disable |
|---|---|
| 0 | Interrupt servicing enabled |
| 1 | Interrupt servicing disabled |

| xxISMn | Interrupt servicing mode specification |
|---|---|
| 0 | Vectored interrupt servicing/context switching processing |
| 1 | Macro service processing |

| xxCSEn | Context switching processing specification |
|---|---|
| 0 | Processing as vectored interrupt |
| 1 | Processing by 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 |
|---|---|---|---|---|---|---|---|---|
| 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 |
| SERIC2 | SERIF2 | SERMK2 | SERISM2 | SERCSE2 | 0 | 0 | SERPR21 | SERPR20 |
| SRIC2 | SRIF2 | SRMK2 | SRISM2 | SRCSE2 | 0 | 0 | SRPR21 | SRPR20 |
| STIC2 | STIF2 | STMK2 | STISM2 | STCSE2 | 0 | 0 | STPR21 | STPR20 |
| TMIC3 | TMIF3 | TMMK3 | TMISM3 | TMCSE3 | 0 | 0 | TMPR31 | TMPR30 |
| 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 servicing enable/disable |
|---|---|
| 0 | Interrupt servicing enabled |
| 1 | Interrupt servicing disabled |

| xxISMn | Interrupt servicing mode specification |
|---|---|
| 0 | Vectored interrupt servicing/context switching processing |
| 1 | Macro service processing |

| xxCSEn | Context switching processing specification |
|---|---|
| 0 | Processing as vectored interrupt |
| 1 | Processing by 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 0FFF6H, 0FFF9H  After reset: 43H  R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| TMIC1 | TMIF1 | TMMK1 | TMISM1 | TMCSE1 | 0 | 0 | TMPR11 | TMPR10 |
| TMIC2 | TMIF2 | TMMK2 | TMISM2 | TMCSE2 | 0 | 0 | TMPR21 | TMPR20 |
| ADIC | ADIF | ADMK | ADISM | ADCSE | 0 | 0 | ADPR01 | ADPR00 |
| TMIC5 | TMIF5 | TMMK5 | TMISM5 | TMCSE5 | 0 | 0 | TMPR51 | TMPR50 |
| TMIC6 | TMIF6 | TMMK6 | TMISM6 | TMCSE6 | 0 | 0 | TMPR61 | TMPR60 |
| WTIC | WTIF | WTMK | WTISM | WTCSE | 0 | 0 | WTPR1 | WTPR0 |

| xxIFn | Interrupt request generation |
|---|---|
| 0 | No interrupt request (interrupt signal is not generated) |
| 1 | Interrupt request (interrupt signal is generated) |

| xxMKn | Interrupt servicing enable/disable |
|---|---|
| 0 | Interrupt servicing enabled |
| 1 | Interrupt servicing disabled |

| xxISMn | Interrupt servicing mode specification |
|---|---|
| 0 | Vectored interrupt servicing/context switching processing |
| 1 | Macro service processing |

| xxCSEn | Context switching processing specification |
|---|---|
| 0 | Processing as vectored interrupt |
| 1 | Processing by 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 registers are composed of interrupt mask flags. MK0 and MK1 are 16-bit registers that can be manipulated in 16-bit units. In addition 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 MK0 and MK1 can be manipulated individually with a bit manipulation instruction in 1-bit units. 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 MK0 and MK1 is the same flag as the interrupt mask flag in the interrupt control register. MK0 and MK1 are provided for blanket control of interrupt masking.

After $\overline{\text{RESET}}$ input, 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 | 1 | PMK5 | PMK4 | PMK3 | PMK2 | PMK1 | PMK0 | WDTMK |
| MK0H | TMMK3 | STMK2 | SRMK2 | SERMK2 | STMK1 | SRMK1 | SERMK1 | CSIMK0 |
| MK1L | 1 | TMMK6 | TMMK5 | ADMK | TMMK2 | TMMK1 | TMMK01 | TMMK00 |
| MK1H | 1 | 1 | 1 | 1 | 1 | 1 | WTMK | 1 |

| xxMKn | Interrupt request enable/disable |
|---|---|
| 0 | Interrupt servicing enabled |
| 1 | Interrupt servicing disabled |

**<Word access>**

Address: 0FFACH, 0FFAEH After reset: FFFFH R/W

| Symbol | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| MK0 | TMMK3 | STMK2 | SRMK2 | SERMK2 | STMK1 | SRMK1 | SERMK1 | CSIMK0 |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | 1 | PMK5 | PMK4 | PMK3 | PMK2 | PMK1 | PMK0 | WDTMK |
| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MK1 | 1 | 1 | 1 | 1 | 1 | 1 | WTMK | 1 |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | 1 | TMMK6 | TMMK5 | ADMK | TMMK2 | TMMK1 | TMMK01 | TMMK00 |

| xxMKn | Interrupt request enable/disable |
|---|---|
| 0 | Interrupt servicing enabled |
| 1 | Interrupt servicing disabled |

### 22.3.3 In-service priority register (ISPR)

The ISPR shows the priority level of the maskable interrupt currently being serviced and the non-maskable interrupt being processed. When a maskable interrupt request is acknowledged, the bit corresponding to the priority of that interrupt request is set (1), and remains set until the service program ends. When a non-maskable interrupt is acknowledged, the bit corresponding to the priority of that non-maskable interrupt is set (1), and remains set until the service program ends.

When the RETI or RETCS instruction is executed, the bit among those set (1) in the ISPR that corresponds to the highest-priority interrupt request is automatically cleared (0) by hardware.

The contents of the ISPR are not changed by execution of the RETB or RETCSB instruction.

$\overline{\text{RESET}}$ input clears the ISPR 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 servicing 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 by 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 interrupts (lowest level) |
|---|---|
| 0 | Interrupts with level 3 (lowest level) can be nested. |
| 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 the 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), initialize the system by 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 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 clears 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 Interrupt selection control register (SNMI)

SNMI selects whether to use interrupt request signals from the watchdog timer and inputs from the P02 pin as maskable interrupts or non-maskable interrupts.

Since the bits of this register can be set (1) only once after reset, the bits should be cleared (0) by reset.

SNMI is set with a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input clears SNMI to 00H.

**Figure 22-6. Format of Interrupt Selection Control Register (SNMI)**

Address: 0FFA9H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| SNMI | 0 | 0 | 0 | 0 | 0 | 0 | SWDT | SNMI |

| SWDT | Watchdog timer interrupt selection |
|---|---|
| 0 | Use as non-maskable interrupt.<br>Interrupt servicing cannot be disabled with interrupt mask register. |
| 1 | Use as maskable interrupt.<br>Vectored interrupts and macro servicing can be used. Interrupt servicing can be disabled with interrupt mask register. |

| SNMI | P02 pin function selection |
|---|---|
| 0 | Use as INTP2.<br>Vectored interrupts and macro servicing can be used.<br>Interrupt servicing can be disabled with interrupt mask register.<br>At this time, the standby mode set by the P02 pin is released with a maskable interrupt. |
| 1 | Use as NMI.<br>Interrupt servicing cannot be disabled with interrupt mask register.<br>At this time, the standby mode set by the P02 pin is released with an NMI. |

### 22.3.7 Program status word (PSW)

The PSW is a register that holds the current status of instruction execution results and interrupt requests. The IE flag that sets enabling/disabling of maskable interrupts is mapped in the lower 8 bits of the PSW (PSWL).

PSWL can be read or written to with an 8-bit manipulation instruction, and can also be manipulated with a bit manipulation instruction or dedicated instruction (EI/DI).

When a vectored interrupt is acknowledged or the BRK instruction is executed, PSWL is saved to the stack and the IE flag is cleared (0). PSWL is also saved to the stack by the PUSH PSW instruction, and is restored from the stack by the RETI, RETB and POP PSW instructions.

When context switching or the BRKCS instruction is executed, PSWL is saved to a fixed area in the register bank, and the IE flag is cleared (0). PSWL is restored from the fixed area in the register bank by the RETCSI or RETCSB instruction.

$\overline{\text{RESET}}$ input clears PSWL to 00H.

**Figure 22-7. Format of Program Status Word (PSWL)**

After reset: 00H

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

| | |
|---|---|
| S | Used for normal instruction execution |
| Z | |
| RSS | |
| AC | |

| IE | Enable or disable interrupt acknowledgment |
|---|---|
| 0 | Disabled |
| 1 | Enabled |

| | |
|---|---|
| P/V | Used for normal instruction execution |
| CY | |

## 22.4 Software Interrupt Acknowledgment Operations

A software interrupt is acknowledged in response to execution of the BRK or BRKCS instruction. Software interrupts cannot be disabled.

### 22.4.1 BRK instruction software interrupt acknowledgment operation

When the 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 lower 16 bits of the PC, and 0000B into the higher 4 bits, and a branch is performed (the start of the service program must be in the base area).

The RETB instruction must be used to return from a BRK instruction software interrupt.

**Caution The RETI instruction must not be used to return from a BRK instruction software interrupt.**

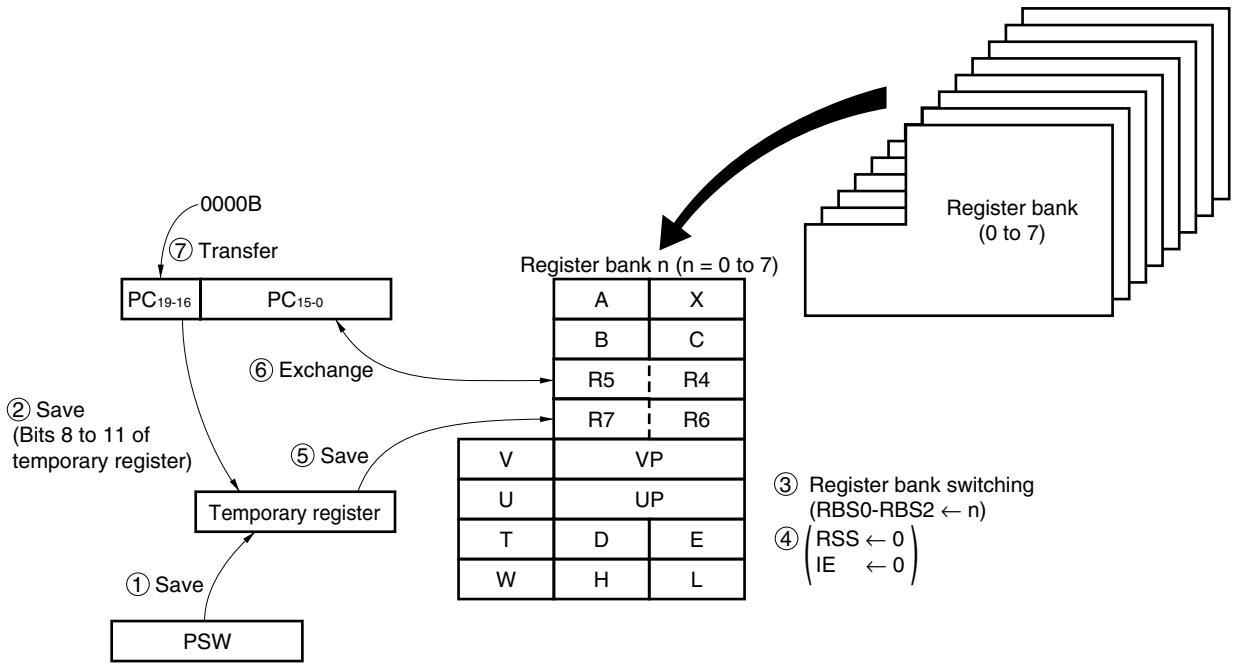
### 22.4.2 BRKCS instruction software interrupt (software context switching) acknowledgment operation

The context switching function can be initiated by executing the BRKCS instruction.

The register bank to be used after context switching is specified by the BRKCS instruction operand.

When the 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.

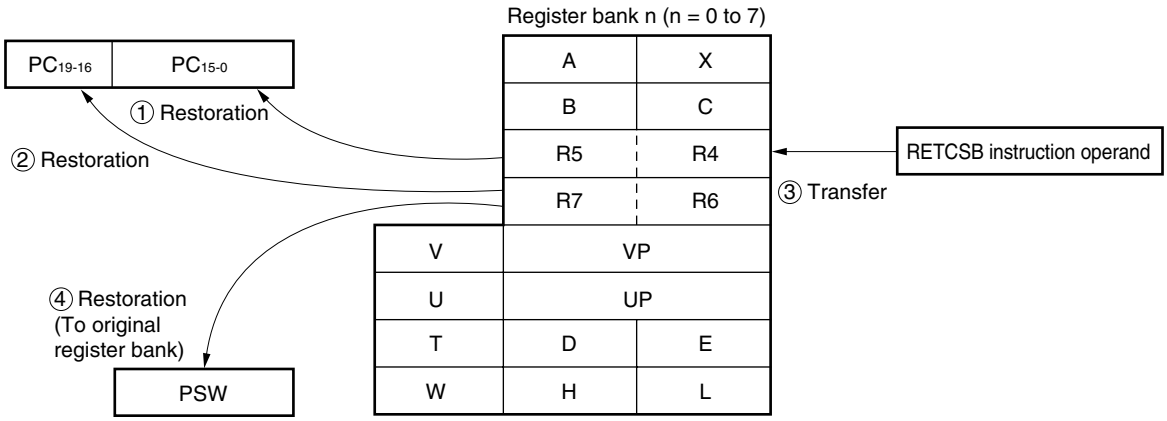
**Figure 22-8. Context Switching Operation by Execution of BRKCS Instruction**




The RETCSB instruction is used to return from a software interrupt generated by the BRKCS instruction. The RETCSB instruction must specify the start address of the interrupt service program for when context switching is next performed by the 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 22-9. Return from BRKCS Instruction Software Interrupt (RETCSB Instruction Operation)**




## 22.5 Operand Error Interrupt Acknowledgement Operation

An operand error interrupt is generated when the data obtained by inverting all the bits of the 3rd byte of the operand of the MOV STBC, #byte instruction or LOCATION instruction or the MOV WDM,#byte instruction does not match the 4th byte of the operand. Operand error interrupts cannot be disabled.

When an operand error interrupt is generated, the program status word (PSW) and the start address of the instruction that caused the error are saved to the stack, the IE flag is cleared (0), the vector table value is loaded into the program counter (PC), and a branch is performed (within the base area only).

As the address saved to the stack is the start address of the instruction in which the error occurred, simply writing an RETB instruction at the end of the operand error interrupt service program will result in generation of another operand error interrupt. You should therefore either process the address in the stack or initialize the program by referring to **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.

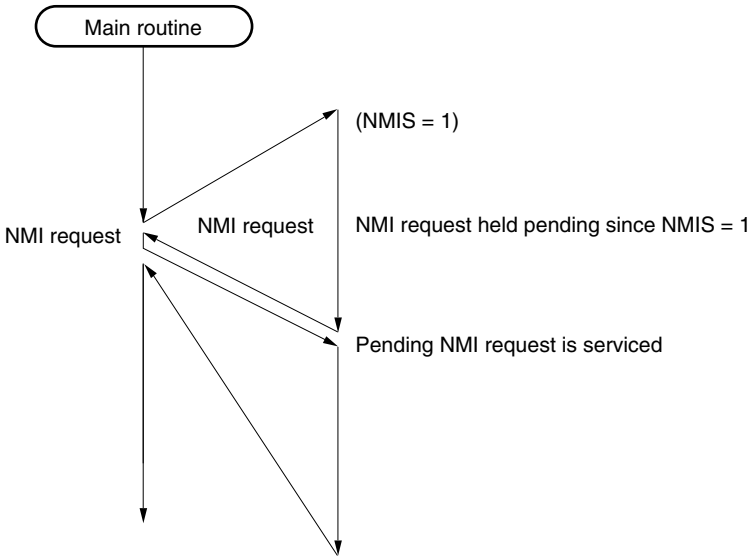
The relative priorities of non-maskable interrupts are set by the WDT4 bit of the watchdog timer mode register (WDM) (see **22.3.5 Watchdog timer mode register (WDM)**).

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 (0), the in-service priority register (ISPR) bit corresponding to the acknowledged non-maskable interrupt is set (1), the vector table contents are loaded into the PC, and a branch is performed. The ISPR bit that is set (1) is the NMIS bit in the case of a non-maskable interrupt due to edge input to the NMI pin, and the WDTS bit in the case of watchdog timer overflow.

When the non-maskable interrupt service program is executed, non-maskable interrupt requests of the same priority as the non-maskable interrupt currently being executed and non-maskable interrupts of lower priority than the non-maskable interrupt currently being executed are held pending. A pending non-maskable interrupt is acknowledged 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 22-10. Non-Maskable Interrupt Request Acknowledgment Operations (1/2)**

**(a) When a new NMI request is generated during NMI service program execution**




**(b) When a watchdog timer interrupt request is generated during NMI service program execution (when the watchdog timer interrupt priority is higher (when WDT4 in the WDM = 1))**

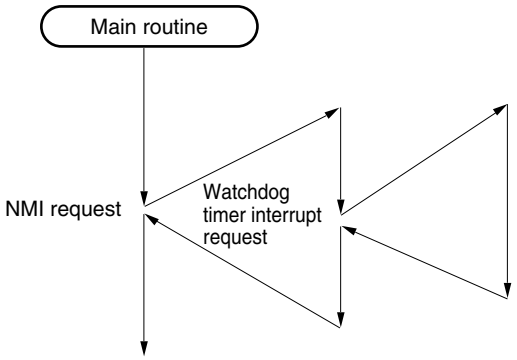
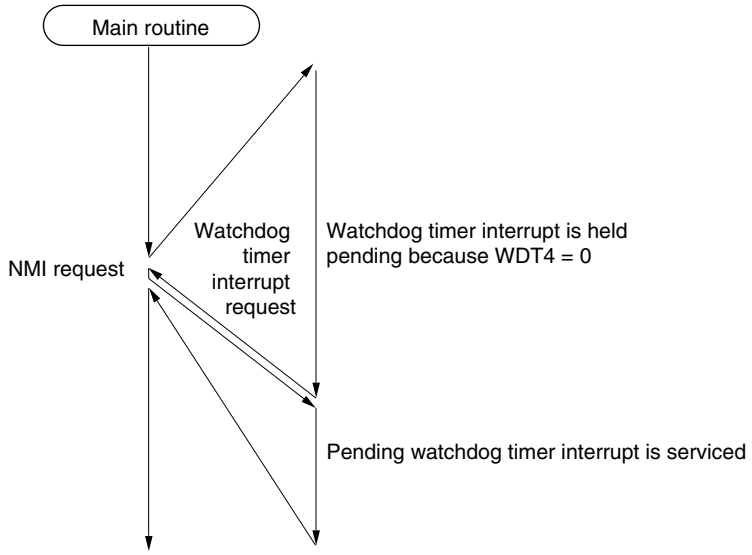
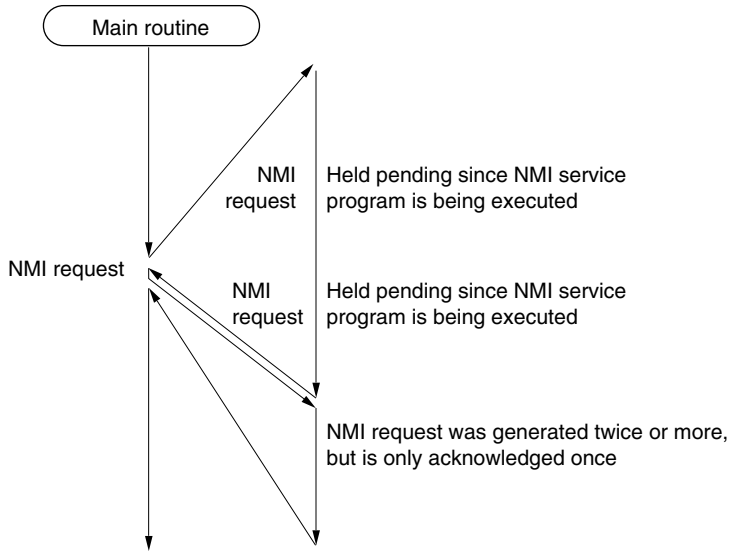


**Figure 22-10. Non-Maskable Interrupt Request Acknowledgment Operations (2/2)**

**(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))**




**(d) When an NMI request is generated twice during NMI service program execution**



**Cautions 1. Macro service requests are acknowledged and serviced even during execution of a non-maskable interrupt service program. To avoid macro service processing being performed during a non-maskable interrupt service program, 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 (SFR) (see Table 3.6 in 3.9 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 at which RAM is not incorporated, with the result that the return from the non-maskable interrupt service program is not performed normally and a software malfunction occurs.**

**Therefore, the program following $\overline{\text{RESET}}$ release must be as shown below.**

```
        CSEG AT 0
        DW   STRT
        CSEG BASE
STRT:
        LOCATION   0FH; or LOCATION 0H
        MOVG SP, #imm24
```

## 22.7 Maskable Interrupt Acknowledgment Operation

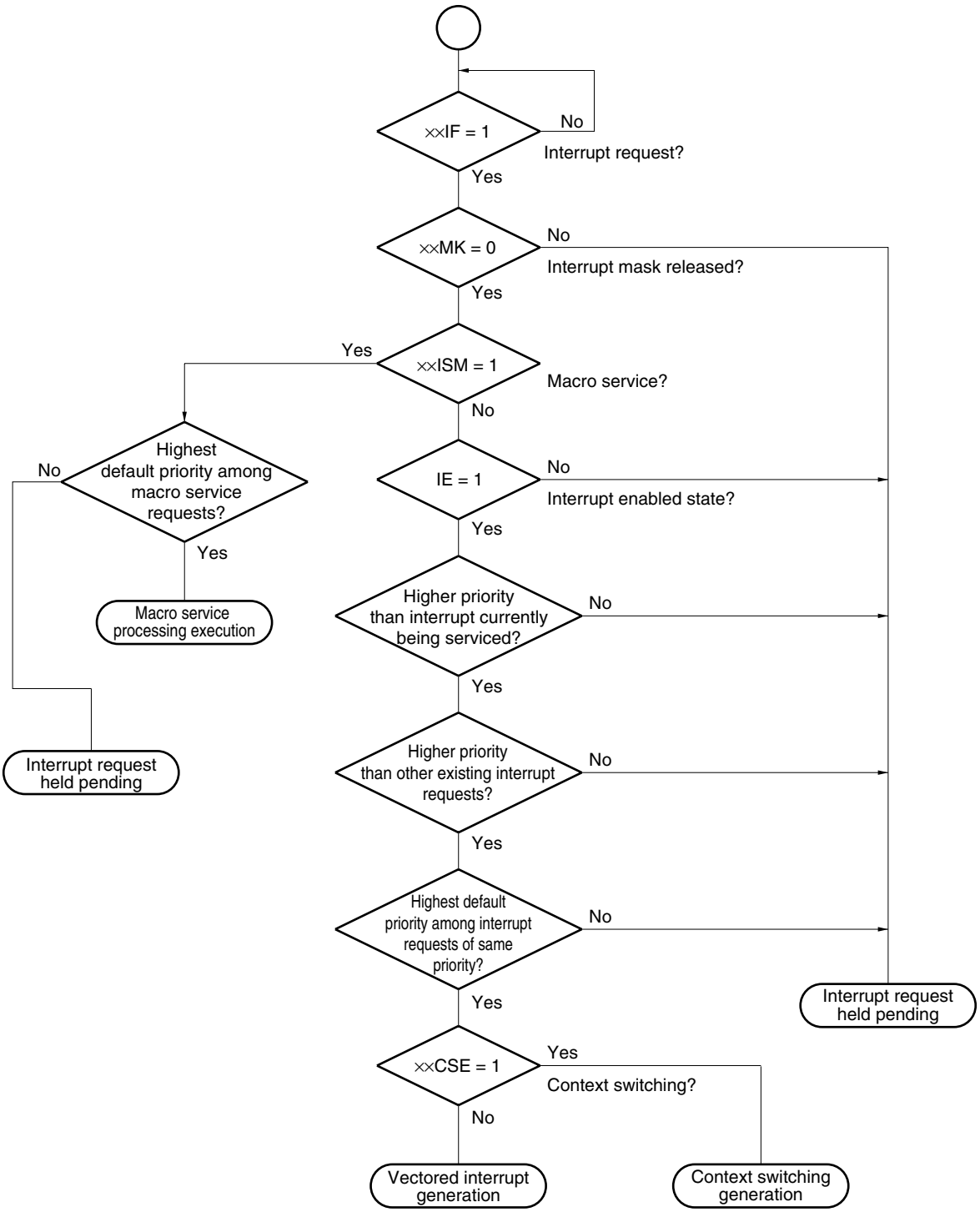
A maskable interrupt can be acknowledged when the interrupt request flag is set (1) and the mask flag for that interrupt is cleared (0). When servicing is performed by a macro service, the interrupt is acknowledged and serviced by the macro service immediately. In the case of vectored interruption and context switching, an interrupt is acknowledged in the interrupt enabled state (when the IE flag is set (1)) if the priority of that interrupt is one for which acknowledgment is permitted.

If maskable interrupt requests are generated simultaneously, the interrupt for which the highest priority is specified by the priority specification flag is acknowledged. If the interrupts have the same priority specified, they are acknowledged in accordance with their default priorities.

A pending interrupt is acknowledged when a state in which it can be acknowledged is established.

The interrupt acknowledgment algorithm is shown in Figure 22-11.

**Figure 22-11. Interrupt Acknowledgment Processing Algorithm**



### 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 (0) (the interrupt disabled status is set), and the in-service priority register (ISPR) bit corresponding to the priority of the acknowledged interrupt is set (1). Also, data in the vector table predetermined for each interrupt request is loaded into the PC, and a branch is performed. The return from a vectored interrupt is performed by means of the RETI instruction.

**Caution When a maskable interrupt is acknowledged by vectored interrupt, the RETI instruction must be used to return from the interrupt. Subsequent interrupt acknowledgment will not be performed normally if a different instruction is used.**

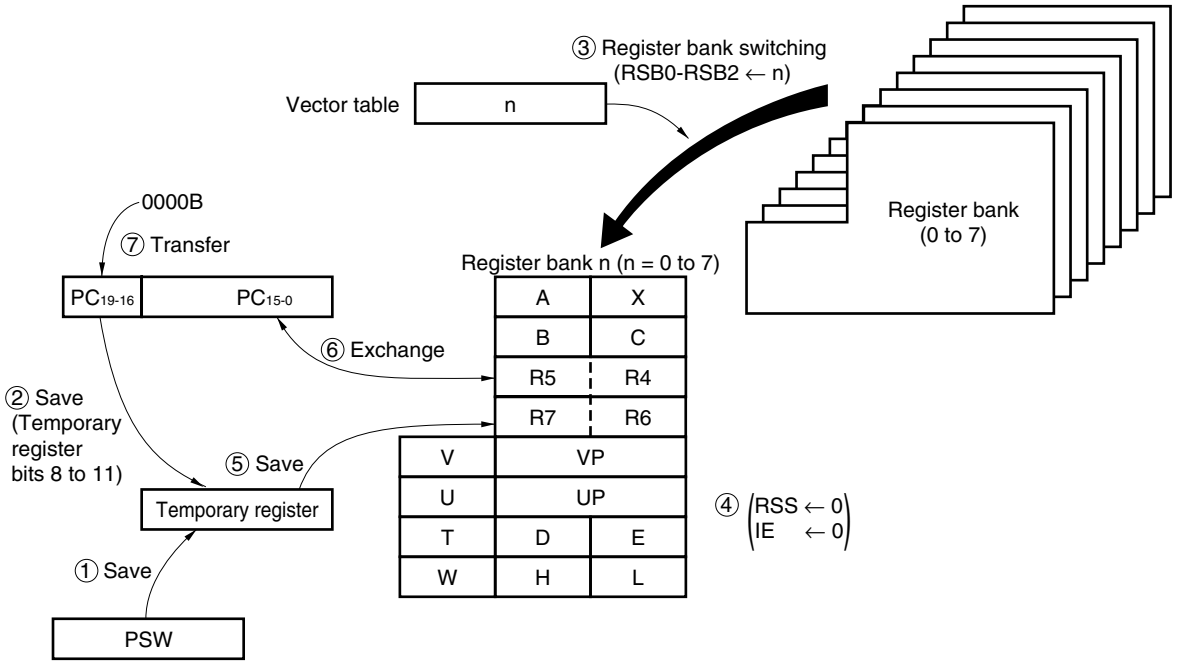
### 22.7.2 Context switching

Initiation of the context switching function is enabled by setting (1) the context switching enable flag of the interrupt control register.

When an interrupt request for which the context switching function is enabled is acknowledged, the register bank specified by the lowest 3 bits of the lower address (even address) of the corresponding vector table address is selected.

The vector address stored beforehand in the selected register bank is transferred to the program counter (PC), and at the same time the contents of the PC and program status word (PSW) up to that time are saved in the register bank and branching is performed to the interrupt service program.

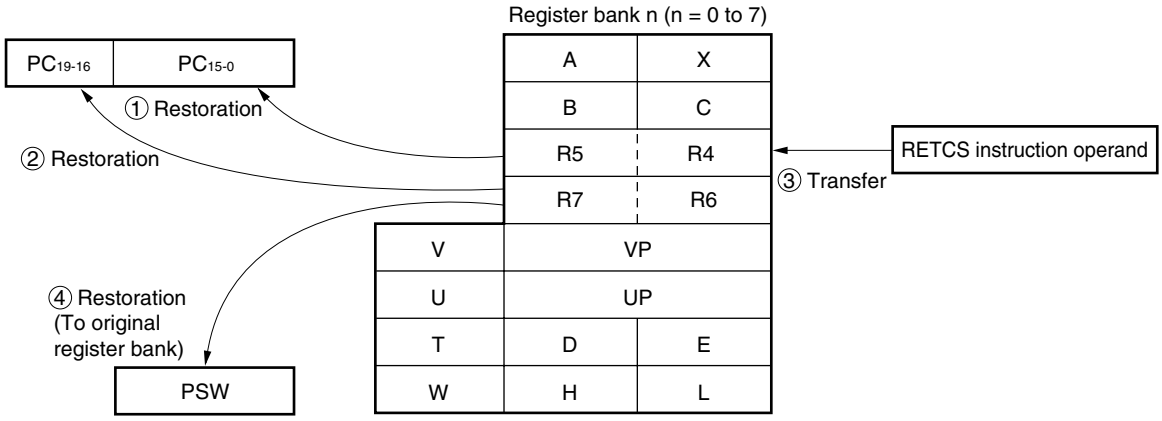
**Figure 22-12. Context Switching Operation by Generation of an Interrupt Request**



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 22-13. Return from Interrupt That Uses Context Switching by Means of RETCS Instruction**



### 22.7.3 Maskable interrupt priority levels

The μPD784225 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 servicing 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 interruption is permitted are shown in Table 22-5.

Since the IE flag is cleared (0) automatically when an interrupt is acknowledged, when multiple interruption is used, the IE flag should be set (1) to enable interrupts by executing the 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 requests only |
| | | 1 | × | • All maskable interrupts |
| 3 | 00001000 | 0 | × | • All macro service requests only |
| | | 1 | 0 | • All maskable interrupts |
| | | 1 | 1 | • All macro service requests<br>• Maskable interrupts specified as priority 0/1/2 |
| 2 | 0000×100 | 0 | × | • All macro service requests only |
| | | 1 | × | • All macro service requests<br>• Maskable interrupts specified as priority 0/1 |
| 1 | 0000××10 | 0 | × | • All macro service requests only |
| | | 1 | × | • All macro service requests<br>• Maskable interrupts specified as priority 0 |
| 0 | 0000×××1 | × | × | • All macro service requests only |
| Non-maskable interrupts | 1000××××<br>0100××××<br>1100×××× | × | × | • All macro service requests only |

**Figure 22-14. Examples of Servicing When Another Interrupt Request Is Generated During Interrupt Servicing (1/3)**

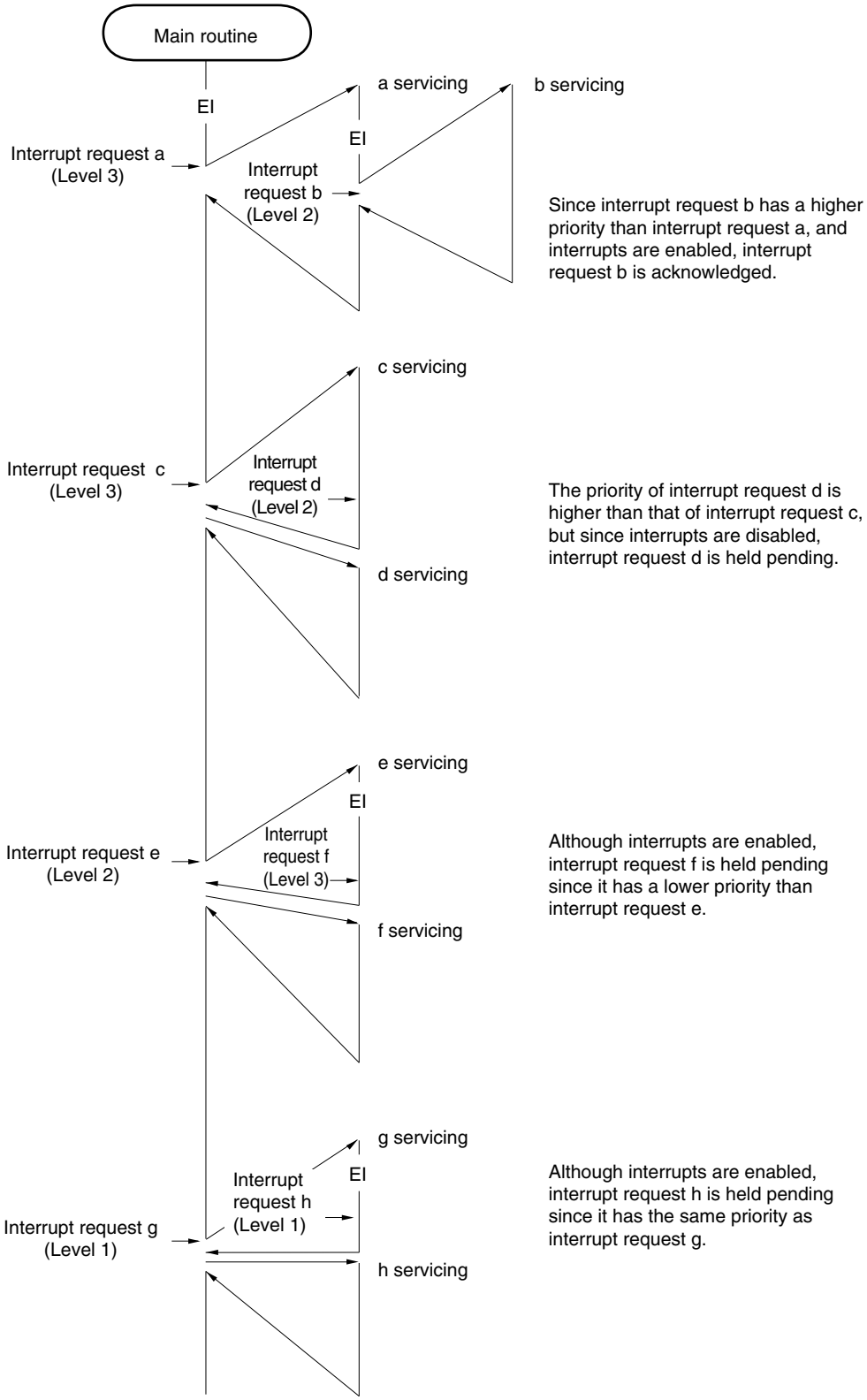
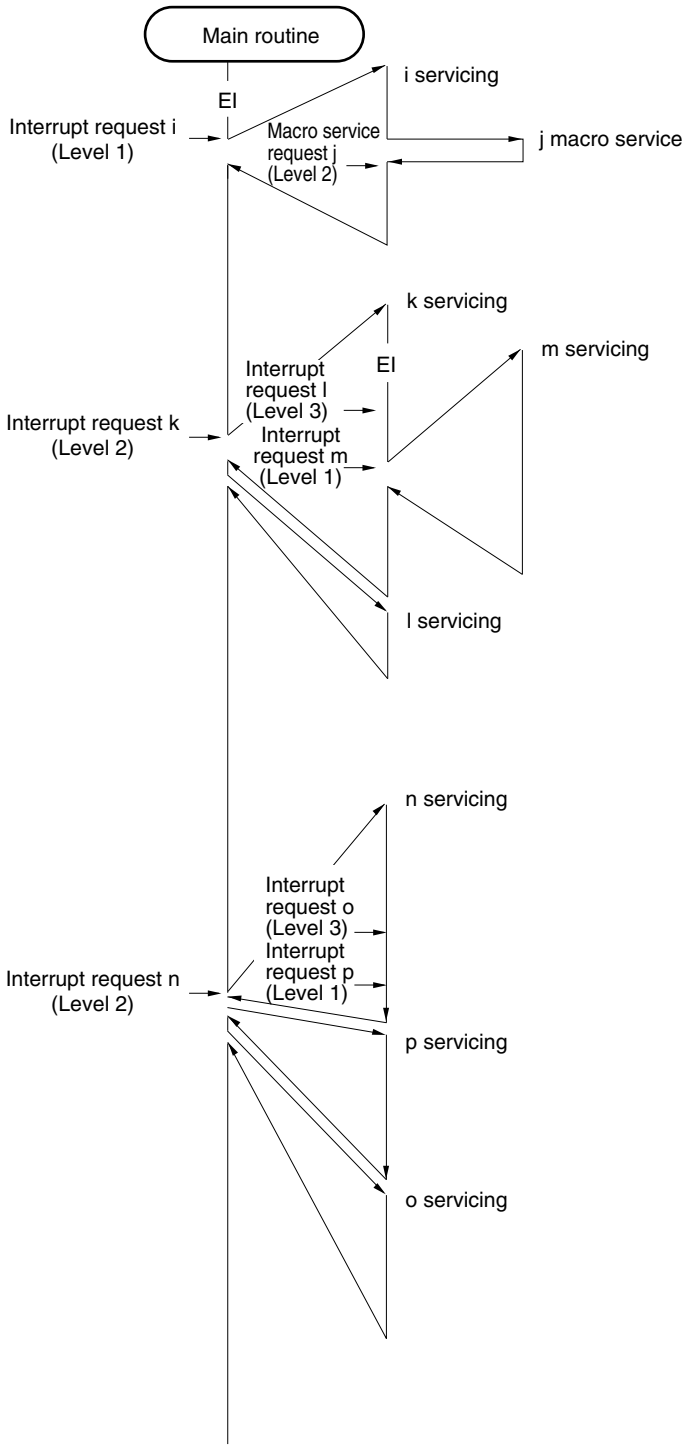


**Figure 22-14. Examples of Servicing When Another Interrupt Request Is Generated During Interrupt Servicing (2/3)**

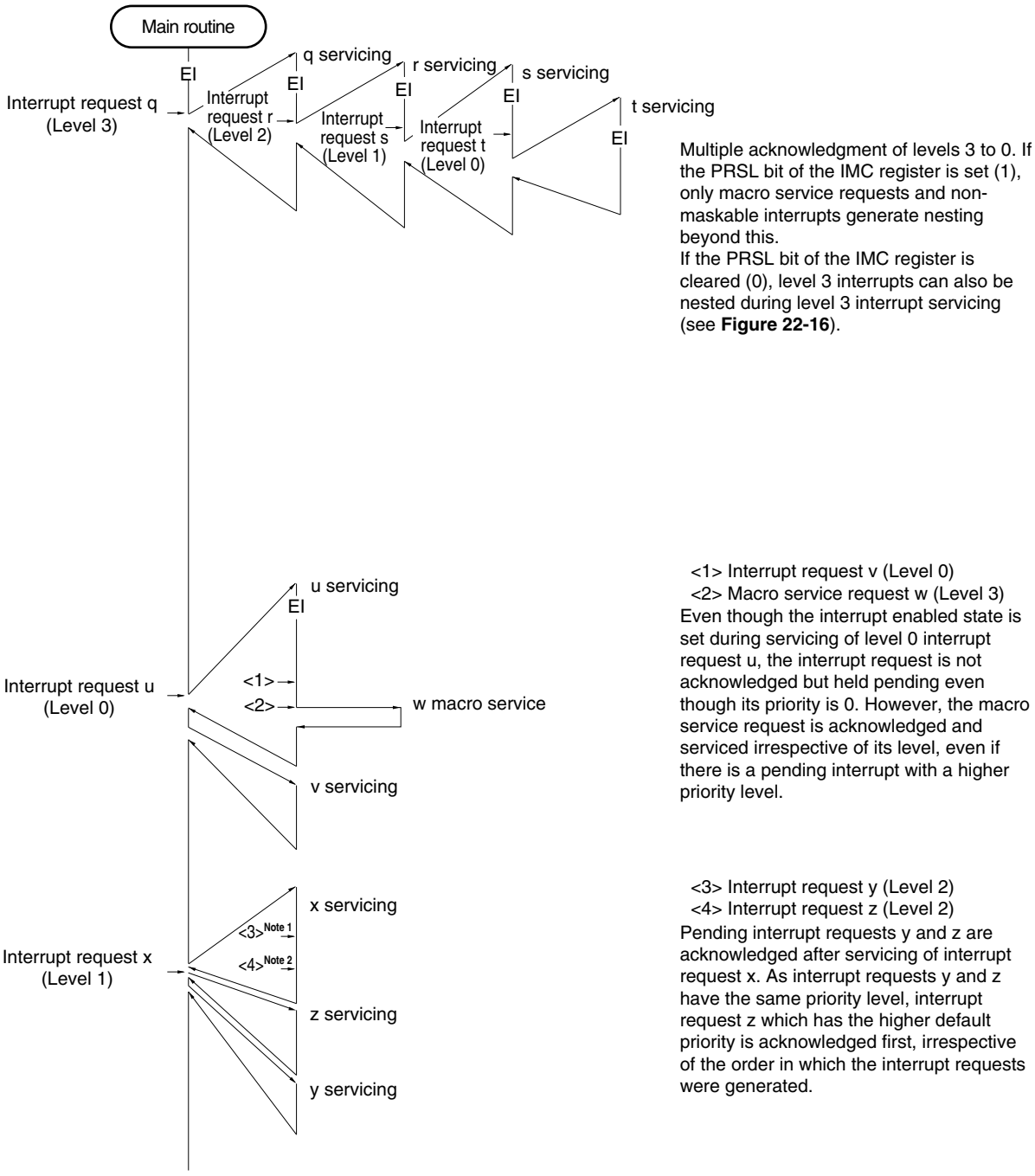



The macro service request is serviced irrespective of interrupt enabling/disabling and priority.

The interrupt request is held pending since it has a lower priority than interrupt request k. Interrupt request m generated after interrupt request l has a higher priority, and is therefore acknowledged first.

Since servicing of interrupt request n performed in the interrupt disabled state, interrupt requests o and p are held pending.
After interrupt request n servicing, the pending interrupt requests are acknowledged. Although interrupt request o was generated first, interrupt request p has a higher priority and is therefore acknowledged first.

**Figure 22-14. Examples of Servicing When Another Interrupt Request Is Generated During Interrupt Servicing (3/3)**



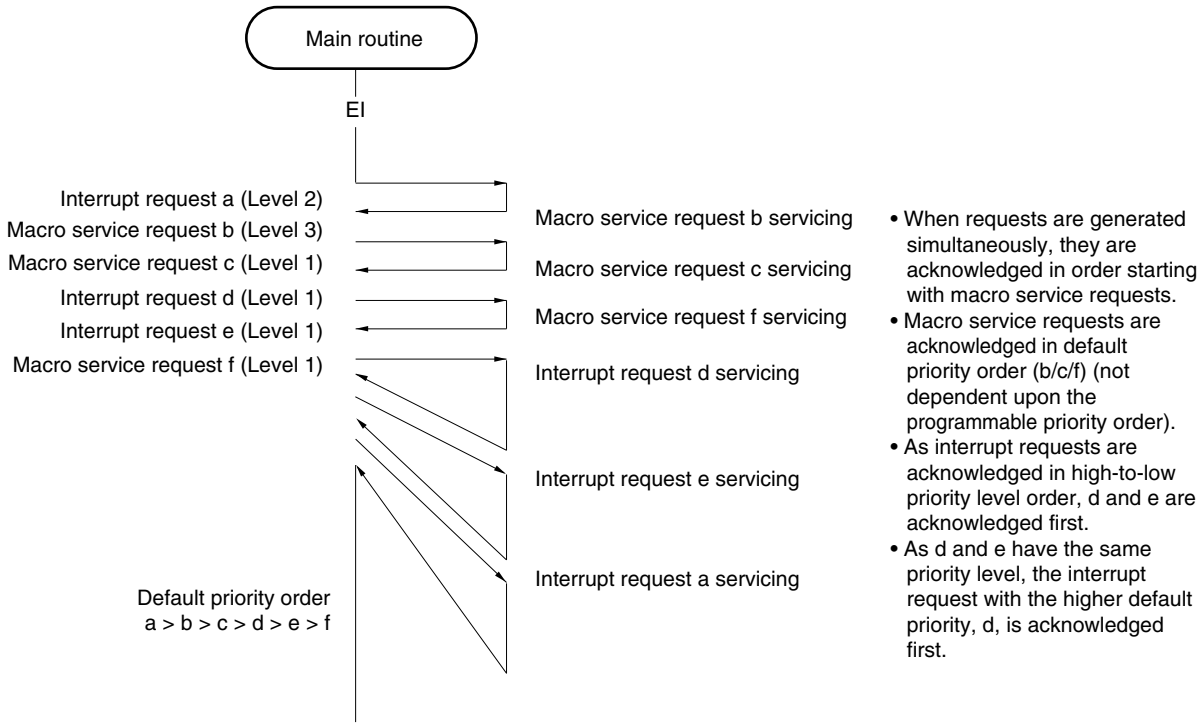

**Notes 1.** Low default priority

**2.** High default priority

**Remarks 1.** “a” to “z” in the figure above are arbitrary names used to differentiate between the interrupt requests and macro service requests.

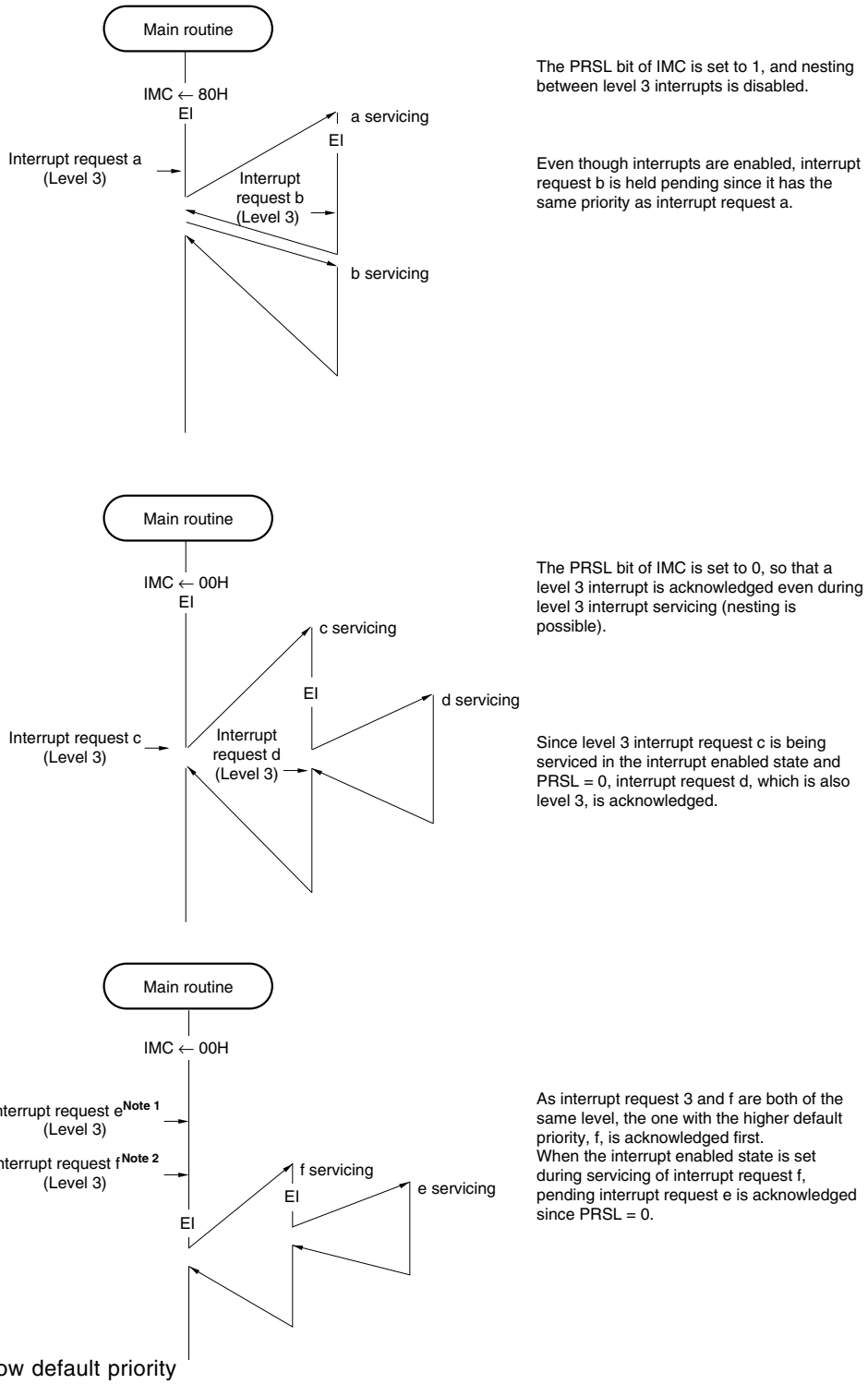
**2.** High/low default priorities in the figure indicate the relative priority levels of the two interrupt requests.

**Figure 22-15. Examples of Servicing of Simultaneously Generated Interrupt Requests**




**Remark** “a” to “f” in the figure above are arbitrary names used to differentiate between the interrupt requests and macro service requests.

**Figure 22-16. Differences in Level 3 Interrupt Acknowledgment According to IMC Register Setting**




**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.

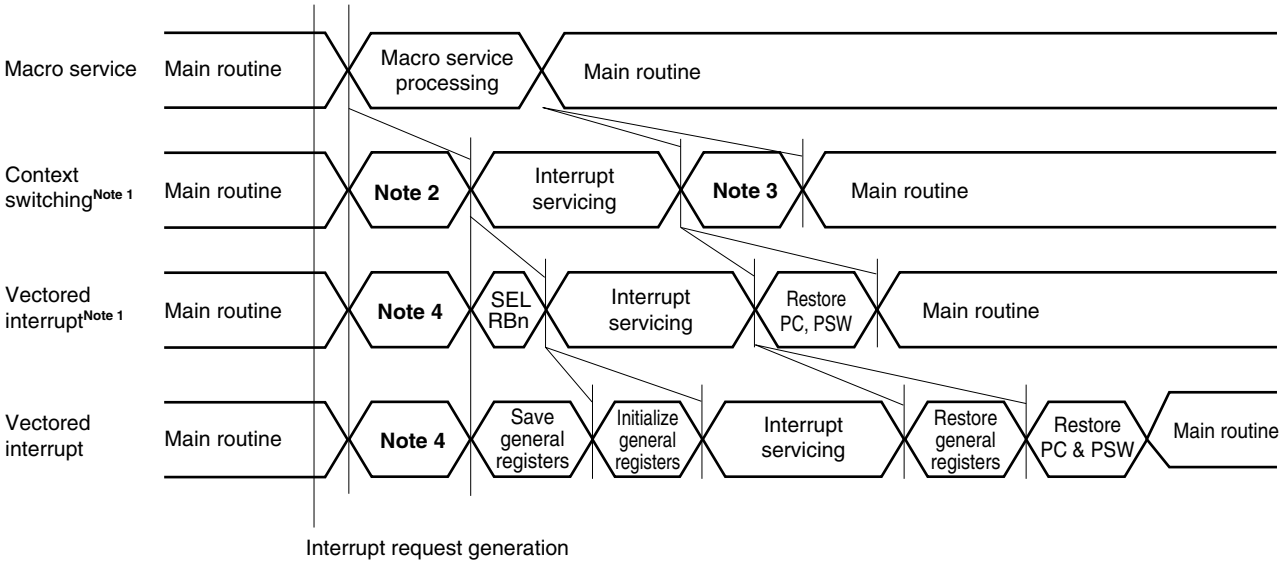
## 22.8 Macro Service Function

### 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 servicing, 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 22-17. Differences Between Vectored Interrupt and Macro Service Processing**




**Notes 1.** When register bank switching is used, and an initial value has been set in the register beforehand

**2.** Register bank switching by context switching, saving of PC and PSW

**3.** Register bank, PC and PSW restoration by context switching

**4.** PC and PSW saved to the stack, vector address loaded into PC

### 22.8.2 Types of macro servicing

Macro servicing can be used with the 23 kinds of interrupts shown in Table 22-6. There are four kinds of operations, selectable according to the application.

**Table 22-6. Interrupts for Which Macro Servicing Can Be Used**

| Default Priority | Interrupt Request Generation Source | Generating Unit | Macro Service Control Word Address |
|---|---|---|---|
| 0 | INTWDTM (Watchdog timer overflow) | Watchdog timer | 0FE06H |
| 1 | INTP0 (Pin input edge detection) | Edge detection | 0FE08H |
| 2 | INTP1 (Pin input edge detection) | | 0FE0AH |
| 3 | INTP2 (Pin input edge detection) | | 0FE0CH |
| 4 | INTP3 (Pin input edge detection) | | 0FE0EH |
| 5 | INTP4 (Pin input edge detection) | | 0FE10H |
| 6 | INTP5 (Pin input edge detection) | | 0FE12H |
| 7 | INTIIC0 (CSI0 $I^2C$ bus transfer end)[Note] | Clocked serial interface | 0FE16H |
| | INTCSI0 (CSI0 3-wire transfer end) | | |
| 8 | INTSER1 (ASI1 UART reception error) | Asynchronous serial interface/ clocked serial interface 1 | 0FE18H |
| 9 | INTSR1 (ASI1 UART reception end) | | 0FE1AH |
| | INTCSI1 (CSI1 3-wire transfer end) | | |
| 10 | INTST1 (ASI1 UART transmission end) | | 0FE1CH |
| 11 | INTSER2 (ASI2 UART reception error) | Asynchronous serial interface/ clocked serial interface 2 | 0FE1EH |
| 12 | INTSR2 (ASI2 UART reception end) | | 0FE20H |
| | INTCSI2 (CSI2 3-wire transfer end) | | |
| 13 | INTST2 (ASI2 UART transmission end) | | 0FE22H |
| 14 | INTTM3 (Reference time interval signal from watch timer) | Watch timer | 0FE24H |
| 15 | INTTM00 (Match signal generation of 16-bit timer register 0 and capture/compare register 00 (CR00)) | Timer/Counter | 0FE26H |
| 16 | INTTM01 (Match signal generation of 16-bit timer register 0 and capture/compare register 01 (CR01)) | | 0FE28H |
| 17 | INTTM1 (Match signal generation of 8-bit timer counter 1) | Timer/counter 1 | 0FE2AH |
| 18 | INTTM2 (Match signal generation of 8-bit timer counter 2) | Timer/counter 2 | 0FE2CH |
| 19 | INTAD (A/D converter conversion end) | A/D converter | 0FE2EH |
| 20 | INTTM5 (Match signal generation of 8-bit timer counter 5) | Timer/counter 5 | 0FE30H |
| 21 | INTTM6 (Match signal generation of 8-bit timer counter 6) | Timer/counter 6 | 0FE32H |
| 22 | INTWT (Watch timer overflow) | Watch timer | 0FE38H |

**Note** $\mu$PD784225Y Subseries only

**Remarks 1.** The default priority is a fixed number. This indicates the priority order when macro service requests are generated simultaneously.

**2.** ASI: Asynchronous serial interface
CSI: Clocked serial interface

There are four kinds of macro services, 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 0H 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 MB 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 servicing, not only are data transfers performed to two locations in response to a single interrupt request, but it is also possible to add output data ring control and a function that automatically adds data to a compare register. The entire 1 MB memory space can be used.

Type C is mainly used with the INTTM1 and INTTM2 interrupts, and is used for stepper motor control, etc., by macro service, with RTBL or RTBH, CR10, and CR20 used as the SFRs to which data is transferred.

**(4) Counter mode**

This mode is used to decrement the macro service counter (MSC) when an interrupt occurs and is used to count the division operation of an interrupt and interrupt generator.

When MSC is 0, a vectored interrupt can be generated.

To restart the macro service, MSC must be set again.

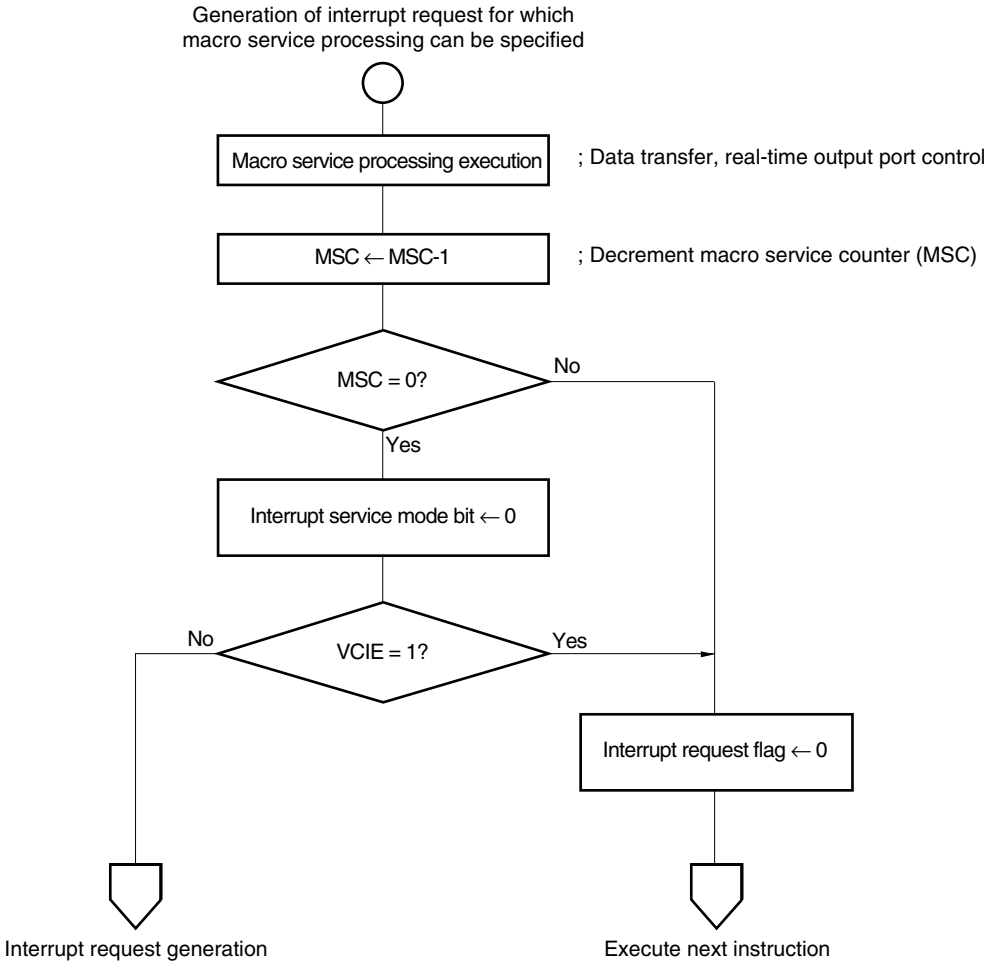
MSC is fixed to 16 bits and cannot be used as an 8-bit counter.

### 22.8.3 Basic macro service operation

Interrupt requests for which the macro service processing generated by the algorithm shown in Figure 22-11 can be specified are basically serviced in the sequence shown in Figure 22-18.

Interrupt requests for which macro service processing can be specified are not affected by the status of the IE flag, but are disabled by setting (1) an interrupt mask flag in the interrupt mask register (MK0). Macro service processing can be executed in the interrupt disabled state and during execution of an interrupt service program.

**Figure 22-18. Macro Service Processing Sequence**




The macro service type and transfer direction are determined by the value set in the macro service control word mode register. Transfer processing is then performed using the macro service channel specified by the channel pointer according to the macro service type.

The macro service channel is memory which contains the macro service counter which records the number of transfers, the transfer destination and transfer source pointers, and data buffers, and can be located at any address in the range FE00H to FEFFH when the LOCATION 0H 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 servicing, processing is performed the number of times specified during execution of another program. The 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.

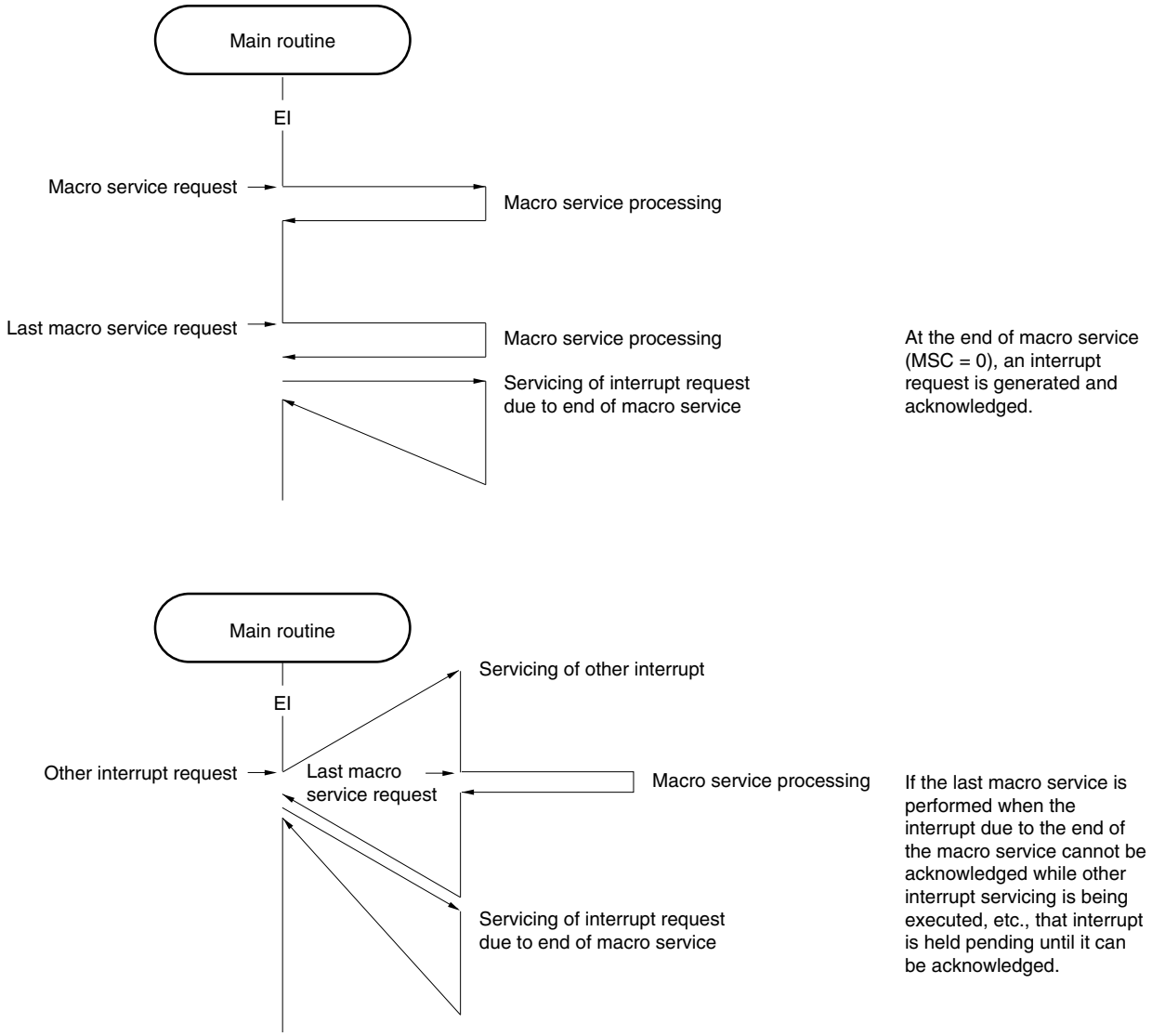
**(1) When VCIE bit is 0**

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, etc. It is mainly used in the following cases.

- Asynchronous serial interface receive data buffering (INTSR1/INTSR2)
- 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, INTTM1, INTTM2, INTTM5, and INTTM6)

**Figure 22-19. Operation at End of Macro Service When VCIE = 0**



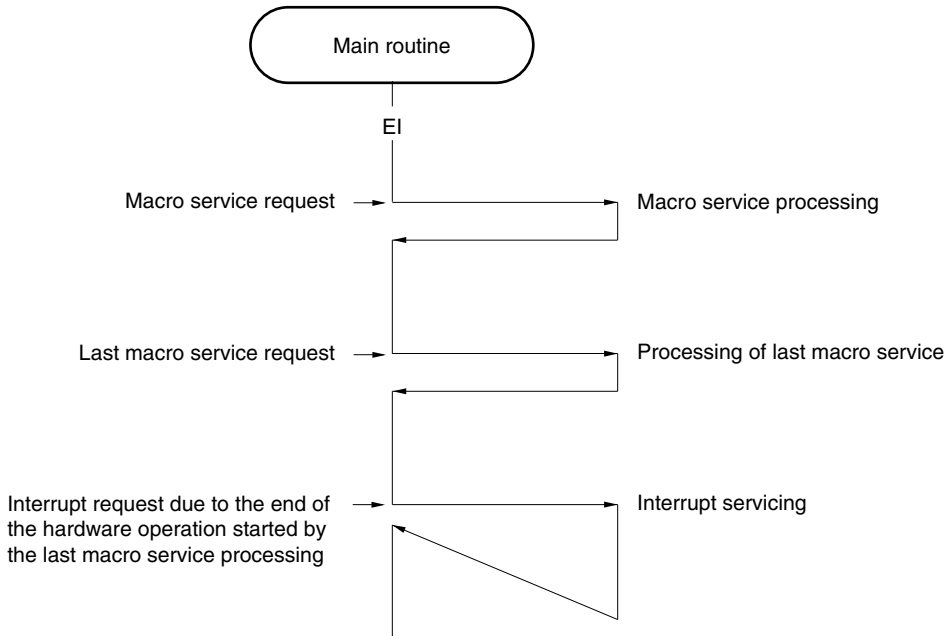
**(2) When VCIE bit is 1**

In this mode, an interrupt is not generated after macro servicing ends. Figure 22-20 shows an example of macro service and interrupt acknowledgment operations when the VCIE bit is 1.

This mode is used when the final operation is to be started by the last macro service processing performed, etc. It is mainly used in the following cases.

- Clocked serial interface receive data transfers (INTCSI0, INTCSI1/INTCSI2)
- Asynchronous serial interface data transfers (INTST1, INTST2)
- To stop a stepper motor in the case of stepping motor control by means of macro service type C using the real-time output port and timer/counter (INTTM1, INTTM2).

**Figure 22-20. Operation at End of Macro Service When VCIE = 1**



### 22.8.5 Macro service control registers

**(1) Macro service control word**

The μPD784225'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 22-21 for each macro service as the macro service control word.

When macro service processing is performed, the macro service mode register and channel pointer values corresponding to the interrupt requests for which macro service processing can be specified must be set beforehand.

★ **Figure 22-21. Format of Macro Service Control Word**

| Reserved word | Address | | Source |
|---|---|---|---|
| WTCHP | 0FE39H | Channel pointer | INTWT |
| WTMMD | 0FE38H | Mode register | |
| CCHP6 | 0FE33H | Channel pointer | INTTM6 |
| CMMD6 | 0FE32H | Mode register | |
| CCHP5 | 0FE31H | Channel pointer | INTTM5 |
| CMMD5 | 0FE30H | Mode register | |
| ADCHP | 0FE2FH | Channel pointer | INTAD |
| ADMMD | 0FE2EH | Mode register | |
| CCHP2 | 0FE2DH | Channel pointer | INTTM2 |
| CMMD2 | 0FE2CH | Mode register | |
| CCHP1 | 0FE2BH | Channel pointer | INTTM1 |
| CMMD1 | 0FE2AH | Mode register | |
| CCHP01 | 0FE29H | Channel pointer | INTTM01 |
| CMMD01 | 0FE28H | Mode register | |
| CCHP00 | 0FE27H | Channel pointer | INTTM00 |
| CMMD00 | 0FE26H | Mode register | |
| CCHP3 | 0FE25H | Channel pointer | INTTM3 |
| CMMD3 | 0FE24H | Mode register | |
| STCHP2 | 0FE23H | Channel pointer | INTST2 |
| STMMD2 | 0FE22H | Mode register | |
| SRCHP2/CSICHP2 | 0FE21H | Channel pointer | INTSR2/INTCSI2 |
| SRMMD2/CSIMMD2 | 0FE20H | Mode register | |
| SERCHP2 | 0FE1FH | Channel pointer | INTSER2 |
| SERMMD2 | 0FE1EH | Mode register | |
| STCHP1 | 0FE1DH | Channel pointer | INTST1 |
| STMMD1 | 0FE1CH | Mode register | |
| SRCHP1/CSICHP1 | 0FE1BH | Channel pointer | INTSR1/INTCSII |
| SRMMD1/CSIMMD1 | 0FE1AH | Mode register | |
| SERCHP1 | 0FE19H | Channel pointer | INTSER1 |
| SERMMD1 | 0FE18H | Mode register | |
| IICCHP[Note]/CSICHP0 | 0FE17H | Channel pointer | INTIIC0[Note]/INTCSI0 |
| IICMMD[Note]/CSIMMD0 | 0FE16H | Mode register | |
| PCHP5 | 0FE13H | Channel pointer | INTP5 |
| PMMD5 | 0FE12H | Mode register | |
| PCHP4 | 0FE11H | Channel pointer | INTP4 |
| PMMD4 | 0FE10H | Mode register | |
| PCHP3 | 0FE0FH | Channel pointer | INTP3 |
| PMMD3 | 0FE0EH | Mode register | |
| PCHP2 | 0FE0DH | Channel pointer | INTP2 |
| PMMD2 | 0FE0CH | Mode register | |
| PCHP1 | 0FE0BH | Channel pointer | INTP1 |
| PMMD1 | 0FE0AH | Mode register | |
| PCHP0 | 0FE09H | Channel pointer | INTP0 |
| PMMD0 | 0FE08H | Mode register | |
| WDTCHP | 0FE07H | Channel pointer | INTWDTM |
| WDTMMD | 0FE06H | Mode register | |

**Note** μPD784225Y Subseries only

**(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 22-21**).

The format of the macro service mode register is shown in Figure 22-22.

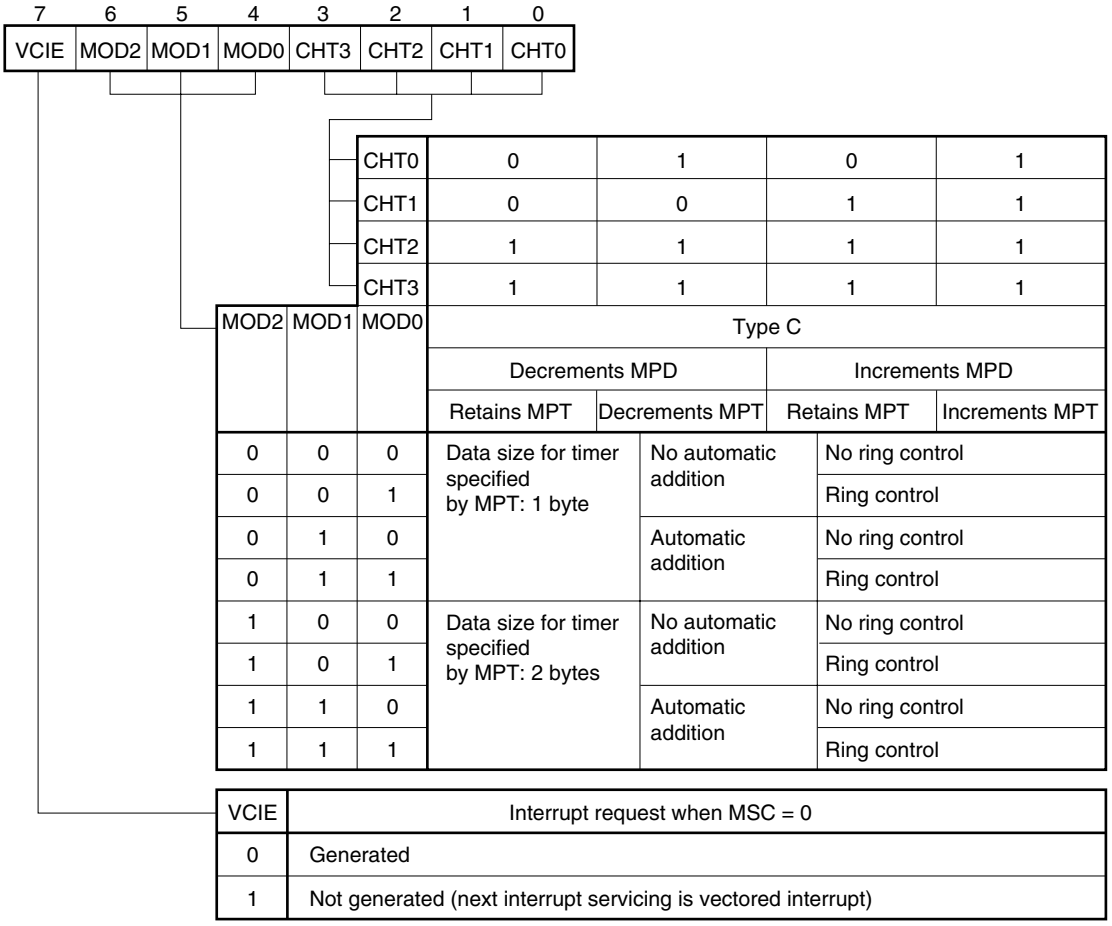
**Figure 22-22. Format of Macro Service Mode Register (1/2)**

| 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 22-22. Format of Macro Service Mode Register (2/2)**



| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| VCIE | MOD2 | MOD1 | MOD0 | CHT3 | CHT2 | CHT1 | CHT0 |

| | | | | |
|---|---|---|---|---|
| CHT0 | 0 | 1 | 0 | 1 |
| CHT1 | 0 | 0 | 1 | 1 |
| CHT2 | 1 | 1 | 1 | 1 |
| CHT3 | 1 | 1 | 1 | 1 |

| MOD2 | MOD1 | MOD0 | Type C | | | |
|---|---|---|---|---|---|---|
| | | | Decrements MPD | | Increments MPD | |
| | | | Retains MPT | Decrements MPT | Retains MPT | Increments MPT |
| 0 | 0 | 0 | Data size for timer specified by MPT: 1 byte | No automatic addition | No ring control | |
| 0 | 0 | 1 | | | Ring control | |
| 0 | 1 | 0 | | Automatic addition | No ring control | |
| 0 | 1 | 1 | | | Ring control | |
| 1 | 0 | 0 | Data size for timer specified by MPT: 2 bytes | No automatic addition | No ring control | |
| 1 | 0 | 1 | | | Ring control | |
| 1 | 1 | 0 | | Automatic addition | No ring control | |
| 1 | 1 | 1 | | | Ring control | |

| VCIE | Interrupt request when MSC = 0 |
|---|---|
| 0 | Generated |
| 1 | Not generated (next interrupt servicing 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 0H instruction is executed, or FFE00H to FFEFFH when the LOCATION 0FH instruction is executed, and the higher 16 bits of the address are fixed. Therefore, the lower 8 bits of the data stored to the highest address of the macro service channel are set in the macro service channel pointer.

### 22.8.6 Macro service type A

**(1) Operation**

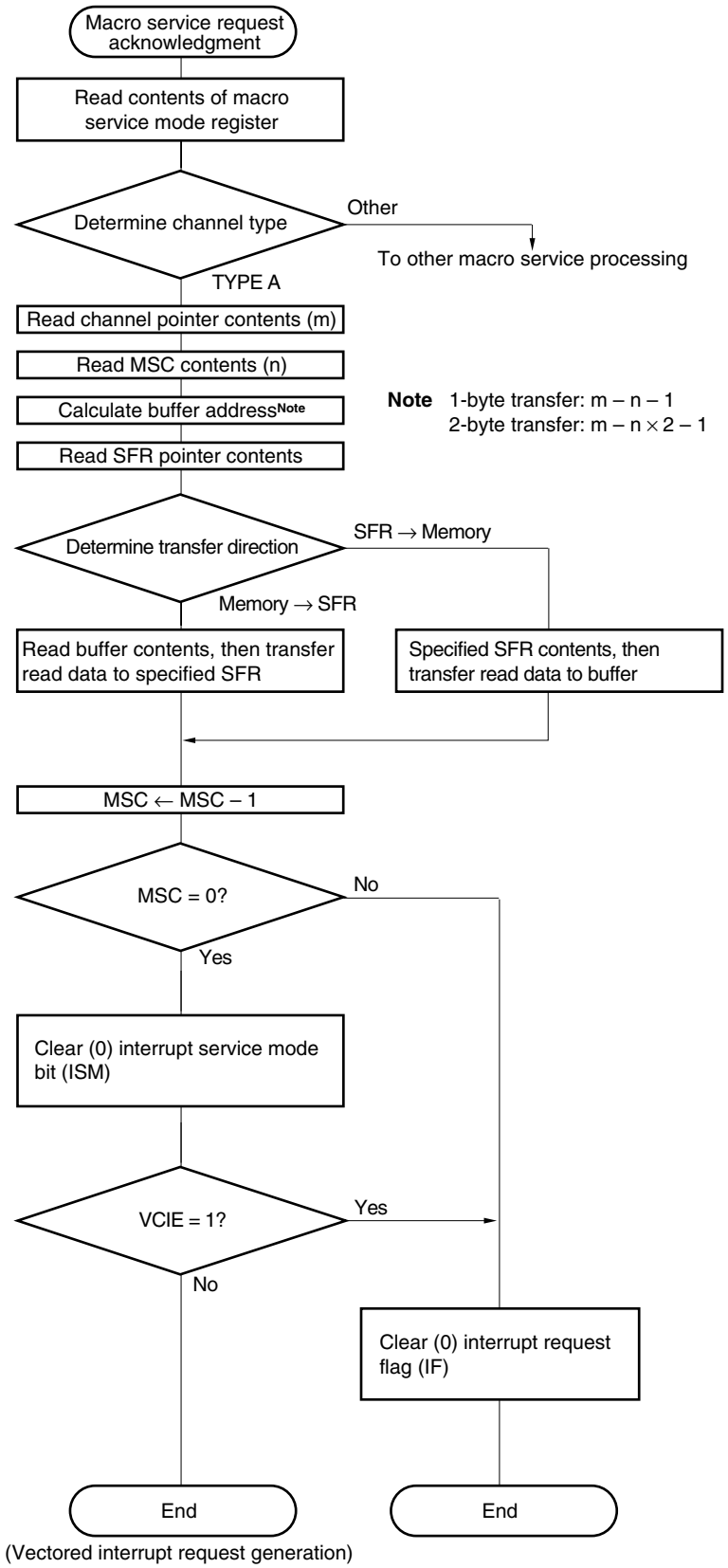
Data transfers are performed between buffer memory in the macro service channel and an SFR specified in the macro service channel.

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 22-23. Macro Service Data Transfer Processing Flow (Type A)**

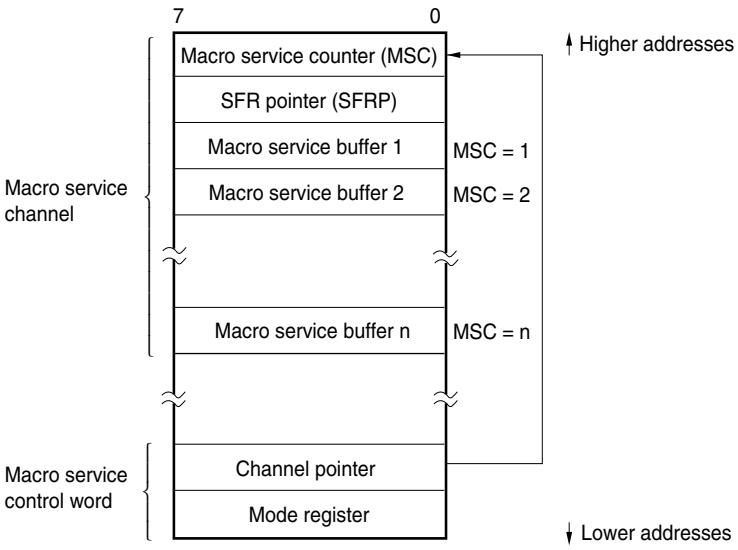


**(2) Macro service channel configuration**

The channel pointer and 8-bit macro service counter (MSC) indicate the buffer address in internal RAM (FE00H to FEFFH when the LOCATION 0H 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 22-24**). In the channel pointer, the lower 8 bits of the address are written to the macro service counter in the macro service channel. The SFR involved with the access is specified by the SFR pointer (SFRP). The lower 8 bits of the SFR address are written to SFRP.

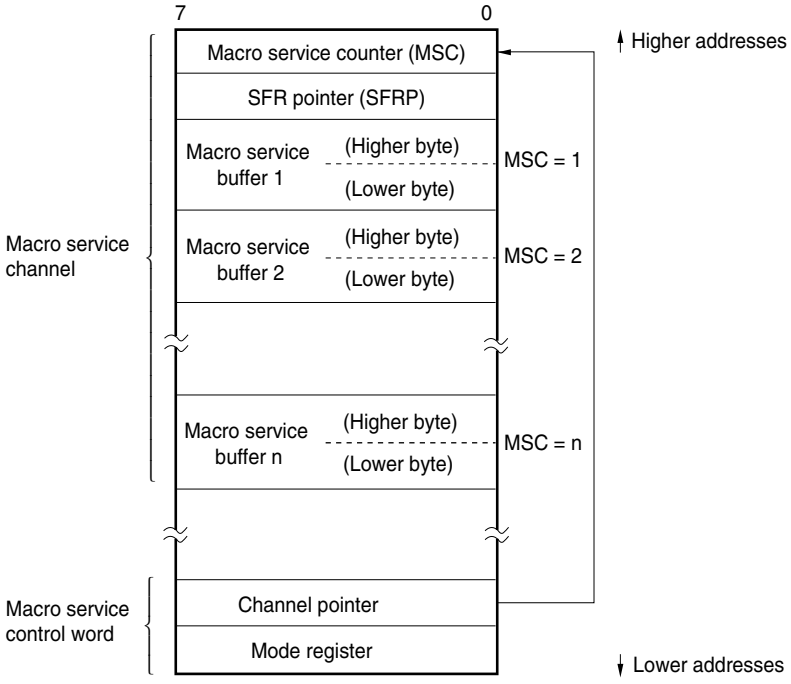
**Figure 22-24. Type A Macro Service Channel**

**(a) 1-byte transfers**




Macro service buffer address = (channel pointer) – (macro service counter) – 1

**(b) 2-byte transfers**

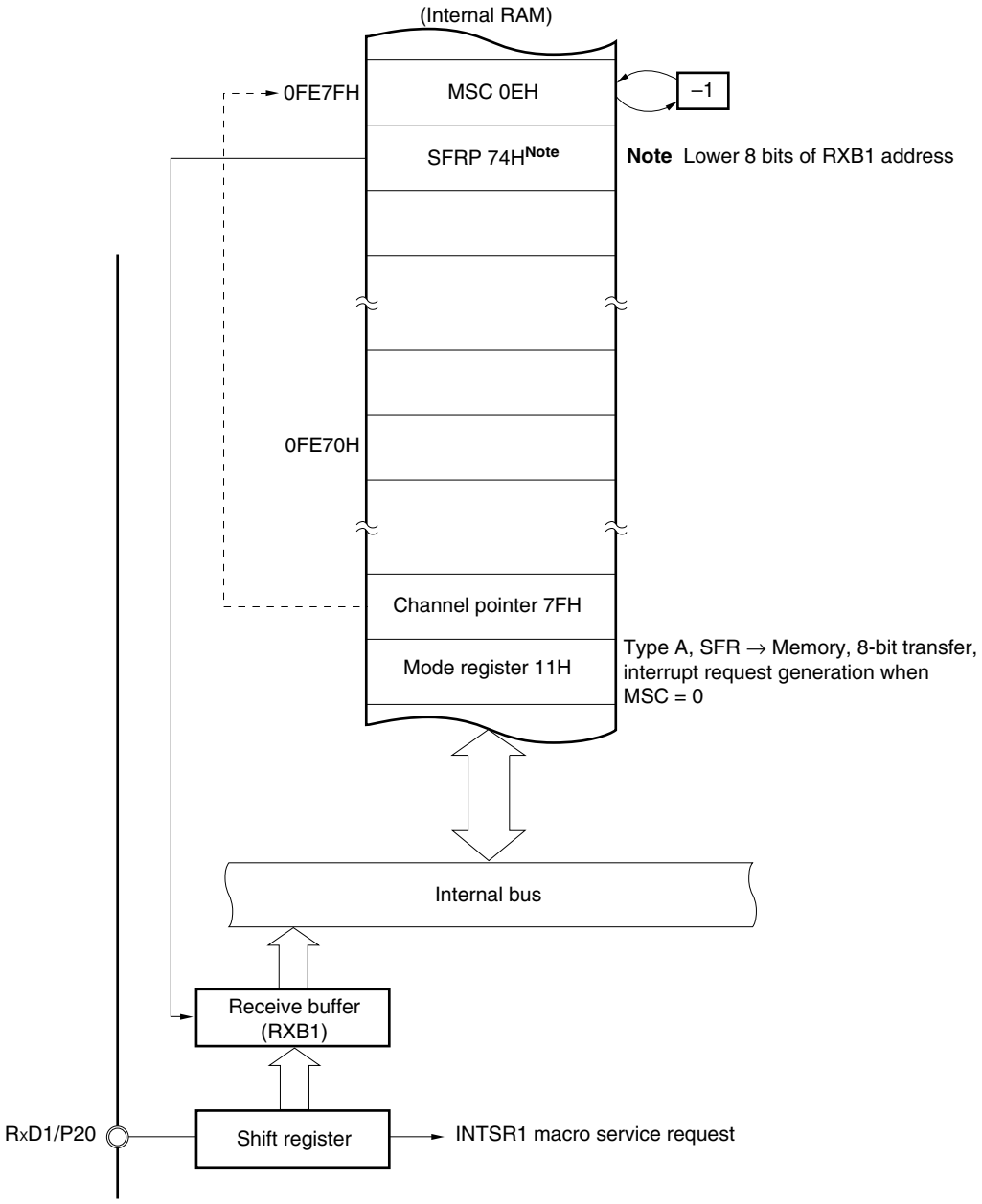



Macro service buffer address = (channel pointer) – (macro service counter) × 2 – 1

**(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 internal RAM.

**Figure 22-25. Asynchronous Serial Reception**




**Remark** Addresses in the figure are the values when the LOCATION 0H instruction is executed.
When the LOCATION 0FH instruction is executed, 0F0000H should be added to the values in the figure.

### 22.8.7 Macro service type B

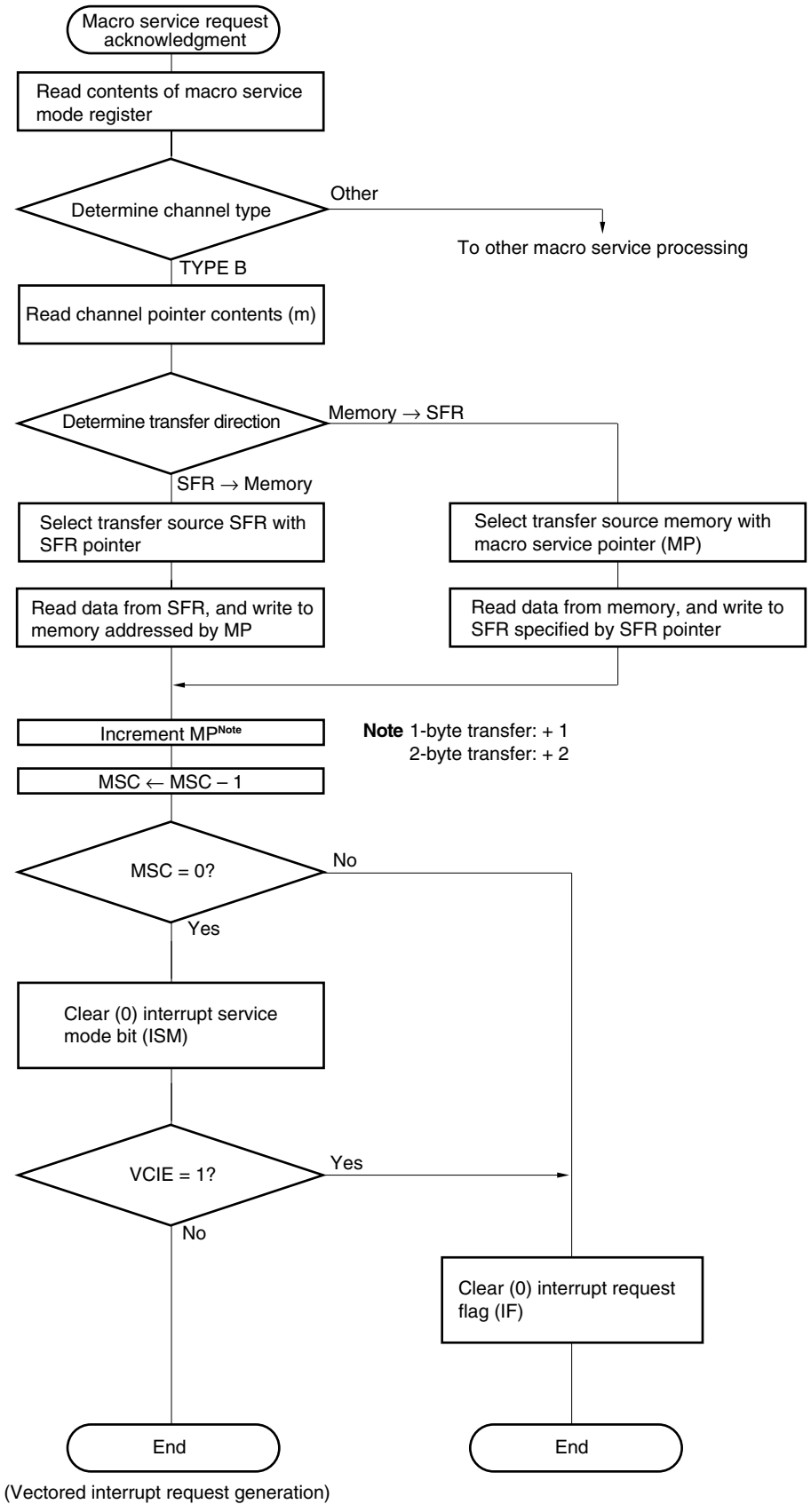
**(1) Operation**

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 KB of data buffer area when 8-bit data is transferred or 128 KB of data buffer area when 16-bit data is transferred can be set in any address space of 1 MB.

**Figure 22-26. Macro Service Data Transfer Processing Flow (Type B)**



**(2) Macro service channel configuration**

The macro service pointer (MP) indicates the data buffer area in the 1 MB memory space that is the transfer destination or transfer source.

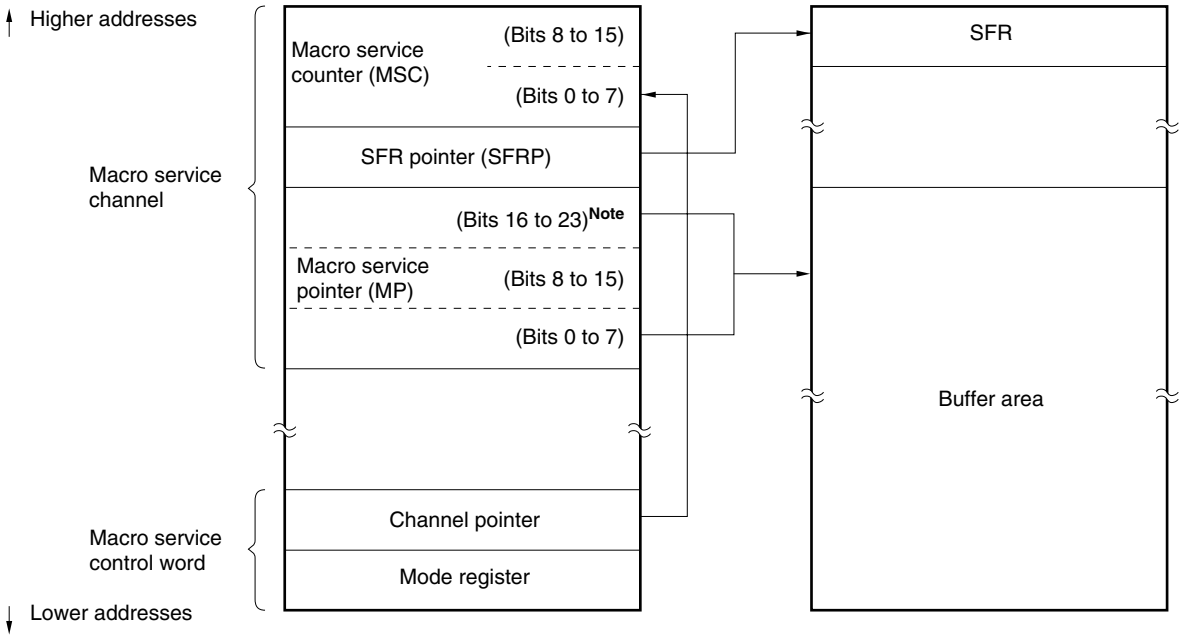
The lower 8 bits of the SFR that is the transfer destination or transfer source is written to the SFR pointer (SFRP).

The macro service counter (MSC) is a 16-bit counter that specifies the number of data transfers.

The macro service channel that stores the MP, SFRP and MSC is located in internal RAM space addresses 0FE00H to 0FEFFH when the LOCATION 0H 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 22-27. In the channel pointer, the lower 8 bits of the address are written to the macro service counter in the macro service channel.

**Figure 22-27. Type B Macro Service Channel**



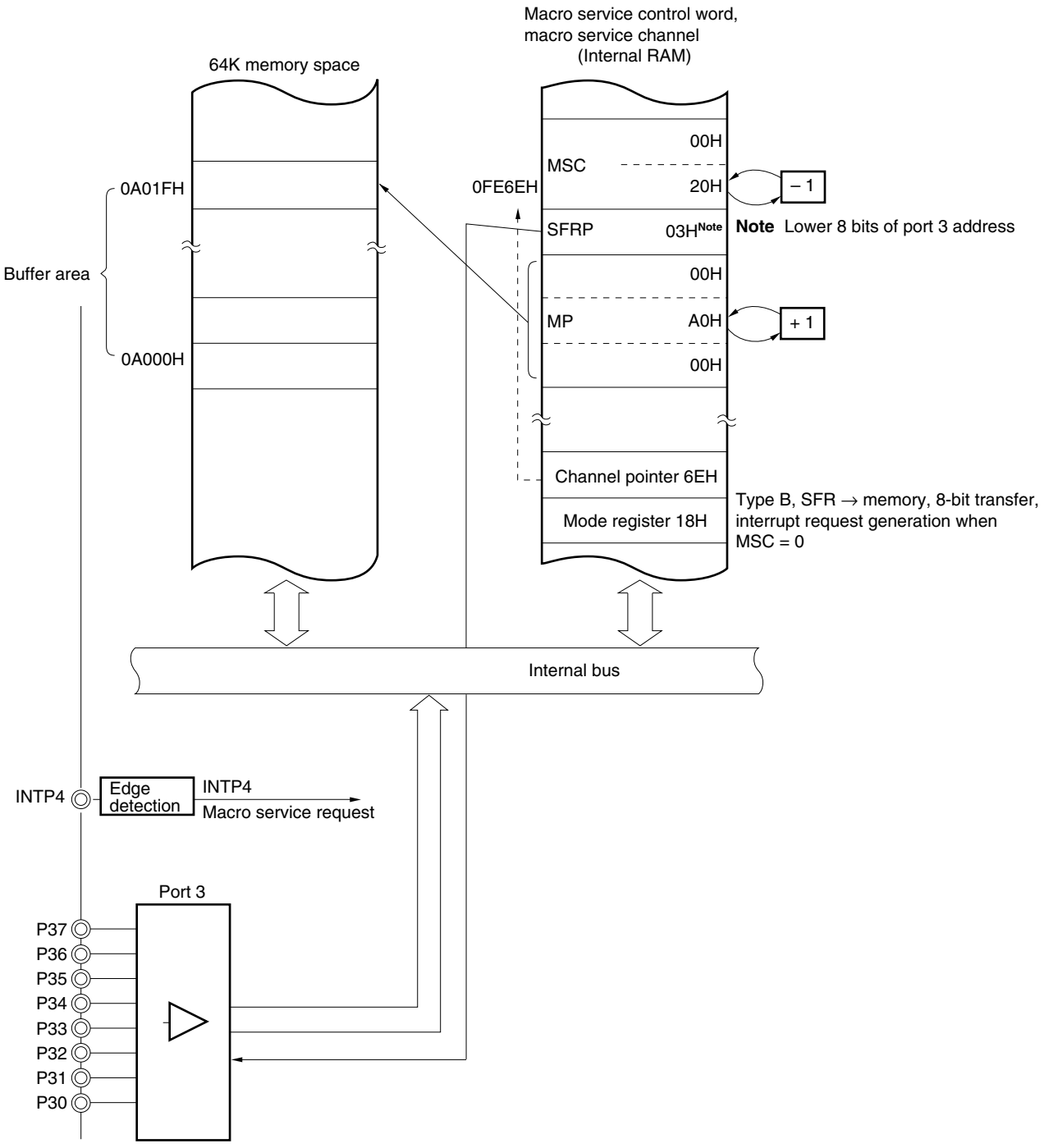

Macro service buffer address = macro service pointer

**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.

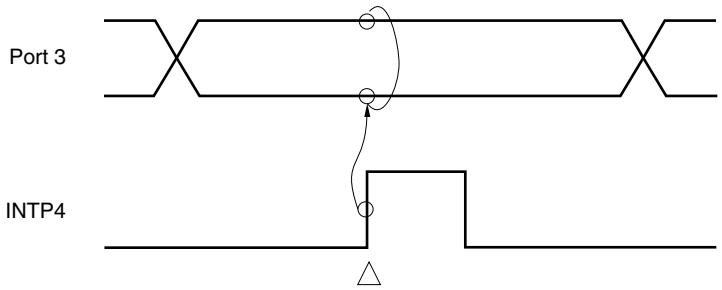
**Figure 22-28. Parallel Data Input Synchronized with External Interrupts**




**Remark** Macro service channel addresses in the figure are the values when the LOCATION 0H instruction is executed.

When the LOCATION 0FH instruction is executed, 0F0000H should be added to the values in the figure.

**Figure 22-29. Parallel Data Input Timing**



### 22.8.8 Macro service type C

**(1) Operation**

For the type C macro service, data in the memory specified by the macro service channel is transferred to two SFRs, for timer use and data use, specified by the macro service channel in response to a single interrupt request (the SFRs can be freely selected). An 8-bit or 16-bit timer SFR can be selected.

In addition to the basic data transfers described above, the following functions can be added to the 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. 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 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 22-30. Macro Service Data Transfer Processing Flow (Type C) (1/2)**

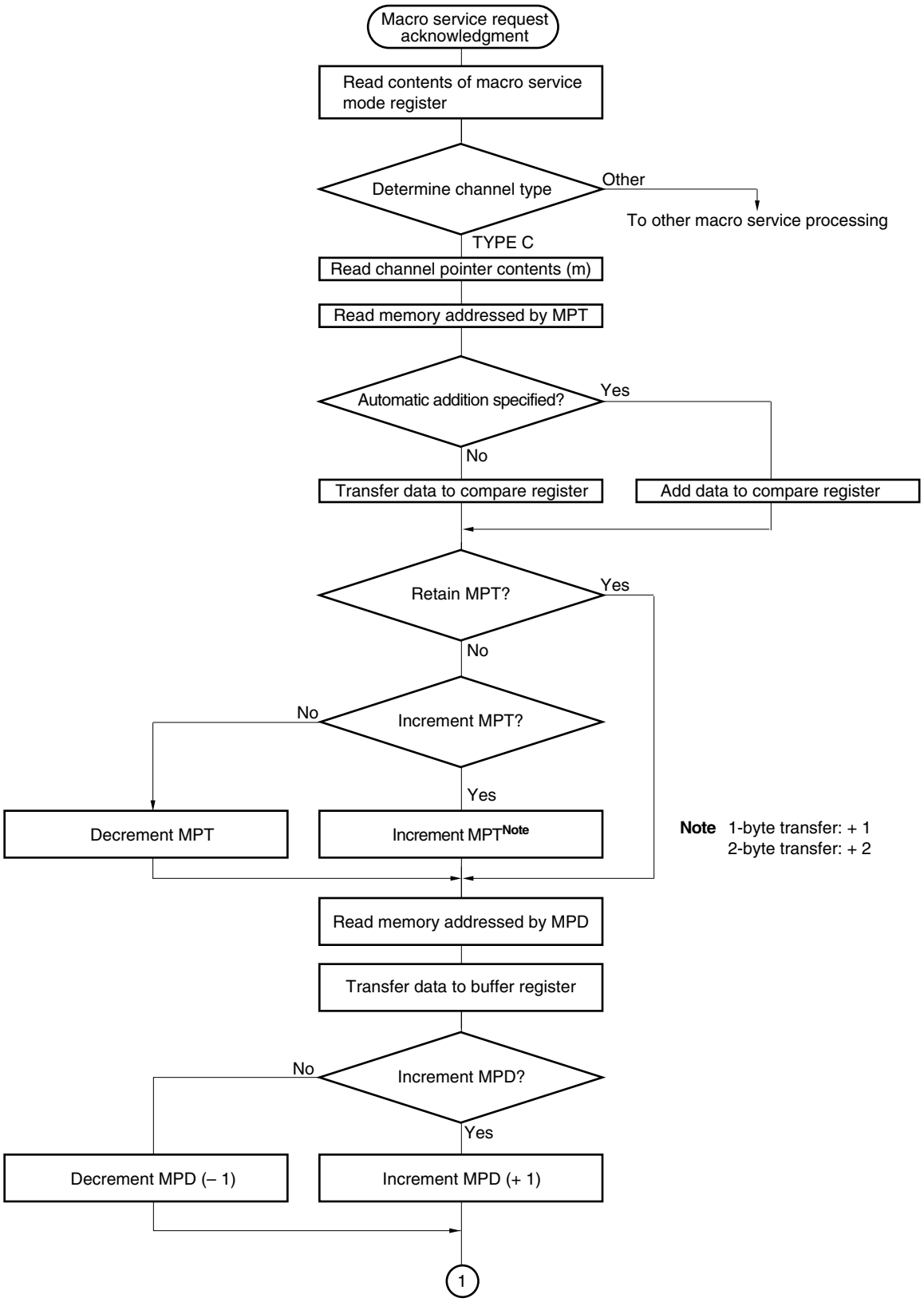
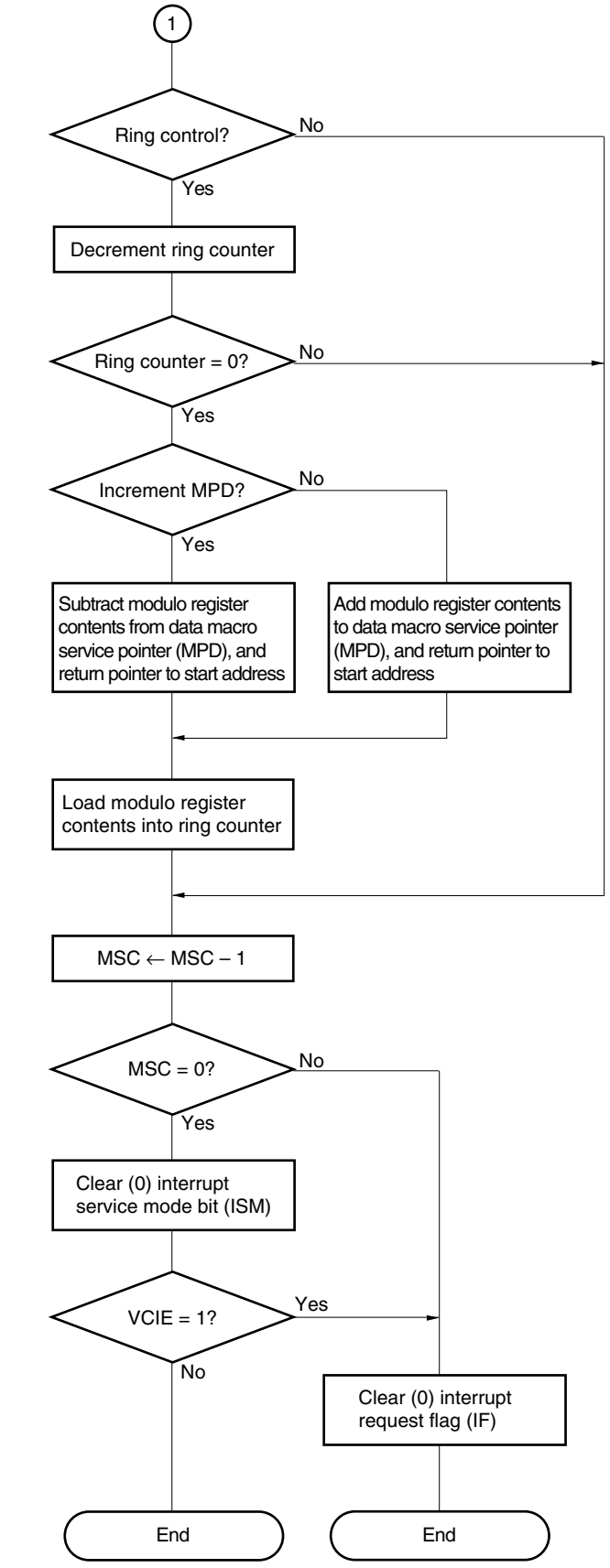


**Figure 22-30. Macro Service Data Transfer Processing Flow (Type C) (2/2)**




(Vectored interrupt request generation)

**(2) Macro service channel configuration**

There are two kinds of type C macro service channels, as shown in Figure 22-31.

The timer macro service pointer (MPT) mainly indicates the data buffer area in the 1 MB memory space to be transferred or added to the timer/counter compare register.

The data macro service pointer (MPD) indicates the data buffer area in the 1 MB 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.

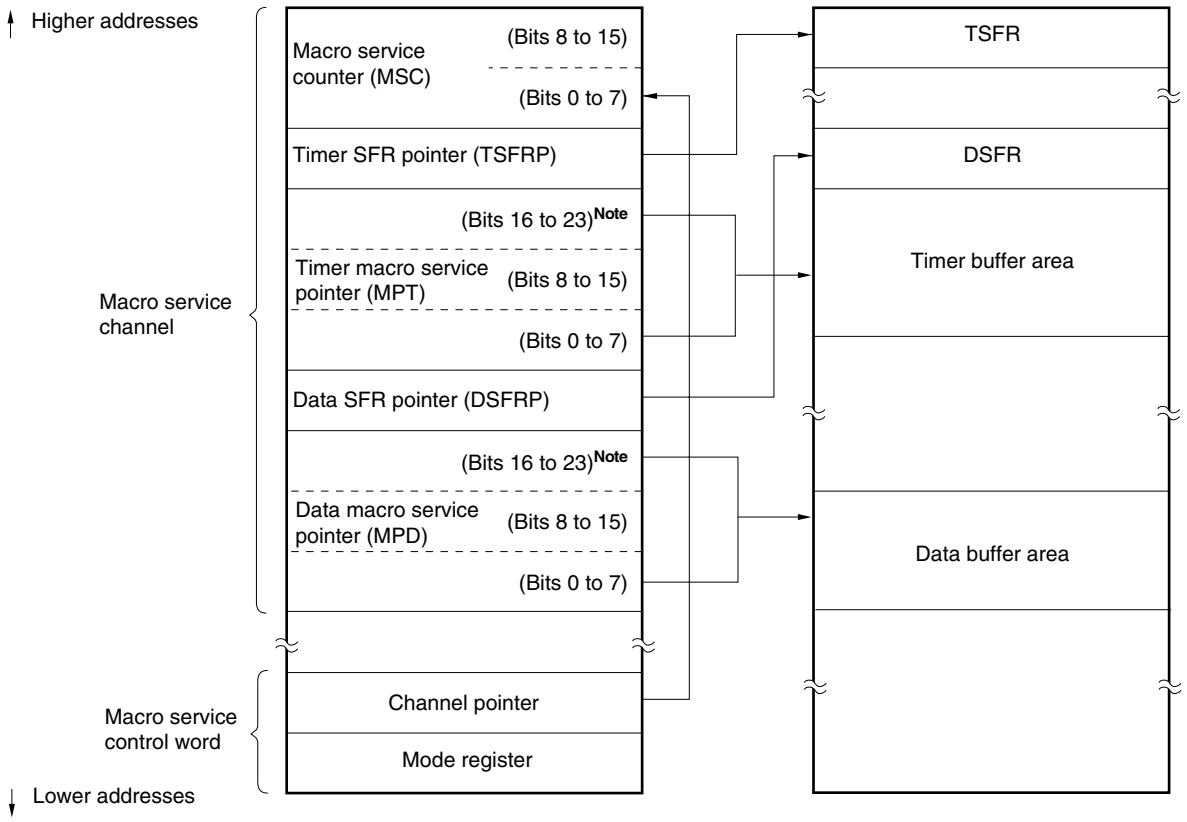
The macro service counter (MSC) is a 16-bit counter that specifies the number of data transfers.

The lower 8 bits of the SFR that is the transfer destination is written to the timer SFR pointer (TSFRP) and data SFR pointer (DSFRP).

The macro service channel that stores these pointers and counters is located in internal RAM space addresses 0FE00H to 0FEFFH when the LOCATION 0H 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 22-31. In the channel pointer, the lower 8 bits of the address are written to the macro service counter in the macro service channel.

**Figure 22-31. Type C Macro Service Channel (1/2)**

**(a) No ring control**

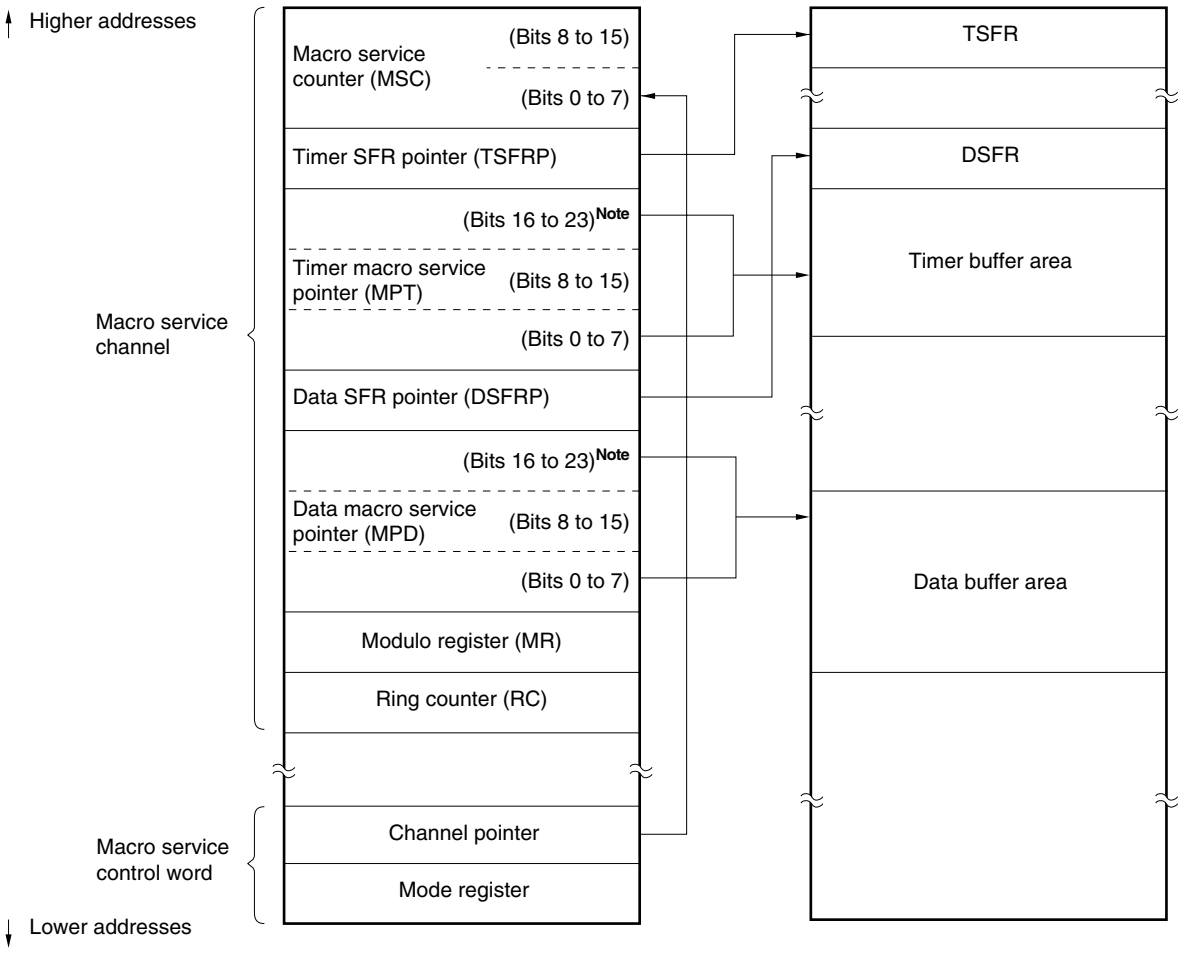



Macro service buffer address = macro service pointer

**Note** Bits 20 to 23 must be set to 0.

**Figure 22-31. 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.

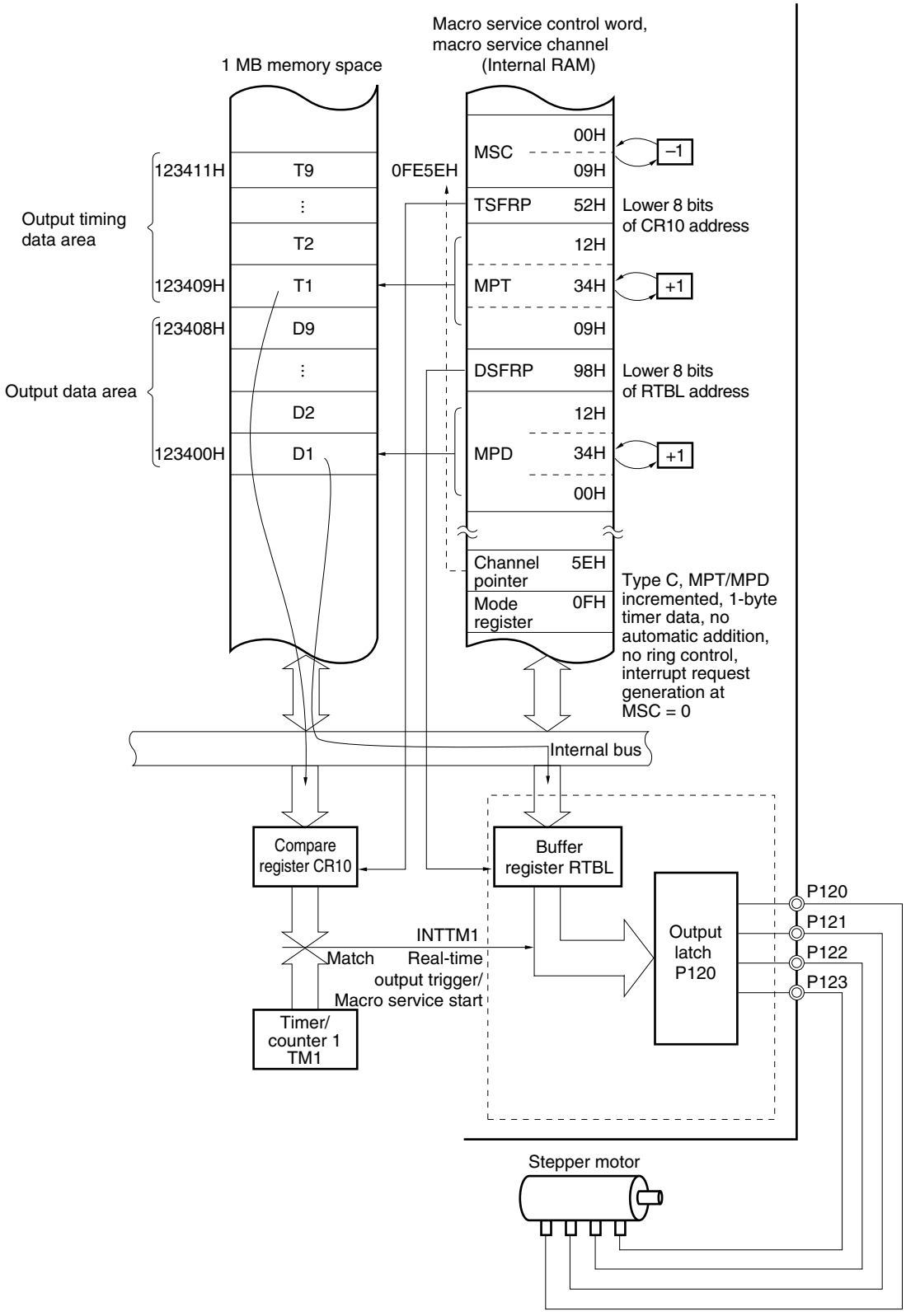
### (3) Examples of use of type C

#### (a) Basic operation

An example is shown below in which the output pattern to the real-time output port and the output interval are directly controlled.

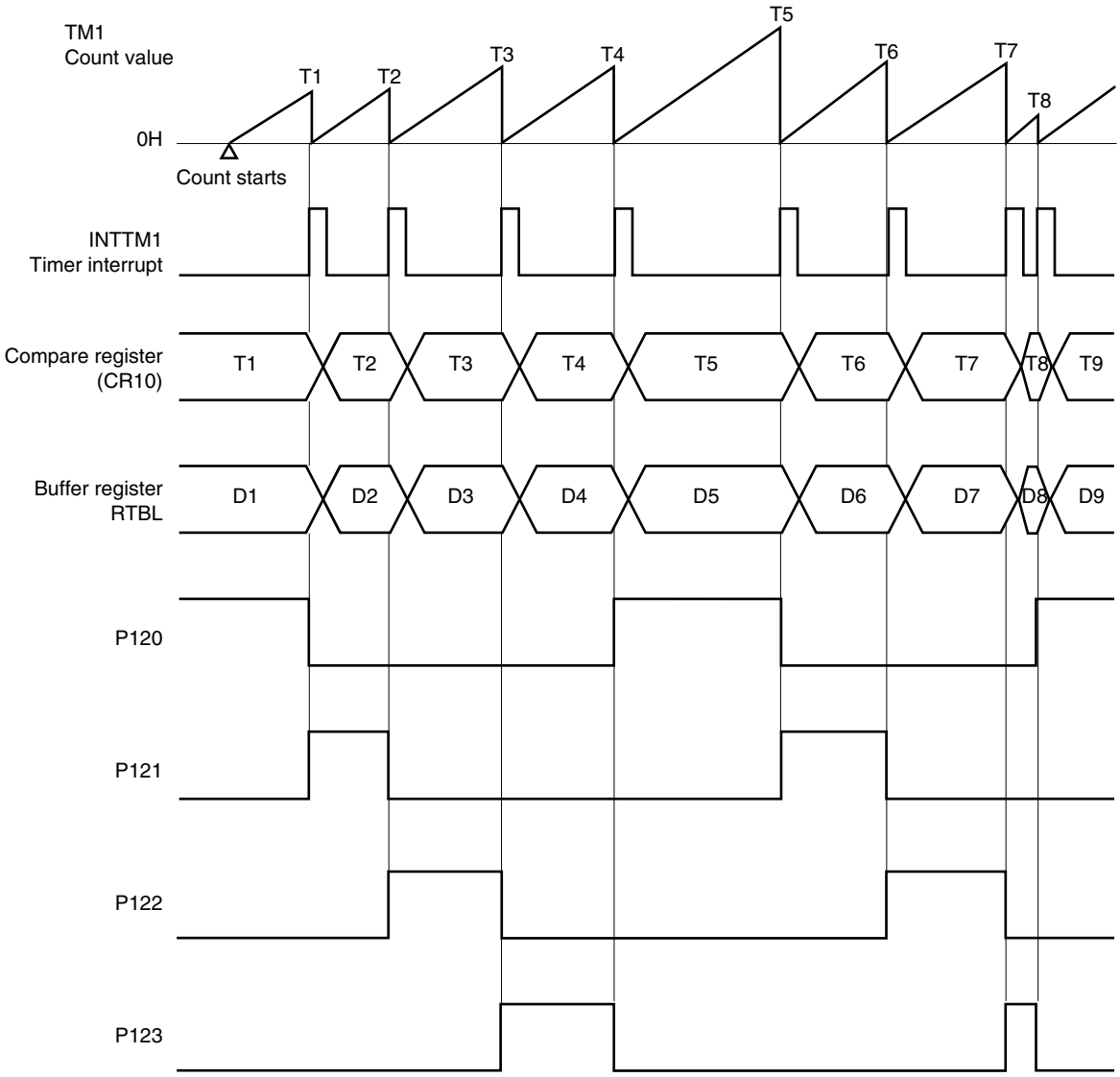
Update data is transferred from the two data storage areas set in the 1 MB space beforehand to the real-time output function buffer register (RTBL) and the compare register (CR10).

**Figure 22-32. Stepper Motor Open Loop Control by Real-Time Output Port**




**Remark** Internal RAM addresses in the figure are the values when the LOCATION 0H instruction is executed. When the LOCATION 0FH instruction is executed, 0F0000H should be added to the values in the figure.

**Figure 22-33. Data Transfer Control Timing**



### (b) Examples of use of automatic addition control and ring control

#### (i) Automatic addition control

The output timing data ($\Delta$t) specified by the macro service pointer (MPT) is added to the contents of the compare register, and the result is written back to the compare register.

Use of this automatic addition control eliminates the need to calculate the compare register setting value in the program each time.

#### (ii) Ring control

With ring control, predetermined output patterns are 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 to 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 stepper motor, for example, the output patterns will vary depending on the configuration of the stepper 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 stepper motor are shown in Figures 22-34 and 22-35.

**Figure 22-34. Single-Phase Excitation of 4-Phase Stepper Motor**

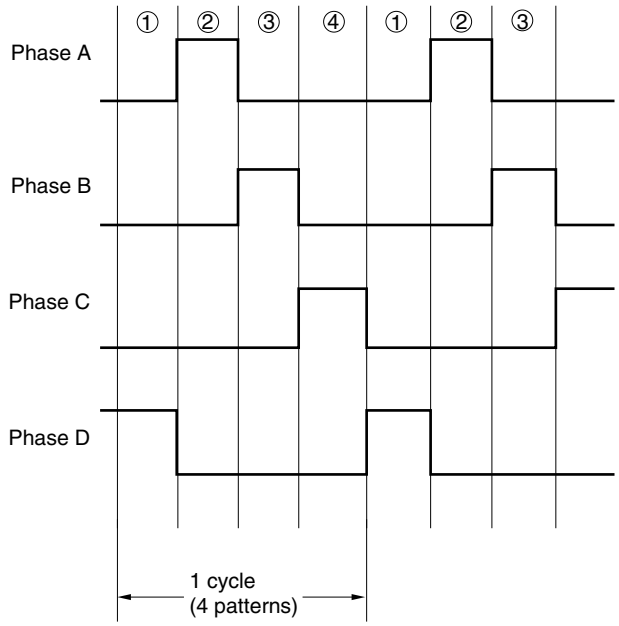



**Figure 22-35. 1-2-Phase Excitation of 4-Phase Stepper Motor**

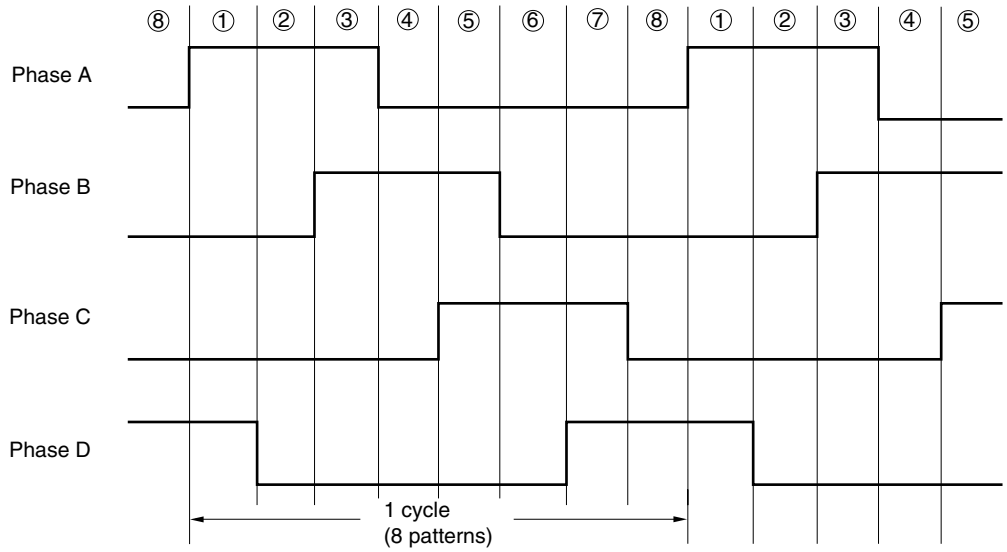
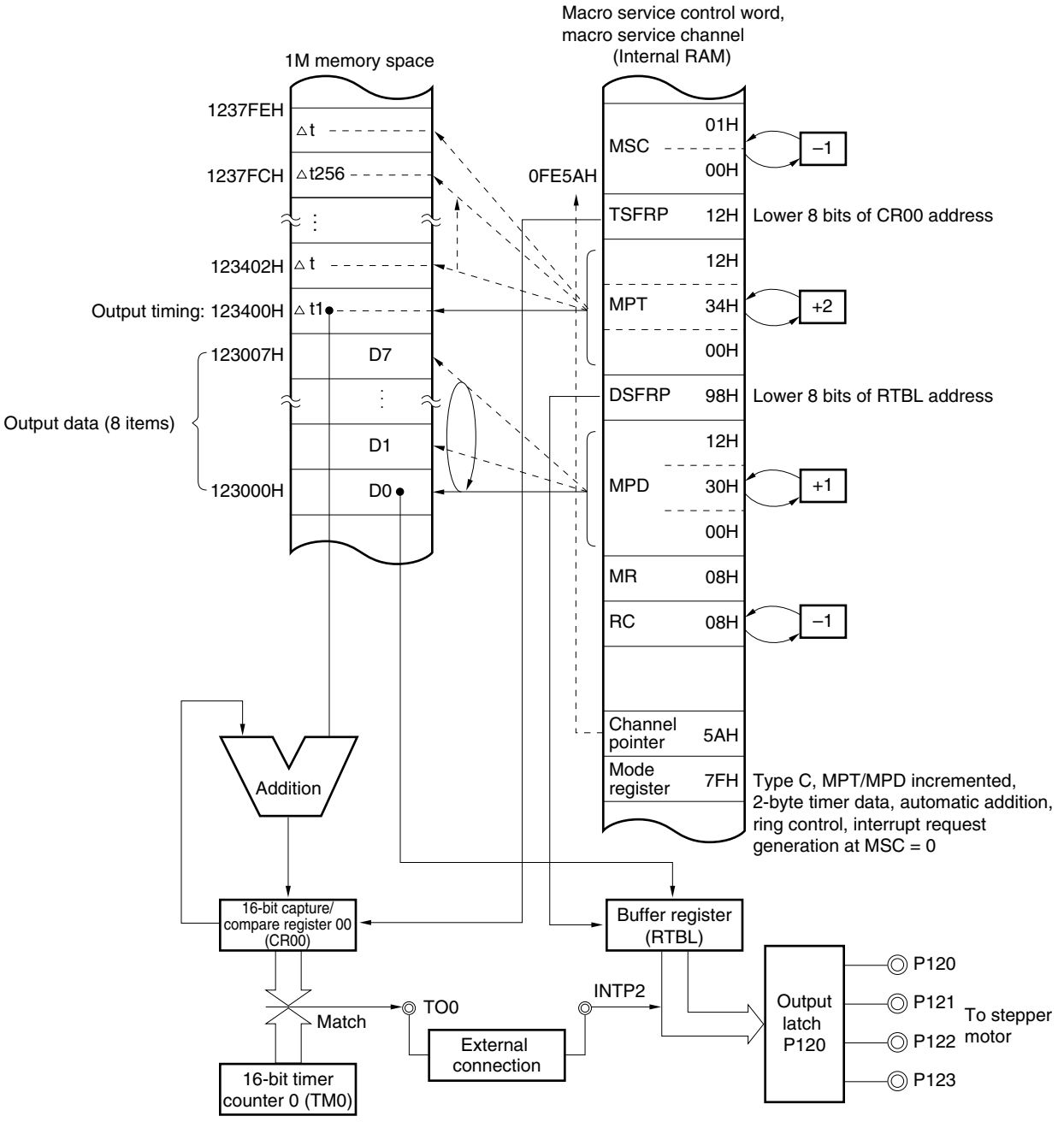
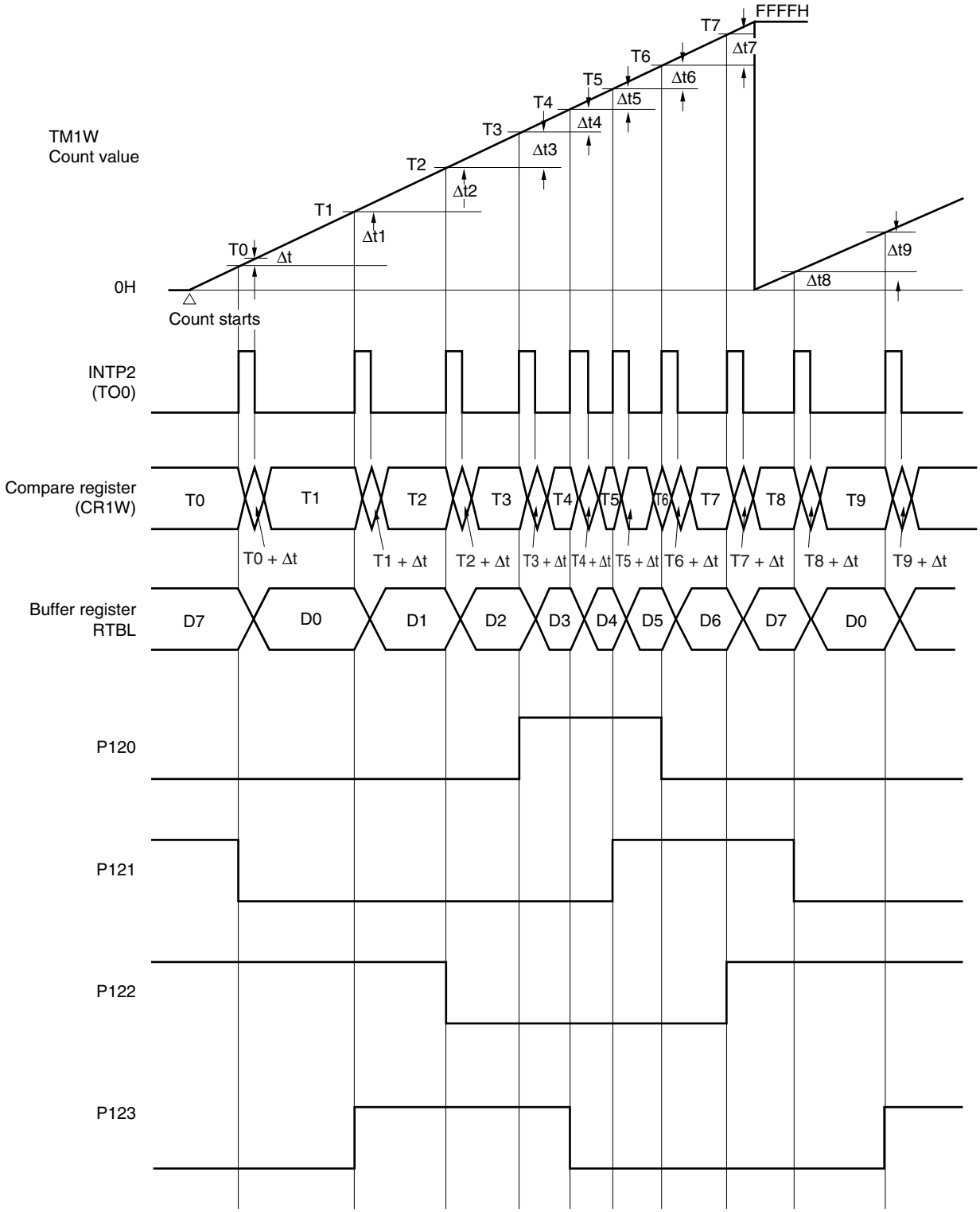


**Figure 22-36. Automatic Addition Control + Ring Control Block Diagram 1 (When Output Timing Varies with 1-2-Phase Excitation)**



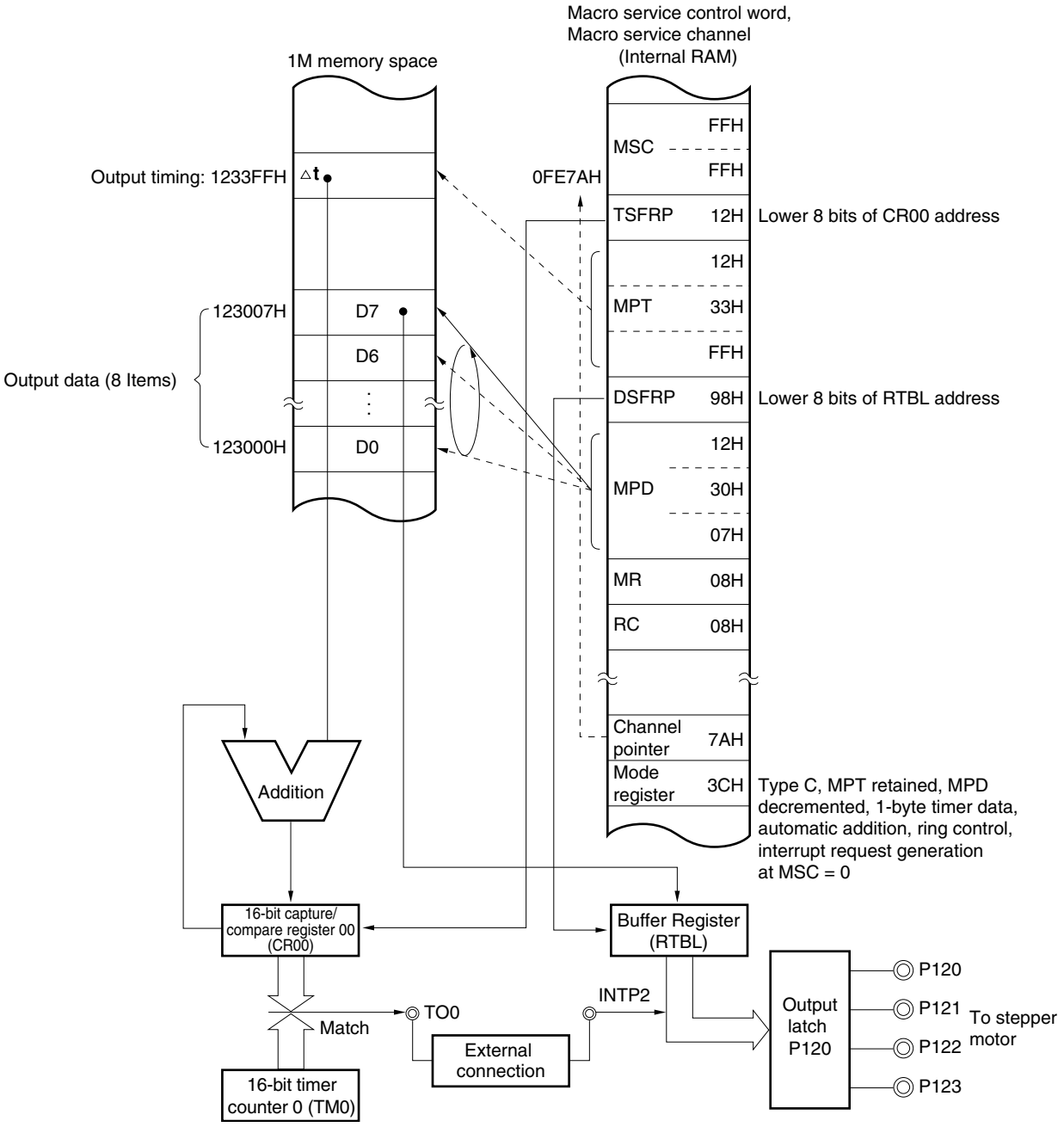

**Remark** Internal RAM addresses in the figure are the values when the LOCATION 0H instruction is executed. When the LOCATION 0FH instruction is executed, 0F0000H should be added to the values in the figure.

**Figure 22-37. Automatic Addition Control + Ring Control Timing Diagram 1 (When Output Timing Varies with 1-2-Phase Excitation)**



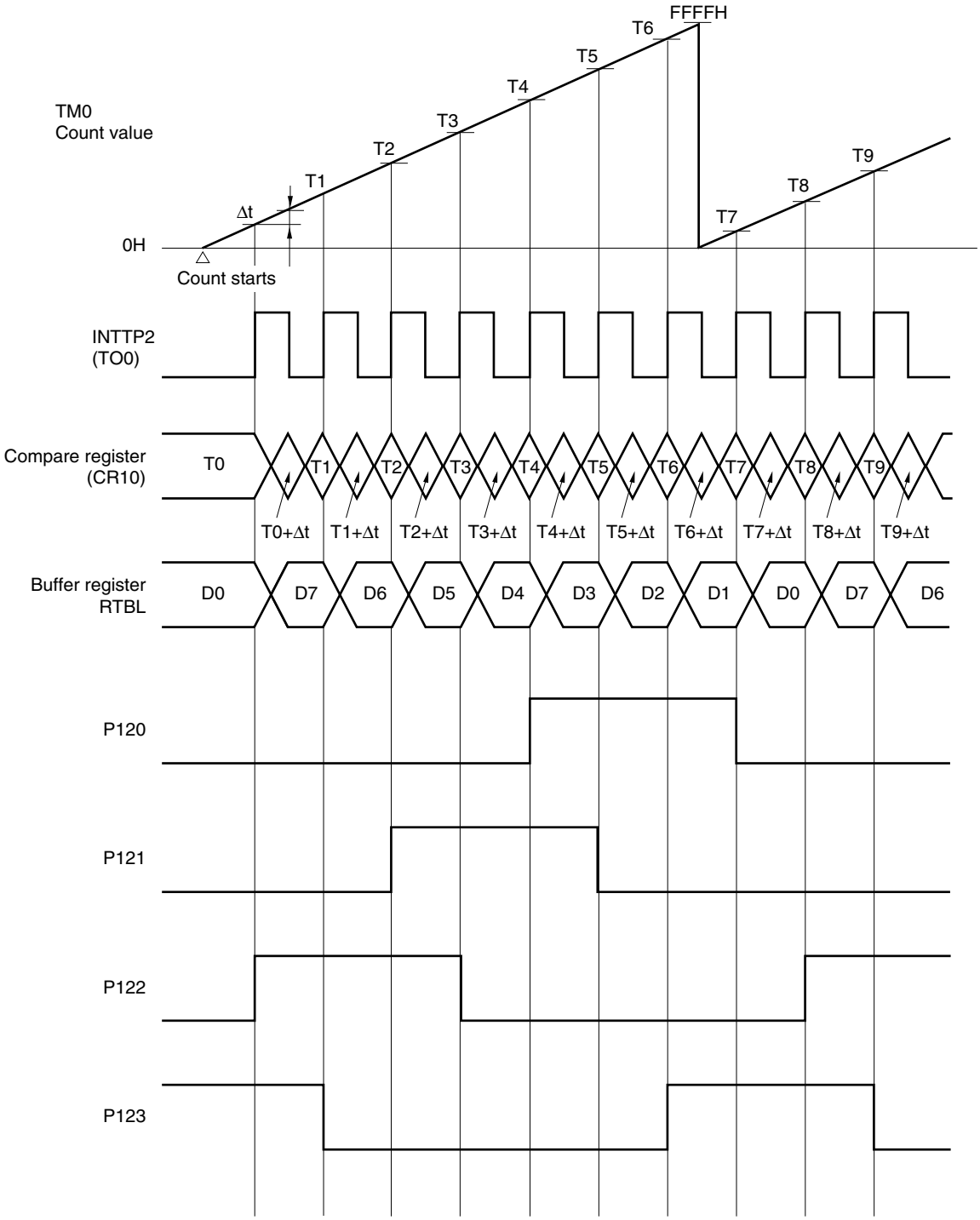

**Note** For the INTP2 high-/low-level width, refer to the data sheet.

**Figure 22-38. 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 0H instruction is executed. When the LOCATION 0FH instruction is executed, 0F0000H should be added to the values in the figure.

**Figure 22-39. Automatic Addition Control + Ring Control Timing Diagram 2 (1-2-Phase Excitation Constant-Velocity Operation)**




**Note** For the INTP2 high-/low-level width, refer to the data sheet.

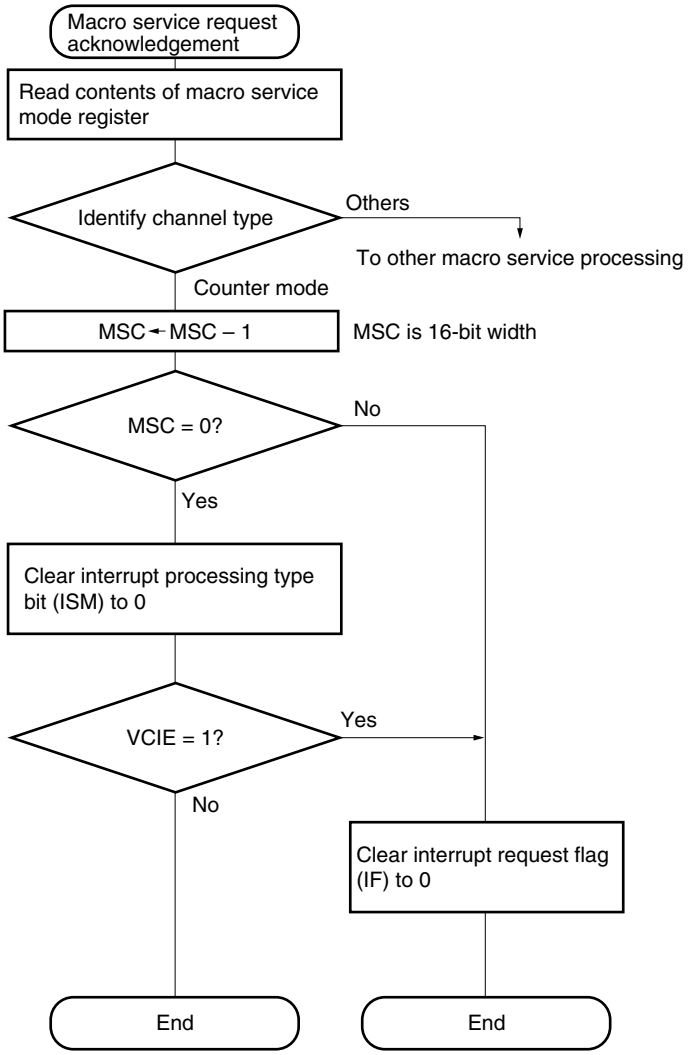
### 22.8.9 Counter mode

**(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 22-40. Macro Service Data Transfer Processing Flow (Counter Mode)**

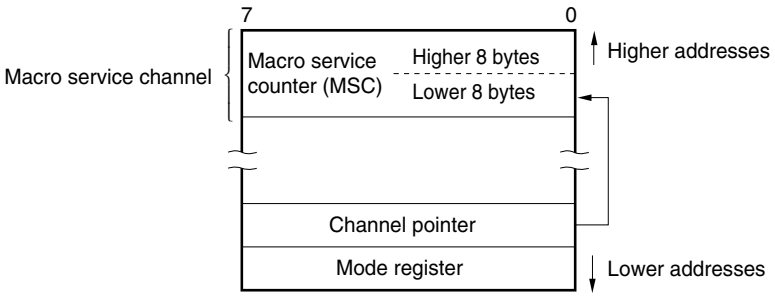



(Vectored interrupt request is generated)

**(2) Configuration of macro service channel**

The macro service channel consists of only a 16-bit macro service counter (MSC). The lower 8 bits of the address of the MSC are written to the channel pointer.

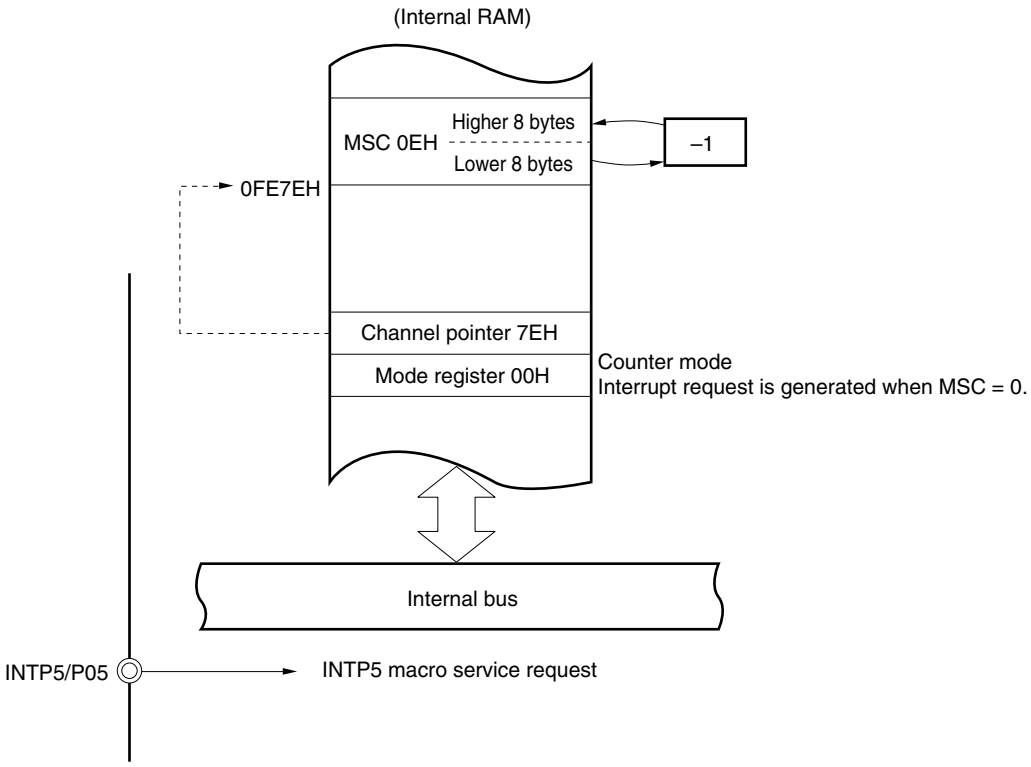
**Figure 22-41. Counter Mode**




**(3) Example of using counter mode**

Here is an example of counting the number of edges input to external interrupt pin INTP5.

**Figure 22-42. Counting Number of Edges**




**Remark** The internal RAM address in the figure above is the value when the LOCATION 0H instruction is executed.

When the LOCATION 0FH instruction is executed, add 0F0000H to this value.

## 22.9 When Interrupt Requests and Macro Service Are Temporarily Held Pending

When the following instructions are executed, interrupt acknowledgment and macro service processing are 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, ISPR, and SNMI.
PSW bit manipulation instruction
(Excluding the BT PSWL.bit, $addr20 instruction, BF PSWL.bit, $addr20 instruction, BT PSWH.bit, $addr20 instruction, BF PSWH.bit, $addr20 instruction, SET1 CY instruction, NOT1 CY instruction, and CLR1 CY instruction)

**Note** Interrupt control registers: WDTIC, PIC0, PIC1, PIC2, PIC3, PIC4, PIC5, CSIIC0, SERIC1, SRIC1, STIC1, SERIC2, SRIC2, STIC2, TMIC3, TMIC00, TMIC01, TMIC1, TMIC2, ADIC, TMIC5, TMIC6, WTIC

**Caution If problems are caused by a long pending period for interrupts and macro servicing when the corresponding instructions are used in succession, a time at which interrupts and macro service requests can be acknowledged should be provided by inserting an NOP instruction, etc., in the series of instructions.**

## 22.10 Instructions Whose Execution Is Temporarily Suspended by Interrupt or Macro Service

Execution of the following instructions is temporarily suspended by an acknowledgeable interrupt request or macro service request, and the interrupt or macro service request is acknowledged. The suspended instruction is resumed after completion of the interrupt service program or macro service processing.

Temporarily suspended instructions:

MOVM, XCHM, MOVBK, XCHBK
CMPME, CMPMNE, CMPMC, CMPMNC
CMPBKE, CMPBKNE, CMPBKC, CMPBKNC
SACW

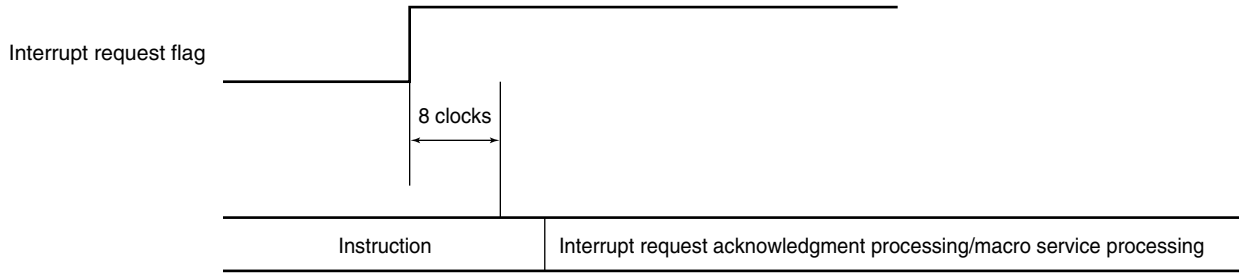
## 22.11 Interrupt and Macro Service Operation Timing

Interrupt requests are generated by hardware. The generated interrupt request sets (1) an interrupt request flag.

When the interrupt request flag is set (1), it takes of 8 clocks (0.64 $\mu$s: $f_{XX}$ = 12.5 MHz) 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 holds interrupts and macro servicing pending, the interrupt request is acknowledged after the following instruction (refer to **22.9 When Interrupt Requests and Macro Service Are Temporarily Held Pending** for these instructions).

**Figure 22-43. Interrupt Request Generation and Acknowledgment (Unit: Clock = 1/$f_{CLK}$)**



### 22.11.1 Interrupt acknowledge processing time

The time shown in Table 22-7 is required to acknowledge an interrupt request. After the time shown in this table has elapsed, execution of the interrupt processing program is started.

**Table 22-7. Interrupt Acknowledge Processing Time**

(Unit: Clock = 1/$f_{CLK}$)

| Vector Table | IROM | | | | | | EMEM | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Branch destination | IROM, PRAM | | | EMEM | | | PRAM | | | EMEM | | |
| Stack | IRAM | PRAM | EMEM | IRAM | PRAM | EMEM | IRAM | PRAM | EMEM | IRAM | PRAM | EMEM |
| Vectored interrupts | 26 | 29 | 37 + 4n | 27 | 30 | 38 + 4n | 30 | 33 | 41 + 4n | 31 | 34 | 42 + 4n |
| Context switching | 22 | – | – | 23 | – | – | 22 | – | – | 23 | – | – |

**Remarks 1.** IROM: Internal ROM (with high-speed fetch specified)
PRAM: Peripheral RAM of internal RAM (only when LOCATION 0H instruction is executed in the case of branch destination)
IRAM: Internal high-speed RAM
EMEM: Internal ROM when external memory and high-speed fetch are not specified

**2.** n is the number of wait states per byte necessary for writing data to the stack (the number of wait states is the sum of the number of address wait states and the number of access wait states).

**3.** If the vector table is EMEM, and if wait states are inserted in reading the vector table, add 2 m to the value of the vectored interrupt in the above table, and add m to the value of context switching, where m is the number of wait states per byte necessary for reading the vector table.

**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

The macro service processing time differs depending on the type of the macro service, as shown in Table 22-8.

**Table 22-8. Macro Service Processing Time**

(Units: Clock = 1/$f_{CLK}$)

| Processing Type of Macro Service | | | Data Area | |
|---|---|---|---|---|
| | | | IRAM | Other |
| Type A | SFR → memory | 1 byte | 24 | – |
| | | 2 bytes | 25 | – |
| | Memory → SFR | 1 byte | 24 | – |
| | | 2 bytes | 26 | – |
| Type B | SFR → memory | | 33 | 35 |
| | Memory → SFR | | 34 | 36 |
| Type C | | | 49 | 53 |
| Counter mode | MSC ≠ 0 | | 17 | – |
| | 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: The 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. Add 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 μPD784225, 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
NEXT    :
```

- It is necessary to ensure that a non-maskable interrupt request is not generated via the NMI pin during execution of this program.
- After this, on-chip peripheral hardware initialization and interrupt control register initialization are performed.
- When interrupt control register initialization is performed, the interrupt request flags must be cleared (0).

## 22.13 Cautions

(1) The in-service priority register (ISPR) is read-only. Writing to this register may result in a 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 the BRK instruction. Use the RETB instruction.

(4) The RETCS instruction must not be used to return from a software interrupt caused by the BRKCS instruction. Use the RETCSB instruction.

(5) When a maskable interrupt is acknowledged by vectored interruption, the RETI instruction must be used to return from the interrupt. Subsequent interrupt-related operations will not be performed normally if a different instruction is used.

(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. To avoid macro service processing being performed during a non-maskable interrupt service program, 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 acknowledgement 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-priority non-maskable interrupt request is generated during execution of a low-priority non-maskable interrupt service program) and for a certain period after execution of the special instructions shown in **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 **3.9 Special Function Registers (SFRs)**), and the CPU becomes deadlocked, or an unexpected signal output from a pin, or PC and PSW are written to an address is which RAM is not incorporated, with the result that the return from the non-maskable interrupt service program is not performed normally and a software malfunction 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 the following instructions are executed, interrupt acknowledgement and macro service processing are held pending for 8 system clocks. 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, #imm24
Write instruction and bit manipulation instruction to interrupt control registers[Note], MK0, MK1, IMC, ISPR, or SNM1 register (excluding BT, BF instructions)
PSW bit manipulation instructions (excluding 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, CLR1 CY instruction)

**Note** Interrupt control registers: WDTIC, PIC0, PIC1, PIC2, PIC3, PIC4, PIC5, PIC6, CSIIC0, SERIC1, SRIC1, STIC1, SERIC2, SRIC2, STIC2, TMIC3, TMIC00, TMIC01, TMIC1, TMIC2, ADIC, TMIC5, TMIC6, TMIC7, TMIC8, WTIC, KRIC

**Caution If problems are caused by a long pending period for interrupts and macro servicing when the corresponding instructions are used in succession, a time at which interrupts and macro service requests can be acknowledged should be provided by inserting an NOP instruction, etc., in the series of instructions.**

# CHAPTER 23 LOCAL BUS INTERFACE FUNCTIONS

## 23.1 External Memory Expansion Function

The external memory expansion function connects external memory to areas other than the internal ROM, RAM, and SFR.

A time-divided address/data bus is used to connect external memory. A 256-byte expansion mode and a 1 MB expansion mode are supported for the external memory expansion function.

When external memory is connected, ports 4 to 6 are used. Ports 4 to 6 control address/data and read/write strobes, and wait and address strobes, etc.

**Table 23-1. Pin Functions in External Memory Expansion Mode**

| Pin Functions When External Device Connected | | Alternate Functions |
|---|---|---|
| Name | Function | |
| AD0 to AD7 | Multiplexed address/data bus | P40 to P47 |
| A8 to A15 | Middle address bus | P50 to P57 |
| A16 to A19 | High address bus | P60 to P63 |
| $\overline{RD}$ | Read strobe | P64 |
| $\overline{WR}$ | Write strobe | P65 |
| $\overline{WAIT}$ | Wait signal | P66 |
| ASTB | Address strobe | P67 |

**Table 23-2. Pin States in Ports 4 to 6 in External Memory Expansion Mode**

| Port / External Expansion Mode | Port 4 | Port 5 | | | | | | | | Port 6 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 to 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Single-chip mode | Port | Port | | | | | | | | Port | | | | | | | |
| 256 KB expansion mode | Address/data | Address | | | | | | | | Address | | | Port | $\overline{RD}$, $\overline{WR}$, $\overline{WAIT}$, ASTB | | | |
| 1 MB expansion mode | Address/data | Address | | | | | | | | Address | | | | $\overline{RD}$, $\overline{WR}$, $\overline{WAIT}$, ASTB | | | |

**Caution When the external wait function is not used, the $\overline{WAIT}$ pin can be used as the port in all of the modes.**

## 23.2 Control Registers

### (1) Memory expansion mode register (MM)

MM is an 8-bit register that controls the external expanded memory, sets the number of address waits, and controls the internal fetch cycle.

MM can be read or written by a 1-bit or 8-bit memory manipulation instruction. Figure 23-1 shows the MM format.

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

**Figure 23-1. Format of Memory Expansion Mode Register (MM)**

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

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| MM | IFCH | 0 | AW | 0 | MM3 | MM2 | MM1 | MM0 |

| IFCH | Internal ROM fetch |
|---|---|
| 0 | Fetch at the same speed as from external memory.<br>All of the wait control settings are valid. |
| 1 | High-speed fetch<br>The wait control settings are invalid. |

| AW | Address wait setting |
|---|---|
| 0 | An address wait is not inserted. |
| 1 | A one-clock address wait is inserted at the address output timing. |

| MM3 | MM2 | MM1 | MM0 | Mode | Port 4 (P40 to P47) | Port 5 (P50 to P57) | P60 to P63 | | P64 | P65 | P66 | P67 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Single-chip mode | Port | | | | | | | |
| 1 | 0 | 0 | 0 | 256 KB expansion mode | AD0 to AD7 | A8 to A15 | A16, A17 | Port | $\overline{\text{RD}}$ | $\overline{\text{WR}}$ | $\overline{\text{WAIT}}$ **Note** | ASTB |
| 1 | 0 | 0 | 1 | 1 MB expansion mode | | | A16 to A19 | | | | | |
| Other than above | | | | Setting prohibited | | | | | | | | |

**Note** When the external wait function is not used, the $\overline{\text{WAIT}}$ pin can be used as a port.

**(2) Programmable wait control register 1 (PWC1)**

PWC1 is an 8-bit register that sets the number of waits.

The insertion of wait cycles is controlled by PWC1 over the entire space.

PWC1 can be read and written by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets PWC1 to AAH.

**Figure 23-2. Format of Programmable Wait Control Register 1 (PWC1)**

Address: 0FFC7H After reset: AAH R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PWC1 | × | × | × | × | × | × | PW01 | PW00 |

| PW01 | PW00 | Insertion wait cycles | Data access cycles, fetch cycles |
|---|---|---|---|
| 0 | 0 | 0 | 3 |
| 0 | 1 | 1 | 4 |
| 1 | 0 | 2 | 5 |
| 1 | 1 | Low-level period that is input at the $\overline{\text{WAIT}}$ pin | – |

**Remarks 1.** The insertion of wait cycles is controlled by the entire address space (except for the peripheral RAM area).

**2.** ×: Don't care

**(3) Programmable wait control register 2 (PWC2)**

In the μPD784225 Subseries, wait cycle insertion control can be performed for the entire address space in one operation using programmable wait control register 1 (PWC1). However, in the in-circuit emulator, the address space is partitioned, and wait control is performed for each area separately (using both the PWC1 and PWC2 registers).

Consequently, when performing programmable debugging using the in-circuit emulator, wait control must be performed by setting both the PWC1 register and programmable wait control register 2 (PWC2). Set as shown in Table 23-3 below. Note that the settings in PWC2 and PWC1 (except bits 1 and 0) are invalid in the μPD784225 Subseries, and therefore have no negative effect.

**Table 23-3. Settings of Program Wait Control Register 2 (PWC2)**

| Inserted Wait Cycle | μPD784225 Subseries | In-Circuit Emulator | |
|---|---|---|---|
| | PWC1 | PWC1 | PWC2 |
| 0 | ××××  ××00B | 00H | 0000H |
| 1 | ××××  ××01B | 55H | 5555H |
| 2 | ××××  ××10B | AAH | AAAAH |
| Low-level time input to $\overline{\text{WAIT}}$ pin | ××××  ××11B | FFH | FFFFH |

## 23.3 Memory Map for External Memory Expansion

Figures 23-4 and 23-5 show the memory map during memory expansion. Even during memory expansion, an external device at the same address as the internal ROM area, internal RAM area, or SFR area (except for the external SFR area (0FFD0H to 0FFDFH)) cannot be accessed. If these areas are accessed, the memory and SFR in $\mu$PD784225 are accessed by priority, and the ASTB, $\overline{RD}$, and $\overline{WD}$ signals are not output (remaining at the inactive level). The output level of the address bus remains at the previous output level. The output of the address/data bus has a high impedance.

Except in the 1 MB expansion mode, an address for external output is output in the state that masked the higher side of the address set by the program.

**<Example 1>**

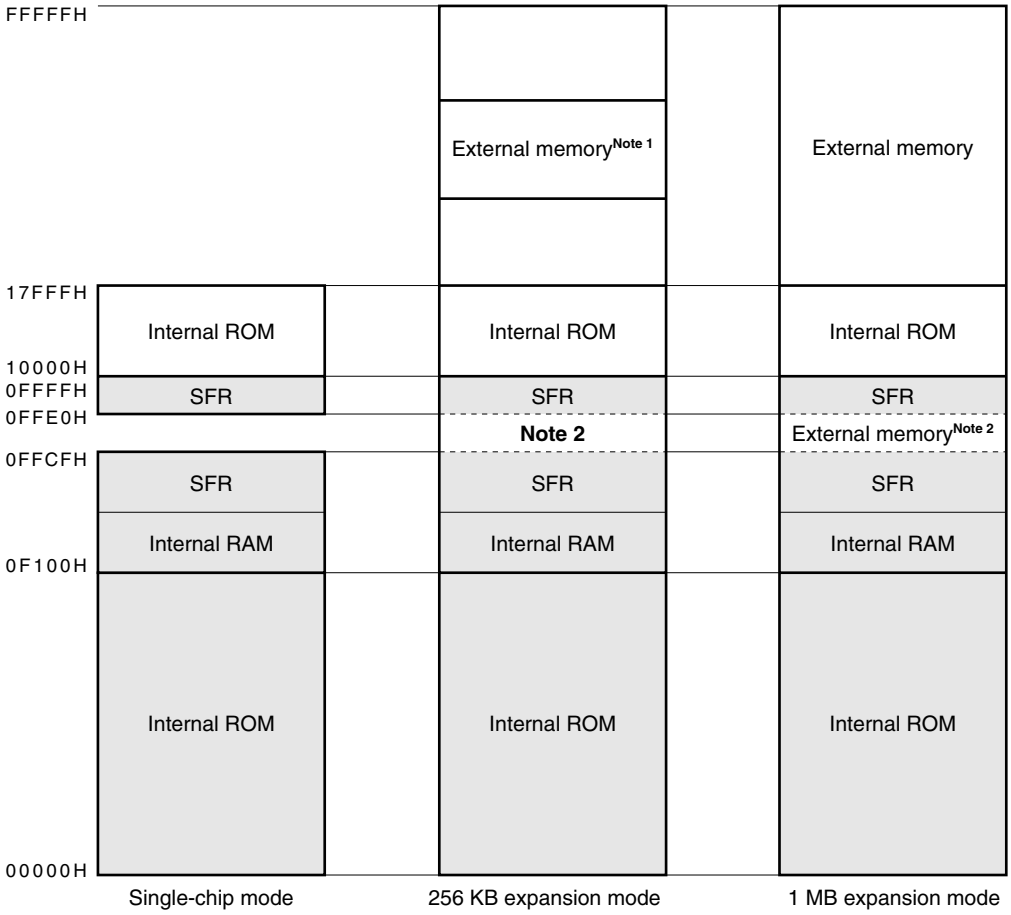
When address 54321H is accessed in the program in the 256 KB expansion mode, the address that is output becomes 14321H.

**<Example 2>**

When address 67821H is accessed in the program in the 256 KB expansion mode, the address that is output becomes 27821H.

**Figure 23-3. μPD784224 Memory Map (1/2)**

**(a) When executing the LOCATION 0H instruction**

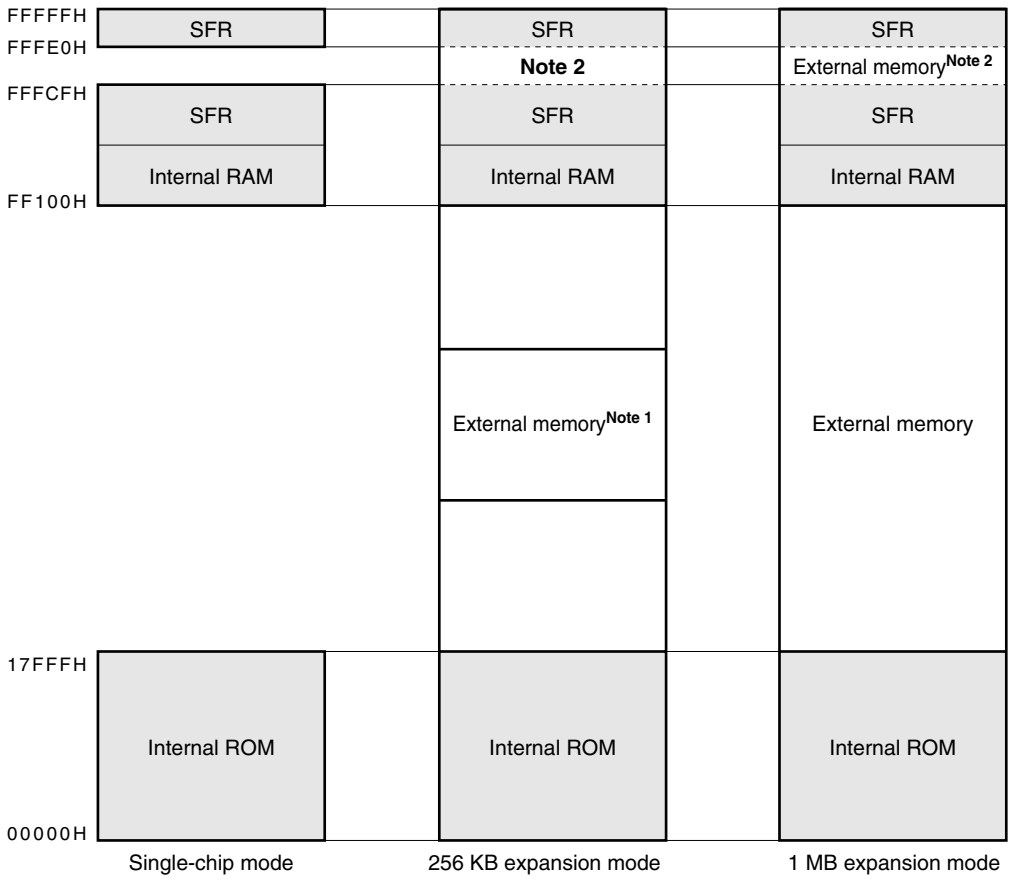



**Notes 1.** Area having any expanded size in the unshaded parts

**2.** External SFR area

**Figure 23-3. $\mu$PD784224 Memory Map (2/2)**

**(b) When executing the LOCATION 0FH instruction**

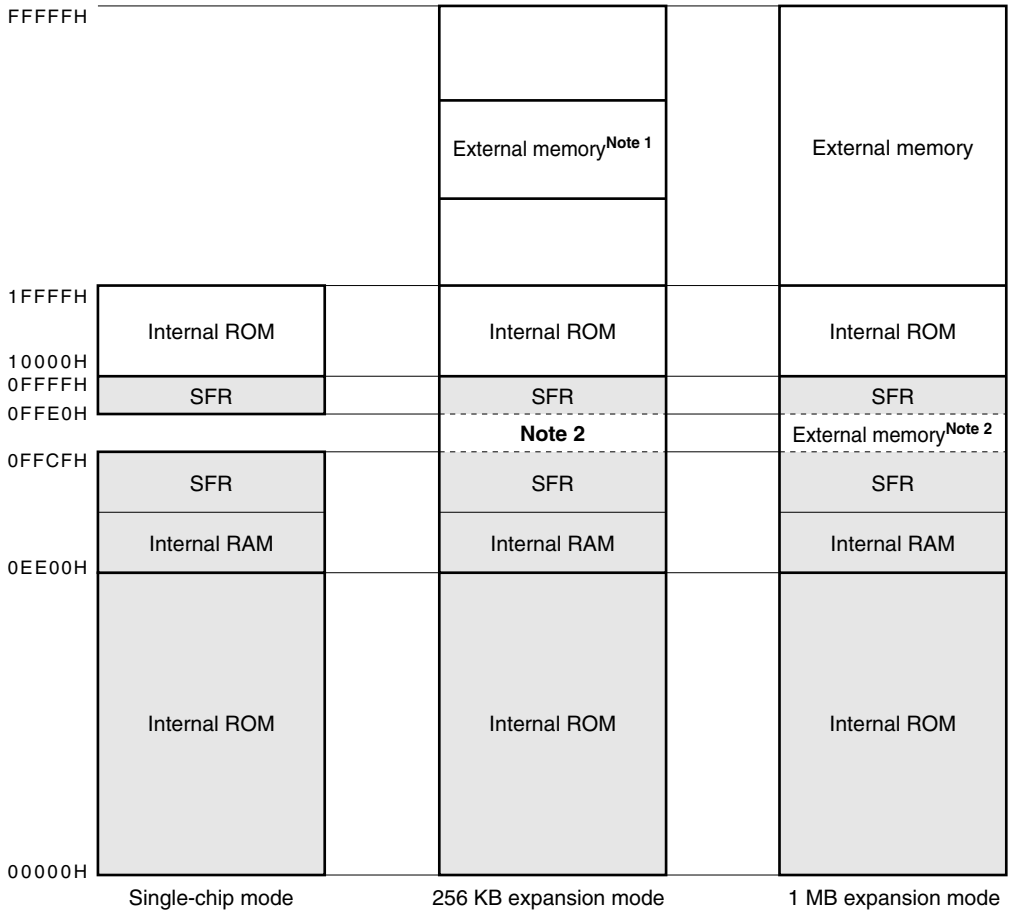



**Notes 1.** Area having any expanded size in the unshaded parts

**2.** External SFR area

**Figure 23-4. μPD784225 Memory Map (1/2)**

**(a) When executing the LOCATION 0H instruction**

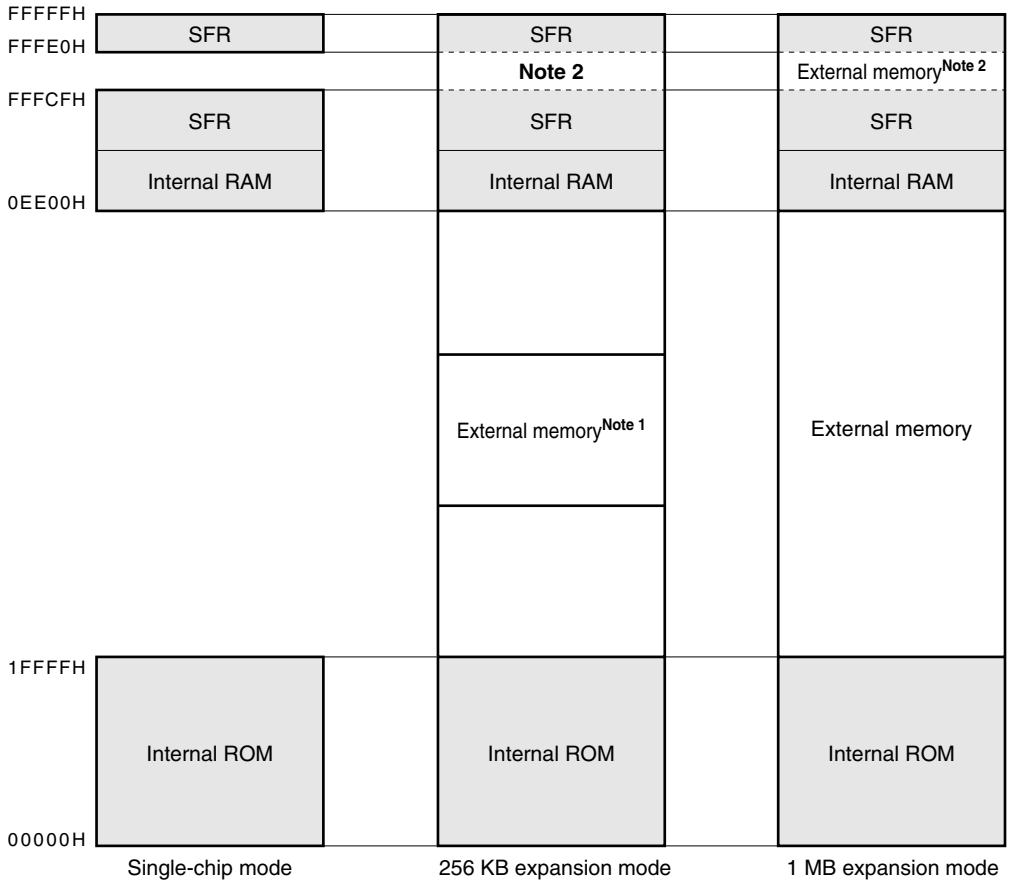



**Notes 1.** Area having any expanded size in the unshaded parts

**2.** External SFR area

**Figure 23-4. $\mu$PD784225 Memory Map (2/2)**

**(b) When executing the LOCATION 0FH instruction**




**Notes 1.** Area having any expanded size in the unshaded parts

**2.** External SFR area

## 23.4 Timing of External Memory Expansion Functions

The timing control signal output pins in the external memory expansion mode are described below.

**(1) $\overline{\text{RD}}$ pin (shared by: P64)**

This pin outputs the read strobe during an instruction fetch or a data access from external memory.
During an internal memory access, the read strobe is not output (held at the high level).

**(2) $\overline{\text{WR}}$ pin (shared by: P65)**

This pin outputs the write strobe during a data access to external memory.
During an internal memory access, the write strobe is not output (held at the high level).

**(3) $\overline{\text{WAIT}}$ pin (shared by: P66)**

This pin inputs the external wait signal.
When the external wait is not used, the $\overline{\text{WAIT}}$ pin can be used as an I/O port.
During an internal memory access, the external wait signal is ignored.

**(4) ASTB pin (shared by: P67)**

This pin always outputs the address strobe in any instruction fetch or data access from external memory.
During an internal memory access, the address strobe is not output (held at the low level).

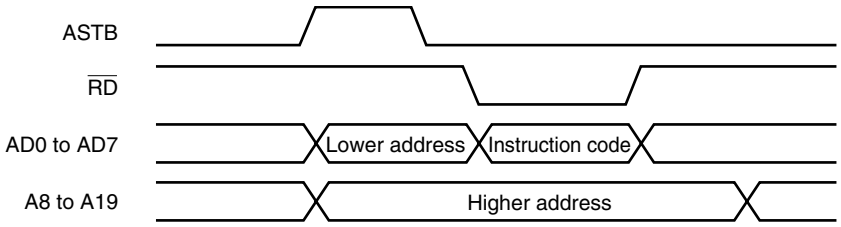
**(5) AD0 to AD7, A8 to A15, A16 to A19 pins (shared by: P40 to P47, P50 to P57, P60 to P63)**

These pins output the address and data signals. When an instruction is fetched or data is accessed from external memory, valid signals are output or input.
During an internal memory access, the signals do not change.

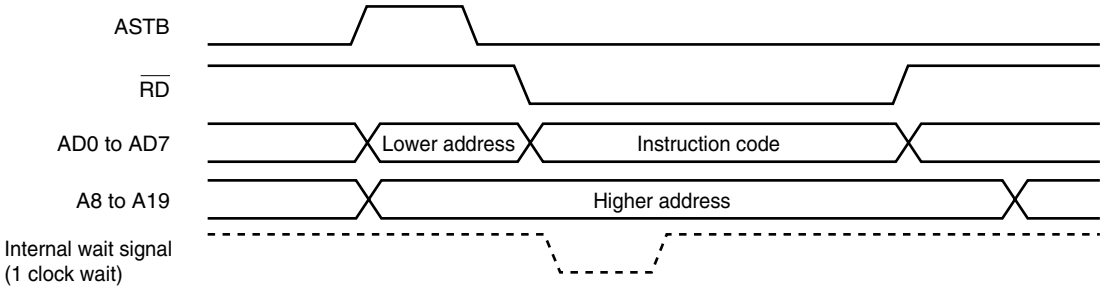
Figures 23-5 to 23-8 are the timing charts.

**Figure 23-5. Instruction Fetch from External Memory in External Memory Expansion Mode**

**(a) Setting 0 wait cycles (PW01, PW00 = 0, 0)**




**(b) Setting 1 wait cycle (PW01, PW00 = 0, 1)**




**(c) Setting an external wait (PW01, PW00 = 1, 1)**

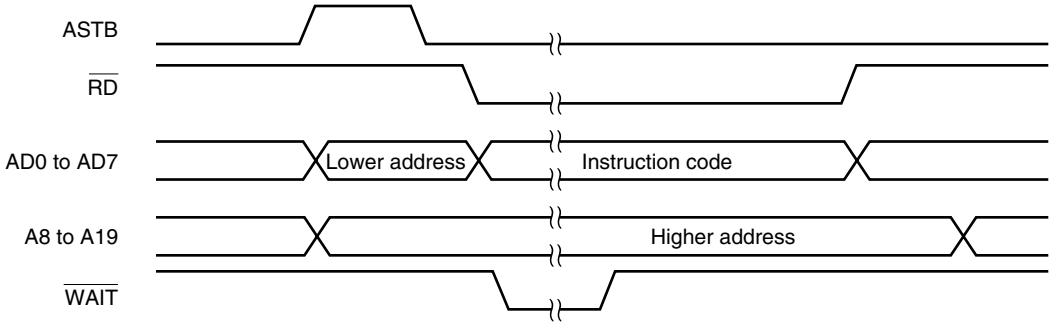
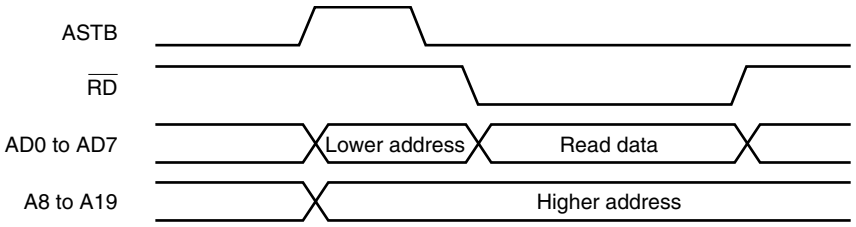
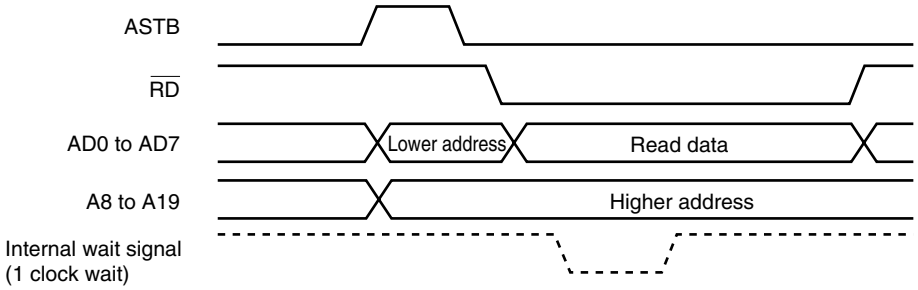


**Figure 23-6. Read Timing for External Memory in External Memory Expansion Mode**

**(a) Setting 0 wait cycles (PW01, PW00 = 0, 0)**




**(b) Setting 1 wait cycle (PW01, PW00 = 0, 1)**




**(c) Setting an external wait (PW01, PW00 = 1, 1)**

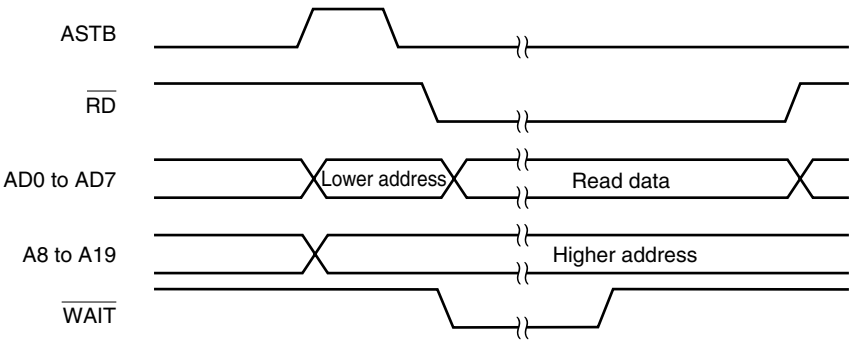
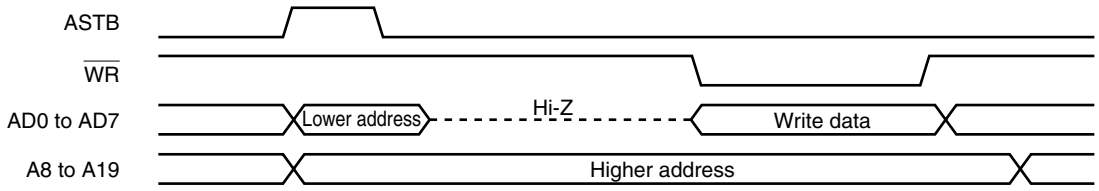
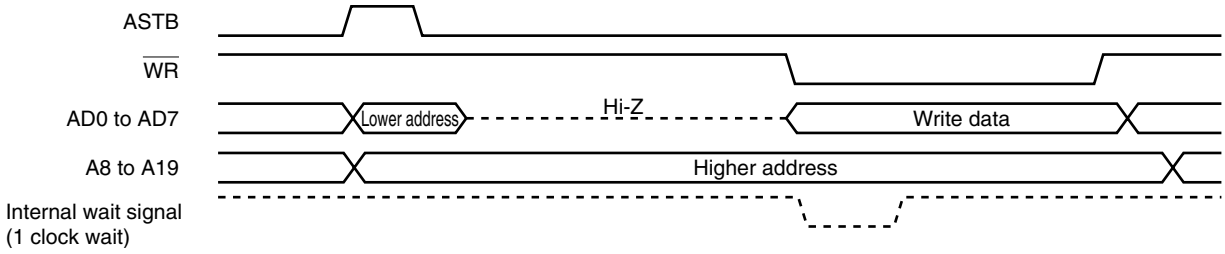


**Figure 23-7. External Write Timing for External Memory in External Memory Expansion Mode**

**(a) Setting 0 wait cycles (PW01, PW00 = 0, 0)**




**(b) Setting 1 wait cycle (PW01, PW00 = 0, 1)**




**(c) Setting an external wait (PW01, PW00 = 1, 1)**

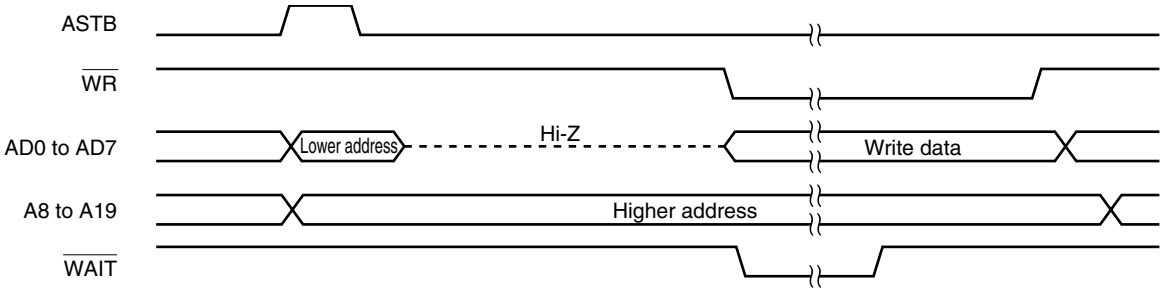
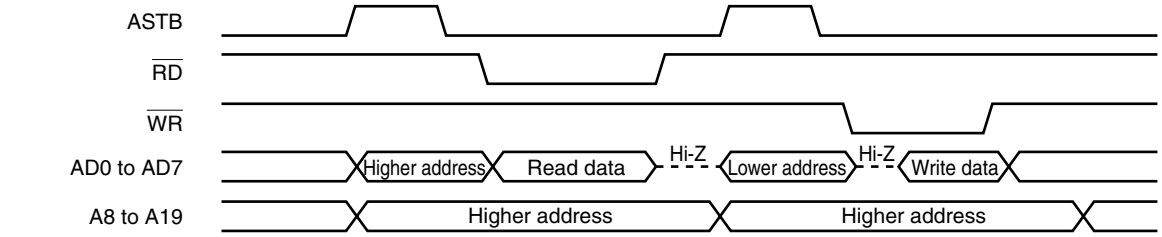
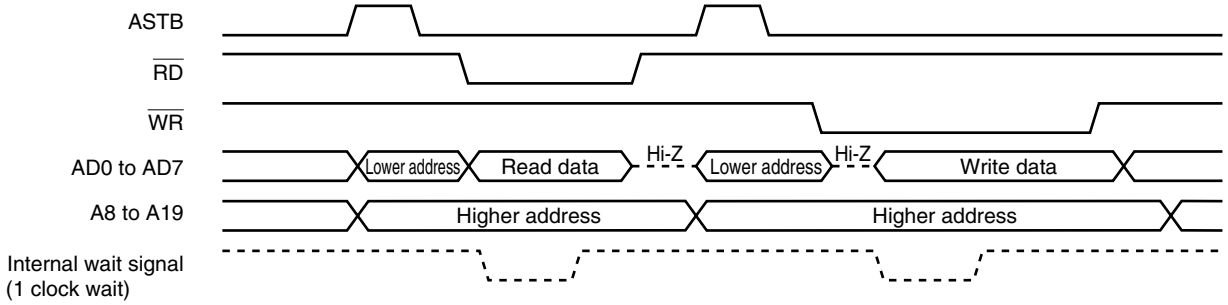


**Figure 23-8. Read Modify Write Timing for External Memory in External Memory Expansion Mode**

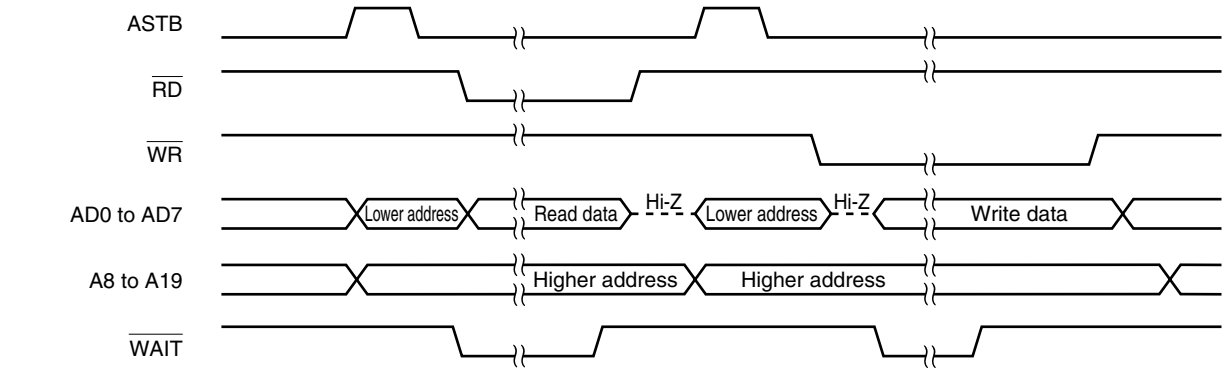
**(a) Setting 0 wait cycles (PW01, PW00 = 0, 0)**




**(b) Setting 1 wait cycle (PW01, PW00 = 0, 1)**




**(c) Setting an external wait (PW01, PW00 = 1, 1)**



## 23.5 Wait Functions

If slow memory and I/O are connected externally to the μPD784225, waits can be inserted in the external memory access cycle.

The wait cycle includes an address wait to guarantee the address decoding time and an access wait to guarantee the access time.

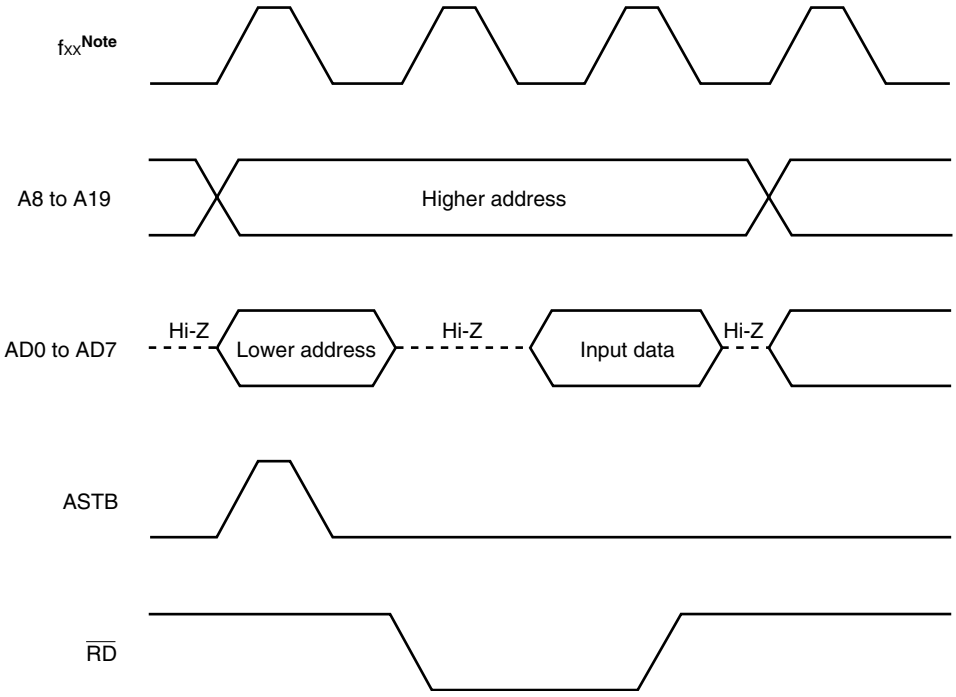
### 23.5.1 Address wait

The address wait guarantees the address decoding time. By setting the AW bit in the memory expansion mode register (MM) to 1, an address wait is inserted into the entire memory access time**Note**. When the address wait is inserted, the high level period of the ASTB signal is lengthened by one system clock (when 80 ns, fxx = 12.5 MHz).

**Note** This excludes the internal RAM, internal SFR, and internal ROM during a high-speed fetch. When the internal ROM access is set to have the same cycle as an external ROM access, an address wait is inserted during an internal ROM access.

**Figure 23-9. Read/Write Timing by Address Wait Function (1/3)**

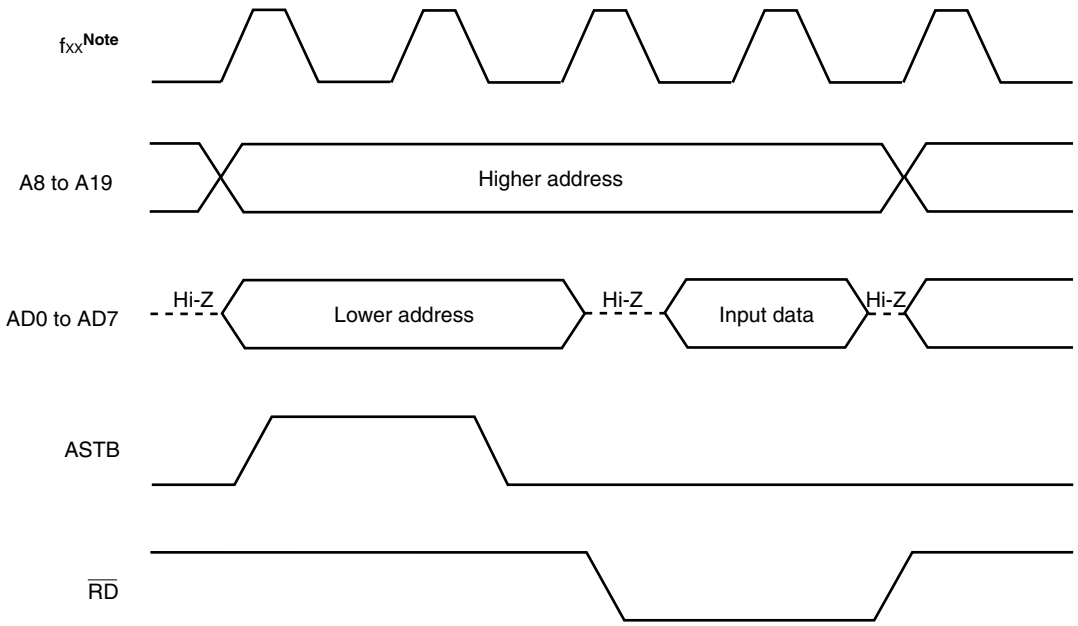
**(a) Read timing when an address wait is not inserted**




**Note** fxx: Main system clock frequency. This signal is only in the μPD784225.

**Figure 23-9. Read/Write Timing by Address Wait Function (2/3)**

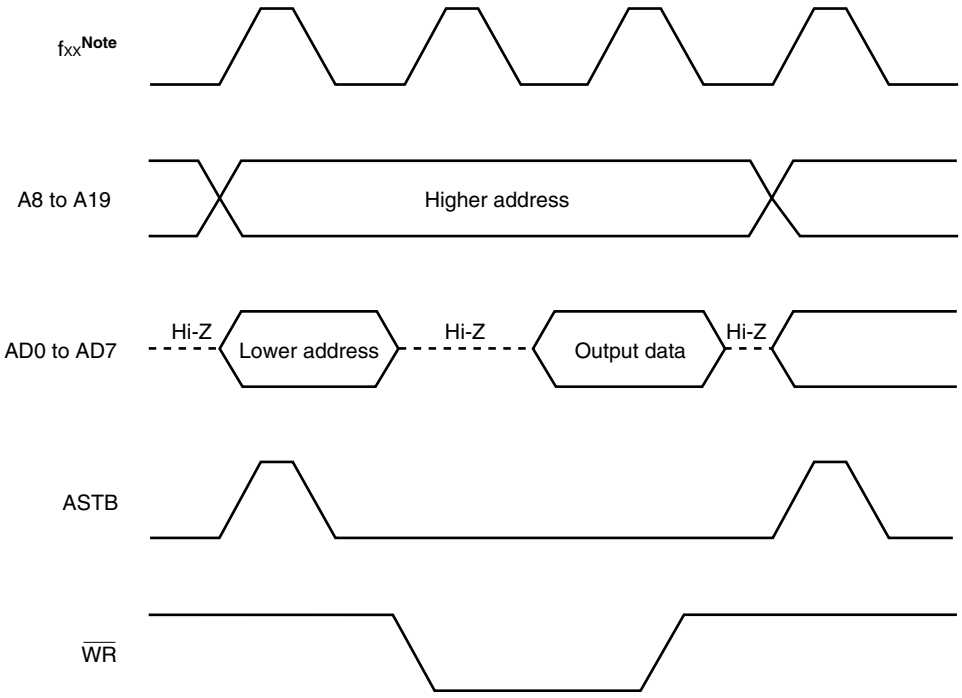
**(b) Read timing when an address wait is inserted**



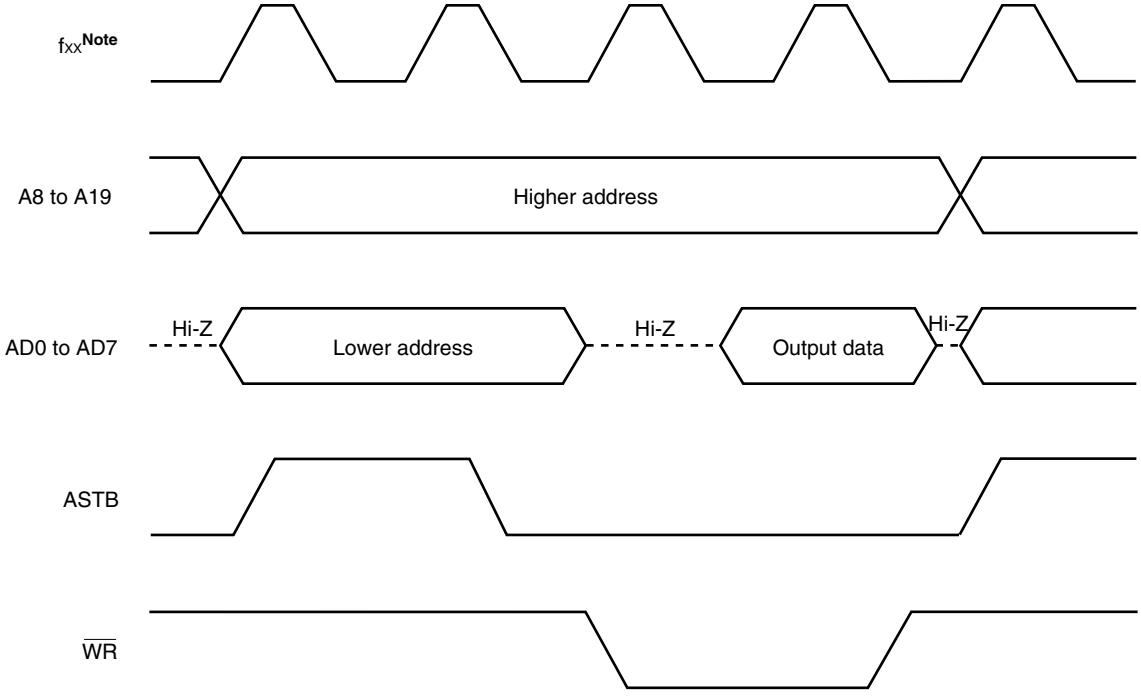

**Note** fxx: Main system clock frequency. This signal is only in the μPD784225.

**Figure 23-9. Read/Write Timing by Address Wait Function (3/3)**

**(c) Write timing when an address wait is not inserted**




**(d) Write timing when an address wait is inserted**




**Note** fxx: Main system clock frequency. This signal is only in the μPD784225.

### 23.5.2 Access wait

An access wait is inserted during low $\overline{RD}$ and $\overline{WR}$ signals. The low level is lengthened by 1/fxx (80 ns, fxx = 12.5 MHz) per cycle.

The wait insertion methods are the programmable wait function that automatically inserts a preset number of cycles and the external wait function that is controlled from the outside by the wait signal.

Wait cycle insertion control is set by the programmable wait control register (PWC1) for the 1 MB memory space. If an internal ROM or internal RAM is accessed during a high-speed fetch, a wait is not inserted. If accessing an internal SFR, a wait is inserted based on the required timing unrelated to this setting.

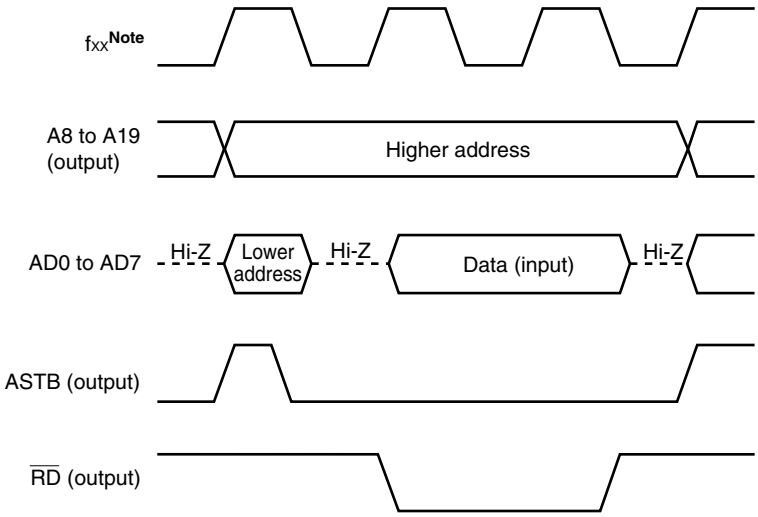
If set so that an access has the same number of cycles as for an external ROM, a wait is also inserted in an internal ROM access in accordance with the PWC1 setting.

If there is space that was externally selected to be controlled by the wait signal by PWC1, pin P66 acts as the $\overline{WAIT}$ signal input pin. $\overline{RESET}$ input makes pin P66 act as an ordinary I/O port.

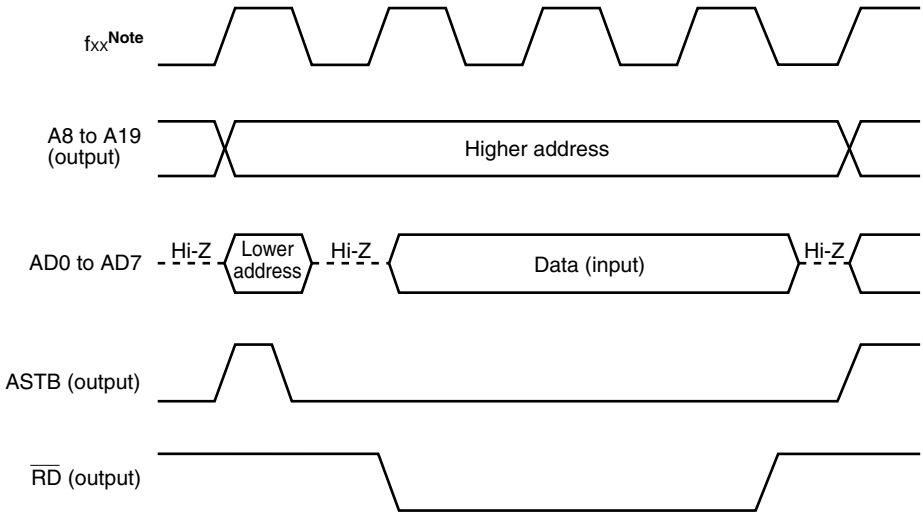
Figures 23-10 to 23-12 show the bus timing when an access wait is inserted.

**Figure 23-10. Read Timing by Access Wait Function (1/2)**

**(a) Setting 0 wait cycles (PW01, PW00 = 0, 0)**



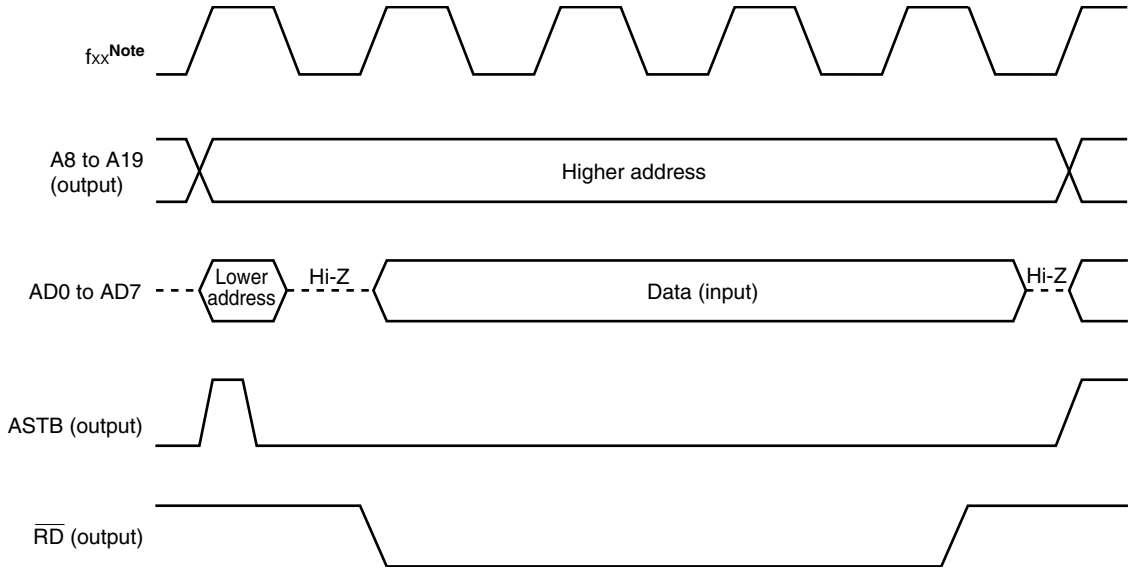

**(b) Setting 1 wait cycle (PW01, PW00 = 0, 1)**




**Note** fxx: Main system clock frequency. This signal is only in the μPD784225.

**Figure 23-10. Read Timing by Access Wait Function (2/2)**

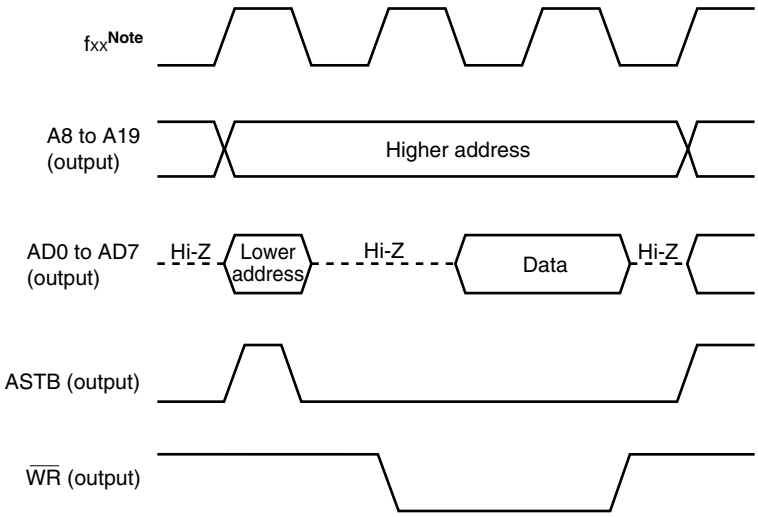
**(c) Setting 2 wait cycles (PW01, PW00 = 1, 0)**



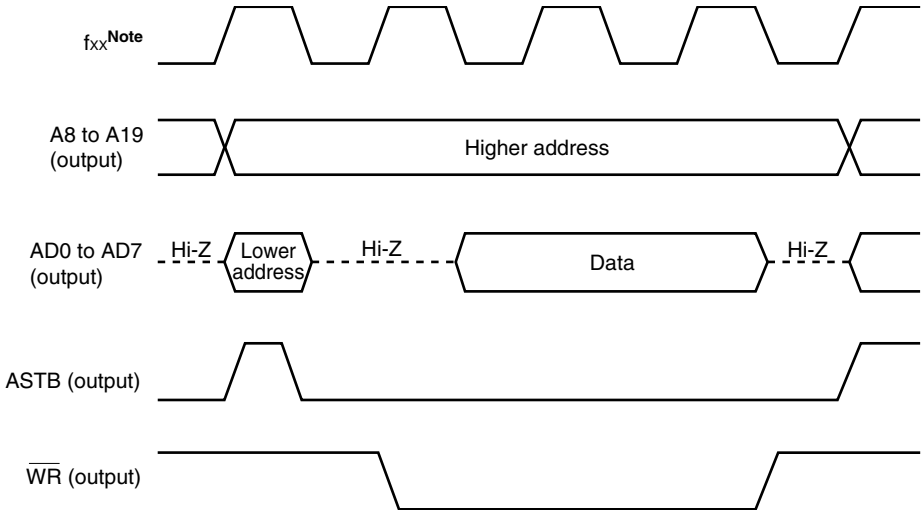

**Note** fxx: Main system clock frequency. This signal is only in the μPD784225.

**Figure 23-11. Write Timing by Access Wait Function (1/2)**

**(a) Setting 0 wait cycles (PW01, PW00 = 0, 0)**



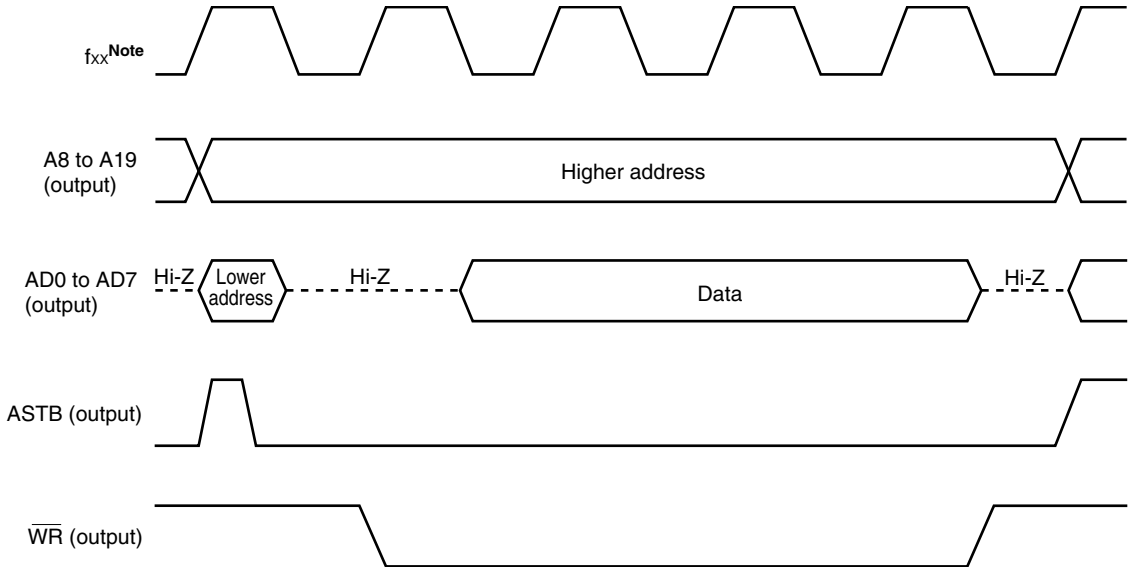

**(b) Setting 1 wait cycle (PW01, PW00 = 0, 1)**




**Note** fxx: Main system clock frequency. This signal is only in the μPD784225.

**Figure 23-11. Write Timing by Access Wait Function (2/2)**

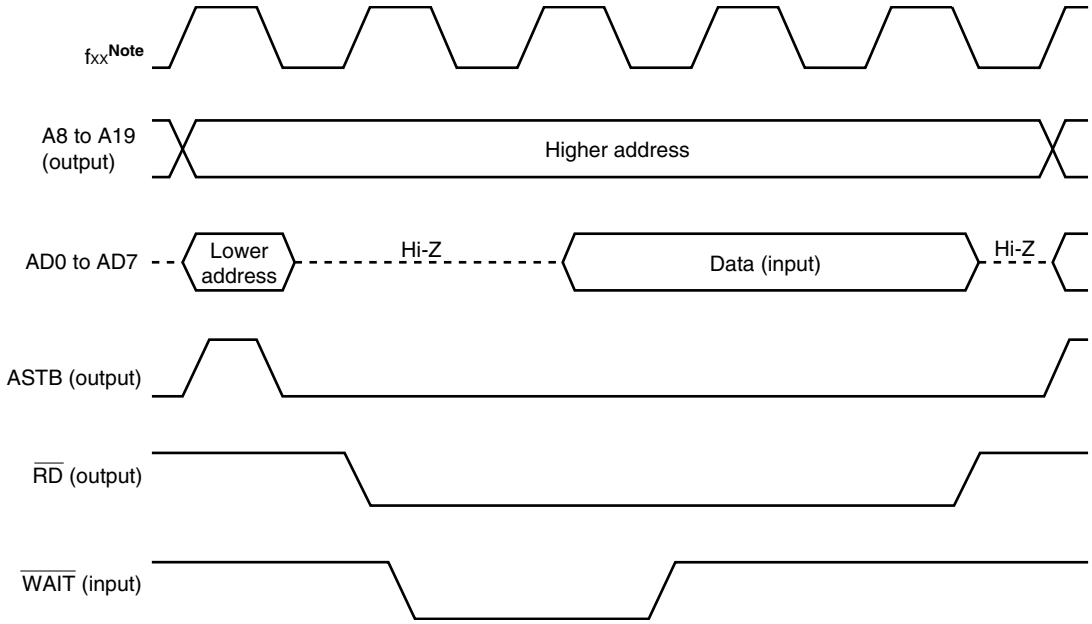
**(c) Setting 2 wait cycles (PW01, PW00 = 1, 0)**



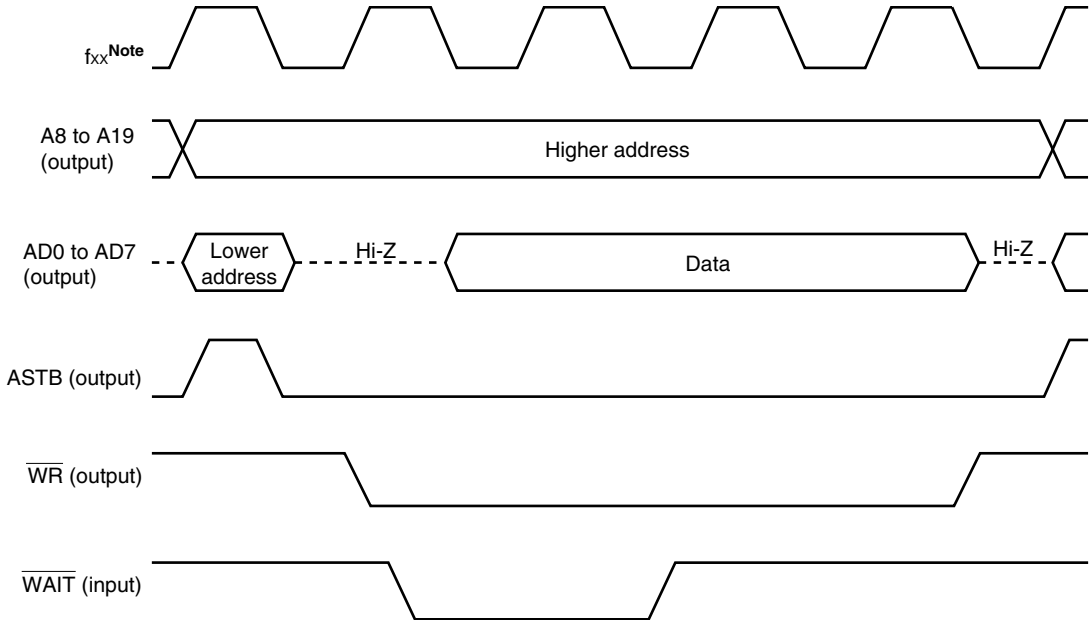

**Note** fxx: Main system clock frequency. This signal is only in the μPD784225.

**Figure 23-12. Timing by External Wait Signal**

**(a) Read Timing (PW01, PW00 = 1, 1)**




**(b) Write timing (PW01, PW00 = 1, 1)**




**Note** fxx: Main system clock frequency. This signal is only in the μPD784225.

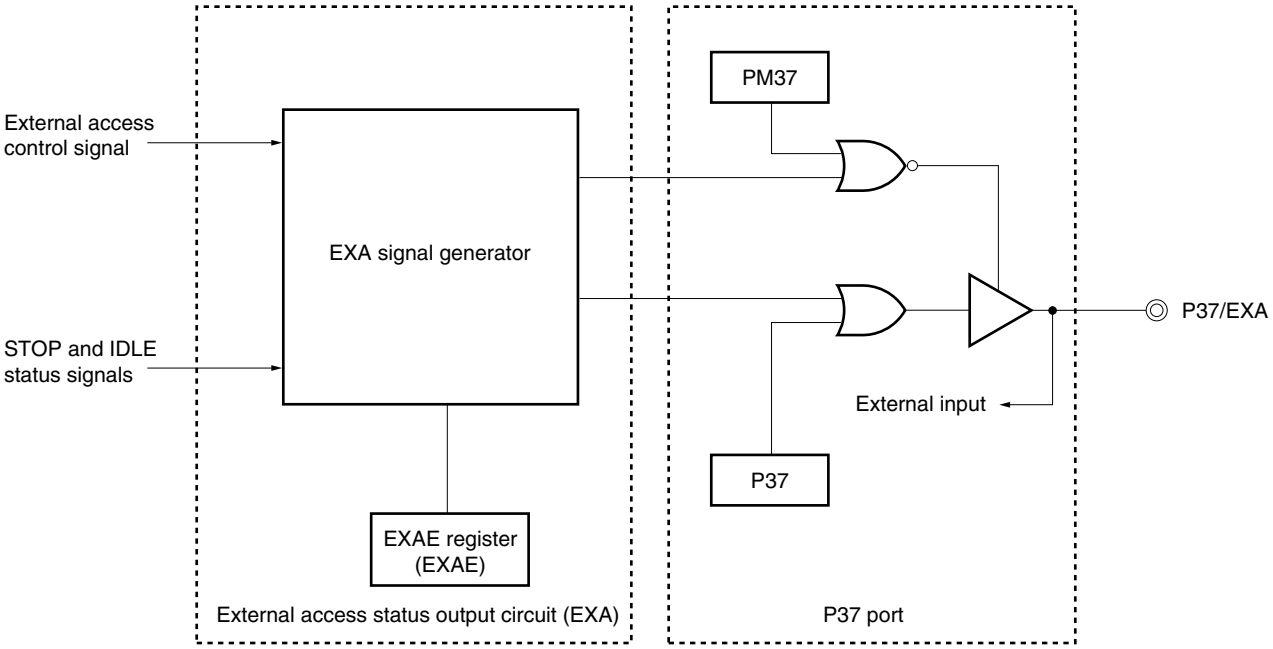
## 23.6 External Access Status Output Function

### 23.6.1 Summary

The external access status signal is output from the P37/EXA pin. This signal is output at the moment of external access when use of the external bus interface function has been enabled. This signal detects the external access status of other devices connected to the external bus, prohibits other devices from outputting data to the external bus, and enables reception.

### 23.6.2 Configuration of external access status output function

**Figure 23-13. Configuration of External Access Status Output Function**



### 23.6.3 External access status enable register

The external access status enable register (EXAE) controls the EXA signal output indicated during external access.

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

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

**Figure 23-14. Format of External Access Status Enable Register (EXAE)**

Address: 0FF8DH After reset: 00H

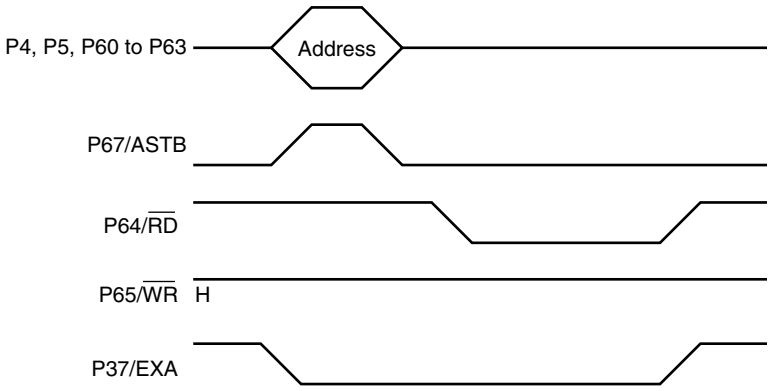
| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| EXAE | 0 | 0 | 0 | 0 | 0 | 0 | 0 | EXAE0 |

| EXAE | P37 function |
|---|---|
| 0 | Port function |
| 1 | External access status function |

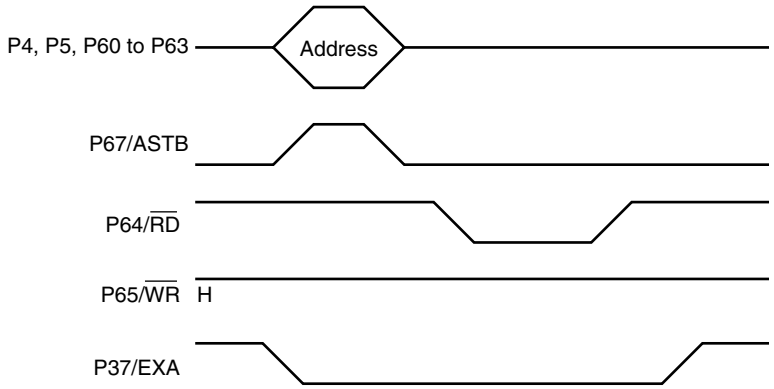
### 23.6.4 External access status signal timing

A timing chart for the P37/EXA and external bus interface pin is shown below. The EXA signal is active-low, and indicates the external access status when “0”.

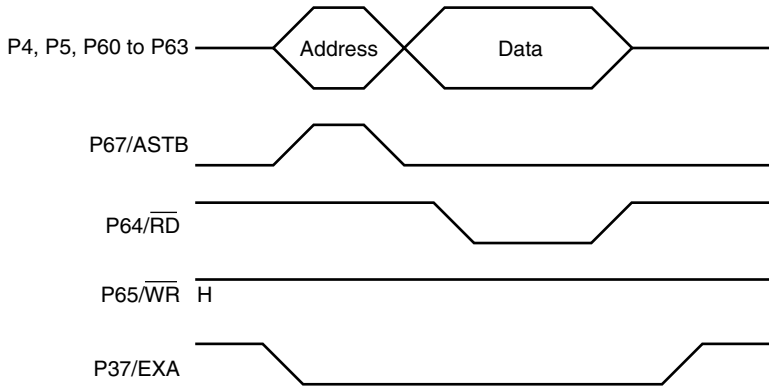
**(a) Data fetch timing**



**(b) Data read timing**




**(c) Data write timing**




### 23.6.5 EXA pin status in each mode

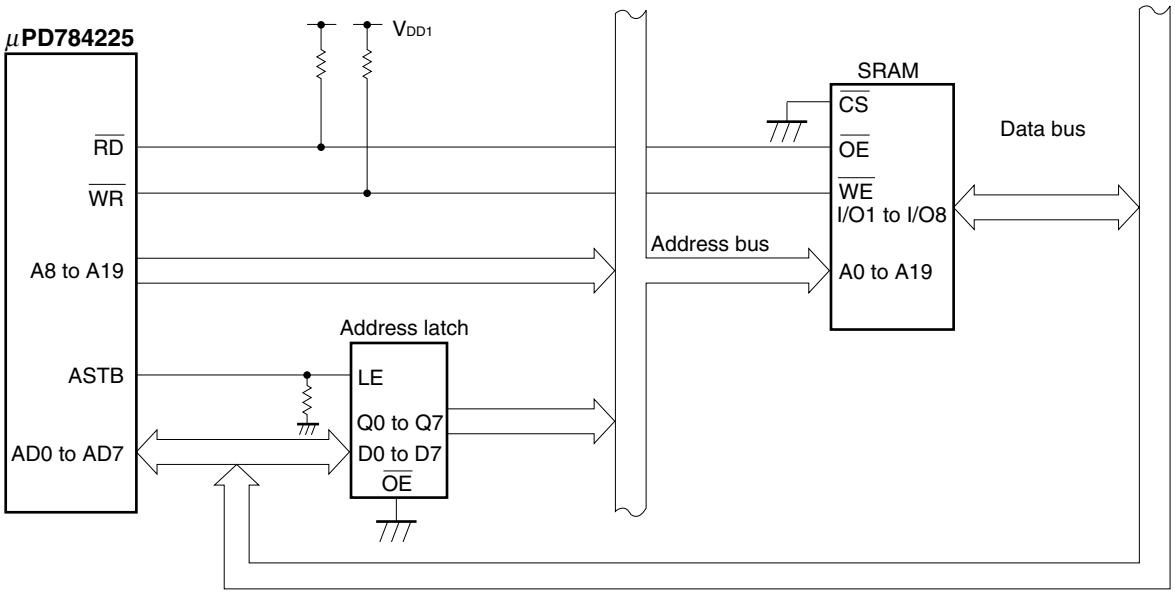
P37/EXA pin status in each mode is shown in Table 23-4.

**Table 23-4. P37/EXA Pin Status in Each Mode**

| Mode | P37/EXA Functions |
|---|---|
| After reset | Hi-Z |
| Normal operation | Hi-Z immediately after the reset is canceled (input mode, PM37 = 1)<br>Port operations when EXAE = 00H with PM37 and P37 = 0<br>EXA signal output enabled when EXAE = 01H with PM37 and P37 = 0 |
| HALT modeStandby | |
| IDLE modeHi-Z | |
| STOP modeHi-Z | |

## 23.7 External Memory Connection Example

**Figure 23-15. Example of Local Bus Interface (Multiplexed Bus)**



# CHAPTER 24 STANDBY FUNCTION

## 24.1 Configuration and Function

The μPD784225 has a standby function that can decrease the system's power consumption. The standby function has the following six modes.

**Table 24-1. Standby Function Modes**

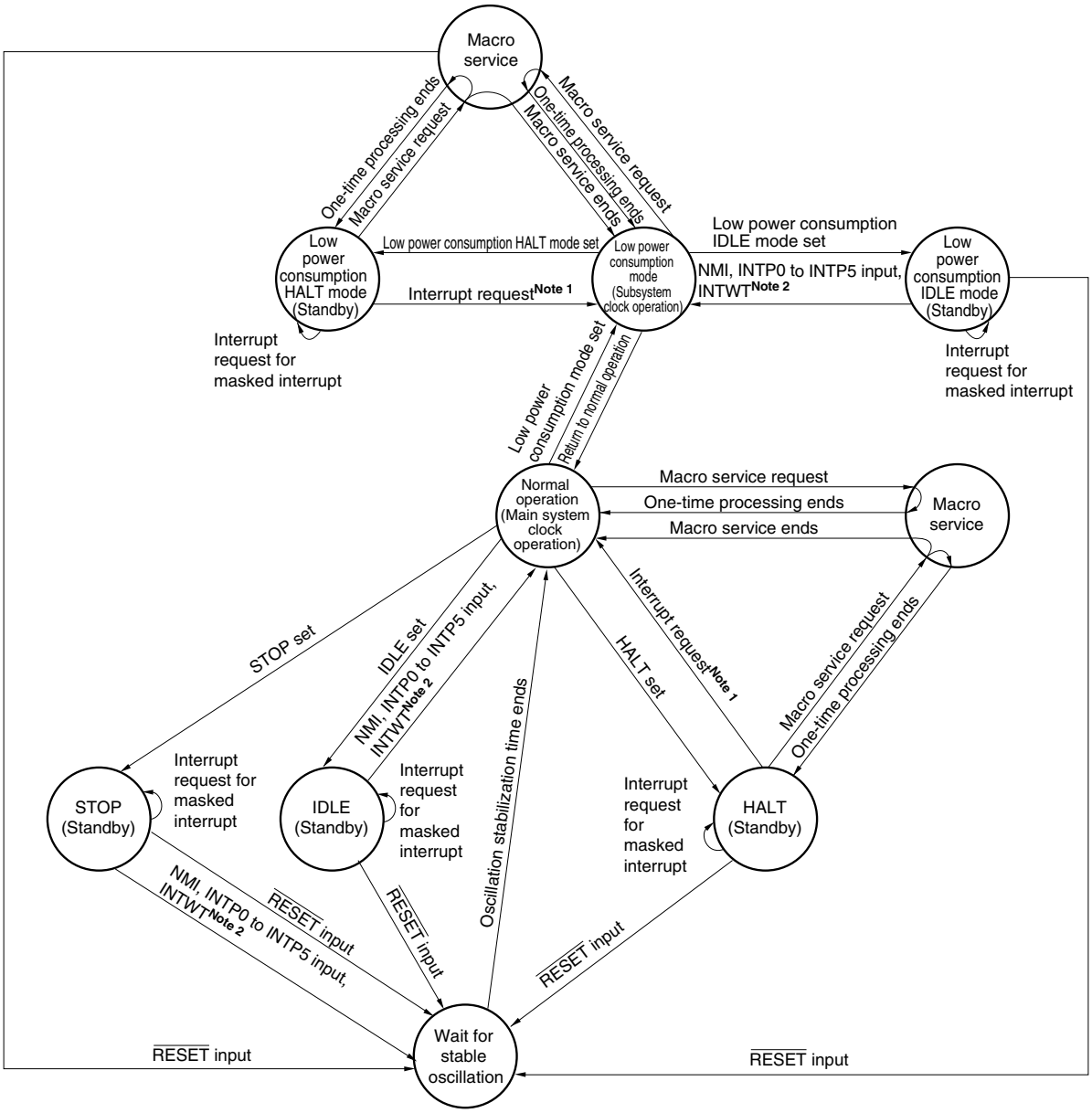
| | |
|---|---|
| HALT mode | Stops the CPU operating clock. The average power consumption can be reduced by intermittent operation during normal operation chips internal. |
| STOP mode | Stops the main system clock. All of the chip's internal operations are stopped, and an extremely low power consumption state of only leakage current is entered. |
| IDLE mode | In this mode, the oscillator continues operating while the rest of the system stops. Power consumption in the IDLE mode is close to that in the STOP mode, but normal program operation can be restored in about the same time as it takes from the HALT mode. |
| Low power consumption mode | The subsystem clock is used as the system clock, and the main system clock is stopped. The CPU can operate with the subsystem clock to reduce power consumption. |
| Low power consumption HALT mode | This is a standby function in the low power consumption mode. In this mode the CPU operating clock is stopped to decrease power consumption for the entire system. |
| Low power consumption IDLE mode | This is a standby function in the low power consumption mode. In this mode the oscillator continues operating while the rest of the system is stopped, decreasing power consumption for the entire system. |

These modes are programmable.

Macro servicing can be started from the HALT mode and the low power consumption HALT mode. After macro service execution, the device is returned to the HALT mode.

Figure 24-1 shows the standby function state transitions.

★ **Figure 24-1. Standby Function State Transition**




**Notes 1**. Only unmasked interrupt requests

**2.** When INTP0 to INTP5 and the watch timer interrupt (INTWT) are not masked

**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.)

## 24.2 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 an inadvertent program loop, this register can only be written by a special instruction. This special instruction is MOV STBC, #byte which has a special code structure (4 bytes). This register can only be written when the third and fourth byte opcodes are mutual 1's complements.

If the third and fourth byte opcodes are not mutual 1's complements, the register is not written and an operand error interrupt is generated. In this case, the return address that is saved on the stack is the address of the instruction that caused the error. Therefore, the address that caused the error can be determined from the return address saved on the stack.

If the RETB instruction is used to simply return from an operand error, an infinite loop occurs.

Since an operand error interrupt is generated only when the program inadvertently loops (only the correct instruction is generated when MOV STBC, #byte is specified in the RA78K4 NEC assembler), make the program initialize the system.

Other write instructions (i.e., MOV STBC, A; STBC, #byte; 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 24-2 shows the STBC format.

**Figure 24-2. Format of Standby Control Register (STBC)**

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

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| STBC | SBK | CK2 | CK1 | CK0 | 0 | MCK | STP | HLT |

| SBK | Subsystem clock oscillation control |
|---|---|
| 0 | Oscillator operating (internal feedback resistors used) |
| 1 | Oscillator stopped (internal feedback resistors not used) |

| CK2 | CK1 | CK0 | CPU clock selection |
|---|---|---|---|
| 0 | 0 | 0 | $f_{XX}$ |
| 0 | 0 | 1 | $f_{XX}/2$ |
| 0 | 1 | 0 | $f_{XX}/4$ |
| 0 | 1 | 1 | $f_{XX}/8$ |
| 1 | 1 | 1 | $f_{XT}$ (recommended) |
| 1 | – | – | $f_{XT}$ |

| MCK | Main system clock oscillation control |
|---|---|
| 0 | Oscillator operating (internal feedback resistors used) |
| 1 | Oscillator stopped (internal feedback resistors not used) |

| STP | HLT | Operation setting flag |
|---|---|---|
| 0 | 0 | Normal operation mode |
| 0 | 1 | HALT mode<br>(automatically cleared when the HALT mode is released) |
| 1 | 0 | STOP mode<br>(automatically cleared when the STOP mode is released) |
| 1 | 1 | IDLE mode<br>(automatically cleared when the IDLE mode is released) |

**Cautions 1. If the STOP mode is used when an external clock is input, set the STOP mode after setting bit EXTC in the oscillation stabilization time specification register (OSTS) to 1. Using the STOP mode in the state where bit EXTC of OSTS is cleared (0) while the external clock is input may destroy the μPD784225 or reduce reliability. When the EXTC bit of OSTS is set to 1, always input to pin X2 the clock that has the inverse phase of the clock input at pin X1.**

**2. Execute three NOP instructions after the standby instruction (after releasing standby). If this is not done, when the execution of a standby instruction conflicts with an interrupt request, the standby instruction is not executed, and interrupts are acknowledged after executing multiple instructions that follow a standby instruction. The instruction that is executed before acknowledging the interrupt starts being executed within a maximum of six clocks after the standby instruction is executed.**

```
Example  MOV STBC, #byte
         NOP
         NOP
         NOP
           :
```

**Cautions 3. When CK2 = 0, even if MCK = 1, the oscillation of the main system clock does not stop (refer to 4.5.1 Main system clock operations).**

**Remarks 1.** $f_{XX}$: Main system clock oscillation frequency ($f_X$ or $f_X/2$)
$f_X$: Main system clock oscillation frequency
$f_{XT}$: Subsystem clock oscillation frequency
**2.** ×: Don't care

**(2) Clock status register (PCS)**

PCS is an 8-bit read-only register that shows the operating state of the CPU clock. When bits 2 and 4 to 7 in PCS are read, the corresponding bits in the standby control register (STBC) can be read.
PCS is set by a 1-bit or 8-bit memory manipulation instruction.
$\overline{\text{RESET}}$ input sets PCS to 32H.

**Figure 24-3. Format of Clock Status Register (PCS)**

Address: 0FFCEH After reset: 32H R

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PCS | SBK | CK2 | CK1 | CK0 | 0 | MCK | 1 | CST |

| SBK | Feedback resistor state for subsystem clock |
|---|---|
| 0 | Internal feedback resistors used |
| 1 | Internal feedback resistors not used |

| CK2 | CK1 | CK0 | CPU clock operating frequency |
|---|---|---|---|
| 0 | 0 | 0 | $f_{XX}$ |
| 0 | 0 | 1 | $f_{XX}/2$ |
| 0 | 1 | 0 | $f_{XX}/4$ |
| 0 | 1 | 1 | $f_{XX}/8$ |
| 1 | 1 | 1 | $f_{XT}$ (recommended) |
| 1 | – | – | $f_{XT}$ |

| MCK | Main system clock oscillation control |
|---|---|
| 0 | Oscillator operating |
| 1 | Oscillator stopped |

| CST | CPU clock state |
|---|---|
| 0 | Main system clock operation |
| 1 | Subsystem clock operation |

**Remark** ×: Don't care

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

The OSTS register sets the oscillator operation and the oscillation stabilization time when the STOP mode is released. Whether a crystal/ceramic oscillator or an external clock will be used is set by the EXTC bit of OSTS. If only the EXTC bit is set to 1, the STOP mode can also be set when the external clock is input.

Bits OSTS0 to OSTS2 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 oscillator and at least 4 ms when using a ceramic oscillator.

The time until the oscillation stabilizes is affected by the crystal/ceramic oscillator 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 oscillator.

OSTS can be set by a 1-bit or 8-bit memory manipulation instruction.

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

Figure 24-4 shows the OSTS format.

**Figure 24-4. Format of Oscillation Stabilization Time Specification Register (OSTS)**

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

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

| EXTC | External clock selection |
|---|---|
| 0 | Crystal/ceramic oscillation used |
| 1 | External clock used |

| EXTC | OSTS2 | OSTS1 | OSTS0 | Oscillation stabilization time selection |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | $2^{19}/f_{XX}$ (42.0 ms) |
| 0 | 0 | 0 | 1 | $2^{18}/f_{XX}$ (21.0 ms) |
| 0 | 0 | 1 | 0 | $2^{17}/f_{XX}$ (10.5 ms) |
| 0 | 0 | 1 | 1 | $2^{16}/f_{XX}$ (5.3 ms) |
| 0 | 1 | 0 | 0 | $2^{15}/f_{XX}$ (2.6 ms) |
| 0 | 1 | 0 | 1 | $2^{14}/f_{XX}$ (1.3 ms) |
| 0 | 1 | 1 | 0 | $2^{13}/f_{XX}$ (0.7 ms) |
| 0 | 1 | 1 | 1 | $2^{12}/f_{XX}$ (0.4 ms) |
| 1 | × | × | × | $512/f_{XX}$ (41.0 $\mu$s) |

**Cautions 1. When using crystal/ceramic oscillation, always clear the EXTC bit to 0. When the EXTC bit is set to 1, oscillation stops.**

**2. If the STOP mode is used when an external clock is input, always set the EXTC bit to 1 and then set the STOP mode. Using the STOP mode in the state where the EXTC bit is cleared (0) while the external clock is input may destroy $\mu$PD784225 or reduce reliability.**

**3. When the EXTC bit is set to 1 when an external clock is input, input to pin X2 a clock that has the inverse phase of the clock input to pin X1. If the EXTC bit is set to 1, $\mu$PD784225 only operates with the clock that is input to the X2 pin.**

**Remarks 1.** Figures in parentheses apply to operation at $f_{XX}$ = 12.5 MHz.

**2.** ×: Don't care

## 24.3 HALT Mode

### 24.3.1 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 interrupts are enabled (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. Inserting NOP instructions may change, the order relationship between the interrupt servicing and instruction execution, so to prevent problems caused by changes in the execution order, be sure to take the measures described earlier.

The system clock when setting the HALT mode can be set to either the main system clock or the subsystem clock.

The operating states in the HALT mode are described next.

**Table 24-2. Operating States in HALT Mode**

| HALT Mode Setting / Item | | HALT Mode Setting During Main System Clock Operation | | HALT Mode Setting During Subsystem Clock Operation | |
|---|---|---|---|---|---|
| | | No subsystem clock **Note 1** | Subsystem clock **Note 2** | When the main system clock continues oscillating | When the main system clock stops oscillating |
| Clock generator | | Both the main system clock and subsystem clock can oscillate. The clock supply to the CPU stops. | | | |
| CPU | | Operation disabled | | | |
| Port (output latch) | | Holds the state before the HALT mode was set. | | | |
| 16-bit timer/counter | | Operation enabled | | | Operational when the watch timer output is selected as the count clock. (Select $f_{XT}$ as the count clock of the watch timer.) |
| 8-bit timer/counters 1, 2 | | Operation enabled | | | Operational when TI1 and TI2 are selected as the count clocks |
| 8-bit timer/counters 5, 6 | | Operation enabled | | | Operational when TI5 and TI6 are selected as the count clocks |
| Watch timer | | Operational when $f_{XX}/2^7$ is selected as the count clock | Operation enabled | | Operational when $f_{XT}$ is selected as the count clock |
| Watchdog timer | | Operation disabled (initializing counter) | | | |
| A/D converter | | Operation enabled | | | Operation disabled |
| D/A converter | | Operation enabled | | | |
| Real-time output port | | Operation enabled | | | |
| Serial interface | | Operation enabled | | | Operational during external SCK |
| External interrupt | INTP0 to INTP5 | Operation enabled | | | |
| Bus lines during external expansion | AD0 to AD7 | High impedance | | | |
| | A8 to A19 | Holds the state before the HALT mode was set. | | | |
| | ASTB | Low level | | | |
| | $\overline{WR}$, $\overline{RD}$ | High level | | | |
| | $\overline{WAIT}$ | Holds input status | | | |

**Notes 1.** Including when an external clock is not supplied.

**2.** Including when an external clock is supplied.

### 24.3.2 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 24-3 lists the release sources and describes the operation after release. Operations following the release of the HALT mode are also shown in Figure 24-5.

**Table 24-3. Releasing HALT Mode 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 | × | × | • Not executing a non-maskable interrupt service program<br>• Executing a low-priority non-maskable interrupt service program | Acknowledges interrupt requests |
| | | | • Executing the service program for the same request<br>• Executing a high-priority non-maskable interrupt service program | The instruction following the MOV STBC, #byte instruction is executed.<br>(The interrupt request that released the HALT mode is held pending[Note 3].) |
| Maskable interrupt request (except macro service request) | 0 | 1 | • Not executing an interrupt service program<br>• Executing a low-priority maskable interrupt service program<br>• 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<br>(This excludes executing an interrupt service program in priority level 3 when the PRSL bit[Note 4] is cleared to 0.)<br>• Executing a high-priority interrupt service program | The instruction following MOV STBC, #byte is executed.<br>(The interrupt request that released the HALT mode is held pending[Note 3].) |
| | 0 | 0 | – | |
| | 1 | × | – | Holds the HALT mode |
| Macro service request | 0 | × | – | Macro service processing execution<br>End condition is not satisfied → End<br>HALT mode condition is satisfied again<br>→ When VCIE[Note 5] = 1:<br>HALT mode again<br>When VCIE[Note 5] = 0:<br>Same as release by 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)

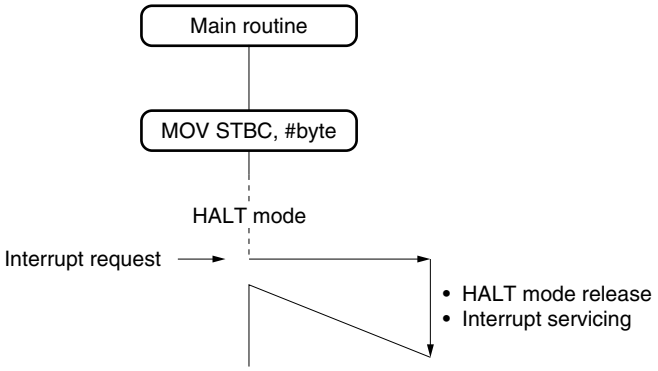
**3.** The pending interrupt request is acknowledged when acknowledgement is enabled.

**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 24-5. Operations After Releasing HALT Mode (1/4)**

**(1) Interrupt after HALT mode**




**(2) Reset after HALT mode**

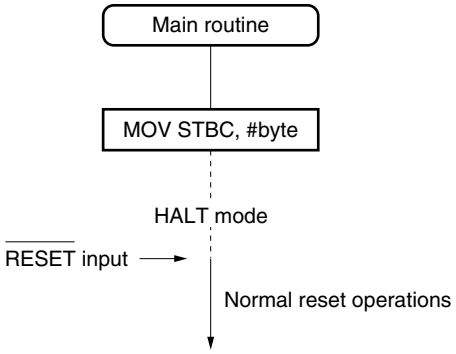
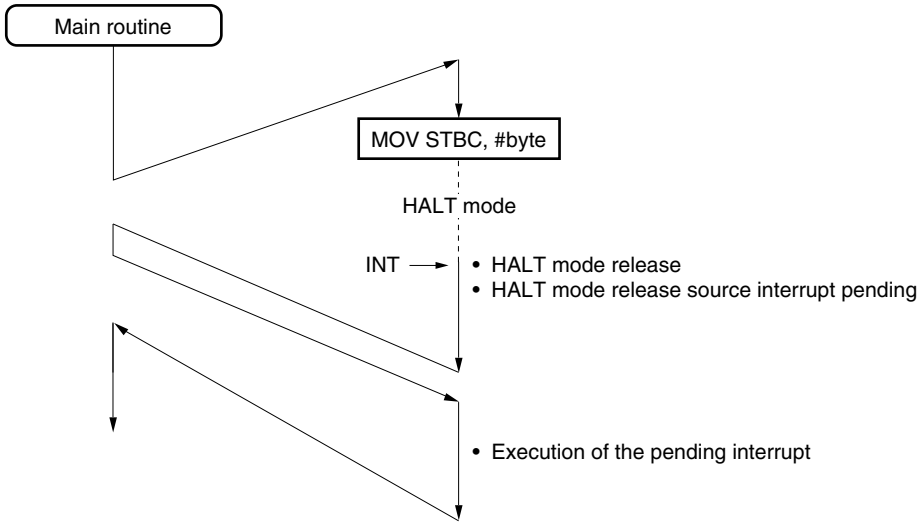


**Figure 24-5. Operations After Releasing HALT Mode (2/4)**

**(3) HALT mode during interrupt servicing routine whose priority is higher than or equal to release source interrupt**




**(4) HALT mode during interrupt servicing routine whose priority is lower than release source interrupt**

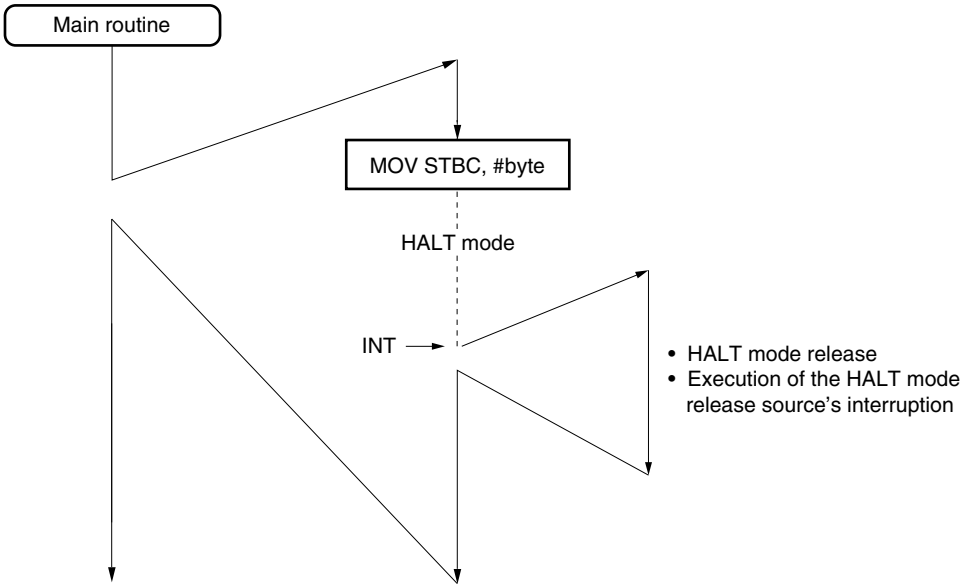
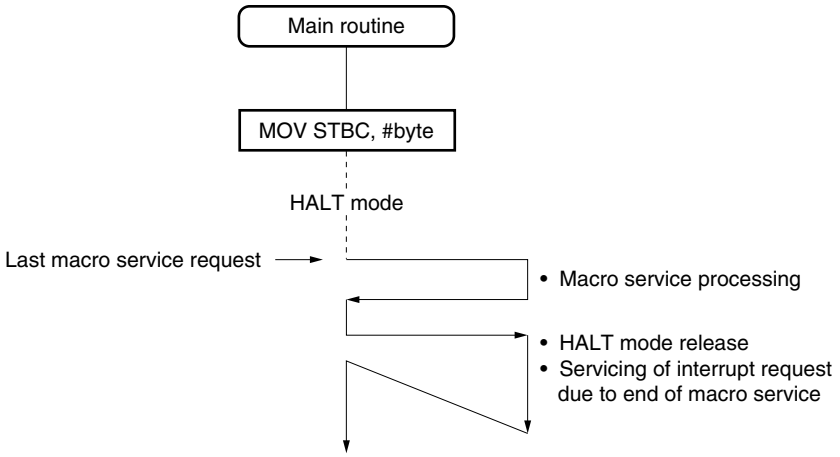


**Figure 24-5. Operations After Releasing HALT Mode (3/4)**

**(5) Macro service request in HALT mode**

**(a) Interrupt request is issued (VCIE = 0) immediately after macro service end condition is satisfied.**




**(b) Macro service end condition is not satisfied, or interrupt request is not issued (VCIE = 1) after macro service end condition is satisfied.**

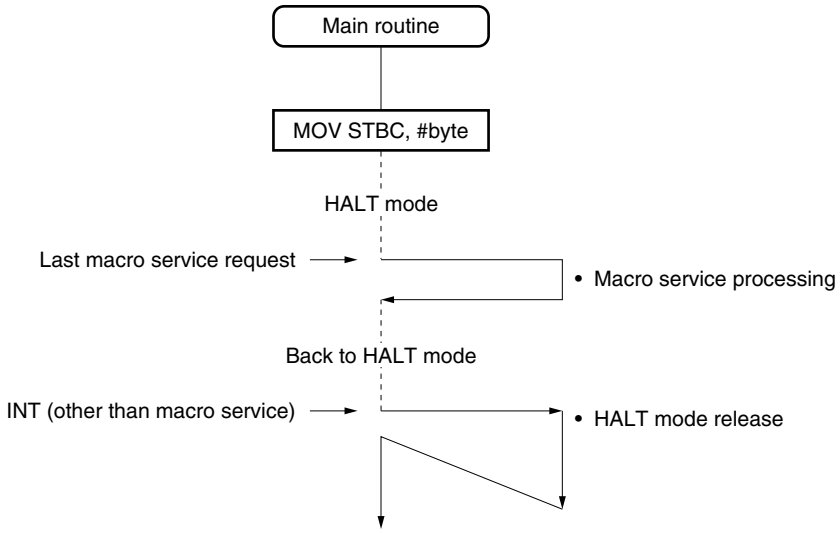
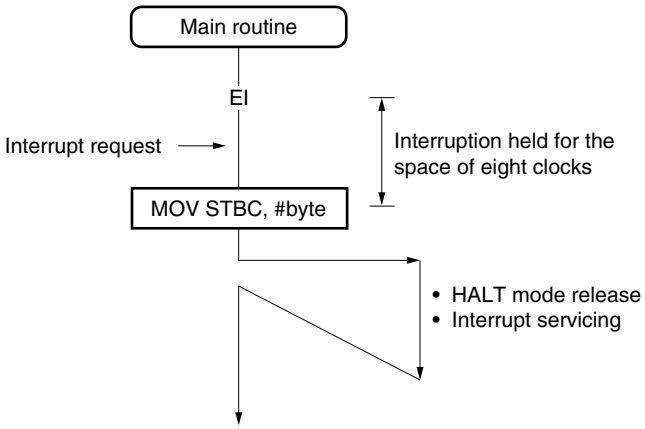
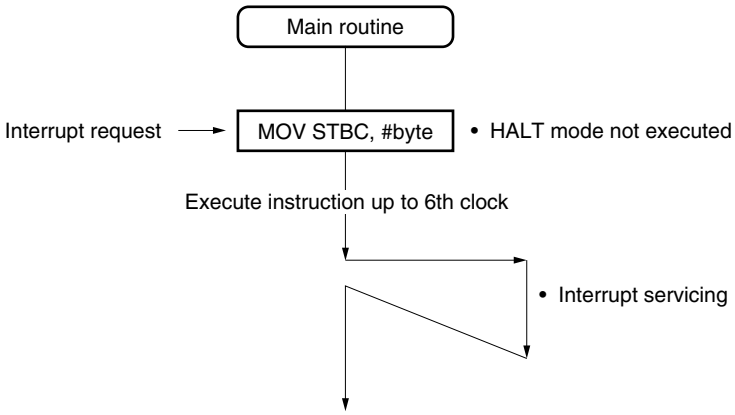


**Figure 24-5. Operations After Releasing HALT Mode (4/4)**

**(6) HALT mode which the interrupt is held, which is enabled in an instruction that interrupt requests are temporarily held**




**(7) Conflict between HALT mode setting instruction and interrupt**



**(1) Releasing HALT mode 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 acknowledgement.

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 acknowledgement is possible. For details about non-maskable interrupt acknowledgement, refer to **22.6 Non-Maskable Interrupt Acknowledgment Operation**.

**Caution The HALT mode cannot be released by the watchdog timer.**

**(2) Releasing HALT mode by a maskable interrupt request**

The HALT mode can only be released by a maskable interrupt request when that interrupt's 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 acknowledgement is not possible, execution restarts from the next instruction that sets the HALT mode. For details about interrupt acknowledgement, refer to **22.7 Maskable Interrupt Acknowledgment Operation**.

A macro service temporarily releases the HALT mode, performs one-time processing, and returns again to the HALT mode. If the macro service is only specified a few 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 24-4. 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 | • Not executing an interrupt service program<br>• Executing a low-priority maskable interrupt service program<br>• 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.)<br>• 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 held pending[Note 3].) |
| | 0 | 0 | – | |
| | 1 | × | – | Holds the HALT mode |
| Macro service request | 0 | × | – | Macro service processing execution End condition is not satisfied → End HALT mode condition is satisfied again<br>→ When VCIE[Note 5] = 1:<br>HALT mode again<br>When VCIE[Note 5] = 0:<br>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 pending interrupt request is acknowledged when acknowledgement is possible.

**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) Releasing HALT mode by $\overline{\text{RESET}}$ input**

After branching to the reset vector address as in a normal reset, the program is executed. However, the contents of the internal RAM are held at the value before the HALT mode was set.

## 24.4 STOP Mode

### 24.4.1 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 be written with 8-bit data by a special instruction. Therefore, the STOP mode is set by the MOV STBC, #byte instruction.

When interrupts are enabled (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. Inserting NOP instructions may change the order relationship between the interrupt servicing and instruction execution, so to prevent problems caused by changes in the execution order, be sure to take the measures described earlier.

The system clock when setting the STOP mode can only be set to the main system clock.

**Caution Since an interrupt request signal is used when releasing the standby mode, when an interrupt source that sets the interrupt request flag or resets the interrupt mask flag is generated, even though the standby mode is entered, it is immediately released. When the STOP mode setting instruction conflicts with the setting of an unmasked interrupt request flag or a non-maskable interrupt request, either of following two statuses are entered.**

**(1) Status in which STOP mode is set once, and then released**

**(2) Status in which STOP mode is not set**

**The oscillation stabilization time after releasing STOP mode is inserted only for the status in which STOP mode is set once and then released.**

The operating states in the STOP mode are described next.

**Table 24-5. Operating States in STOP Mode**

| STOP Mode Setting / Item | | With Subsystem Clock | Without Subsystem Clock |
|---|---|---|---|
| Clock generator | | Only main system clock stops oscillating | |
| CPU | | Operation disabled | |
| Port (output latch) | | Holds the state before the STOP mode was set | |
| 16-bit timer/counter | | Operational when the watch timer output is selected as the count clock (select $f_{XT}$ as the count clock of the watch timer) | Operation disabled |
| 8-bit timer/counters 1, 2 | | Operational only when TI1 and TI2 are selected as the count clocks | |
| 8-bit timer/counters 5, 6 | | Operational only when TI5 and TI6 are selected as the count clocks | |
| Watch timer | | Operational only when $f_{XT}$ is selected as the count clock | Operation disabled |
| Watchdog timer | | Operation disabled (initializing counter) | |
| A/D converter | | Operation disabled | |
| D/A converter | | Operation enabled | |
| Real-time output port | | Operational when an external trigger is used or TI1 and TI2 are selected as the count clocks of 8-bit timer counters 1 and 2 | |
| Serial interface | Except $I^2C$ bus mode | Operational only when an external input clock is selected as the serial clock | |
| | $I^2C$ bus mode | Operation disabled | |
| External interrupt | INTP0 to INTP5 | Operation enabled | |
| Bus lines during external expansion | AD0 to AD7 | High impedance | |
| | A8 to A19 | High impedance | |
| | ASTB | High impedance | |
| | $\overline{WR}$, $\overline{RD}$ | High impedance | |
| | $\overline{WAIT}$ | Holds input status | |

**Caution In the STOP mode, only external interrupts (INTP0 to INTP5) and the watch timer interrupt (INTWT) can release the STOP mode and be acknowledged. All other interrupt requests are held pending, and acknowledged after the STOP mode has been released through NMI input, INTP0 to INTP5 input or INTWT.**

### 24.4.2 Releasing STOP mode

The STOP mode is released by NMI input, INTP0 to INTP5 input, the watch timer interrupt (INTWT), or $\overline{\text{RESET}}$ input.

An outline of the release sources and operations following release are shown in Table 24-6. Operations following release of the STOP mode are also shown in Figure 24-6.

**Table 24-6. 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 | × | × | × | • Not executing a non-maskable interrupt service program<br>• Executing a low-priority non-maskable interrupt service program | Acknowledges interrupt requests |
| | | | | • Executing the service program for the NMI pin input<br>• 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 held pending[Note 4].) |
| INTP0 to INTP5 pin input, watch timer interrupt | 0 | 0 | 1 | • Not executing an interrupt service program<br>• Executing a low-priority maskable interrupt service program<br>• The PRSL bit[Note 5] is cleared to 0 while an interrupt service program at priority level 3 is being executed. | 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 5] is cleared to 0.)<br>• Executing a high-priority interrupt service program | The instruction following MOV STBC, #byte instruction is executed. (The interrupt request that released the STOP mode is held pending[Note 4].) |
| | 0 | 0 | 0 | – | |
| | 1 | 0 | × | – | Holds the STOP mode |
| | × | 1 | × | | |

**Notes 1.** Interrupt mask bit in each interrupt request source

**2.** Macro service enable flag that is in each interrupt request source

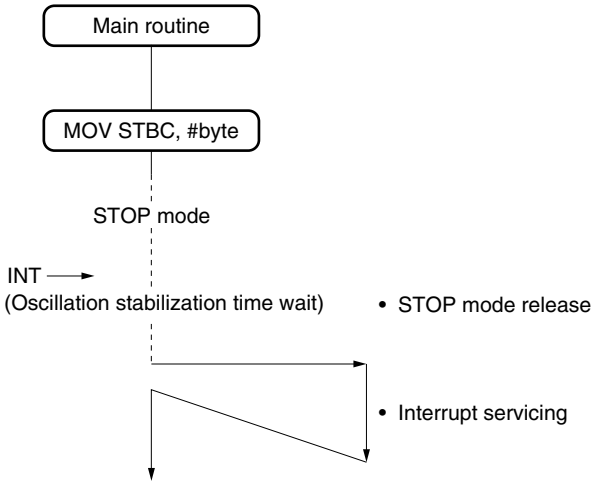
**3.** Interrupt enable flag in the program status word (PSW)

**4.** The pending interrupt request is acknowledged when acknowledgement is possible.

**5.** Bit in the interrupt mode control register (IMC)

**Figure 24-6. Operations After Releasing STOP Mode (1/3)**

**(1) Interrupt after STOP mode**




**(2) Reset after STOP mode**

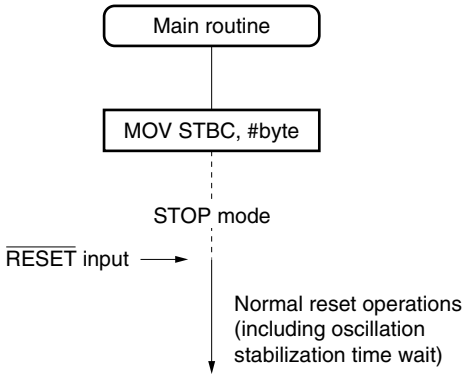
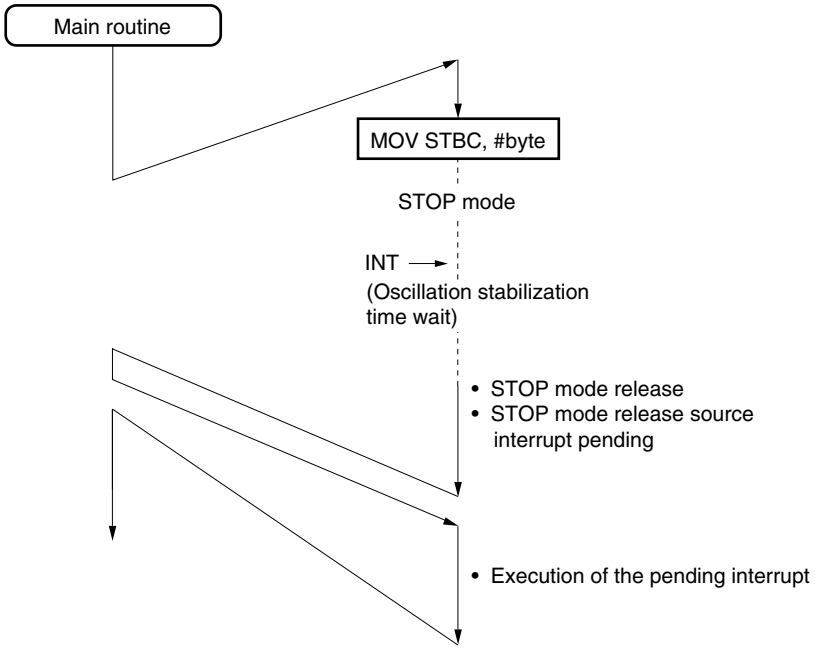


**Figure 24-6. Operations After Releasing STOP Mode (2/3)**

**(3) STOP mode during servicing routine of interrupt whose priority is higher than or equal to release source interrupt**




**(4) STOP mode during servicing routine of interrupt whose priority is lower than release source interrupt**

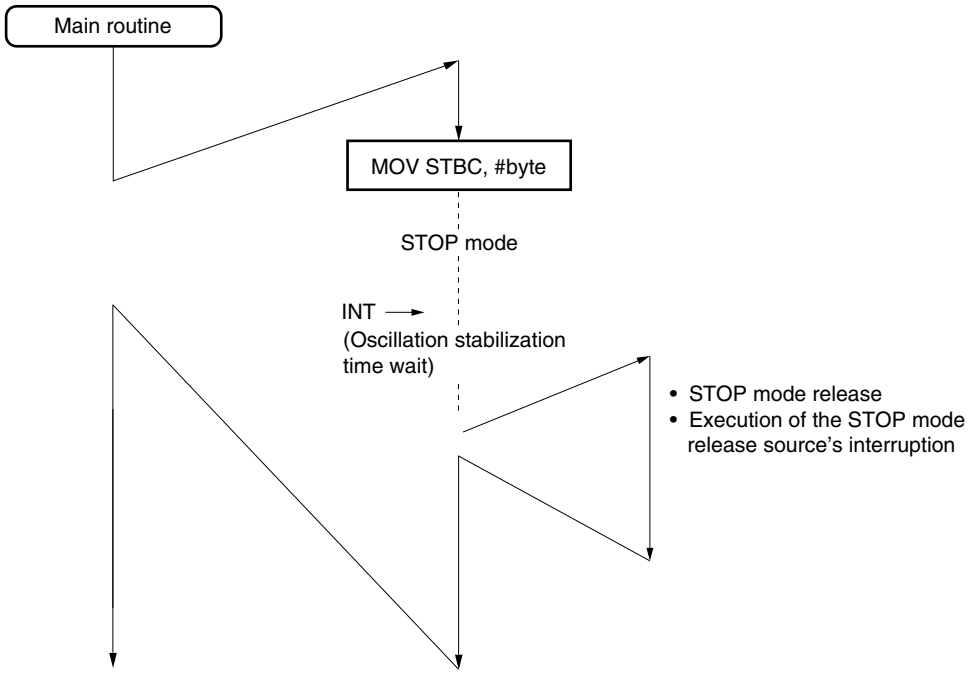
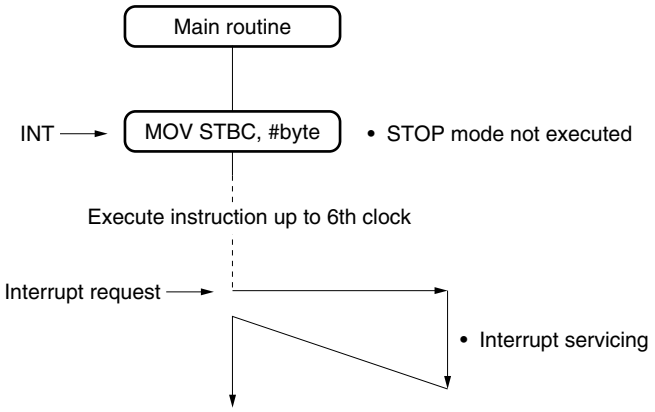


**Figure 24-6. Operations After Releasing STOP Mode (3/3)**

**(5) Conflict between STOP mode setting instruction and interrupt**



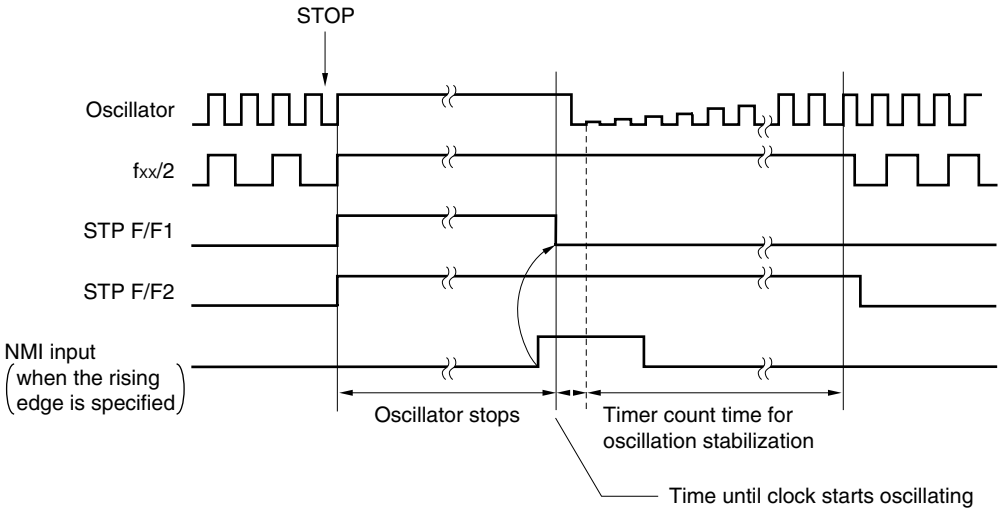
**(1) Releasing STOP mode by NMI input**

When the valid edge specified by external interrupt edge enable register 0 (EGP0, EGN0) is input by NMI input, the oscillator starts oscillating again. The STOP mode is then released after the oscillation stabilization time set by the oscillation stabilization time specification register (OSTS) has elapsed.

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 acknowledgement is not possible (such as when the STOP mode has been set in the NMI interrupt service program), execution starts again from the instruction following the instruction that set the STOP mode. When acknowledgement is enabled, execution branches to the NMI interrupt service program (by executing the RETI instruction).

For details of NMI interrupt acknowledgment, refer to **22.6 Non-Maskable Interrupt Acknowledgment Operation**.

**Figure 24-7. Releasing STOP Mode by NMI Input**



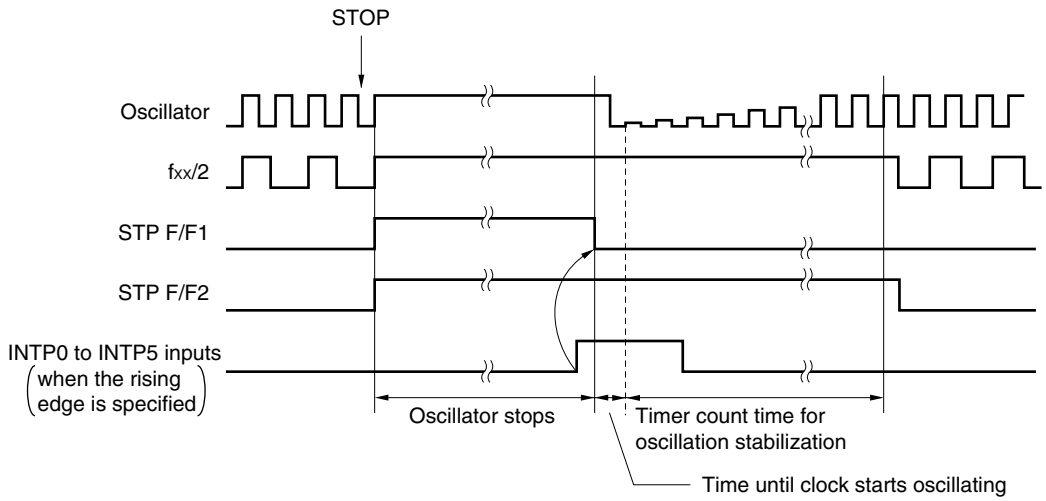
**(2) Releasing STOP mode by INTP0 to INTP5 input and watch timer interrupt**

If interrupt masking is released through INTP0 to INTP5 input and macro servicing is disabled, the oscillator restarts oscillating when the valid edge specified by external interrupt edge enable register 0 (EGP0, EGN0) is input to INTP0 to INTP5. At the same time, a watch timer overflow will occur and the STOP mode will be released when the watch timer interrupt mask is released and macro services are disabled.

If interrupts can be acknowledged when the STOP mode is released and the interrupt enable flag (IE) is set to 1, execution branches to the interrupt service program. If the IE flag is cleared to 0 when acknowledgement is not possible, execution starts again from the instruction following the instruction that set the STOP mode.

For details of interrupt acknowledgement, refer to **22.7 Maskable Interrupt Acknowledgment Operation**.

**Figure 24-8. Example of Releasing STOP Mode by INTP0 to INTP5 Input**




**(3) Releasing STOP mode by $\overline{\text{RESET}}$ input**

When the $\overline{\text{RESET}}$ input rises from low to high and the reset is released, the oscillator starts oscillating. Oscillation stops for the $\overline{\text{RESET}}$ active period. After the oscillation stabilization time elapses, normal operation starts.

The difference from the normal reset operation is that the data memory saves the contents before setting the STOP mode.

## 24.5 IDLE Mode

### 24.5.1 Settings and operating states of IDLE mode

The IDLE mode is set by setting both the STP and HLT bits 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 interrupts are enabled (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. Inserting NOP instructions may change the order relationship between the interrupt servicing and the instruction execution, so to prevent problems caused by changes in the execution order, be sure to take the measures described earlier.

The system clock when setting the IDLE mode can be set to either the main system clock or the subsystem clock.

The operating states in the IDLE mode are described next.

**Table 24-7. Operating States in IDLE Mode**

| IDLE Mode Setting / Item | | With Subsystem Clock | Without Subsystem Clock |
|---|---|---|---|
| Clock generator | | The oscillators for both the main system clock and subsystem clock continue operating. The clock supply to both the CPU and peripherals is stopped. | |
| CPU | | Operation disabled | |
| Port (output latch) | | Holds the state before the IDLE mode was set | |
| 16-bit timer/event counter | | Operational when the watch timer output is selected as the count clock (select $f_{XT}$ as the count clock of the watch timer.) | Operation disabled |
| 8-bit timer/event counters 1, 2 | | Operational only when TI1 and TI2 are selected as the count clocks | |
| 8-bit timers 5 and 6 | | Operational only when TI5 and TI6 are selected as the count clocks | |
| Watch timer | | Operational only when $f_{XT}$ is selected as the count clock | Operation disabled |
| Watchdog timer | | Operation disabled | |
| A/D converter | | Operation disabled | |
| D/A converter | | Operation enabled | |
| Real-time output port | | Operational when an external trigger is used or TI1 and TI2 are selected as the count clocks of 8-bit timer/event counters 1 and 2 | |
| Serial interface | Except $I^2C$ bus mode | Operational only when an external input clock is selected as the serial clock | |
| | $I^2C$ bus mode | Operation disabled | |
| External interrupt | INTP0 to INTP5 | Operation enabled | |
| Bus lines during external expansion | AD0 to AD7 | High impedance | |
| | A8 to A19 | High impedance | |
| | ASTB | High impedance | |
| | $\overline{WR}$, $\overline{RD}$ | High impedance | |
| | $\overline{WAIT}$ | Holds input status | |

**Caution In the IDLE mode, only external interrupts (INTP0 to INTP5) and the watch timer interrupt (INTWT) can release the IDLE mode and be acknowledged as interrupt requests. All other interrupt requests are held pending, and acknowledged after the IDLE mode has been released through NMI input, INTP0 to INTP5 input or INTWT.**

### 24.5.2 Releasing IDLE mode

The IDLE mode is released by NMI input, INTP0 to INTP5 input, the watch timer interrupt (INTWT), or $\overline{\text{RESET}}$ input.

An outlines of the release sources and operations following release are shown in Table 24-8. Operations following release of the IDLE mode are also shown in Figure 24-9.

**Table 24-8. 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 | × | × | × | • Not executing a non-maskable interrupt service program<br>• Executing a low-priority non-maskable interrupt service program | Acknowledges interrupt requests |
| | | | | • Executing the service program for the NMI pin input<br>• 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 held pending[Note 4].) |
| INTP0 to INTP5 pin input, watch timer interrupt | 0 | 0 | 1 | • Not executing an interrupt service program<br>• Executing a low-priority maskable interrupt service program<br>• The PRSL bit[Note 5] 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 5] is cleared to 0.)<br>• Executing a high-priority interrupt service program | Execute the instruction following the MOV STBC, #byte instruction. (The interrupt request that released the IDLE mode is held pending[Note 4].) |
| | 0 | 0 | 0 | – | |
| | 1 | 0 | × | – | Holds the IDLE mode |
| | × | 1 | × | | |

**Notes 1.** Interrupt mask bit in each interrupt request source

**2.** Macro service enable flag that is in each interrupt request source

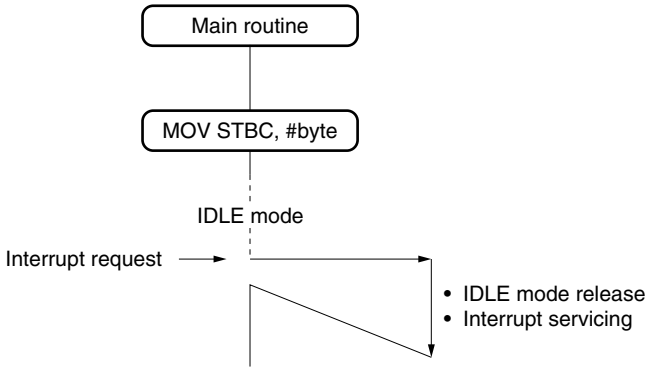
**3.** Interrupt enable flag in the program status word (PSW)

**4.** The pending interrupt request is acknowledged when acknowledgement is possible.

**5.** Bit in the interrupt mode control register (IMC)

**Figure 24-9. Operations After Releasing IDLE Mode (1/2)**

**(1) Interrupt after IDLE mode**




**(2) Reset after IDLE mode**

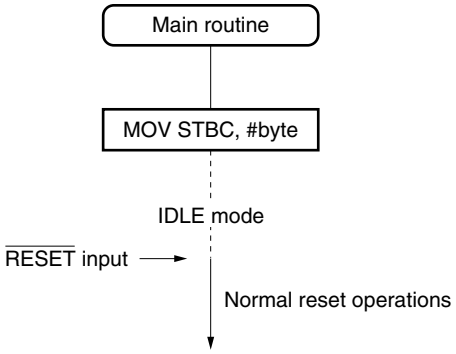
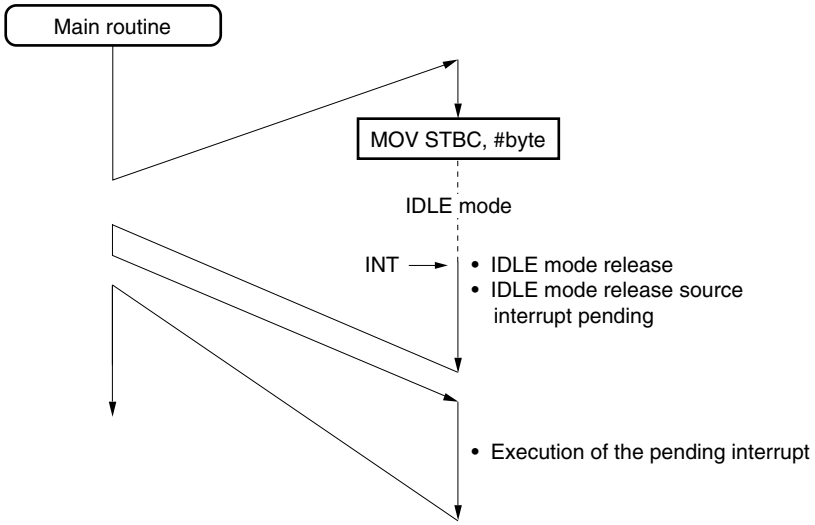


**Figure 24-9. Operations After Releasing IDLE Mode (2/2)**

**(3) IDLE mode during interrupt servicing routine whose priority is higher than or equal to release source interrupt**




**(4) IDLE mode during interrupt servicing routine whose priority is lower than release source interrupt**

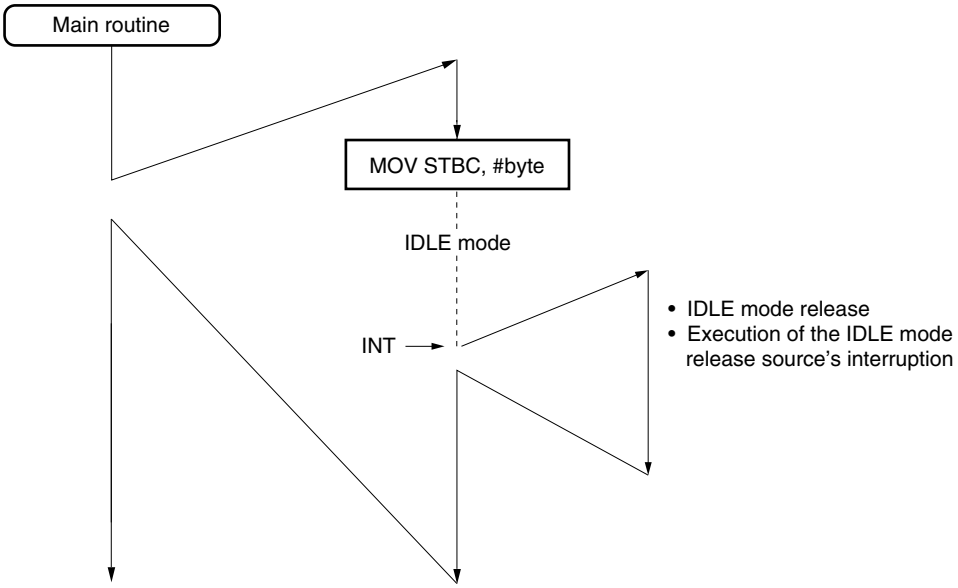
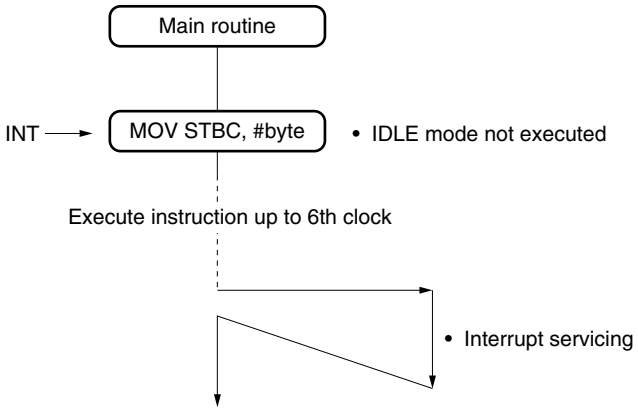


**Figure 24-9. Operations After Releasing IDLE Mode (3/3)**

**(5) Conflict between IDLE mode setting instruction and interrupt**



**(1) Releasing IDLE mode by NMI input**

When the valid edge specified by external interrupt edge enable register 0 (EGP0, EGN0) is input by NMI input, the IDLE mode is released.

When the IDLE mode is released and non-maskable interrupts from the NMI pin input can be acknowledged, execution branches to the NMI interrupt service program. If acknowledgement is not possible (such as when the IDLE mode has been set in the NMI interrupt service program), execution starts again from the instruction following the instruction that set the IDLE mode. When acknowledgement is enabled, execution branches to the NMI interrupt service program (by executing the RETI instruction).

For details of NMI interrupt acknowledgement, refer to **22.6 Non-Maskable Interrupt Acknowledgment Operation**.

**(2) Releasing IDLE mode by INTP0 to INTP5 input and watch timer interrupt**

If interrupt masking by INTP0 to INTP5 input is released and macro servicing is disabled and the valid edge specified by external interrupt edge enable register 0 (EGP0, EGN0) is input to INTP0 to INTP5, the IDLE mode is released. At the same time, a watch timer overflow will occur and the IDLE mode will be released when the watch timer interrupt mask is released and macro services are disabled.

If interrupts can be acknowledged when the IDLE mode is released and the interrupt enable flag (IE) is set to 1, execution branches to the interrupt service program. If the IE flag is cleared to 0 when acknowledgement is not possible, execution starts again from the instruction following the instruction that set the IDLE mode.

For details of interrupt acknowledgement, refer to **22.7 Maskable Interrupt Acknowledgment Operation**.

**(3) Releasing of IDLE mode by $\overline{\text{RESET}}$ input**

When the $\overline{\text{RESET}}$ input rises from low to high and the reset condition is released, the oscillator starts oscillating. Oscillation stops for the $\overline{\text{RESET}}$ active period. After the oscillation stabilization time elapses, normal operation starts.

The difference from the normal reset operation is the data memory saves the contents before setting the IDLE mode.

## 24.6 Check Items When Using STOP or IDLE Mode

The following points must be checked to decrease the current consumption when using the STOP mode or IDLE mode.

**(1) Is the output level of each output pin appropriate?**

The appropriate output level of each pin differs depending on the circuit in the next stage. Select the output level so that the current consumption 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 current consumption 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 suppress the current consumption 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 current consumption sometimes increases when a low level is input. In this case, output a high level or high impedance to reduce the current consumption.

- 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 current consumption 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 pull-up or pull-down resistors.

The setting method for the output level differs depending on 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 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 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 a range from the $V_{SS}$ voltage to the $V_{DD}$ voltage. If a voltage outside of this range is applied, not only does the current consumption increase, but the reliability of the $\mu$PD784225 is negatively affected.

In addition, do not increase the middle voltage.

**(3) Are internal pull-up resistors needed?**

Unnecessary pull-up resistors increase the current consumption and are another cause of device latch-up. Set the pull-up resistors to the mode in which they are used only for the required parts.

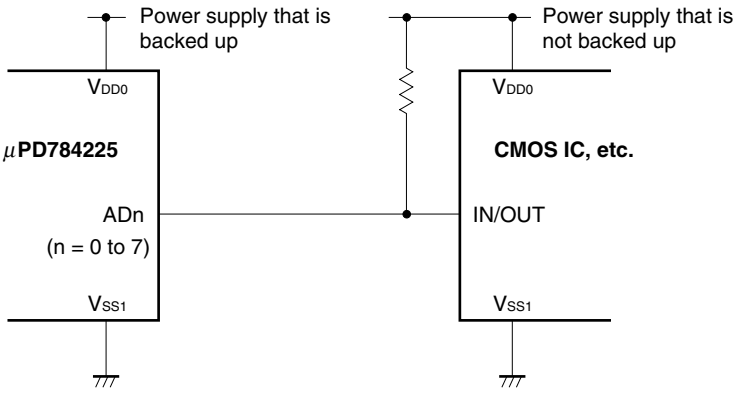
When the parts needing pull-up resistors and the parts not needing them are mixed together, externally connect the pull-up resistors where they are needed and set the mode in which the internal pull-up resistors are not used.

**(4) Are the address bus, the address/data bus, etc. handled appropriately?**

The address bus, address/data bus, and $\overline{RD}$ and $\overline{WR}$ pins have a high impedance in the STOP and IDLE modes. Normally, these pins are pulled up by pull-up resistors. If the pull-up resistors are connected to the power supply that is backed up, the current flows through the pull-up resistors when the low input impedance of the circuit connected to the power supply that is not backed up, and the current consumption increases. Therefore, connect the pull-up resistors on the power supply side that is not backed up as shown in Figure 24-10.

The ASTB pin has a high impedance in both the STOP and IDLE modes. Handle in the manner described in (1) above.

**Figure 24-10. Example of Handling Address/Data Bus**




Set the input voltage level applied to the $\overline{WAIT}$ pin in a range from the $V_{SS1}$ voltage to the $V_{DD0}$ voltage. If a voltage outside of this range is applied, not only does the current consumption increase, but the reliability of the μPD784225 is negatively affected.

**(5) A/D converter**

The current flowing through the $AV_{DD}$ pin can be reduced by clearing the ADCS bit, that is bit 7 in the A/D converter mode register (ADM), to 0.

The $AV_{DD}$ pin must always have the same voltage as the $V_{DD}$ pin. If current is not supplied to the $AV_{DD}$ pin in the STOP mode, not only does the current consumption increase, but the reliability of the μPD784225 is negatively affected.

**(6) D/A converter**

The D/A converter consumes a constant current in the STOP and IDLE modes. By clearing the DACEn (n = 0, 1) bits in the D/A converter mode registers (DAM0, DAM1) to 0, the output of ANOn (n = 0, 1) has a high impedance, and the current consumption can be decreased.

Do not apply an external voltage to the ANOn pins. If an external voltage is applied, not only is the current consumption increased, but the μPD784225 may be destroyed or the reliability decreased.

## 24.7 Low Power Consumption Mode

### 24.7.1 Setting low power consumption mode[Note]

When the low power consumption mode is entered, set 70H in the standby control register (STBC). This setting switches the system clock from the main system clock to the subsystem clock.

Whether the system clock switched to the subsystem clock can be verified from the data read from the CST bit in the clock status register (PCS) (refer to **Figure 24-3**).

To check whether switching has ended, set 74H in STBC to stop the oscillation of the main system clock. Then switch to the backup power supply from the main power supply.

**Note** The low power consumption mode is a state in which the subsystem clock is used as the system clock, and the main system clock is stopped.

Figure 24-11 shows the flow for setting subsystem clock operation. Figure 24-12 shows the setting timing diagram.

**Figure 24-11. Flow for Setting Subsystem Clock Operation**

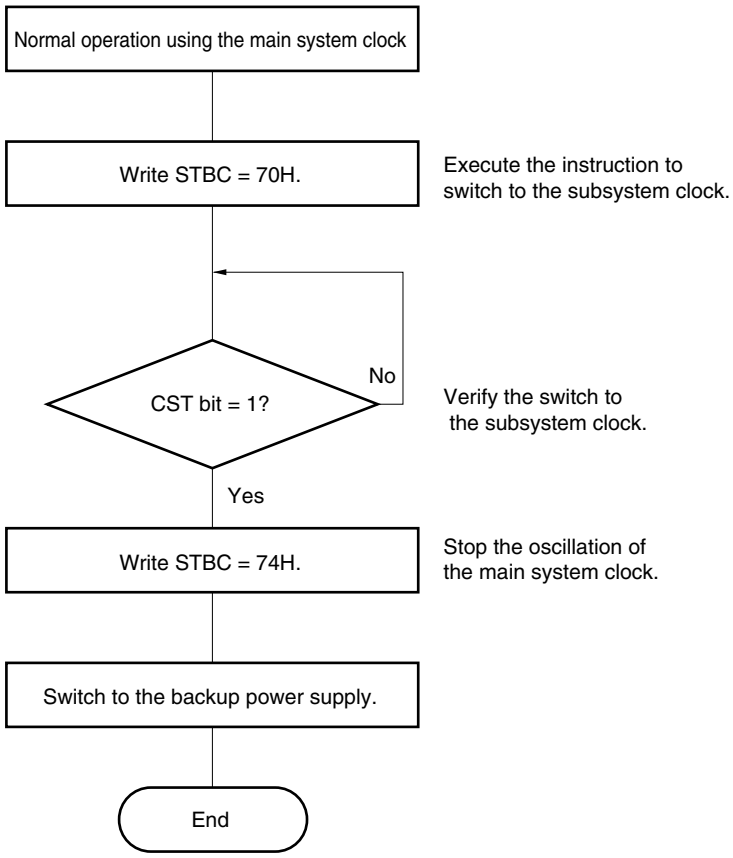
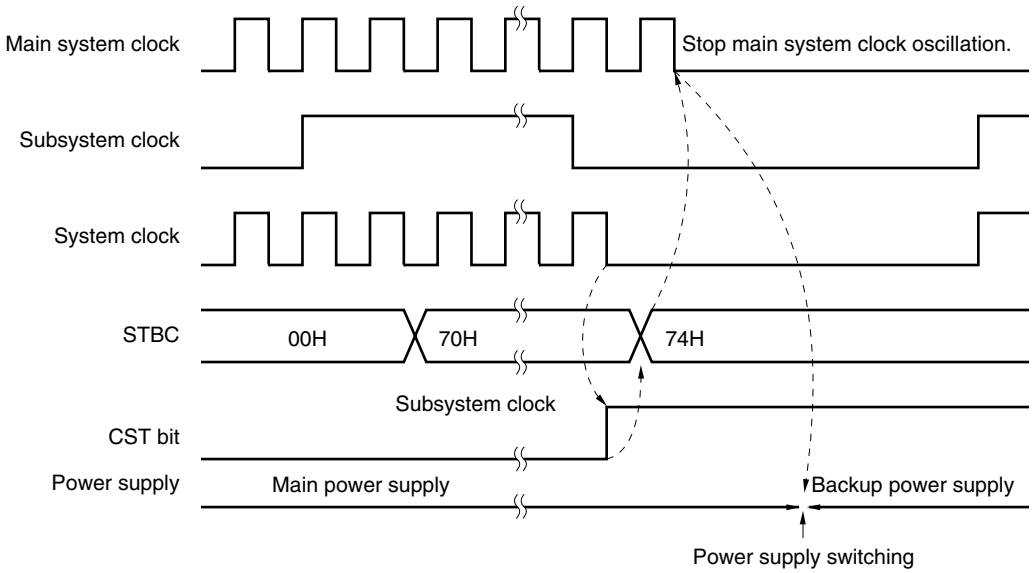


**Figure 24-12. Setting Timing for Subsystem Clock Operation**




### 24.7.2 Returning to main system clock operation

When returning to main system clock operation from subsystem clock operation, the system power supply first switches to the main power supply and enables the oscillation of the main system clock (set STBC = 70H). Then the software waits for the oscillation stabilization time of the main system clock, and the system clock switches to the main system clock (set STBC to 00H).

**Cautions 1. When returning from subsystem clock operation (stopped oscillation of the main system clock) to main system clock operation, do not simultaneously specify MCK = 0 and CK2 = 0 by write instructions to STBC.**

**2. The oscillation stabilization time specification register (OSTS) specifies the oscillation stabilization time after the STOP mode is released, except when released by $\overline{\text{RESET}}$, when the system clock is the main system clock. This cannot be used when the system clock is restored from the subsystem clock to the main system clock.**

Figure 24-13 is the flow for restoring main system clock operation, and Figure 24-14 is the restore timing diagram.

**Figure 24-13. Flow to Restore Main System Clock Operation**

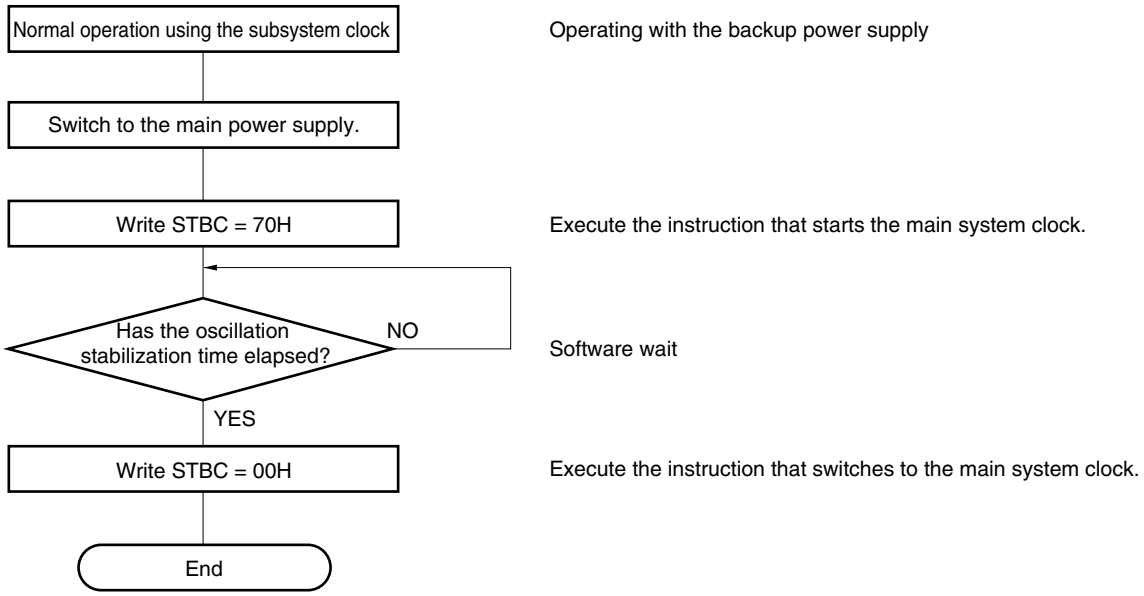
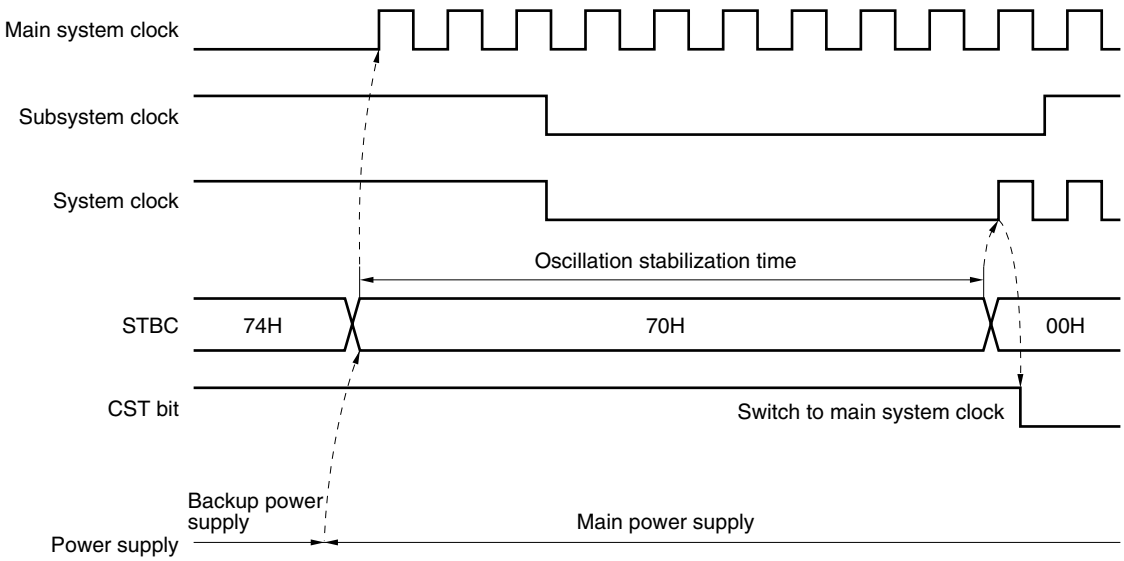



**Figure 24-14. Timing for Restoring Main System Clock Operation**




### 24.7.3 Standby function in low power consumption mode

The standby function in the low power consumption mode has a HALT mode and an IDLE mode.

### (1) HALT mode

#### (a) Settings and operating states of HALT mode

When the HALT mode is set in the low power consumption mode, set 75H in STBC. Table 24-9 shows the operating states in the HALT mode.

**Table 24-9. Operating States in HALT Mode**

| Item | | Operating State |
|---|---|---|
| Clock generator | | The clock supplied to the CPU stops, and only the main system clock stops oscillating. |
| CPU | | Operation disabled |
| Port (output latch) | | Holds the state before the HALT mode was set. |
| 16-bit timer/event counter | | Operational when the watch timer output is selected as the count clock (select $f_{XT}$ as the count clock of the watch timer) |
| 8-bit timer/event counters 1, 2 | | Operational when TI1 and TI2 are selected as the count clocks |
| 8-bit timers 5 and 6 | | Operational when TI5 and TI6 are selected as the count clocks |
| Watch timer | | Operational only when $f_{XT}$ is selected as the count clock |
| Watchdog timer | | Operation disabled (counter is initialized) |
| A/D converter | | Operation disabled |
| D/A converter | | Operation enabled |
| Real-time output port | | Operational when an external trigger is used or TI1 and TI2 are selected as the count clocks of 8-bit timer counters 1 and 2 |
| Serial interface | Except $I^2C$ bus mode | Operational only when an external input clock is selected as the serial clock |
| | $I^2C$ bus mode | Operation disabled |
| External interrupt | INTP0 to INTP5 | Operation enabled |
| Bus lines during external expansion | AD0 to AD7 | High impedance |
| | A8 to A19 | Holds the state before the HALT mode was set |
| | ASTM | Low level |
| | $\overline{WR}$, $\overline{RD}$ | High level |
| | $\overline{WAIT}$ | Input state is retained |

**(b) Releasing the HALT mode**

**(i) Releasing HALT mode by NMI input**

When the valid edge specified by external interrupt edge enable register 0 (EGP0, EGN0) is input to the NMI input, the HALT mode is released.

When the HALT mode is released, if non-maskable interrupts by NMI pin input can be acknowledged, execution branches to the NMI interrupt service program. If interrupts cannot be acknowledged (when the HALT mode has been set in the NMI interrupt service program), execution starts again from the instruction following the instruction that set the HALT mode. When interrupts can be acknowledged (by executing the RETI instruction), execution branches to the NMI interrupt service program.

For details of NMI interrupt acknowledgement, refer to **22.6 Non-Maskable Interrupt Acknowledgment Operation**.

**(ii) Releasing HALT mode by a maskable interrupt request**

An unmasked maskable interrupt request is generated to release the HALT mode.

When the HALT mode is released and the interrupt enable flag (IE) is set to 1, if interrupts can be acknowledged, execution branches to interrupt service program. When interrupts cannot be acknowledged and when the IE flag is cleared to 0, execution restarts from the instruction following the instruction that set the HALT mode.

For details of interrupt acknowledgement, refer to **22.7 Maskable Interrupt Acknowledgment Operation**.

**(iii) Releasing HALT mode by $\overline{\text{RESET}}$ input**

When the $\overline{\text{RESET}}$ input rises from low to high and the reset condition is released, the oscillator starts oscillating. Oscillation stops for the $\overline{\text{RESET}}$ active period. After the oscillation stabilization time elapses, normal operation starts.

The difference from the normal reset operation is the data memory saves the contents before setting the HALT mode.

### (2) IDLE mode

#### (a) Settings and operating states of IDLE mode

When the low power consumption mode is set in the IDLE mode, set 77H in STBC.

Table 24-10 shows the operating states in the IDLE mode.

**Table 24-10. Operating States in IDLE Mode**

| Item | | Operating State |
|---|---|---|
| Clock generator | | The main system clock stops oscillating. The oscillator of the subsystem clock continues operating. The clock supplied to the CPU and the peripherals stops. |
| CPU | | Operation disabled |
| Port (output latch) | | Holds the state before the IDLE mode was set |
| 16-bit timer/event counter | | Operational when the watch timer output is selected as the count clock (select $f_{XT}$ as the count clock of the watch timer) |
| 8-bit timer/event counters 1, 2 | | Operational when TI1 and TI2 are selected as the count clocks |
| 8-bit timers 5 and 6 | | Operational when TI5 and TI6 are selected as the count clocks |
| Watch timer | | Operational only when $f_{XT}$ is selected as the count clock |
| Watchdog timer | | Operation disabled |
| A/D converter | | Operation disabled |
| D/A converter | | Operation enabled |
| Real-time output port | | Operational when an external trigger is used or TI1 and TI2 are selected as the count clocks of 8-bit timer counters 1 and 2 |
| Serial interface | Except $I^2C$ bus mode | Operational only when an external input clock is selected as the serial clock |
| | $I^2C$ bus mode | Operation disabled |
| External interrupt | INTP0 to INTP5 | Operation enabled |
| Bus lines during external expansion | AD0 to AD7 | High impedance |
| | A8 to A19 | High impedance |
| | ASTB | High impedance |
| | $\overline{WR}$, $\overline{RD}$ | High impedance |
| | $\overline{WAIT}$ | Input state is retained |

**Caution In the IDLE mode, only external interrupts (INTP0 to INTP5) and the watch timer interrupt (INTWT) can release the IDLE mode and be acknowledged as interrupt requests. All other interrupt requests are held pending, and acknowledged after the IDLE mode has been released through NMI input, INTP0 to INTP5 input, and INTWT.**

**(b) Releasing the IDLE mode**

**(i) Releasing IDLE mode by NMI input**

When the valid edge set by external interrupt edge enable register 0 (EGP0, EGN0) is input by NMI input, the IDLE mode is released.

When the IDLE mode is released and non-maskable interrupts by NMI pin input can be acknowledged, execution branches to the NMI interrupt service program. When interrupts cannot be acknowledged (when the IDLE mode has been set in the NMI interrupt service program), execution restarts from the instruction following the instruction that set the IDLE mode. When interrupts can be acknowledged (by executing the RETI instruction), execution branches to the NMI interrupt service program.

For details of NMI interrupt acknowledgement, refer to **22.6 Non-Maskable Interrupt Acknowledgment Operation**.

**(ii) Releasing IDLE mode by INTP0 to INTP5 input and watch timer interrupt**

If interrupt masking is released through INTP0 to INTP5 input and macro servicing is disabled, the oscillator restarts oscillating when the valid edge specified by external interrupt edge enable register 0 (EGP0, EGN0) is input to INTP0 to INTP5. At the same time, a watch timer overflow will occur and the IDLE mode will be released when the watch timer interrupt mask is released and macro services are disabled.

When the IDLE mode is released and the interrupt enable flag (IE) is set to 1, if interrupts can be acknowledged, execution branches to the interrupt service program. When interrupts cannot be acknowledged and when the IE flag is cleared to 0, execution restarts from the instruction following the instruction that set the IDLE mode.

For details of interrupt acknowledgement, refer to **22.7 Maskable Interrupt Acknowledgment Operation**.

**(iii) Releasing IDLE mode by $\overline{\text{RESET}}$ input**

When the $\overline{\text{RESET}}$ input rises from low to high and the reset condition is released, the oscillator starts oscillating. Oscillation stops for the $\overline{\text{RESET}}$ active period. After the oscillation stabilization time elapses, normal operation starts.

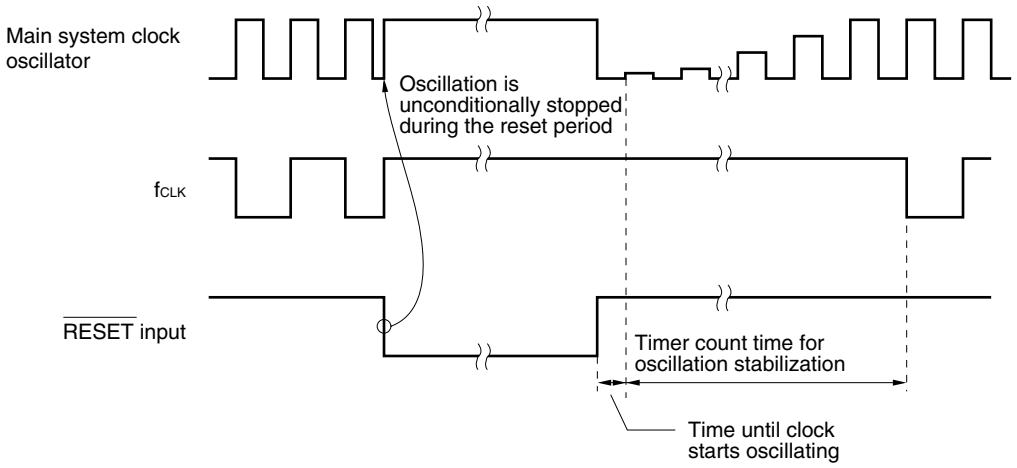
The difference from the normal reset operation is the data memory saves the contents before setting the IDLE mode.

# CHAPTER 25 RESET FUNCTION

When a low level is input to the $\overline{\text{RESET}}$ input pin, a system reset is performed. The hardware enters the states listed in Table 25-1. Since the oscillation of the main system clock unconditionally stops during the reset period, the current consumption of the entire system can be reduced.

When the $\overline{\text{RESET}}$ input goes from low to high, the reset state is released. After the count time of the timer for oscillation stabilization (41.9 ms: at 12.5 MHz operation), the contents of the reset vector table are set in the program counter (PC). Execution branches to the address set in the PC, and program execution starts from the branch destination address. Therefore, a reset can be started from any address.

**Figure 25-1. Oscillation of Main System Clock in Reset Period**




To prevent erroneous operation caused by noise, a noise eliminator based on analog delay is incorporated at the $\overline{\text{RESET}}$ input pin.

**Figure 25-2. Receiving Reset Signal**

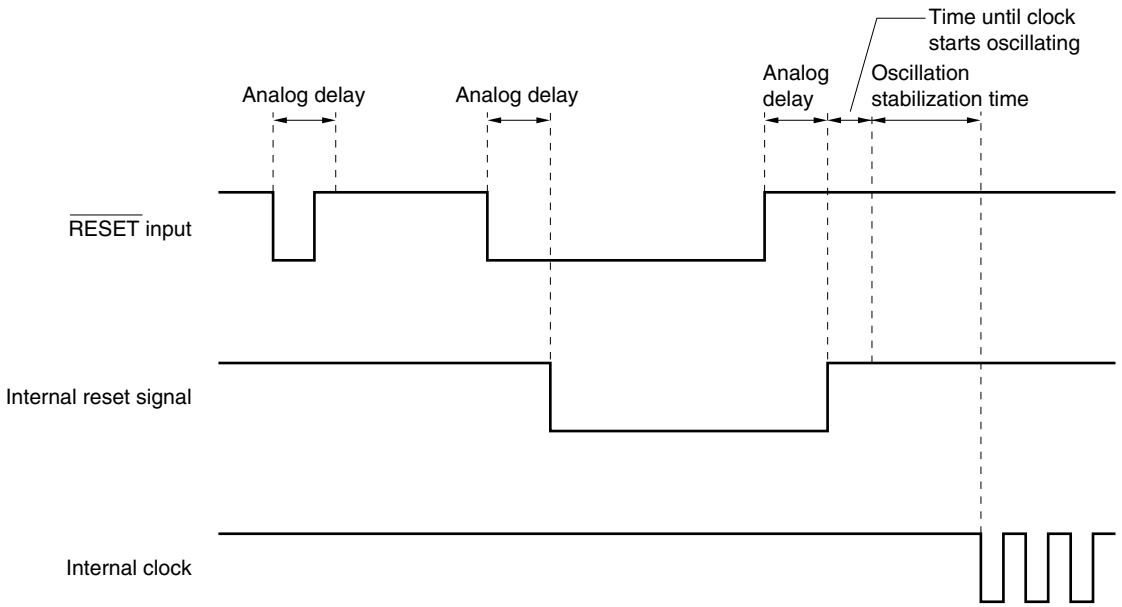



**Table 25-1. State of Hardware During and After Reset**

| Hardware | State During Reset ($\overline{\text{RESET}}$ = L) | State After Reset ($\overline{\text{RESET}}$ = H) |
|---|---|---|
| Main system clock oscillator | Oscillation stops | Oscillation starts |
| Subsystem clock oscillator | Not affected by reset | |
| Program counter (PC) | Undefined | Value in reset vector table set. |
| Stack pointer (SP) | Undefined | |
| Program status word (PSW) | Initialized to 0000H. | |
| Internal RAM | Undefined. However, when the standby state is released by a reset, the value is saved before setting standby. | |
| I/O lines | Input and output buffers off. | High impedance |
| Other hardware | Initialized to the fixed state**Note**. | |

**Note** See **Table 3-6 Special Function Register (SFR) List** when resetting.

# CHAPTER 26 ROM CORRECTION

## 26.1 ROM Correction Functions

In the μPD784224, 784225, 784224Y and 784225Y, part of the program in the mask ROM can be replaced by the program in the internal expansion RAM.

The use of ROM correction enables command bugs discovered in the mask ROM to be repaired, and the flow of the program to be changed.

ROM correction can be used in a maximum of four located within the internal ROM (program).

**Caution Note that ROM correction cannot be emulated by the in-circuit emulator (IE-784000-R, IE-784000-R-EM).**

Specifically, the command addresses that require repair from the inactive memory externally connected to a microcontroller by a user program and the repair command codes are loaded into the peripheral RAM.

The above addresses and the internal ROM access addresses are compared by the comparator built into the microcontroller during execution of internal ROM programs (during command fetch), and the internal ROM's output data is then converted to call command (CALLT) codes and output when a match is determined.

When the CALLT command codes are changed to valid commands by the CPU and executed, the CALLT table is referenced, and the process routine and other peripheral RAM are branched. At this point, a CALLT table is prepared for each repair address for referencing purposes. Four repair address can be set for the μPD784225.

| | |
|---|---|
| Match with address pointer 0: | CALLT table (0078H)<br>Conversion command code: FCH |
| Match with address pointer 1: | CALLT table (007AH)<br>Conversion command code: FDH |
| Match with address pointer 2: | CALLT table (007CH)<br>Conversion command code: FEH |
| Match with address pointer 3: | CALLT table (007EH)<br>Conversion command code: FFH |

**Caution As it is necessary to reserve four locations for the CALLT tables when the ROM correction function is used (0078H, 007AH, 007CH, 007EH), ensure that these are not used for other applications. However, the CALLT tables can be used if the ROM correction function is not being used.**

The differences between 78K/IV ROM correction and 78K/0 ROM correction are shown in Table 26-1.

**Table 26-1. Differences Between 78K/IV ROM Correction and 78K/0 ROM Correction**

| Difference | 78K/IV | 78K/0 |
|---|---|---|
| Generated command codes | CALLT instruction<br>(1-byte instruction:<br>FCH, FDH, FEH, FFH) | Branch instruction for peripheral RAM<br>(3-byte instruction) |
| Change of stack pointer | Yes (3-byte save) | None |
| Address comparison conditions | Instruction fetch only | Instruction fetch only |
| Correction status flag | None<br>As there is a possibility that the addresses match owing to an invalid fetch, the status is not necessary | Yes |
| Jump destination address during correction | CALLT table<br>0078H, 007AH, 007CH, 007EH | Fixed address on peripheral RAM |

## 26.2 ROM Correction Configuration

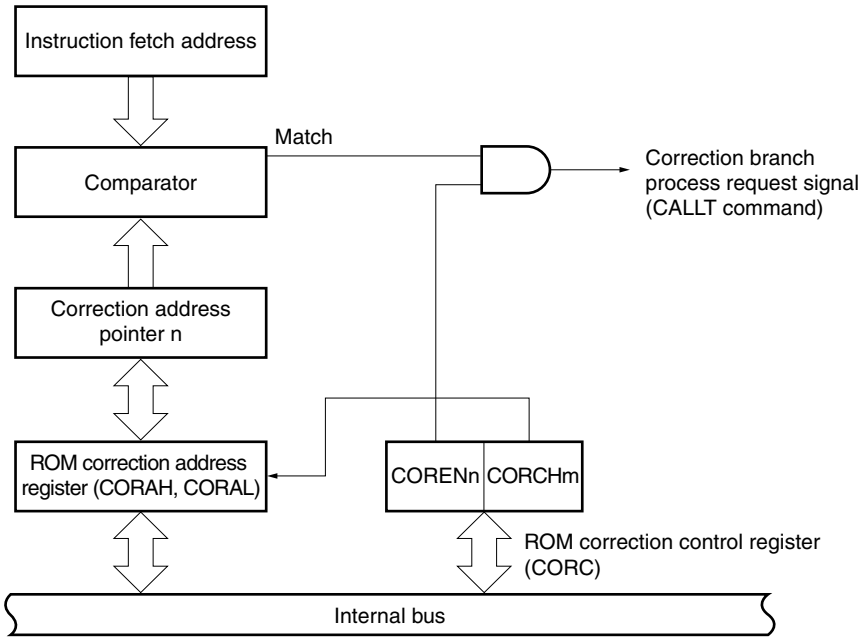
ROM correction includes the following hardware.

**Table 26-2. ROM Correction Configuration**

| Item | Configuration |
|---|---|
| Register | ROM correction address register H, L (CORAH, CORAL) |
| Control register | ROM correction control register (CORC) |

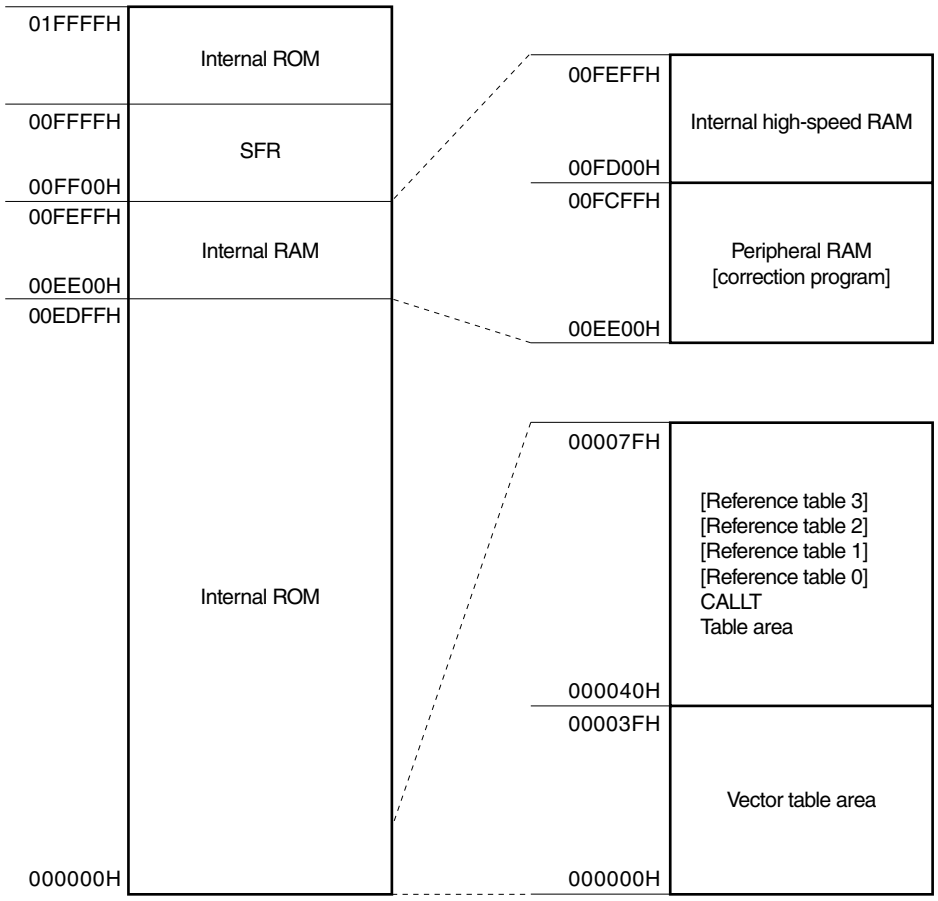
A ROM correction block diagram is shown in Figure 26-1, and Figure 26-2 shows an example of memory mapping.

**Figure 26-1. ROM Correction Block Diagram**




**Remark** n = 0 to 3, m = 0, 1

**Figure 26-2. Memory Mapping Example ($\mu$PD784225)**




**(1) ROM correction address register (CORAH, CORAL)**

This register sets the header address (correction address) of the command in the mask ROM that needs to be repaired. A maximum of four program locations can be repaired with ROM correction. First of all, the channel is selected with bit 0 (CORCH0) and bit 1 (CORCH1) of the ROM correction control register (CORC), and the address is then set in the specified channel's address pointer when the address is written in CORAH and CORAL.

**Figure 26-3. Format of ROM Correction Address Register (CORAH, CORAL)**

| | 7 | 0 | Address | After reset | R/W |
|---|---|---|---|---|---|
| CORAH | | | FF89H | 00H | R/W |

| | 15 | 0 | Address | After reset | R/W |
|---|---|---|---|---|---|
| CORAL | | | FF8AH | 0000H | R/W |

**(2) Comparator**

ROM correction address registers H and L (CORAH, CORAL) normally compare the corrected address value with the fetch register value. If any of the ROM correction control register (CORC) bits between bit 4 and bit 7 (COREN0 to 3) are 1 and the correct address matches the fetch address value, a table reference instruction (CALLT) is issued from the ROM correction circuit.

## 26.3 Control Register for ROM Correction

ROM correction is controlled by the ROM correction control register (CORC).

**(1) ROM correction control register (CORC)**

The register that controls the issuance of the table reference instruction (CALLT) when the correct address set in ROM correction address registers H and L (CORAH, CORAL) match the value of the fetch address.

This register is composed of a correction enable flag (COREN0 to 3) that enables or disables match detection with the comparator, and four channel correction pointers.

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

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

**Figure 26-4. Format of ROM Correction Control Register (CORC)**

Address: 0FF88H After reset: 00H R/W

| Symbol | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CORC | COREN3 | COREN2 | COREN1 | COREN0 | 0 | 0 | CORCH1 | CORCH0 |

| CORENn | Controls the match detection for the ROM correction address register and the fetch address. |
|---|---|
| 0 | Disabled |
| 1 | Enabled |

| CORCH1 | CORCH0 | Channel selection |
|---|---|---|
| 0 | 0 | Address pointer channel 0 |
| 0 | 1 | Address pointer channel 1 |
| 1 | 0 | Address pointer channel 2 |
| 1 | 1 | Address pointer channel 3 |

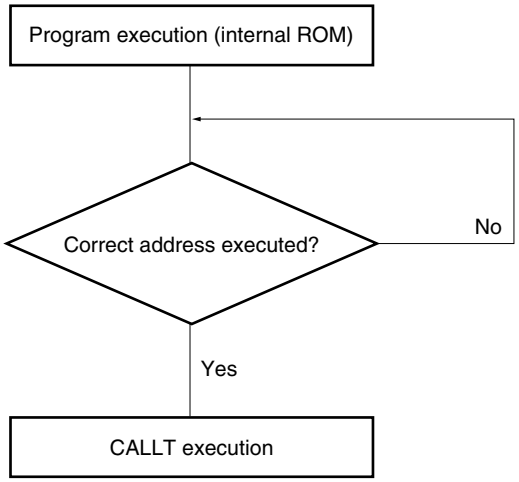
**Remark** n = 0 to 3

## 26.4 Usage of ROM Correction

<1> The correct address and post-correction instruction (correction program) are stored in the microcontroller external inactive memory (EEPROM™).

<2> A substitute instruction is read from the inactive memory using a serial interface (etc.) when the initialization program is running after being reset, and this is stored in the peripheral RAM and external memory. The correction channel is then selected, the address for the command that requires correction is read and set in ROM correction address registers H, L (CORAH, CORAL), and the correction enable flag (COREN0 to 3) is set to 1. A maximum of four locations can be set.

<3> The CALLT instruction is then executed during execution of the corrected address.




<4> CALLT routine branch
When matched with address pointer 0: CALLT table (0078H)
When matched with address pointer 1: CALLT table (007AH)
When matched with address pointer 2: CALLT table (007CH)
When matched with address pointer 3: CALLT table (007EH)

<5> Substitute instruction execution

<6> Addition of +3 to the stack pointer (SP)

<7> Restoration to any addresses with the branch instruction (BR)

## 26.5 Conditions for Executing ROM Correction

In order to use the ROM correction function, it is necessary for the external environment and program to satisfy the following conditions.

**(1) External environment**

Must be connected externally to an inactive memory, and be configured to read that data.

**(2) Target program**

The data setting instruction for CORC, CORAH and CORAL must be previously annotated in the target program (program stored in the ROM).

The setting data (the items written in lower case in the setting example below) must be read from the external inactive memory, and the correct number of required correction pointers must be set.

Example of four pointer settings

```
                MOV     CORC,   #00H        ; Specified channel 0
                MOVW    CORAL,  #ch0_datal  ; Sets the channel 0 match address
                MOV     CORAH,  #ch0_datah  ; Sets the channel 0 match address
                MOV     CORC,   #01H        ; Specified channel 1
                MOVW    CORAL,  #ch1_datal  ; Sets the channel 1 match address
                MOV     CORAH,  #ch1_datah  ; Sets the channel 1 match address
                MOV     CORC,   #02H        ; Specified channel 2
                MOVW    CORAL,  #ch2_datal  ; Sets the channel 2 match address
                MOV     CORAH,  #ch2_datah  ; Sets the channel 2 match address
                MOV     CORC,   #03H        ; Specified channel 3
                MOVW    CORAL,  #ch3_datah  ; Sets the channel 3 match address
                MOV     CORAH,  #ch3_datal  ; Sets the channel 3 match address
                MOV     CORC,   #romcor en
                                            ; Sets 00H when correction is disabled
                                            ; Sets F0H when correction is operated
                BR      $NORMAL
                BR      ! ! COR_ADDR0       ; Specifies the address of the correction program (channel 0)
                BR      ! ! COR_ADDR1       ; Specifies the address of the correction program (channel 1)
                BR      ! ! COR_ADDR2       ; Specifies the address of the correction program (channel 2)
                BR      ! ! COR_ADDR3       ; Specifies the address of the correction program (channel 3)
                ;                             (two-level branch)
NOMAL instruction                           ; Next instruction
```

**(3) Setting branch instructions in the CALLT table**

For the CALLT table that corresponds to each channel, in the case of the above program, the header addresses for the BR!!COR_ADDR0, BR!!COR_ADDR1, BR!!COR_ADDR2, and BR!!COR_ADDR3 instructions are specified (COR_ADDR0 to COR_ADDR3 indicate the address where the correction program is located).

These instructions are branched by the CALLT instruction and BR instruction into two levels because only the base area can be branched to with CALLT. There is no need to branch these instructions into two levels when they are to be attached to the RAM base area with the LOCATION instruction.

# CHAPTER 27 μPD78F4225 AND μPD78F4225Y PROGRAMMING

The flash memory versions in the μPD784225 and 784225Y Subseries are the μPD78F4225 and 78F4225Y. The μPD78F4225 and μPD78F4225Y are described in this chapter using the μPD78F4225 as the representative product.

The μPD78F4225 and 78F4225Y are versions with on-chip flash memories that enable programs to be written, deleted and overwritten while mounted on the substrate. The differences between the flash memory versions (μPD78F4225 and 78F4225Y) and mask ROM versions (μPD784224, 784225, 784224Y and 784225Y) are shown in Table 27-1.

**Table 27-1. Differences Between μPD78F4225/78F4225Y and Mask ROM Versions**

| Item | μPD78F4225, 78F4225Y | Mask ROM Versions |
|---|---|---|
| Internal ROM structure | Flash memory | Mask ROM |
| Internal ROM capacity | 128 KB | μPD784224, 784224Y: 96 KB<br>μPD784225, 784225Y: 128 KB |
| Internal RAM capacity | 4,352 bytes | μPD784224, 784224Y: 3,584 bytes<br>μPD784225, 784225Y: 4,352 bytes |
| Changing internal ROM capacity using the internal memory size switching register (IMS) | Possible[Note] | Not possible |
| TEST pin | None | Provided |
| $V_{PP}$ pin | Provided | None |

**Note** The capacity of the flash memory will become 128 KB and the internal RAM capacity will become 4,352 bytes by $\overline{\text{RESET}}$ input.

**Caution There are differences in noise immunity and noise radiation between the flash memory and mask ROM versions. When pre-producing an application set with the flash memory version and then mass-producing it with the mask ROM version, be sure to conduct sufficient evaluations for the commercial samples (not engineering samples) of the mask ROM version.**

## 27.1 Internal Memory Size Switching Register (IMS)

IMS is a register used to prevent a certain part of the internal memory from being used by software. By setting IMS, it is possible to establish a memory map that is the same as the mask ROM product's memory map for an internal memory (ROM, RAM) of a different capacity.

IMS is set by an 8-bit memory manipulation instruction.

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

**Figure 27-1. Format of Internal Memory Size Switching Register (IMS)**

Address: 0FFFCH After reset: FFH W

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

| ROM1 | ROM0 | Internal ROM capacity selection |
|---|---|---|
| 1 | 0 | 96 KB |
| 1 | 1 | 128 KB |
| Other than above | | Setting prohibited |

| RAM1 | RAM0 | Internal high-speed RAM capacity selection |
|---|---|---|
| 1 | 0 | 3,072 bytes |
| 1 | 1 | 3,840 bytes |
| Other than above | | Setting prohibited |

**Caution IMS is not available in the mask ROM versions (μPD784224, 784225, 784224Y, and 784225Y).**

The IMS settings to create the same memory map as mask ROM versions are shown in Table 27-2.

**Table 27-2. Internal Memory Size Switching Register (IMS) Settings**

| Relevant Mask ROM Version | IMS Setting |
|---|---|
| μPD784224, 784224Y | EEH |
| μPD784225, 784225Y | FFH |

## 27.2 Flash Memory Overwriting

The μPD78F4225 is equipped with an on-chip 128 KB flash memory. The following method is available for overwriting the on-chip flash memory.

- On-board overwrite mode: Overwriting performed using a flash programmer.

Flash memory overwriting can be performed 20 times.
A voltage of +10 V is required for deleting and writing the flash memory.

## 27.3 On-Board Overwrite Mode

The on-board overwrite mode is used with the target system mounted (on-board). Overwriting is performed by connecting a special flash programmer (Flashpro III (part No.: FL-PR3, PG-FP3) ) to the host machine or target system. Overwriting is controlled via the serial interface.
Writing to the flash memory can be performed using the flash memory write adapter connected to Flashpro III.

**Remark** Flashpro III is a product of Naito Densei Machida Mfg. Co., Ltd.

On-board overwrite mode settings are performed by controlling the TEST/$V_{PP}$ pin and $\overline{\text{RESET}}$ pin. The serial interface is selected according to the number of pulses applied to the TEST/$V_{PP}$ pin.

### 27.3.1 Selecting communication mode

Flashpro III writes to the flash memory by serial communication. The communication mode is selected from the modes shown in Table 27-3, then writing is performed. The selection of the communication mode has the format shown in Figure 27-2. Each communication mode is selected according to the number of $V_{PP}$ pulses shown in Table 27-3.

**Table 27-3. Communication Modes**

| Communication Mode | No. of Channels | Pins Used[Note 1] | No. of $V_{PP}$ Pulses |
|---|---|---|---|
| 3-wire serial I/O | 3 | SCK0/P27/SCL0[Note 2]<br>SO0/P26<br>SI0/P25/SDA0[Note 2] | 0 |
| | | $\overline{SCK1}$/ASCK1/P22<br>SO1/TxD1/P21<br>SI1/RxD1/P20 | 1 |
| | | $\overline{SCK2}$/ASCK2/P72<br>SO2/TxD2/P71<br>SI2/RxD2/P70 | 2 |
| 3-wire serial I/O<br>(handshake[Note 3]) | 1 | SCK0/P27/SCL0[Note 2]<br>SO0/P26<br>SI0/P25/SDA0[Note 2]<br>P24/BUZ | 3 |
| UART | 2 | TxD1/SO1/P21<br>RxD1/SI1/P20 | 8 |
| | | TxD2/SO2/P71<br>RxD2/SI2/P70 | 9 |

**Notes 1.** Shifting to the flash memory programming mode sets all pins not used for flash memory programming to the same state as immediately after reset. Therefore, if the external devices do not acknowledge the port state immediately after reset, handling such as connecting to $V_{DD}$ via a resistor or connecting to $V_{SS}$ via a resistor is required.

**2.** μPD78F4225Y only

**3.** Other than K standard

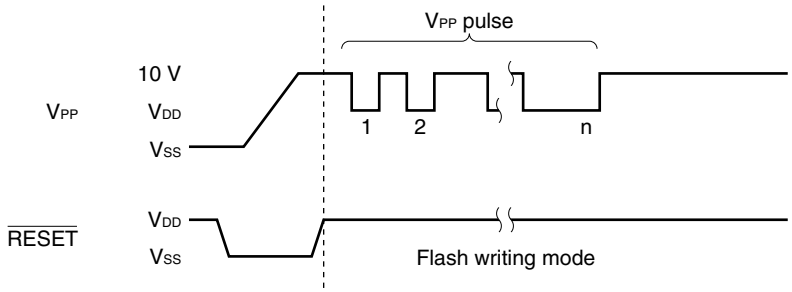
**Caution Select the communication mode by using the number of $V_{PP}$ pulses given in Table 27-3.**

**Remarks 1.** The fifth digit from the left in the lot number indicates the standard.

**2.** Handshake mode is the CSI writing mode using P24. This mode is available for PG-FR3 and FL-PR3.

**3.** The I standard is applicable only for ES (engineering sample) products, so the operation cannot be guaranteed.

**Figure 27-2. Format of Communication Mode Selection**




### 27.3.2 On-board overwrite mode 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 27-4 shows the major functions.

**Table 27-4. Major Functions of On-Board Overwrite Mode**

| Function | Description |
|---|---|
| Area erase | Erase the contents of the specified memory area. |
| Area blank check | Checks the erase state of the specified area. |
| Data write | Writes to the flash memory based on the start write address and the number of data written (the number of bytes). |
| Area verify | Compares the data input to the contents of the specified memory area. |

Flash memory verification 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 relevant area 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 using the verification function.

### 27.3.3 Connecting Flashpro III

The connection between Flashpro III and the μPD78F4225 differs depending on the communication mode (3-wire serial I/O or UART). Figures 27-3 to 27-5 show the connection diagrams in each case.

**Figure 27-3. Connection of Flashpro III in 3-Wire Serial I/O Mode (When Using 3-Wire Serial I/O 0)**

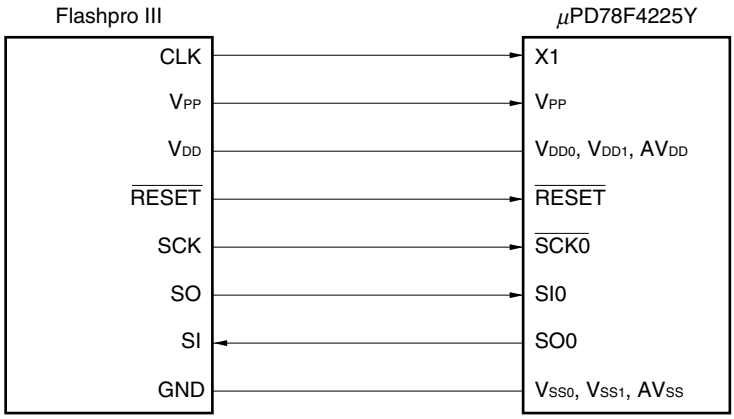
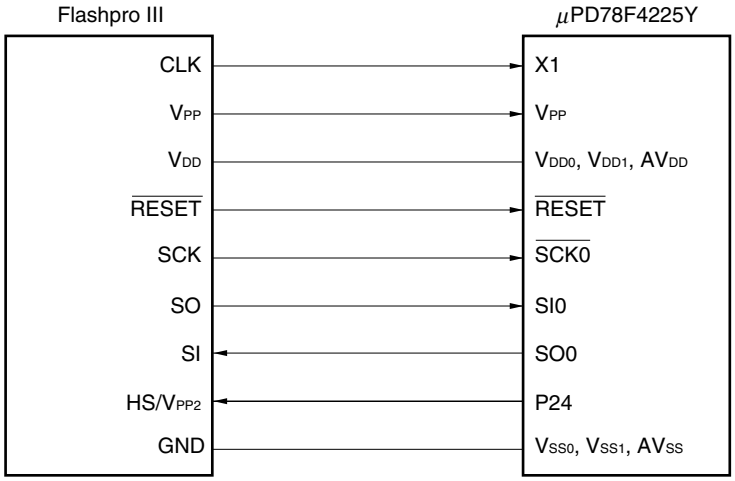
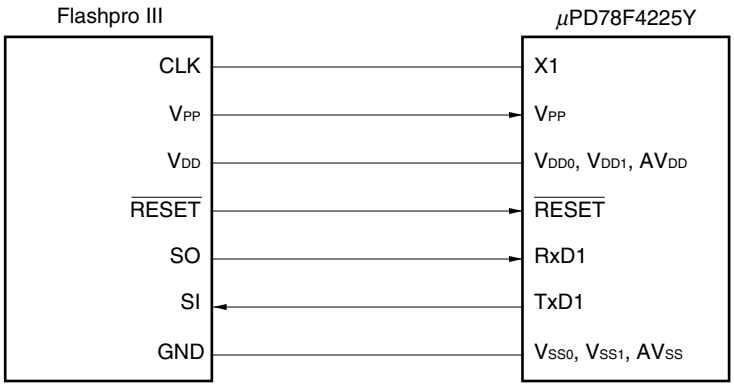



**Figure 27-4. Connection of Flashpro III in 3-Wire Serial I/O Mode (When Using Handshake)**




**Note** n = 1, 2

**Figure 27-5. Connection of Flashpro III in UART Mode (When Using UART1)**




**Note** n = 1, 2

**Caution Connect the $V_{PP}$ pin directly to $V_{SS}$ or pull down. For the pull-down connection, use of resistors with a resistance between 470 Ω and 10 kΩ is recommended.**

# CHAPTER 28 INSTRUCTION OPERATION

## 28.1 Conventions

### (1) Operand format and descriptions (1/2)

| 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) |
| r1Note 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) |
| rp1Note 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) |
| postNote 2 | AX(RP0), BC(RP1), RP2, RP3, VP(RP4), UP(RP5)/PSW, DE(RP6), HL(RP7)<br>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<br>[TDE+byte], [WHL+byte], [SP+byte], [UUP+byte], [VVP+byte]: based addressing<br>imm24[A], imm24[B], imm24[DE], imm24[HL]: indexed addressing<br>[TDE+A], [TDE+B], [TDE+C], [WHL+A], [WHL+B], [WHL+C], [VVP+DE], [VVP+HL]: based indexed addressing |
| mem1 | Everything under mem except [WHL+] and [WHL–] |
| mem2 | [TDE], [WHL] |
| mem3 | [AX], [BC], [RP2], [RP3], [VVP], [UUP], [TDE], [WHL] |

**Notes 1.** By setting the RSS bit to 1, R4 to R7 can be used as X, A, C, and B. Use this function only when 78K/III Series programs are also used.

**2.** By setting the RSS bit to 1, RP2 and RP3 can be used as AX and BC. Use this function only when 78K/III Series programs are also used.

### (1) Operand format and descriptions (2/2)

| Format | Description |
|---|---|
| **Note** | |
| saddr, saddr' | FD20H to FF1FH Immediate data or label |
| saddr1 | FE00H to FEFFH Immediate data or label |
| saddr2 | FD20H to FDFFH, FF00H to FF1FH Immediate data or label |
| saddrp | FD20H to FF1EH Immediate data or label (when manipulating 16 bits) |
| saddrp1 | FE00H to FEFFH Immediate data or label (when manipulating 16 bits) |
| saddrp2 | FD20H to FDFFH, FF00H to FF1EH Immediate data or label (when manipulating 16 bits) |
| saddrg | FD20H to FEFDH Immediate data or label (when manipulating 24 bits) |
| saddrg1 | FE00H to FEFDH Immediate data or label (when manipulating 24 bits) |
| saddrg2 | FD20H to FDFFH Immediate data or label (when manipulating 24 bits) |
| addr24 | 0H to FFFFFFH Immediate data or label |
| addr20 | 0H to FFFFFH Immediate data or label |
| addr16 | 0H to FFFFH Immediate data or label |
| addr11 | 800H to FFFH Immediate data or label |
| addr8 | 0FE00H to 0FEFFH[Note] Immediate data or label |
| addr5 | 40H to 7EH Immediate data or label |
| imm24 | 24-bit immediate data or label |
| word | 16-bit immediate data or label |
| byte | 8-bit immediate data or label |
| bit | 3-bit immediate data or label |
| n | 3-bit immediate data |
| locaddr | 0H or 0FH |

**Note** When 0H is set by the LOCATION instruction, these addresses become the addresses shown here. When 0FH is set by the LOCATION instruction, the values of the addresses shown here with F0000H added 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 0. |
| 1 | Set to 1. |
| × | Set or clear based on the result. |
| P | Operate with the P/V flag as the parity flag. |
| V | Operate with the P/V flag as the overflow flag. |
| R | |

### (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 part.

| 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).

## 28.2 List of Operations

### (1) 8-bit data transfer instruction: MOV

| Mnemonic | Operand | Bytes | Operation | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 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 | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 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 | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 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 | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 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 | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 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 | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 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 | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 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 ← ((saddrp)) + 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 | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 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 | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 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 | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 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 | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 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 | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | S | Z | AC | P/V | CY |
| OR | A, #byte | 2 | A ← A∨byte | × | × | | P | |
| | r, #byte | 3 | r ← r∨byte | × | × | | P | |
| | saddr, #byte | 3/4 | (saddr) ← (saddr) ∨byte | × | × | | P | |
| | sfr, #byte | 4 | sfr ← sfr∨byte | × | × | | P | |
| | r, r' | 2/3 | r ← r∨r' | × | × | | P | |
| | A, saddr2 | 2 | A ← A∨ (saddr2) | × | × | | P | |
| | r, saddr | 3 | r ← r∨ (saddr) | × | × | | P | |
| | saddr, r | 3 | (saddr) ← (saddr) ∨r | × | × | | P | |
| | r, sfr | 3 | r ← r∨sfr | × | × | | P | |
| | sfr, r | 3 | sfr ← sfr∨r | × | × | | P | |
| | saddr, saddr' | 4 | (saddr) ← (saddr) ∨ (saddr') | × | × | | P | |
| | A, [saddrp] | 3/4 | A ← A∨ ((saddrp)) | × | × | | P | |
| | A, [%saddrg] | 3/4 | A ← A∨ ((saddrg)) | × | × | | P | |
| | [saddrp], A | 3/4 | ((saddrp)) ← ((saddrp)) ∨A | × | × | | P | |
| | [%saddrg], A | 3/4 | ((saddrg)) ← ((saddrg)) ∨A | × | × | | P | |
| | A, !addr16 | 4 | A ← A∨ (saddr16) | × | × | | P | |
| | A, !!addr24 | 5 | A ← A∨ (saddr24) | × | × | | 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 | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 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 | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 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 | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | S | Z | AC | P/V | CY |
| ADDG | rg, rg' | 2 | rg, CY ← rg + rg' | × | × | × | V | × |
| | rg, #imm24 | 5 | rg, CY ← rg + imm24 | × | × | × | V | × |
| | WHL, saddrg | 3 | WHL, CY ← WHL + (saddrg) | × | × | × | V | × |
| SUBG | rg, rg' | 2 | rg, CY ← rg – rg' | × | × | × | V | × |
| | rg, #imm24 | 5 | rg, CY ← rg – imm24 | × | × | × | V | × |
| | WHL, saddrg | 3 | WHL, CY ← WHL – (saddrg) | × | × | × | V | × |

### (9) Multiplication/division instructions: MULU, MULUW, MULW, DIVUW, DIVUX

| Mnemonic | Operand | Bytes | Operation | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | S | Z | AC | P/V | CY |
| MULU | r | 2/3 | AX ← AXr | | | | | |
| MULUW | rp | 2 | AX (high order), rp (low order) ← AXXrp | | | | | |
| MULW | rp | 2 | AX (high order), rp (low order) ← AXXrp | | | | | |
| 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 | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | S | Z | AC | P/V | CY |
| MACW | byte | 3 | AXDE ← (B) X (C) + AXDE, B ← B + 2,<br>C ← C + 2, byte ← byte – 1<br>End if (byte = 0 or P/V = 1) | × | × | × | V | × |
| MACSW | byte | 3 | AXDE ← (B) X (C) + AXDE, B ← B + 2,<br>C ← C + 2, byte ← byte – 1<br>if byte = 0 then End<br>if P/V = 1 then if overflow AXDE ← 7FFFFFFFH, End<br>if underflow AXDE ← 80000000H, End | × | × | × | V | × |
| SACW | [TDE+], [WHL+] | 4 | AX ← ǀ (TDE) – (WHL) ǀ + AX,<br>TDE ← TDE + 2, WHL ← WHL + 2<br>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 | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 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 | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 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<br>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 | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | S | Z | AC | P/V | CY |
| ROR | r, n | 2/3 | (CY, $r_7 \leftarrow r_0$, $r_{m-1} \leftarrow r_m$) × n n = 0 to 7 | | | | P | × |
| ROL | r, n | 2/3 | (CY, $r_0 \leftarrow r_7$, $r_{m+1} \leftarrow r_m$) × n n = 0 to 7 | | | | P | × |
| RORC | r, n | 2/3 | (CY $\leftarrow r_0$, $r_7 \leftarrow$ CY, $r_{m-1} \leftarrow r_m$) × n n = 0 to 7 | | | | P | × |
| ROLC | r, n | 2/3 | (CY $\leftarrow r_7$, $r_0 \leftarrow$ CY, $r_{m+1} \leftarrow r_m$) × n n = 0 to 7 | | | | P | × |
| SHR | r, n | 2/3 | (CY $\leftarrow r_0$, $r_7 \leftarrow 0$, $r_{m-1} \leftarrow r_m$) × n n = 0 to 7 | × | × | 0 | P | × |
| SHL | r, n | 2/3 | (CY $\leftarrow r_7$, $r_0 \leftarrow 0$, $r_{m+1} \leftarrow r_m$) × n n = 0 to 7 | × | × | 0 | P | × |
| SHRW | rp, n | 2 | (CY $\leftarrow rp_0$, $rp_{15} \leftarrow 0$, $rp_{m-1} \leftarrow rp_m$) × n n = 0 to 7 | × | × | 0 | P | × |
| SHLW | rp, n | 2 | (CY $\leftarrow rp_{15}$, $rp_0 \leftarrow 0$, $rp_{m+1} \leftarrow rp_m$) × 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 | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 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 | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 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 ∧ $\overline{PSW_L.bit}$ | | | | | × |
| | CY, PSWH.bit | 2 | CY ← CY ∧ $PSW_H$.bit | | | | | × |
| | CY, /PSWH.bit | 2 | CY ← CY ∧ $\overline{PSW_H.bit}$ | | | | | × |
| | 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 ∨ $\overline{PSW_L.bit}$ | | | | | × |
| | CY, PSWH.bit | 2 | CY ← CY ∨ $PSW_H$.bit | | | | | × |
| | CY, /PSWH.bit | 2 | CY ← CY ∨ $\overline{PSW_H.bit}$ | | | | | × |
| | 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 | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 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 ∀ $PSW_L$.bit | | | | | × |
| | CY, PSWH.bit | 2 | CY ← CY ∀ $PSW_H$.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 ← $\overline{PSW_L.bit}$ | × | × | × | × | × |
| | PSWH.bit | 2 | PSWH.bit ← $\overline{PSW_H.bit}$ | | | | | |
| | !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 | $PSW_L$.bit ← 1 | × | × | × | × | × |
| | PSWH.bit | 2 | $PSW_H$.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 | $PSW_L$.bit ← 0 | × | × | × | × | × |
| | PSWH.bit | 2 | $PSW_H$.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 | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 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 | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | S | Z | AC | P/V | CY |
| CALL | !addr16 | 3 | (SP – 3) ← (PC + 3), SP ← SP – 3,<br>$PC_{HW}$ ← 0, $PC_{LW}$ ← addr16 | | | | | |
| | !!addr20 | 4 | (SP – 3) ← (PC + 4), SP ← SP – 3,<br>PC ← addr20 | | | | | |
| | rp | 2 | (SP – 3) ← (PC + 2), SP ← SP – 3,<br>$PC_{HW}$ ← 0, $PC_{LW}$ ← rp | | | | | |
| | rg | 2 | (SP – 3) ← (PC + 2), SP ← SP – 3,<br>PC ← rg | | | | | |
| | [rp] | 2 | (SP – 3) ← (PC + 2), SP ← SP – 3,<br>$PC_{HW}$ ← 0, $PC_{LW}$ ← (rp) | | | | | |
| | [rg] | 2 | (SP – 3) ← (PC + 2), SP ← SP – 3,<br>PC ← (rg) | | | | | |
| | $!addr20 | 3 | (SP – 3) ← (PC + 3), SP ← SP – 3,<br>PC ← PC + 3 + jdisp16 | | | | | |
| CALLF | !addr11 | 2 | (SP – 3) ← (PC + 2), SP ← SP – 3<br>$PC_{19-12}$ ← 0, $PC_{11}$ ← 1, $PC_{10-0}$ ← addr11 | | | | | |
| CALLT | [addr5] | 1 | (SP – 3) ← (PC + 1), SP ← SP – 3<br>$PC_{HW}$ ← 0, $PC_{LW}$ ← (addr5) | | | | | |
| BRK | | 1 | (SP – 2) ← PSW, $(SP - 1)_{0-3}$ ←, $(PC + 1)_{HW}$,<br>$(SP - 4)$ ← $(PC + 1)_{LW}$,<br>SP ← SP – 4<br>$PC_{HW}$ ← 0, $PC_{LW}$ ← (003EH) | | | | | |
| BRKCS | RBn | 2 | $PC_{LW}$ ← RP2, RP3 ← PSW, RBS2 – 0 ← n,<br>RSS ← 0, IE ← 0, $RP3_{8-11}$ ← $PC_{HW}$, $PC_{HW}$ ← 0 | | | | | |
| RET | | 1 | PC ← (SP), SP ← SP + 3 | | | | | |
| RETI | | 1 | $PC_{LW}$ ← (SP), $PC_{HW}$ ← $(SP + 3)_{0-3}$,<br>PSW ← (SP + 2), SP ← SP + 4<br>The flag with the highest priority that is set to 1 in the ISPR is cleared to 0. | R | R | R | R | R |
| RETB | | 1 | $PC_{LW}$ ← (SP), $PC_{HW}$ ← $(SP + 3)_{0-3}$,<br>PSW ← (SP + 2), SP ← SP + 4 | R | R | R | R | R |
| RETCS | !addr16 | 3 | PSW ← RP3, $PC_{LW}$ ← RP2, RP2 ← addr16,<br>$PC_{HW}$ ← $RP3_{8-11}$<br>The flag with the highest priority that is set to 1 in the ISPR is cleared to 0. | R | R | R | R | R |
| RETCSB | !addr16 | 4 | PSW ← RP3, $PC_{LW}$ ← RP2, RP2 ← addr16,<br>$PC_{HW}$ ← $RP3_{8-11}$ | R | R | R | R | R |

### (17) Unconditional branch instruction: BR

| Mnemonic | Operand | Bytes | Operation | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | S | Z | AC | P/V | CY |
| BR | !addr16 | 3 | $PC_{HW} \leftarrow 0$, $PC_{LW} \leftarrow$ addr16 | | | | | |
| | !!addr20 | 4 | PC ← addr20 | | | | | |
| | rp | 2 | $PC_{HW} \leftarrow 0$, $PC_{LW} \leftarrow$ rp | | | | | |
| | rg | 2 | PC ← rg | | | | | |
| | [rp] | 2 | $PC_{HW} \leftarrow 0$, $PC_{LW} \leftarrow$ (rp) | | | | | |
| | [rg] | 2 | PC ← (rg) | | | | | |
| | $addr20 | 2 | PC ← PC + 2 + jdisp8 | | | | | |
| | $!addr20 | 3 | PC ← PC + 3 + jdisp16 | | | | | |

### (18) Conditional branch instructions: BNZ, BNE, BZ, BE, BNC, BNL, BC, BL, BNV, BPO, BV, BPE, BP, BN, BLT, BGE, BLE, BGT, BNH, BH, BF, BT, BTCLR, BFSET, DBNZ

| Mnemonic | Operand | Bytes | Operation | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 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[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 | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | S | Z | AC | P/V | CY |
| BT | saddr.bit, $addr20 | 3/4 | PC ← PC + 3[Note 1] + jdisp8 if (saddr.bit) = 1 | | | | | |
| | sfr.bit, $addr20 | 4 | PC ← PC + 4 + jdisp8 if sfr.bit = 1 | | | | | |
| | X.bit, $addr20 | 3 | PC ← PC + 3 + jdisp8 if X.bit = 1 | | | | | |
| | A.bit, $addr20 | 3 | PC ← PC + 3 + jdisp8 if A.bit = 1 | | | | | |
| | PSWL.bit, $addr20 | 3 | PC ← PC + 3 + jdisp8 if $PSW_L$.bit = 1 | | | | | |
| | PSWH.bit, $addr20 | 3 | PC ← PC + 3 + jdisp8 if $PSW_H$.bit = 1 | | | | | |
| | !addr16.bit, $addr20 | 6 | PC ← PC + 3 + jdisp8 if !addr16.bit = 1 | | | | | |
| | !!addr24.bit, $addr20 | 3 | PC ← PC + 3 + jdisp8 if !!addr24.bit = 1 | | | | | |
| | mem2.bit, $addr20 | 3 | PC ← PC + 3 + jdisp8 if mem2.bit = 1 | | | | | |
| BTCLR | saddr.bit, $addr20 | 4/5 | {PC ← PC + 4[Note 2] + jdisp8, (saddr.bit) ← 0} if (saddr.bit = 1) | | | | | |
| | sfr.bit, $addr20 | 4 | {PC ← PC + 4 + jdisp8, sfr.bit ← 0} if sfr. bit = 1 | | | | | |
| | X.bit, $addr20 | 3 | {PC ← PC + 3 + jdisp8, X.bit ← 0} if X.bit = 1 | | | | | |
| | A.bit, $addr20 | 3 | {PC ← PC + 3 + jdisp8, A.bit ← 0} if A.bit = 1 | | | | | |
| | PSWL.bit, $addr20 | 3 | {PC ← PC + 3 + jdisp8, $PSW_L$.bit ← 0} if $PSW_L$.bit = 1 | × | × | × | × | × |
| | PSWH.bit, $addr20 | 3 | {PC ← PC + 3 + jdisp8, $PSW_H$.bit ← 0} if $PSW_H$.bit = 1 | | | | | |
| | !addr16.bit, $addr20 | 6 | {PC ← PC + 3 + jdisp8, !addr16.bit ← 0} if !addr16.bit = 1 | | | | | |
| | !!addr24.bit, $addr20 | 3 | {PC ← PC + 3 + jdisp8, !!addr24.bit ← 0} if !!addr24.bit = 1 | | | | | |
| | mem2.bit, $addr20 | 3 | {PC ← PC + 3 + jdisp8, mem2.bit ← 0} if mem2.bit = 1 | | | | | |

**Notes 1.** 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 | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | S | Z | AC | P/V | CY |
| BFSET | saddr.bit, $addr20 | 4/5 | {PC ← PC + 4[Note 2] + jdisp8, (saddr.bit) ← 1} if (saddr.bit = 0) | | | | | |
| | sfr.bit, $addr20 | 4 | {PC ← PC + 4 + jdisp8, sfr.bit ← 1} if sfr. bit = 0 | | | | | |
| | X.bit, $addr20 | 3 | {PC ← PC + 3 + jdisp8, X.bit ← 1} if X.bit = 0 | | | | | |
| | A.bit, $addr20 | 3 | {PC ← PC + 3 + jdisp8, A.bit ← 1} if A.bit = 0 | | | | | |
| | PSWL.bit, $addr20 | 3 | {PC ← PC + 3 + jdisp8, $PSW_L$.bit ← 1} if $PSW_L$.bit = 0 | × | × | × | × | × |
| | PSWH.bit, $addr20 | 3 | {PC ← PC + 3 + jdisp8, $PSW_H$.bit ← 1} if $PSW_H$.bit = 0 | | | | | |
| | !addr16.bit, $addr20 | 6 | {PC ← PC + 3 + jdisp8, !addr16.bit ← 1} if !addr16.bit = 0 | | | | | |
| | !!addr24.bit, $addr20 | 3 | {PC ← PC + 3 + jdisp8, !!addr24.bit ← 1} if !!addr24.bit = 0 | | | | | |
| | mem2.bit, $addr20 | 3 | {PC ← PC + 3 + jdisp8, mem2.bit ← 1} if mem2.bit = 0 | | | | | |
| DBNZ | B, $addr20 | 2 | B ← B – 1, PC ← PC + 2 + jdisp8 if B ≠ 0 | | | | | |
| | C. $addr20 | 2 | C ← C – 1, PC ← PC + 2 + jdisp8 if C ≠ 0 | | | | | |
| | saddr, $addr20 | 3/4 | (saddr) ← (saddr) – 1, PC ← PC + 3[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 | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 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 ← $\overline{RSS}$ | | | | | |
| 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 | Flag | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | 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 | × |

## 28.3 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 28-1. 8-Bit Addressing Instructions**

| First operand \ Second operand | #byte | A | r<br>r' | saddr<br>saddr' | sfr | !addr16<br>!!addr24 | mem<br>[saddrp]<br>[%saddrg] | r3<br>PSWL<br>PSWH | [WHL+]<br>[WHL–] | n | None[Note 2] |
|---|---|---|---|---|---|---|---|---|---|---|---|
| A | (MOV)<br>ADD[Note 1] | (MOV)<br>(XCH)<br>(ADD)[Note 1] | MOV<br>XCH<br>(ADD)[Note 1] | (MOV)[Note 6]<br>(XCH)[Note 6]<br>(ADD)[Notes 1, 6] | MOV<br>(XCH)<br>(ADD)[Note 1] | (MOV)<br>(XCH)<br>ADD[Note 1] | MOV<br>XCH<br>ADD[Note 1] | MOV | (MOV)<br>(XCH)<br>(ADD)[Note 1] | | |
| r | MOV<br>ADD[Note 1] | (MOV)<br>(XCH)<br>(ADD)[Note 1] | MOV<br>XCH<br>ADD[Note 1] | MOV<br>XCH<br>ADD[Note 1] | MOV<br>XCH<br>ADD[Note 1] | MOV<br>XCH | | | | ROR[Note 3] | MULU<br>DIVUW<br>INC<br>DEC |
| saddr | MOV<br>ADD[Note 1] | (MOV)[Note 6]<br>(ADD)[Note 1] | MOV<br>ADD[Note 1] | MOV<br>XCH<br>ADD[Note 1] | | | | | | | INC<br>DEC<br>DBNZ |
| sfr | MOV<br>ADD[Note 1] | MOV<br>(ADD)[Note 1] | MOV<br>ADD[Note 1] | | | | | | | | PUSH<br>POP |
| !addr16<br>!!addr24 | MOV | MOV<br>ADD[Note 1] | MOV | | | | | | | | |
| mem<br>[saddrp]<br>[%saddrg] | | MOV<br>ADD[Note 1] | | | | | | | | | |
| mem3 | | | | | | | | | | | ROR4<br>ROL4 |
| r3<br>PSWL<br>PSWH | MOV | MOV | | | | | | | | | |
| B, C | | | | | | | | | | | DBNZ |
| STBC, WDM | MOV | | | | | | | | | | |
| [TDE+]<br>[TDE–] | | (MOV)<br>(ADD)[Note 1]<br>MOVM[Note 4] | | | | | | | MOVBK[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 28-2. 16-Bit Addressing Instructions**

| First operand \ Second operand | #word | AX | rp<br>rp' | saddrp<br>saddrp' | sfrp | !addr16<br>!!addr24 | mem<br>[saddrp]<br>[%saddrg] | [WHL+] | byte | n | None[Note 2] |
|---|---|---|---|---|---|---|---|---|---|---|---|
| AX | (MOVW)<br>ADDW[Note 1] | (MOVW)<br>(XCHW)<br>(ADD)[Note 1] | (MOVW)<br>(XCHW)<br>(ADDW)[Note 1] | (MOVW)[Note 3]<br>(XCHW)[Note 3]<br>(ADDW)[Notes 1, 3] | MOVW<br>(XCHW)<br>(ADDW)[Note 1] | (MOVW)<br>XCHW | MOVW<br>XCHW | (MOVW)<br>(XCHW) | | | |
| rp | MOVW<br>ADDW[Note 1] | (MOVW)<br>(XCHW)<br>(ADDW)[Note 1] | MOVW<br>XCHW<br>ADDW[Note 1] | MOVW<br>XCHW<br>ADDW[Note 1] | MOVW<br>XCHW<br>ADDW[Note 1] | MOVW | | | | SHRW<br>SHLW | MULW[Note 4]<br>INCW<br>DECW |
| saddrp | MOVW<br>ADDW[Note 1] | (MOVW)[Note 3]<br>(ADDW)[Note 1] | MOVW<br>ADDW[Note 1] | MOVW<br>XCHW<br>ADDW[Note 1] | | | | | | | INCW<br>DECW |
| sfrp | MOVW<br>ADDW[Note 1] | MOVW<br>(ADDW)[Note 1] | MOVW<br>(ADDW)[Note 1] | | | | | | | | PUSH<br>POP |
| !addr16<br>!!addr24 | MOVW | (MOVW) | MOVW | | | | | | MOVTBLW | | |
| mem<br>[saddrp]<br>[%saddrg] | | MOVW | | | | | | | | | |
| PSW | | | | | | | | | | | PUSH<br>POP |
| SP | ADDWG<br>SUBWG | | | | | | | | | | |
| post | | | | | | | | | | | PUSH<br>POP<br>PUSHU<br>POPU |
| [TDE+] | | (MOVW) | | | | | | SACW | | | |
| byte | | | | | | | | | | | MACW<br>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 28-3. 24-Bit Addressing Instructions**

| First operand \ Second operand | #imm24 | WHL | rg<br>rg' | saddrg | !!addr24 | mem1 | [%saddrg] | SP | None[Note] |
|---|---|---|---|---|---|---|---|---|---|
| WHL | (MOVG)<br>(ADDG)<br>(SUBG) | (MOVG)<br>(ADDG)<br>(SUBG) | (MOVG)<br>(ADDG)<br>(SUBG) | (MOVG)<br>ADDG<br>SUBG | (MOVG) | MOVG | MOVG | MOVG | |
| rg | MOVG<br>ADDG<br>SUBG | (MOVG)<br>(ADDG)<br>(SUBG) | MOVG<br>ADDG<br>SUBG | MOVG | MOVG | | | | INCG<br>DECG<br>PUSH<br>POP |
| saddrg | | (MOVG) | MOVG | | | | | | |
| !!addr24 | | (MOVG) | MOVG | | | | | | |
| mem1 | | MOVG | | | | | | | |
| [%saddrg] | | MOVG | | | | | | | |
| SP | MOVG | MOVG | | | | | | | INCG<br>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 28-4. Bit Manipulation Instruction Addressing Instructions**

| First operand \ Second operand | CY | saddr.bit sfr.bit<br>A.bit X.bit<br>PSWL.bit PSWH.bit<br>mem2.bit<br>!addr16.bit<br>!!addr24.bit | /saddr.bit /sfr.bit<br>/A.bit /X.bit<br>/PSWL.bit /PSWH.bit<br>/mem2.bit<br>/!addr16.bit<br>/!!addr24.bit | None[Note] |
|---|---|---|---|---|
| CY | | MOV1<br>AND1<br>OR1<br>XOR1 | AND1<br>OR1 | NOT1<br>SET1<br>CLR1 |
| saddr.bit<br>sfr.bit<br>A.bit<br>X.bit<br>PSWL.bit<br>PSWH.bit<br>mem2.bit<br>!addr16.bit<br>!!addr24.bit | MOV1 | | | NOT1<br>SET1<br>CLR1<br>BF<br>BT<br>BTCLR<br>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 28-5. 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>BR | CALL<br>BR | CALL<br>BR<br>RETCS<br>RETCSB | CALL<br>BR | CALL<br>BR | CALL<br>BR | CALL<br>BR | CALL<br>BR | CALLF | CALLT | BRKCS | BRK<br>RET<br>RETI<br>RETB |
| Composite instructions | BF<br>BT<br>BTCLR<br>BFSET<br>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

★ CHAPTER 29 ELECTRICAL SPECIFICATIONS

## 29.1 Electrical Specifications of μPD784224, 784225, 784224Y, and 784225Y

For the timing charts, refer to **29.3 Timing Charts**.

**Absolute Maximum Ratings ($T_A$ = 25°C)**

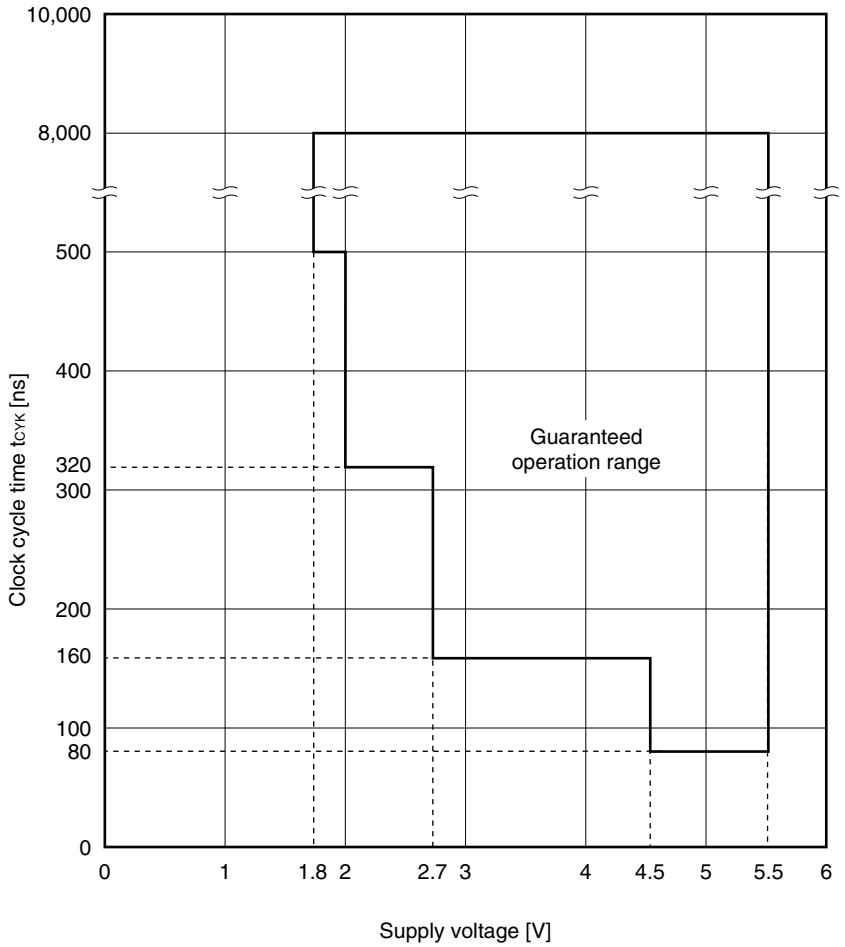
| Parameter | Symbol | Conditions | Ratings | Unit |
|---|---|---|---|---|
| Supply voltage | $V_{DD}$ | $V_{DD} = V_{DD0} = V_{DD1}$ | –0.3 to +6.5 | V |
| | $AV_{DD}$ | | –0.3 to $V_{DD0}$ + 0.3 | V |
| | $AV_{SS}$ | | –0.3 to $V_{SS0}$ + 0.3 | V |
| | $AV_{REF1}$ | D/A converter reference voltage input | –0.3 to $V_{DD0}$ + 0.3 | V |
| Input voltage | $V_I$ | | –0.3 to $V_{DD0}$ + 0.3 | V |
| Analog input voltage | $V_{AN}$ | Analog input pin | $AV_{SS}$ – 0.3 to $AV_{REF1}$ + 0.3 | V |
| Output voltage | $V_O$ | | –0.3 to $V_{DD}$ + 0.3 | V |
| Output current, low | $I_{OL}$ | Per pin | 15 | mA |
| | | Total of all pins | 100 | mA |
| Output current, high | $I_{OH}$ | Per pin | –10 | mA |
| | | Total of all pins | –40 | mA |
| Operating ambient temperature | $T_A$ | | –40 to +85 | °C |
| Storage temperature | $T_{stg}$ | | –65 to +150 | °C |

**Caution Product quality may suffer if the absolute maximum rating is exceeded even momentarily for any parameter. That is, the absolute maximum ratings are rated values at which the product is on the verge of suffering physical damage, and therefore the product must be used under conditions that ensure that the absolute maximum ratings are not exceeded.**

### Operating Conditions

- Operating ambient temperature ($T_A$): −40 to +85°C
- Power supply voltage and clock cycle time: see **Figure 29-1**
- Operating voltage during subsystem clock operation: $V_{DD}$ = 1.8 to 5.5 V

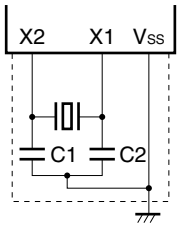
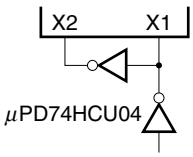
**Figure 29-1. Power Supply Voltage and Clock Cycle Time (CPU Clock Frequency: $f_{CPU}$)**




### Capacitance ($T_A$ = 25°C, $V_{DD}$ = $V_{SS}$ = 0 V)

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Input capacitance | $C_I$ | f = 1 MHz | | | 15 | pF |
| Output capacitance | $C_O$ | Unmeasured pins returned to 0 V. | | | 15 | pF |
| I/O capacitance | $C_{IO}$ | | | | 15 | pF |

**Main System Clock Oscillator Characteristics ($T_A$ = −40 to +85°C)**

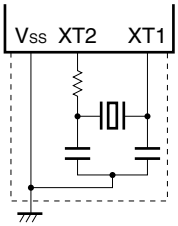
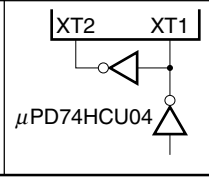
| Resonator | Recommended Circuit | Parameter | Conditions | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|---|
| Ceramic resonator or crystal resonator |  | Oscillation frequency ($f_X$) | ENMP = 0 | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | 4 | | 25 | MHz |
| | | | | 2.7 V ≤ $V_{DD}$ < 4.5 V | 4 | | 12.5 | |
| | | | | 2.0 V ≤ $V_{DD}$ < 2.7 V | 4 | | 6.25 | |
| | | | | 1.8 V ≤ $V_{DD}$ < 2.0 V | 4 | | 4 | |
| | | | ENMP = 1 | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | 2 | | 12.5 | MHz |
| | | | | 2.7 V ≤ $V_{DD}$ < 4.5 V | 2 | | 6.25 | |
| | | | | 2.0 V ≤ $V_{DD}$ < 2.7 V | 2 | | 3.125 | |
| | | | | 1.8 V ≤ $V_{DD}$ < 2.0 V | 2 | | 2 | |
| External clock |  | X1 input frequency ($f_X$) | ENMP = 0 | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | 4 | | 25 | MHz |
| | | | | 2.7 V ≤ $V_{DD}$ < 4.5 V | 4 | | 12.5 | |
| | | | | 2.0 V ≤ $V_{DD}$ < 2.7 V | 4 | | 6.25 | |
| | | | | 1.8 V ≤ $V_{DD}$ < 2.0 V | 4 | | 4 | |
| | | | ENMP = 1 | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | 2 | | 12.5 | MHz |
| | | | | 2.7 V ≤ $V_{DD}$ < 4.5 V | 2 | | 6.25 | |
| | | | | 2.0 V ≤ $V_{DD}$ < 2.7 V | 2 | | 3.125 | |
| | | | | 1.8 V ≤ $V_{DD}$ < 2.0 V | 2 | | 2 | |
| | | X1 input high-/low-level width ($t_{WXH}$, $t_{WXL}$) | | | 15 | | 250 | ns |
| | | X1 input rising/falling time ($t_{XR}$, $t_{XF}$) | | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | 0 | | 5 | ns |
| | | | | 2.7 V ≤ $V_{DD}$ < 4.5 V | 0 | | 10 | |
| | | | | 2.0 V ≤ $V_{DD}$ < 2.7 V | 0 | | 20 | |
| | | | | 1.8 V ≤ $V_{DD}$ < 2.0 V | 0 | | 30 | |

**Cautions 1. When using the main system clock oscillator, wire as follows in the area enclosed by the broken lines in the above figures to avoid an adverse effect from wiring capacitance.**

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

**2. When the main system clock is stopped and the device is operating on the subsystem clock, wait until the oscillation stabilization time has been secured by the program before switching back to the main system clock.**

**Subsystem Clock Oscillator Characteristics ($T_A$ = −40 to +85°C)**

| Resonator | Recommended Circuit | Parameter | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Crystal resonator |  | Oscillation frequency ($f_{XT}$) | | 32 | 32.768 | 35 | kHz |
| | | Oscillation stabilization time[Note] | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | | 1.2 | 2 | s |
| | | | 1.8 V ≤ $V_{DD}$ < 4.5 V | | | 10 | |
| External clock |  | XT1 input frequency ($f_{XT}$) | | 32 | | 35 | kHz |
| | | XT1 input high-/low-level width ($t_{XTH}$, $t_{XTL}$) | | 14.3 | | 15.6 | μs |

**Note** Time required to stabilize oscillation after applying the supply voltage ($V_{DD}$).

**Cautions 1. When using the subsystem clock oscillator, wire as follows in the area enclosed by the broken lines in the above figures to avoid an adverse effect from wiring capacitance.**

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

**2. When the main system clock is stopped and the device is operating on the subsystem clock, wait until the oscillation stabilization time has been secured by the program before switching back to the main system clock.**

**Remark** For the resonator selection and oscillator constant, customers are required to either evaluate the oscillation themselves or apply to the resonator manufacturer for evaluation.

**Recommended Oscillator Constants**

**Main system clock: Ceramic resonator connection ($T_A$ = −40 to +85°C)**

| Manufacturer | Product Name | Oscillation Frequency $f_{XX}$ (MHz) | Recommended Oscillator Constants | | Oscillation Voltage Range | | Oscillation Stabilization Time (MAX.) Tost (ms) |
|---|---|---|---|---|---|---|---|
| | | | C1 (pf) | C2 (pf) | MIN. (V) | MAX. (V) | |
| Murata Mfg. | CSA2.00MG040 | 2.0 | 100 | 100 | 1.8 | 5.5 | 0.60 |
| | CST2.00MG040 | 2.0 | On-chip | On-chip | 1.8 | 5.5 | 0.60 |
| | CSA3.00MG | 3.0 | 30 | 30 | 2.0 | 5.5 | 0.25 |
| | CST3.00MGW | 3.0 | On-chip | On-chip | 2.0 | 5.5 | 0.25 |
| | CSA4.00MG | 4.0 | 30 | 30 | 2.7 | 5.5 | 0.15 |
| | CST4.00MGW | 4.0 | On-chip | On-chip | 2.7 | 5.5 | 0.15 |
| | CSA6.00MG | 6.0 | 30 | 30 | 2.7 | 5.5 | 0.25 |
| | CST6.00MGW | 6.0 | On-chip | On-chip | 2.7 | 5.5 | 0.25 |
| | CSA8.00MTZ | 8.0 | 30 | 30 | 4.5 | 5.5 | 0.40 |
| | CST8.00MTW | 8.0 | On-chip | On-chip | 4.5 | 5.5 | 0.40 |
| | CSA12.0MTZ | 12.0 | 30 | 30 | 4.5 | 5.5 | 0.40 |
| | CST12.0MTW | 12.0 | On-chip | On-chip | 4.5 | 5.5 | 0.40 |
| Kyocera | PBRC2.00AR-A | 2.0 | 68 | 68 | 1.8 | 5.5 | 0.34 |
| | PBRC4.00HR | 4.0 | On-chip | On-chip | 2.7 | 5.5 | 0.25 |
| | KBR-4.0MKC | 4.0 | On-chip | On-chip | 2.7 | 5.5 | 0.25 |
| | PBRC8.00HR | 8.0 | On-chip | On-chip | 4.5 | 5.5 | 0.25 |
| | KBR-8.0MKC | 8.0 | On-chip | On-chip | 4.5 | 5.5 | 0.25 |
| | PBRC12.50BR | 12.5 | On-chip | On-chip | 4.5 | 5.5 | 0.14 |
| TDK | FCR4.0MC5 | 4.0 | On-chip | On-chip | 2.7 | 5.5 | 0.22 |
| | FCR6.0MC5 | 6.0 | On-chip | On-chip | 2.7 | 5.5 | 0.18 |
| | FCR8.0MC5 | 8.0 | On-chip | On-chip | 4.5 | 5.5 | 0.16 |

**Caution The oscillator constant and oscillation voltage range indicate conditions of stable oscillation. Oscillation frequency precision is not guaranteed. For applications requiring oscillation frequency precision, the oscillation frequency must be adjusted on the implementation circuit. For details, please contact directly the manufacturer of the resonator you will use.**

**DC Characteristics ($T_A$ = −40 to +85°C, $V_{DD}$ = $AV_{DD}$ = 1.8 to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0 V) (1/2)**

| Parameter | Symbol | Conditions | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Input voltage, low | $V_{IL1}$ | **Note 1** | 2.2 V ≤ $V_{DD}$ ≤ 5.5 V | 0 | | 0.3$V_{DD}$ | V |
| | | | 1.8 V ≤ $V_{DD}$ < 2.2 V | 0 | | 0.2$V_{DD}$ | |
| | $V_{IL2}$ | P00 to P05, P20, P22, P33, P34, P70, P72, $\overline{\text{RESET}}$ | 2.2 V ≤ $V_{DD}$ ≤ 5.5 V | 0 | | 0.2$V_{DD}$ | V |
| | | | 1.8 V ≤ $V_{DD}$ < 2.2 V | 0 | | 0.15$V_{DD}$ | |
| | $V_{IL3}$ | P10 to P17, P130, P131 | 2.2 V ≤ $V_{DD}$ ≤ 5.5 V | 0 | | 0.3$V_{DD}$ | V |
| | | | 1.8 V ≤ $V_{DD}$ < 2.2 V | 0 | | 0.2$V_{DD}$ | |
| | $V_{IL4}$ | X1, X2, XT1, XT2 | 2.2 V ≤ $V_{DD}$ ≤ 5.5 V | 0 | | 0.2$V_{DD}$ | V |
| | | | 1.8 V ≤ $V_{DD}$ < 2.2 V | 0 | | 0.1$V_{DD}$ | |
| | $V_{IL5}$ | P25, P27 | 2.2 V ≤ $V_{DD}$ ≤ 5.5 V | 0 | | 0.3$V_{DD}$ | V |
| | | | 1.8 V ≤ $V_{DD}$ < 2.2 V | 0 | | 0.2$V_{DD}$ | |
| Input voltage, high | $V_{IH1}$ | **Note 1** | 2.2 V ≤ $V_{DD}$ ≤ 5.5 V | 0.7$V_{DD}$ | | $V_{DD}$ | V |
| | | | 1.8 V ≤ $V_{DD}$ < 2.2 V | 0.8$V_{DD}$ | | $V_{DD}$ | |
| | $V_{IH2}$ | P00 to P05, P20, P22, P33, P34, P70, P72, $\overline{\text{RESET}}$ | 2.2 V ≤ $V_{DD}$ ≤ 5.5 V | 0.8$V_{DD}$ | | $V_{DD}$ | V |
| | | | 1.8 V ≤ $V_{DD}$ < 2.2 V | 0.85$V_{DD}$ | | $V_{DD}$ | |
| | $V_{IH3}$ | P10 to P17, P130, P131 | 2.2 V ≤ $V_{DD}$ ≤ 5.5 V | 0.7$V_{DD}$ | | $V_{DD}$ | V |
| | | | 1.8 V ≤ $V_{DD}$ < 2.2 V | 0.8$V_{DD}$ | | $V_{DD}$ | |
| | $V_{IH4}$ | X1, X2, XT1, XT2 | 2.2 V ≤ $V_{DD}$ ≤ 5.5 V | 0.8$V_{DD}$ | | $V_{DD}$ | V |
| | | | 1.8 V ≤ $V_{DD}$ < 2.2 V | 0.85$V_{DD}$ | | $V_{DD}$ | |
| | $V_{IH5}$ | P25, P27 | 2.2 V ≤ $V_{DD}$ ≤ 5.5 V | 0.7$V_{DD}$ | | $V_{DD}$ | V |
| | | | 1.8 V ≤ $V_{DD}$ < 2.2 V | 0.8$V_{DD}$ | | $V_{DD}$ | |
| Output voltage, low | $V_{OL1}$ | For pins other than P40 to P47, P50 to P57, $I_{OL}$ = 1.6 mA**Note 2** | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | | | 0.4 | V |
| | | P40 to P47, P50 to P57 $I_{OL}$ = 8 mA**Note 2** | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | | | 1.0 | V |
| | $V_{OL2}$ | $I_{OL}$ = 400 μA**Note 2** | 1.8 V ≤ $V_{DD}$ ≤ 5.5 V | | | 0.5 | V |
| Output voltage, high | $V_{OH1}$ | $I_{OH}$ = −1 mA**Note 2** | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | $V_{DD}$ − 1.0 | | | V |
| | | $I_{OH}$ = −100 μA**Note 2** | 1.8 V ≤ $V_{DD}$ ≤ 5.5 V | $V_{DD}$ − 0.5 | | | V |
| Input leakage current, low | $I_{LIL1}$ | $V_I$ = 0 V | Except X1, X2, XT1, XT2 | | | −3 | μA |
| | $I_{LIL2}$ | | X1, X2, XT1, XT2 | | | −20 | μA |
| Input leakage current, high | $I_{LIH1}$ | $V_I$ = $V_{DD}$ | Except X1, X2, XT1, XT2 | | | 3 | μA |
| | $I_{LIH2}$ | | X1, X2, XT1, XT2 | | | 20 | μA |
| Output leakage current, low | $I_{LOL1}$ | $V_O$ = 0 V | | | | −3 | μA |
| Output leakage current, high | $I_{LOH1}$ | $V_O$ = $V_{DD}$ | | | | 3 | μA |

**Notes 1.** P21, P23, P24, P26, P30 to P32, P35 to P37, P40 to P47, P50 to P57, P60 to P67, P71, P120 to P127

**2.** Per pin

**DC Characteristics ($T_A$ = −40 to +85°C, $V_{DD}$ = $AV_{DD}$ = 1.8 to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0 V) (2/2)**

| Parameter | Symbol | Conditions | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Supply voltage | $I_{DD1}$ | Operation mode | $f_{XX}$ = 12.5 MHz, $V_{DD}$ = 5.0 V ±10% | | 17 | 40 | mA |
| | | | $f_{XX}$ = 6 MHz, $V_{DD}$ = 3.0 V ±10% | | 5 | 17 | mA |
| | | | $f_{XX}$ = 2 MHz, $V_{DD}$ = 2.0 V ±10% | | 2 | 8 | mA |
| | $I_{DD2}$ | HALT mode | $f_{XX}$ = 12.5 MHz, $V_{DD}$ = 5.0 V ±10% | | 7 | 20 | mA |
| | | | $f_{XX}$ = 6 MHz, $V_{DD}$ = 3.0 V ±10% | | 2 | 8 | mA |
| | | | $f_{XX}$ = 2 MHz, $V_{DD}$ = 2.0 V ±10% | | 0.5 | 3.5 | mA |
| | $I_{DD3}$ | IDLE mode | $f_{XX}$ = 12.5 MHz, $V_{DD}$ = 5.0 V ±10% | | 1 | 2.5 | mA |
| | | | $f_{XX}$ = 6 MHz, $V_{DD}$ = 3.0 V ±10% | | 0.4 | 1.3 | mA |
| | | | $f_{XX}$ = 2 MHz, $V_{DD}$ = 2.0 V ±10% | | 0.2 | 0.9 | mA |
| | $I_{DD4}$ | Operation mode**Note** | $f_{XX}$ = 32 kHz, $V_{DD}$ = 5.0 V ±10% | | 80 | 200 | μA |
| | | | $f_{XX}$ = 32 kHz, $V_{DD}$ = 3.0 V ±10% | | 60 | 110 | μA |
| | | | $f_{XX}$ = 32 kHz, $V_{DD}$ = 2.0 V ±10% | | 30 | 100 | μA |
| | $I_{DD5}$ | HALT mode**Note** | $f_{XX}$ = 32 kHz, $V_{DD}$ = 5.0 V ±10% | | 60 | 160 | μA |
| | | | $f_{XX}$ = 32 kHz, $V_{DD}$ = 3.0 V ±10% | | 20 | 80 | μA |
| | | | $f_{XX}$ = 32 kHz, $V_{DD}$ = 2.0 V ±10% | | 10 | 70 | μA |
| | $I_{DD6}$ | IDLE mode**Note** | $f_{XX}$ = 32 kHz, $V_{DD}$ = 5.0 V ±10% | | 50 | 150 | μA |
| | | | $f_{XX}$ = 32 kHz, $V_{DD}$ = 3.0 V ±10% | | 15 | 70 | μA |
| | | | $f_{XX}$ = 32 kHz, $V_{DD}$ = 2.0 V ±10% | | 5 | 60 | μA |
| Data retention voltage | $V_{DDDR}$ | HALT, IDLE modes | | 1.8 | | 5.5 | V |
| Data retention current | $I_{DDDR}$ | STOP mode | $V_{DD}$ = 2.0 V ±10% | | 2 | 10 | μA |
| | | | $V_{DD}$ = 5.0 V ±10% | | 10 | 50 | μA |
| Pull-up resistor | $R_L$ | $V_I$ = 0 V | | 10 | 30 | 100 | kΩ |

**Note** When the main system clock is stopped and the subsystem clock is operating.

**Remark** Unless otherwise specified, the characteristics of alternate-function pins are the same as those of port pins.

## AC Characteristics ($T_A$ = −40 to +85°C, $V_{DD}$ = $AV_{DD}$ = 1.8 to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0 V)

### (1) Read/write operation (1/3)

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Cycle time | $t_{CYK}$ | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | 80 | | | ns |
| | | 2.7 V ≤ $V_{DD}$ < 4.5 V | 160 | | | ns |
| | | 2.0 V ≤ $V_{DD}$ < 2.7 V | 320 | | | ns |
| | | 1.8 V ≤ $V_{DD}$ < 2.0 V | 500 | | | ns |
| Address setup time (to ASTB↓) | $t_{SAST}$ | $V_{DD}$ = 5.0 V ±10% | (0.5 + a) T − 20 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | (0.5 + a) T − 40 | | | ns |
| | | $V_{DD}$ = 2.0 V ±10% | (0.5 + a) T − 80 | | | ns |
| Address hold time (from ASTB↓) | $t_{HSTLA}$ | $V_{DD}$ = 5.0 V ±10% | 0.5T − 19 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0.5T − 24 | | | ns |
| | | $V_{DD}$ = 2.0 V ±10% | 0.5T − 34 | | | ns |
| ASTB high-level width | $t_{WSTH}$ | $V_{DD}$ = 5.0 V ±10% | (0.5 + a) T − 17 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | (0.5 + a) T − 40 | | | ns |
| | | $V_{DD}$ = 2.0 V ±10% | (0.5 + a) T − 110 | | | ns |
| Address hold time (from $\overline{RD}$↑) | $t_{HRA}$ | $V_{DD}$ = 5.0 V ±10% | 0.5T − 14 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0.5T − 14 | | | ns |
| | | $V_{DD}$ = 2.0 V ±10% | 0.5T − 14 | | | ns |
| Delay time from address to $\overline{RD}$↓ | $t_{DAR}$ | $V_{DD}$ = 5.0 V ±10% | (1 + a) T − 24 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | (1 + a) T − 35 | | | ns |
| | | $V_{DD}$ = 2.0 V ±10% | (1 + a) T − 80 | | | ns |
| Address float time (from $\overline{RD}$↓) | $t_{FAR}$ | $V_{DD}$ = 5.0 V ±10% | | | 0 | ns |
| | | $V_{DD}$ = 3.0 V ±10% | | | 0 | ns |
| | | $V_{DD}$ = 2.0 V ±10% | | | 0 | ns |
| Data input time from address | $t_{DAID}$ | $V_{DD}$ = 5.0 V ±10% | | | (2.5 + a + n) T − 37 | ns |
| | | $V_{DD}$ = 3.0 V ±10% | | | (2.5 + a + n) T − 52 | ns |
| | | $V_{DD}$ = 2.0 V ±10% | | | (2.5 + a + n) T − 120 | ns |
| Data input time from ASTB↓ | $t_{DSTID}$ | $V_{DD}$ = 5.0 V ±10% | | | (2 + n) T − 35 | ns |
| | | $V_{DD}$ = 3.0 V ±10% | | | (2 + n) T − 50 | ns |
| | | $V_{DD}$ = 2.0 V ±10% | | | (2 + n) T − 80 | ns |
| Data input time from $\overline{RD}$↓ | $t_{DRID}$ | $V_{DD}$ = 5.0 V ±10% | | | (1.5 + n) T − 40 | ns |
| | | $V_{DD}$ = 3.0 V ±10% | | | (1.5 + n) T − 50 | ns |
| | | $V_{DD}$ = 2.0 V ±10% | | | (1.5 + n) T − 90 | ns |

**Remark** T: $t_{CYK}$ = 1/$f_{XX}$ ($f_{XX}$: Main system clock frequency)
a: 1 (during address wait), otherwise, 0
n: Number of wait states (n ≥ 0)

### (1) Read/write operation (2/3)

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Delay time from ASTB↓ to $\overline{RD}$↓ | $t_{DSTR}$ | $V_{DD}$ = 5.0 V ±10% | 0.5T – 9 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0.5T – 9 | | | ns |
| | | $V_{DD}$ = 2.0 V ±10% | 0.5T – 20 | | | ns |
| Data hold time (from $\overline{RD}$↑) | $t_{HRID}$ | $V_{DD}$ = 5.0 V ±10% | 0 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0 | | | ns |
| | | $V_{DD}$ = 2.0 V ±10% | 0 | | | ns |
| Address active time from $\overline{RD}$↑ | $t_{DRA}$ | $V_{DD}$ = 5.0 V ±10% | 0.5T – 2 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0.5T – 12 | | | ns |
| | | $V_{DD}$ = 2.0 V ±10% | 0.5T – 35 | | | ns |
| Delay time from $\overline{RD}$↑ to ASTB↑ | $t_{DRST}$ | $V_{DD}$ = 5.0 V ±10% | 0.5T – 9 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0.5T – 9 | | | ns |
| | | $V_{DD}$ = 2.0 V ±10% | 0.5T – 40 | | | ns |
| $\overline{RD}$ low-level width | $t_{WRL}$ | $V_{DD}$ = 5.0 V ±10% | (1.5 + n) T – 25 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | (1.5 + n) T – 30 | | | ns |
| | | $V_{DD}$ = 2.0 V ±10% | (1.5 + n) T – 25 | | | ns |
| Address active time from $\overline{WR}$↑ | $t_{DWA}$ | $V_{DD}$ = 5.0 V ±10% | 0.5T – 2 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0.5T – 12 | | | ns |
| | | $V_{DD}$ = 2.0 V ±10% | 0.5T – 35 | | | ns |
| Delay time from address to $\overline{WR}$↓ | $t_{DAW}$ | $V_{DD}$ = 5.0 V ±10% | (1 + a) T – 24 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | (1 + a) T – 34 | | | ns |
| | | $V_{DD}$ = 2.0 V ±10% | (1 + a) T – 70 | | | ns |
| Address hold time (from $\overline{WR}$↑) | $t_{HWA}$ | $V_{DD}$ = 5.0 V ±10% | 0.5T – 14 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0.5T – 14 | | | ns |
| | | $V_{DD}$ = 2.0 V ±10% | 0.5T – 14 | | | ns |
| Delay time from ASTB↓ to data output | $t_{DSTOD}$ | $V_{DD}$ = 5.0 V ±10% | | | 0.5T + 15 | ns |
| | | $V_{DD}$ = 3.0 V ±10% | | | 0.5T + 30 | ns |
| | | $V_{DD}$ = 2.0 V ±10% | | | 0.5T + 240 | ns |
| Delay time from $\overline{WR}$↓ to data output | $t_{DWOD}$ | $V_{DD}$ = 5.0 V ±10% | | | 0.5T – 30 | ns |
| | | $V_{DD}$ = 3.0 V ±10% | | | 0.5T – 30 | ns |
| | | $V_{DD}$ = 2.0 V ±10% | | | 0.5T – 30 | ns |
| Delay time from ASTB↓ to $\overline{WR}$↓ | $t_{DSTW}$ | $V_{DD}$ = 5.0 V ±10% | 0.5T – 9 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0.5T – 9 | | | ns |
| | | $V_{DD}$ = 2.0 V ±10% | 0.5T – 20 | | | ns |
| Data setup time (to $\overline{WR}$↑) | $t_{SODWR}$ | $V_{DD}$ = 5.0 V ±10% | (1.5 + n) T – 20 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | (1.5 + n) T – 25 | | | ns |
| | | $V_{DD}$ = 2.0 V ±10% | (1.5 + n) T – 70 | | | ns |

**Remark** T: $t_{CYK}$ = 1/fxx (fxx: Main system clock frequency)
a: 1 (during address wait), otherwise, 0
n: Number of wait states ($n \geq 0$)

### (1) Read/write operation (3/3)

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Data hold time (from $\overline{WR}\uparrow$) | $t_{HWOD}$ | $V_{DD}$ = 5.0 V ±10% | 0.5T – 14 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0.5T – 14 | | | ns |
| | | $V_{DD}$ = 2.0 V ±10% | 0.5T – 50 | | | ns |
| Delay time from $\overline{WR}\uparrow$ to ASTB$\uparrow$ | $t_{DWST}$ | $V_{DD}$ = 5.0 V ±10% | 0.5T – 9 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0.5T – 9 | | | ns |
| | | $V_{DD}$ = 2.0 V ±10% | 0.5T – 30 | | | ns |
| $\overline{WR}$ low-level width | $t_{WWL}$ | $V_{DD}$ = 5.0 V ±10% | (1.5 + n) T – 25 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | (1.5 + n) T – 30 | | | ns |
| | | $V_{DD}$ = 2.0 V ±10% | (1.5 + n) T – 30 | | | ns |
| Delay time from address to EXA$\downarrow$ | $t_{ADEXD}$ | $V_{DD}$ = 5.0 V ±10% | 0 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0 | | | ns |
| | | $V_{DD}$ = 2.0 V ±10% | 0 | | | ns |
| Delay time from EXA$\downarrow$ to ASTB$\downarrow$ | $t_{EXTAH}$ | $V_{DD}$ = 5.0 V ±10% | 0.5T – 20 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0.5T – 30 | | | ns |
| | | $V_{DD}$ = 2.0 V ±10% | 0.5T – 40 | | | ns |
| Delay time from $\overline{RD}\uparrow$ to EXA$\uparrow$ | $t_{EXRDS}$ | $V_{DD}$ = 5.0 V ±10% | 0 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0 | | | ns |
| | | $V_{DD}$ = 2.0 V ±10% | 0 | | | ns |
| Delay time from $\overline{WR}\uparrow$ to EXA$\uparrow$ | $t_{EXWDS}$ | $V_{DD}$ = 5.0 V ±10% | T | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | T | | | ns |
| | | $V_{DD}$ = 2.0 V ±10% | T | | | ns |
| Delay time from EXA$\uparrow$ to ASTB$\uparrow$ | $t_{EXADR}$ | $V_{DD}$ = 5.0 V ±10% | 0.5T | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0.5T | | | ns |
| | | $V_{DD}$ = 2.0 V ±10% | 0.5T | | | ns |

**Remark** T: $t_{CYK}$ = 1/fxx (fxx: Main system clock frequency)
n: Number of wait states (n ≥ 0)

### (2) External wait timing (1/2)

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Input time from address to $\overline{WAIT}\downarrow$ | $t_{DAWT}$ | $V_{DD}$ = 5.0 V ±10% | | | (2 + a) T – 40 | ns |
| | | $V_{DD}$ = 3.0 V ±10% | | | (2 + a) T – 60 | ns |
| | | $V_{DD}$ = 2.0 V ±10% | | | (2 + a) T – 300 | ns |
| Input time from ASTB↓ to $\overline{WAIT}\downarrow$ | $t_{DSTWT}$ | $V_{DD}$ = 5.0 V ±10% | | | 1.5T – 40 | ns |
| | | $V_{DD}$ = 3.0 V ±10% | | | 1.5T – 60 | ns |
| | | $V_{DD}$ = 2.0 V ±10% | | | 1.5T – 260 | ns |
| Hold time from ASTB↓ to $\overline{WAIT}$ | $t_{HSTWT}$ | $V_{DD}$ = 5.0 V ±10% | (0.5 + n) T + 5 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | (0.5 + n) T + 10 | | | ns |
| | | $V_{DD}$ = 2.0 V ±10% | (0.5 + n) T + 30 | | | ns |
| Delay time from ASTB↓ to $\overline{WAIT}\uparrow$ | $t_{DSTWTH}$ | $V_{DD}$ = 5.0 V ±10% | | | (1.5 + n) T – 40 | ns |
| | | $V_{DD}$ = 3.0 V ±10% | | | (1.5 + n) T – 60 | ns |
| | | $V_{DD}$ = 2.0 V ±10% | | | (1.5 + n) T – 90 | ns |
| Input time from $\overline{RD}\downarrow$ to $\overline{WAIT}\downarrow$ | $t_{DRWTL}$ | $V_{DD}$ = 5.0 V ±10% | | | T – 40 | ns |
| | | $V_{DD}$ = 3.0 V ±10% | | | T – 60 | ns |
| | | $V_{DD}$ = 2.0 V ±10% | | | T – 70 | ns |
| Hold time from $\overline{RD}\downarrow$ to $\overline{WAIT}$ | $t_{HRWT}$ | $V_{DD}$ = 5.0 V ±10% | nT + 5 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | nT + 10 | | | ns |
| | | $V_{DD}$ = 2.0 V ±10% | nT + 30 | | | ns |
| Delay time from $\overline{RD}\downarrow$ to $\overline{WAIT}\uparrow$ | $t_{DRWTH}$ | $V_{DD}$ = 5.0 V ±10% | | | (1 + n) T – 40 | ns |
| | | $V_{DD}$ = 3.0 V ±10% | | | (1 + n) T – 60 | ns |
| | | $V_{DD}$ = 2.0 V ±10% | | | (1 + n) T – 90 | ns |
| Input time from $\overline{WAIT}\uparrow$ to data | $t_{DWTID}$ | $V_{DD}$ = 5.0 V ±10% | | | 0.5T – 5 | ns |
| | | $V_{DD}$ = 3.0 V ±10% | | | 0.5T – 10 | ns |
| | | $V_{DD}$ = 2.0 V ±10% | | | 0.5T – 30 | ns |
| Delay time from $\overline{WAIT}\uparrow$ to $\overline{RD}\uparrow$ | $t_{DWTR}$ | $V_{DD}$ = 5.0 V ±10% | 0.5T | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0.5T | | | ns |
| | | $V_{DD}$ = 2.0 V ±10% | 0.5T + 5 | | | ns |
| Delay time from $\overline{WAIT}\uparrow$ to $\overline{WR}\uparrow$ | $t_{DWTW}$ | $V_{DD}$ = 5.0 V ±10% | 0.5T | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0.5T | | | ns |
| | | $V_{DD}$ = 2.0 V ±10% | 0.5T + 5 | | | ns |
| Delay time from $\overline{WR}\downarrow$ to $\overline{WAIT}\downarrow$ | $t_{DWWTL}$ | $V_{DD}$ = 5.0 V ±10% | | | T – 40 | ns |
| | | $V_{DD}$ = 3.0 V ±10% | | | T – 60 | ns |
| | | $V_{DD}$ = 2.0 V ±10% | | | T – 90 | ns |

**Remark** T: $t_{CYK}$ = 1/fxx (fxx: Main system clock frequency)
a: 1 (during address wait), otherwise, 0
n: Number of wait states ($n \geq 0$)

### (2) External wait timing (2/2)

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Hold time from $\overline{WR}\downarrow$ to $\overline{WAIT}$ | $t_{HWWT}$ | $V_{DD}$ = 5.0 V ±10% | nT + 5 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | nT + 10 | | | ns |
| | | $V_{DD}$ = 2.0 V ±10% | nT + 30 | | | ns |
| Delay time from $\overline{WR}\downarrow$ to $\overline{WAIT}\uparrow$ | $t_{DWWTH}$ | $V_{DD}$ = 5.0 V ±10% | | | (1 + n) T − 40 | ns |
| | | $V_{DD}$ = 3.0 V ±10% | | | (1 + n) T − 60 | ns |
| | | $V_{DD}$ = 2.0 V ±10% | | | (1 + n) T − 90 | ns |

**Remark** T: $t_{CYK}$ = 1/$f_{XX}$ ($f_{XX}$: Main system clock frequency)
a: 1 (during address wait), otherwise, 0
n: Number of wait states ($n \geq 0$)

### (3) Serial Operation ($T_A$ = −40 to +85°C, $V_{DD}$ = $AV_{DD}$ = 1.8 to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0 V) (1/2)

#### (a) 3-wire serial I/O mode ($\overline{SCK}$: Internal clock output)

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{SCK}$ cycle time | $t_{KCY1}$ | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | 640 | | | ns |
| | | 2.7 V ≤ $V_{DD}$ < 4.5 V | 1,280 | | | ns |
| | | 2.0 V ≤ $V_{DD}$ < 2.7 V | 2,560 | | | ns |
| | | 1.8 V ≤ $V_{DD}$ < 2.0 V | 4,000 | | | ns |
| ~~SCK~~ high-/low-level width | $t_{KH1}$, $t_{KL1}$ | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | 270 | | | ns |
| | | 2.7 V ≤ $V_{DD}$ < 4.5 V | 590 | | | ns |
| | | 2.0 V ≤ $V_{DD}$ < 2.7 V | 1,180 | | | ns |
| | | 1.8 V ≤ $V_{DD}$ < 2.0 V | 1,900 | | | ns |
| SI setup time (to $\overline{SCK}$↑) | $t_{SIK1}$ | 2.7 V ≤ $V_{DD}$ ≤ 5.5 V | 10 | | | ns |
| | | 1.8 V ≤ $V_{DD}$ < 2.7 V | 30 | | | ns |
| SI hold time (from $\overline{SCK}$↑) | $t_{HIK1}$ | | 40 | | | ns |
| SO output delay time (from SCK↓) | $t_{DSO1}$ | | | | 30 | ns |
| SO output hold time (from $\overline{SCK}$↑) | $t_{HSO1}$ | | $0.5t_{KCY1}$ − 50 | | | ns |

#### (b) 3-wire serial I/O mode ($\overline{SCK}$: External clock input)

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{SCK}$ cycle time | $t_{KCY2}$ | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | 640 | | | ns |
| | | 2.7 V ≤ $V_{DD}$ < 4.5 V | 1,280 | | | ns |
| | | 2.0 V ≤ $V_{DD}$ < 2.7 V | 2,560 | | | ns |
| | | 1.8 V ≤ $V_{DD}$ < 2.0 V | 4,000 | | | ns |
| $\overline{SCK}$ high-/low-level width | $t_{KH2}$, $t_{KL2}$ | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | 320 | | | ns |
| | | 2.7 V ≤ $V_{DD}$ < 4.5 V | 640 | | | ns |
| | | 2.0 V ≤ $V_{DD}$ < 2.7 V | 1,280 | | | ns |
| | | 1.8 V ≤ $V_{DD}$ < 2.0 V | 2,000 | | | ns |
| SI setup time (to $\overline{SCK}$↑) | $t_{SIK2}$ | 2.7 V ≤ $V_{DD}$ ≤ 5.5 V | 10 | | | ns |
| | | 1.8 V ≤ $V_{DD}$ < 2.7 V | 30 | | | ns |
| SI hold time (from $\overline{SCK}$↑) | $t_{HIK2}$ | | 40 | | | ns |
| SO output delay time (from $\overline{SCK}$↓) | $t_{DSO2}$ | | | | 30 | ns |
| SO output hold time (from $\overline{SCK}$↑) | $t_{HSO2}$ | | $0.5t_{KCY2}$ − 50 | | | ns |

#### (c) UART mode

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| ASCK cycle time | $t_{KCY3}$ | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | 417 | | | ns |
| | | 2.7 V ≤ $V_{DD}$ < 4.5 V | 833 | | | ns |
| | | 1.8 V ≤ $V_{DD}$ < 2.7 V | 1,667 | | | ns |
| ASCK high-/low-level width | $t_{KH3}$, $t_{KL3}$ | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | 208 | | | ns |
| | | 2.7 V ≤ $V_{DD}$ < 4.5 V | 416 | | | ns |
| | | 1.8 V ≤ $V_{DD}$ < 2.7 V | 833 | | | ns |

### (3) Serial Operation ($T_A$ = −40 to +85°C, $V_{DD}$ = $AV_{DD}$ = 1.8 to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0 V) (2/2)

#### (d) $I^2C$ bus mode ($\mu$PD784224Y, 784225Y only)

| Parameter | | Symbol | Standard Mode | | High-Speed Mode | | Unit |
|---|---|---|---|---|---|---|---|
| | | | MIN. | MAX. | MIN. | MAX. | |
| SCL0 clock frequency | | $f_{CLK}$ | 0 | 100 | 0 | 400 | kHz |
| Bus free time (between stop and start conditions) | | $t_{BUF}$ | 4.7 | – | 1.3 | – | μs |
| Hold time[Note1] | | $t_{HD:STA}$ | 4.0 | – | 0.6 | – | μs |
| Low-level width of SCL0 clock | | $t_{LOW}$ | 4.7 | – | 1.3 | – | μs |
| High-level width of SCL0 clock | | $t_{HIGH}$ | 4.0 | – | 0.6 | – | μs |
| Setup time of start/restart conditions | | $t_{SU:STA}$ | 4.7 | – | 0.6 | – | μs |
| Data hold time | When using CBUS-compatible master | $t_{HD:DAT}$ | 5.0 | – | – | – | μs |
| | When using $I^2C$ bus | | 0[Note 2] | – | 0[Note 2] | 0.9[Note 3] | μs |
| Data setup time | | $t_{SU:DAT}$ | 250 | – | 100[Note 4] | – | ns |
| Rising time of SDA0 and SCL0 signals | | $t_R$ | – | 1,000 | 20 + 0.1Cb[Note 5] | 300 | ns |
| Falling time of SDA0 and SCL0 signals | | $t_F$ | – | 300 | 20 + 0.1Cb[Note 5] | 300 | ns |
| Setup time of stop condition | | $t_{SU:STO}$ | 4.0 | – | 0.6 | – | μs |
| Pulse width of spike restricted by input filter | | $t_{SP}$ | – | – | 0 | 50 | ns |
| Load capacitance of each bus line | | Cb | – | 400 | – | 400 | pF |

**Notes 1.** For the start condition, the first clock pulse is generated after the hold time.

**2.** To fill the undefined area of the SCL0 falling edge, it is necessary for the device to provide an internal SDA0 signal (on $V_{IHmin.}$) with at least 300 ns of hold time.

**3.** If the device does not extend the SCL0 signal low-level hold time ($t_{LOW}$), only the maximum data hold time $t_{HD:DAT}$ needs to be satisfied.

**4.** The high-speed mode $I^2C$ bus can be used in a standard mode $I^2C$ bus system. In this case, the conditions described below must be satisfied.

- If the device does not extend the SCL0 signal low-level hold time
  $t_{SU:DAT} \geq 250$ ns
- If the device extends the SCL0 signal low-level hold time
  Be sure to transmit the data bit to the SDA0 line before the SCL0 line is released
  ($t_{Rmax.} + t_{SU:DAT}$ = 1,000 + 250 = 1,250 ns by standard mode $I^2C$ bus specification)

**5.** Cb: Total capacitance per bus line (unit: pF)

### (4) Clock Output Operation ($T_A$ = −40 to +85°C, $V_{DD}$ = $AV_{DD}$ = 1.8 to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0 V)

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| PCL cycle time | $t_{CYCL}$ | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V, nT | 80 | | 31,250 | ns |
| PCL high-/low-level width | $t_{CLL}$, $t_{CLH}$ | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V, 0.5T − 10 | 30 | | 15,615 | ns |
| PCL rising/falling time | $t_{CLR}$, $t_{CLF}$ | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | | | 5 | ns |
| | | 2.7 V ≤ $V_{DD}$ < 4.5 V | | | 10 | ns |
| | | 1.8 V ≤ $V_{DD}$ < 2.7 V | | | 20 | ns |

**Remark** T: $t_{CYK}$ = 1/$f_{XX}$ ($f_{XX}$: Main system clock frequency)

n: Divided frequency ratio set by software in the CPU

- When using the main system clock: n = 1, 2, 4, 8, 16, 32, 64, 128
- When using the subsystem clock: n = 1

### (5) Other Operations ($T_A$ = −40 to +85°C, $V_{DD}$ = $AV_{DD}$ = 1.8 to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0 V)

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| NMI high-/low-level width | $t_{WNIL}$, $t_{WNIH}$ | | 10 | | | μs |
| INTP input high-/low-level width | $t_{WITL}$, $t_{WITH}$ | INTP0 to INTP5 | 100 | | | ns |
| $\overline{\text{RESET}}$ high-/low-level width | $t_{WRSL}$, $t_{WRSH}$ | | 10 | | | μs |

### A/D Converter Characteristics ($T_A$ = −40 to +85°C, $V_{DD}$ = $AV_{DD}$ = 1.8 to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0 V)

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Resolution | | | 8 | 8 | 8 | bit |
| Overall error[Notes 1, 2] | | 6.25 MHz < $f_{XX}$ ≤ 12.5 MHz, 4.5 V ≤ $V_{DD}$ ≤ 5.5 V, $AV_{DD}$ = $V_{DD}$ | | | ±1.2 | %FSR |
| | | 3.125 MHz < $f_{XX}$ ≤ 6.25 MHz, 2.7 V ≤ $V_{DD}$ ≤ 5.5 V, $AV_{DD}$ = $V_{DD}$ | | | ±1.2 | %FSR |
| | | 2 MHz < $f_{XX}$ ≤ 3.125 MHz, 2.0 V ≤ $V_{DD}$ ≤ 5.5 V, $AV_{DD}$ = $V_{DD}$ | | | ±1.6 | %FSR |
| | | $f_{XX}$ = 2 MHz, 1.8 V ≤ $V_{DD}$ ≤ 5.5 V $AV_{DD}$ = $V_{DD}$ | | | ±1.6 | %FSR |
| Conversion time | $t_{CONV}$ | | 14 | | 144 | μs |
| Sampling time | $t_{SAMP}$ | | 24/$f_{XX}$ | | | μs |
| Analog input voltage | $V_{IAN}$ | | $AV_{SS}$ | | $AV_{DD}$ | V |
| Reference voltage | $AV_{DD}$ | | $V_{DD}$ | $V_{DD}$ | $V_{DD}$ | V |
| Resistance between $AV_{DD}$ and $AV_{SS}$ | $R_{AVREF0}$ | A/D conversion is not performed | | 40 | | kΩ |

**Notes 1.** Excludes quantization error (±0.2%FSR).

**2.** This value is indicated as a ratio to the full-scale value (%FSR).

**Remark** $f_{XX}$: Main system clock frequency

### D/A Converter Characteristics ($T_A$ = −40 to +85°C, $V_{DD}$ = $AV_{DD}$ = 1.8 to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0 V)

| Parameter | Symbol | Conditions | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Resolution | | | | 8 | 8 | 8 | bit |
| Overall error[Notes 1, 2] | | 2.0 V ≤ $V_{DD}$ ≤ 5.5 V R = 10 MΩ, 2.0 V ≤ $AV_{REF1}$ ≤ $AV_{DD}$, $AV_{DD}$ = $V_{DD}$ | | | | ±0.6 | %FSR |
| | | 1.8 V ≤ $V_{DD}$ ≤ 2.0 V R = 10 MΩ, 1.8 V ≤ $AV_{REF1}$ ≤ $AV_{DD}$, $AV_{DD}$ = $V_{DD}$ | | | | ±1.2 | %FSR |
| Settling time | | Load conditions: C = 30 pF | 4.5 V ≤ $AV_{REF1}$ ≤ 5.5 V | | | 10 | μs |
| | | | 2.7 V ≤ $AV_{REF1}$ < 4.5 V | | | 15 | μs |
| | | | 1.8 V ≤ $AV_{REF1}$ < 2.7 V | | | 20 | μs |
| Output resistance | $R_O$ | DACS0, 1 = 55H | | | 8 | | kΩ |
| Reference voltage | $AV_{REF1}$ | | | 1.8 | | $V_{DD}$ | V |
| $AV_{REF1}$ current | $AI_{REF1}$ | For only 1 channel | | | | 2.5 | mA |

**Notes 1.** Excludes quantization error (±0.2%FSR).

**2.** This value is indicated as a ratio to the full-scale value (%FSR).

**Data Retention Characteristics ($T_A$ = −40 to +85°C, $V_{DD}$ = $AV_{DD}$ = 1.8 to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0 V)**

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Data retention voltage | $V_{DDDR}$ | STOP mode | 1.8 | | 5.5 | V |
| Data retention current | $I_{DDDR}$ | $V_{DDDR}$ = 5.0 V ±10% | | 10 | 50 | μA |
| | | $V_{DDDR}$ = 2.0 V ±10% | | 2 | 10 | μA |
| $V_{DD}$ rise time | $t_{RVD}$ | | 200 | | | μs |
| $V_{DD}$ fall time | $t_{FVD}$ | | 200 | | | μs |
| $V_{DD}$ hold time (from STOP mode setting) | $t_{HVD}$ | | 0 | | | ms |
| STOP release signal input time | $t_{DREL}$ | | 0 | | | ms |
| Oscillation stabilization wait time | $t_{WAIT}$ | Crystal resonator | 30 | | | ms |
| | | Ceramic resonator | 5 | | | ms |
| Low-level input voltage | $V_{IL}$ | $\overline{\text{RESET}}$, P00/INTP0 to P05/INTP5 | 0 | | $0.1V_{DDDR}$ | V |
| High-level input voltage | $V_{IH}$ | | $0.9V_{DDDR}$ | | $V_{DDDR}$ | V |

## 29.2 Electrical Specifications of μPD78F4225 and 78F4225Y

For the timing charts, refer to **29.3 Timing Charts**.

**Absolute Maximum Ratings ($T_A$ = 25°C)**

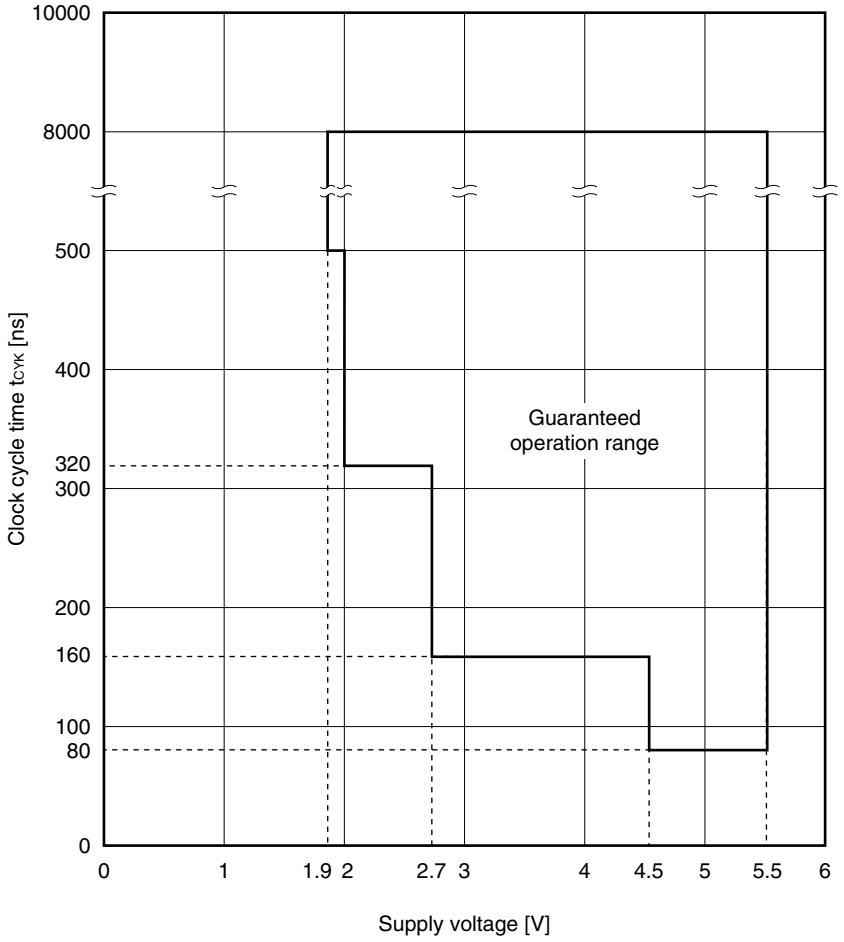
| Parameter | Symbol | Conditions | Ratings | Unit |
|---|---|---|---|---|
| Supply voltage | $V_{DD}$ | $V_{DD} = V_{DD0} = V_{DD1}$ | −0.3 to +6.5 | V |
| | $V_{PP}$ | | −0.3 to +10.5 | V |
| | $AV_{DD}$ | | −0.3 to $V_{DD0}$ + 0.3 | V |
| | $AV_{SS}$ | | −0.3 to $V_{SS0}$ + 0.3 | V |
| | $AV_{REF1}$ | D/A converter reference voltage input | −0.3 to $V_{DD0}$ + 0.3 | V |
| Input voltage | $V_{I1}$ | | −0.3 to $V_{DD0}$ + 0.3 | V |
| | $V_{I2}$ | $V_{PP}$ pin during programming | −0.3 to +10.5 | V |
| Analog input voltage | $V_{AN}$ | Analog input pin | $AV_{SS}$ − 0.3 to $AV_{REF1}$ + 0.3 | V |
| Output voltage | $V_O$ | | −0.3 to $V_{DD}$ + 0.3 | V |
| Output current, low | $I_{OL}$ | Per pin | 15 | mA |
| | | Total for all pins | 100 | mA |
| Output current, high | $I_{OH}$ | Per pin | −10 | mA |
| | | Total for all pins | −40 | mA |
| Operating ambient temperature | $T_A$ | During normal operation | −40 to +85 | °C |
| | | During flash memory programming | +10 to +40 | °C |
| Storage temperature | $T_{stg}$ | | −65 to +125 | °C |

**Caution Product quality may suffer if the absolute maximum rating is exceeded even momentarily for any parameter. That is, the absolute maximum ratings are rated values at which the product is on the verge of suffering physical damage, and therefore the product must be used under conditions that ensure that the absolute maximum ratings are not exceeded.**

### Operating Conditions

- Operating ambient temperature ($T_A$): −40 to +85°C
- Power supply voltage and clock cycle time: See **Figure 29-2**.
- Operating voltage during subsystem clock operation: $V_{DD}$ = 1.9 to 5.5 V

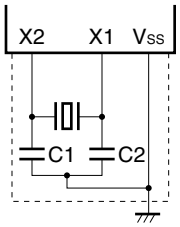
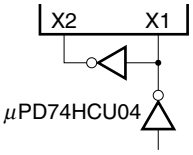
**Figure 29-2. Power Supply Voltage and Clock Cycle Time (CPU Clock Frequency: $f_{CPU}$)**




### Capacitance ($T_A$ = 25°C, $V_{DD}$ = $V_{SS}$ = 0 V)

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Input capacitance | $C_I$ | f = 1 MHz<br>Unmeasured pins returned to 0 V. | | | 15 | pF |
| Output capacitance | $C_O$ | | | | 15 | pF |
| I/O capacitance | $C_{IO}$ | | | | 15 | pF |

**Main System Clock Oscillator Characteristics ($T_A$ = −40 to +85°C)**

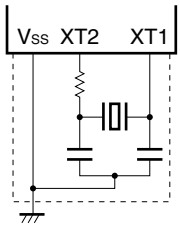
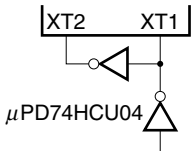
| Resonator | Recommended Circuit | Parameter | Conditions | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|---|
| Ceramic resonator or crystal resonator |  | Oscillation frequency ($f_X$) | ENMP = 0 | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | 4 | | 25 | MHz |
| | | | | 2.7 V ≤ $V_{DD}$ < 4.5 V | 4 | | 12.5 | |
| | | | | 2.0 V ≤ $V_{DD}$ < 2.7 V | 4 | | 6.25 | |
| | | | | 1.9 V ≤ $V_{DD}$ < 2.0 V | 4 | | 4 | |
| | | | ENMP = 1 | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | 2 | | 12.5 | MHz |
| | | | | 2.7 V ≤ $V_{DD}$ < 4.5 V | 2 | | 6.25 | |
| | | | | 2.0 V ≤ $V_{DD}$ < 2.7 V | 2 | | 3.125 | |
| | | | | 1.9 V ≤ $V_{DD}$ < 2.0 V | 2 | | 2 | |
| External clock |  | X1 input frequency ($f_X$) | ENMP = 0 | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | 4 | | 25 | MHz |
| | | | | 2.7 V ≤ $V_{DD}$ < 4.5 V | 4 | | 12.5 | |
| | | | | 2.0 V ≤ $V_{DD}$ < 2.7 V | 4 | | 6.25 | |
| | | | | 1.9 V ≤ $V_{DD}$ < 2.0 V | 4 | | 4 | |
| | | | ENMP = 1 | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | 2 | | 12.5 | MHz |
| | | | | 2.7 V ≤ $V_{DD}$ < 4.5 V | 2 | | 6.25 | |
| | | | | 2.0 V ≤ $V_{DD}$ < 2.7 V | 2 | | 3.125 | |
| | | | | 1.9 V ≤ $V_{DD}$ < 2.0 V | 2 | | 2 | |
| | | X1 input high-/low-level width ($t_{WXH}$, $t_{WXL}$) | | | 15 | | 250 | ns |
| | | X1 input rising/falling time ($t_{XR}$, $t_{XF}$) | | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | 0 | | 5 | ns |
| | | | | 2.7 V ≤ $V_{DD}$ < 4.5 V | 0 | | 10 | |
| | | | | 2.0 V ≤ $V_{DD}$ < 2.7 V | 0 | | 20 | |
| | | | | 1.9 V ≤ $V_{DD}$ < 2.0 V | 0 | | 30 | |

**Cautions 1. When using the main system clock oscillator, wire as follows in the area enclosed by the broken lines in the above figures to avoid an adverse effect from wiring capacitance.**

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

**2. When the main system clock is stopped and the device is operating on the subsystem clock, wait until the oscillation stabilization time has been secured by the program before switching back to the main system clock.**

**Subsystem Clock Oscillator Characteristics ($T_A$ = −40 to +85°C)**

| Resonator | Recommended Circuit | Parameter | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Crystal resonator |  | Oscillation frequency ($f_{XT}$) | | 32 | 32.768 | 35 | kHz |
| | | Oscillation stabilization time[Note] | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | | 1.2 | 2 | s |
| | | | 1.9 V ≤ $V_{DD}$ < 4.5 V | | | 10 | |
| External clock |  | XT1 input frequency ($f_{XT}$) | | 32 | | 35 | kHz |
| | | XT1 input high-/low-level width ($t_{XTH}$, $t_{XTL}$) | | 14.3 | | 15.6 | μs |

**Note** Time required to stabilize oscillation after applying supply voltage ($V_{DD}$).

**Cautions 1. When using the subsystem clock oscillator, wire as follows in the area enclosed by the broken lines in the above figures to avoid an adverse effect from wiring capacitance.**

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

**2. When the main system clock is stopped and the device is operating on the subsystem clock, wait until the oscillation stabilization time has been secured by the program before switching back to the main system clock.**

**Remark** For the resonator selection and oscillator constant, customers are required to either evaluate the oscillation themselves or apply to the resonator manufacturer for evaluation.

### Recommended Oscillator Constants

**Main system clock: Ceramic resonator connection ($T_A$ = –40 to +85°C)**

| Manufacturer | Part Number | Oscillation Frequency $f_{XX}$ (MHz) | Recommended Circuit Constant | | Oscillation Voltage Range | | Oscillation Stabilization Time (MAX.) Tost (ms) |
|---|---|---|---|---|---|---|---|
| | | | C1 (pf) | C2 (pf) | MIN. (V) | MAX. (V) | |
| Murata Mfg. Co., Ltd. | CSTLS2M00G56-B0 | 2.0 | On-chip | On-chip | 1.9 | 5.5 | 0.44 |
| | CSTCC2.00MG0H6 | 2.0 | On-chip | On-chip | 1.9 | 5.5 | 0.40 |
| | CSTCR4M00G55-R0 | 4.0 | On-chip | On-chip | 2.7 | 5.5 | 0.38 |
| | CSTS0400MG06 | 4.0 | On-chip | On-chip | 2.7 | 5.5 | 0.40 |
| | CSTCC4.00MG0H6 | 4.0 | On-chip | On-chip | 2.7 | 5.5 | 0.38 |
| | CSTCR6M00G53-R0 | 6.0 | On-chip | On-chip | 2.7 | 5.5 | 0.28 |
| | CSTS0600MG03 | 6.0 | On-chip | On-chip | 2.7 | 5.5 | 0.24 |
| | CSTCC6.00MG | 6.0 | On-chip | On-chip | 2.7 | 5.5 | 0.23 |
| | CSTS0800MG03 | 8.0 | On-chip | On-chip | 4.5 | 5.5 | 0.22 |
| | CSTCC8.00MG | 8.0 | On-chip | On-chip | 4.5 | 5.5 | 0.22 |
| | CSTS1000MG03 | 10.0 | On-chip | On-chip | 4.5 | 5.5 | 0.23 |
| | CSTCC10.0MG | 10.0 | On-chip | On-chip | 4.5 | 5.5 | 0.22 |
| | CSA12.5MTZ | 12.5 | 30 | 30 | 4.5 | 5.5 | 0.24 |
| | CST12.5MTW | 12.5 | On-chip | On-chip | 4.5 | 5.5 | 0.24 |
| | CSTCV12.5MTJ0C4 | 12.5 | On-chip | On-chip | 4.5 | 5.5 | 0.22 |
| Kyocera Corporation | PBRC4.00HR | 4.0 | On-chip | On-chip | 2.7 | 5.5 | 0.3 |
| | PBRC4.00GR | 4.0 | 33 | 33 | 2.7 | 5.5 | 0.3 |
| | KBR-4.0MKC | 4.0 | On-chip | On-chip | 2.7 | 5.5 | 0.3 |
| | KBR-4.0MSB | 4.0 | 33 | 33 | 2.7 | 5.5 | 0.3 |
| | PBRC8.00HR | 8.0 | On-chip | On-chip | 4.5 | 5.5 | 0.3 |
| | PBRC8.00GR | 8.0 | 33 | 33 | 4.5 | 5.5 | 0.3 |
| | KBR-8.0MKC | 8.0 | On-chip | On-chip | 4.5 | 5.5 | 0.3 |
| | KBR-8.0MSB | 8.0 | 33 | 33 | 4.5 | 5.5 | 0.3 |
| | PBRC10.00BR-A | 10.0 | On-chip | On-chip | 4.5 | 5.5 | 0.2 |
| | PBRC12.50BR-A | 12.5 | On-chip | On-chip | 4.5 | 5.5 | 0.1 |
| TDK | FCR4.0MC5 | 4.0 | On-chip | On-chip | 2.7 | 5.5 | 0.15 |
| | FCR6.0MC5 | 6.0 | On-chip | On-chip | 2.7 | 5.5 | 0.15 |
| | FCR8.0MC5 | 8.0 | On-chip | On-chip | 4.5 | 5.5 | 0.12 |

**Caution The oscillator constant and oscillation voltage range indicate conditions of stable oscillation. Oscillation frequency precision is not guaranteed. For applications requiring oscillation frequency precision, the oscillation frequency must be adjusted on the implementation circuit. For details, please contact directly the manufacturer of the resonator you will use.**

**DC Characteristics ($T_A$ = −40 to +85°C, $V_{DD}$ = $AV_{DD}$ = 1.9 to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0 V) (1/2)**

| Parameter | Symbol | Conditions | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| $V_{PP}$ supply voltage | $V_{PP1}$ | In normal operation | | 0 | | $0.2V_{DD}$ | V |
| Input voltage, low | $V_{IL1}$ | **Note 1** | 2.2 V ≤ $V_{DD}$ ≤ 5.5 V | 0 | | $0.3V_{DD}$ | V |
| | | | 1.9 V ≤ $V_{DD}$ < 2.2 V | 0 | | $0.2V_{DD}$ | |
| | $V_{IL2}$ | P00 to P05, P20, P22, P33, P34, P70, P72, $\overline{\text{RESET}}$ | 2.2 V ≤ $V_{DD}$ ≤ 5.5 V | 0 | | $0.2V_{DD}$ | V |
| | | | 1.9 V ≤ $V_{DD}$ < 2.2 V | 0 | | $0.15V_{DD}$ | |
| | $V_{IL3}$ | P10 to P17, P130, P131 | 2.2 V ≤ $V_{DD}$ ≤ 5.5 V | 0 | | $0.3V_{DD}$ | V |
| | | | 1.9 V ≤ $V_{DD}$ < 2.2 V | 0 | | $0.2V_{DD}$ | |
| | $V_{IL4}$ | X1, X2, XT1, XT2 | 2.2 V ≤ $V_{DD}$ ≤ 5.5 V | 0 | | $0.2V_{DD}$ | V |
| | | | 1.9 V ≤ $V_{DD}$ < 2.2 V | 0 | | $0.1V_{DD}$ | |
| | $V_{IL5}$ | P25, P27 | 2.2 V ≤ $V_{DD}$ ≤ 5.5 V | 0 | | $0.3V_{DD}$ | V |
| | | | 1.9 V ≤ $V_{DD}$ < 2.2 V | 0 | | $0.2V_{DD}$ | |
| Input voltage, high | $V_{IH1}$ | **Note 1** | 2.2 V ≤ $V_{DD}$ ≤ 5.5 V | $0.7V_{DD}$ | | $V_{DD}$ | V |
| | | | 1.9 V ≤ $V_{DD}$ < 2.2 V | $0.8V_{DD}$ | | $V_{DD}$ | |
| | $V_{IH2}$ | P00 to P05, P20, P22, P33, P34, P70, P72, $\overline{\text{RESET}}$ | 2.2 V ≤ $V_{DD}$ ≤ 5.5 V | $0.8V_{DD}$ | | $V_{DD}$ | V |
| | | | 1.9 V ≤ $V_{DD}$ < 2.2 V | $0.85V_{DD}$ | | $V_{DD}$ | |
| | $V_{IH3}$ | P10 to P17, P130, P131 | 2.2 V ≤ $V_{DD}$ ≤ 5.5 V | $0.7V_{DD}$ | | $V_{DD}$ | V |
| | | | 1.9 V ≤ $V_{DD}$ < 2.2 V | $0.8V_{DD}$ | | $V_{DD}$ | |
| | $V_{IH4}$ | X1, X2, XT1, XT2 | 2.2 V ≤ $V_{DD}$ ≤ 5.5 V | $0.8V_{DD}$ | | $V_{DD}$ | V |
| | | | 1.9 V ≤ $V_{DD}$ < 2.2 V | $0.85V_{DD}$ | | $V_{DD}$ | |
| | $V_{IH5}$ | P25, P27 | 2.2 V ≤ $V_{DD}$ ≤ 5.5 V | $0.7V_{DD}$ | | $V_{DD}$ | V |
| | | | 1.9 V ≤ $V_{DD}$ < 2.2 V | $0.8V_{DD}$ | | $V_{DD}$ | |
| Output voltage, low | $V_{OL1}$ | For pins other than P40 to P47, P50 to P57, $I_{OL}$ = 1.6 mA**Note 2** | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | | | 0.4 | V |
| | | P40 to P47, P50 to P57 $I_{OL}$ = 8 mA**Note 2** | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | | | 1.0 | V |
| | $V_{OL2}$ | $I_{OL}$ = 400 μA**Note 2** | 1.9 V ≤ $V_{DD}$ ≤ 5.5 V | | | 0.5 | V |
| Output voltage, high | $V_{OH1}$ | $I_{OH}$ = −1 mA**Note 2** | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | $V_{DD}$ − 1.0 | | | V |
| | | $I_{OH}$ = −100 μA**Note 2** | 1.9 V ≤ $V_{DD}$ ≤ 5.5 V | $V_{DD}$ − 0.5 | | | V |
| Input leakage current, low | $I_{LIL1}$ | $V_I$ = 0 V | Except X1, X2, XT1, XT2 | | | −3 | μA |
| | $I_{LIL2}$ | | X1, X2, XT1, XT2 | | | −20 | μA |
| Input leakage current, high | $I_{LIH1}$ | $V_I$ = $V_{DD}$ | Except X1, X2, XT1, XT2 | | | 3 | μA |
| | $I_{LIH2}$ | | X1, X2, XT1, XT2 | | | 20 | μA |

**Notes 1.** P21, P23, P24, P26, P30 to P32, P35 to P37, P40 to P47, P50 to P57, P60 to P67, P71, P120 to P127

**2.** Per pin

**DC Characteristics ($T_A$ = −40 to +85°C, $V_{DD}$ = $AV_{DD}$ = 1.9 to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0 V) (2/2)**

| Parameter | Symbol | Conditions | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Output leakage current, low | $I_{LOL1}$ | $V_O$ = 0 V | | | | −3 | μA |
| Output leakage current, high | $I_{LOH1}$ | $V_O$ = $V_{DD}$ | | | | 3 | μA |
| Supply voltage | $I_{DD1}$ | Operating mode | $f_{XX}$ = 12.5 MHz, $V_{DD}$ = 5.0 V ±10% | | 24 | 40 | mA |
| | | | $f_{XX}$ = 6 MHz, $V_{DD}$ = 3.0 V ±10% | | 8 | 17 | mA |
| | | | $f_{XX}$ = 2 MHz, $V_{DD}$ = 2.0 V ±5% | | 2 | 10 | mA |
| | $I_{DD2}$ | HALT mode | $f_{XX}$ = 12.5 MHz, $V_{DD}$ = 5.0 V ±10% | | 8 | 20 | mA |
| | | | $f_{XX}$ = 6 MHz, $V_{DD}$ = 3.0 V ±10% | | 2 | 10 | mA |
| | | | $f_{XX}$ = 2 MHz, $V_{DD}$ = 2.0 V ±5% | | 0.7 | 7 | mA |
| | $I_{DD3}$ | IDLE mode | $f_{XX}$ = 12.5 MHz, $V_{DD}$ = 5.0 V ±10% | | 1 | 3 | mA |
| | | | $f_{XX}$ = 6 MHz, $V_{DD}$ = 3.0 V ±10% | | 0.5 | 1.3 | mA |
| | | | $f_{XX}$ = 2 MHz, $V_{DD}$ = 2.0 V ±5% | | 0.3 | 0.9 | mA |
| | $I_{DD4}$ | Operating mode**Note** | $f_{XX}$ = 32 kHz, $V_{DD}$ = 5.0 V ±10% | | 130 | 500 | μA |
| | | | $f_{XX}$ = 32 kHz, $V_{DD}$ = 3.0 V ±10% | | 90 | 350 | μA |
| | | | $f_{XX}$ = 32 kHz, 2.0 V ≤ $V_{DD}$ < 2.7 V | | 80 | 300 | μA |
| | | | $f_{XX}$ = 32 kHz, 1.9 V ≤ $V_{DD}$ < 2.0 V | | 70 | 250 | μA |
| | $I_{DD5}$ | HALT mode**Note** | $f_{XX}$ = 32 kHz, $V_{DD}$ = 5.0 V ±10% | | 60 | 200 | μA |
| | | | $f_{XX}$ = 32 kHz, $V_{DD}$ = 3.0 V ±10% | | 20 | 160 | μA |
| | | | $f_{XX}$ = 32 kHz, 2.0 V ≤ $V_{DD}$ < 2.7 V | | 15 | 120 | μA |
| | | | $f_{XX}$ = 32 kHz, 1.9 V ≤ $V_{DD}$ < 2.0 V | | 10 | 100 | μA |
| | $I_{DD6}$ | IDLE mode**Note** | $f_{XX}$ = 32 kHz, $V_{DD}$ = 5.0 V ±10% | | 50 | 190 | μA |
| | | | $f_{XX}$ = 32 kHz, $V_{DD}$ = 3.0 V ±10% | | 15 | 150 | μA |
| | | | $f_{XX}$ = 32 kHz, 2.0 V ≤ $V_{DD}$ < 2.7 V | | 12 | 110 | μA |
| | | | $f_{XX}$ = 32 kHz, 1.9 V ≤ $V_{DD}$ < 2.0 V | | 7 | 90 | μA |
| Data retention voltage | $V_{DDDR}$ | HALT, IDLE modes | | 1.9 | | 5.5 | V |
| Data retention current | $I_{DDDR}$ | STOP mode | $V_{DD}$ = 2.0 V ±5% | | 2 | 10 | μA |
| | | | $V_{DD}$ = 5.0 V ±10% | | 10 | 50 | μA |
| Pull-up resistor | $R_L$ | $V_I$ = 0 V | | 10 | 30 | 100 | kΩ |

**Note** When the main system clock is stopped and the subsystem clock is operating.

**Remark** Unless otherwise specified, the characteristics of alternate-function pins are the same as those of port pins.

**AC Characteristics ($T_A$ = −40 to +85°C, $V_{DD}$ = $AV_{DD}$ = 1.9 to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0 V)**

**(1) Read/write operation (1/3)**

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Cycle time | $t_{CYK}$ | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | 80 | | | ns |
| | | 2.7 V ≤ $V_{DD}$ < 4.5 V | 160 | | | ns |
| | | 2.0 V ≤ $V_{DD}$ < 2.7 V | 320 | | | ns |
| | | 1.9 V ≤ $V_{DD}$ < 2.0 V | 500 | | | ns |
| Address setup time (to ASTB↓) | $t_{SAST}$ | $V_{DD}$ = 5.0 V ±10% | (0.5 + a) T − 20 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | (0.5 + a) T − 40 | | | ns |
| | | $V_{DD}$ = 2.0 V ±5% | (0.5 + a) T − 80 | | | ns |
| Address hold time (from ASTB↓) | $t_{HSTLA}$ | $V_{DD}$ = 5.0 V ±10% | 0.5T − 19 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0.5T − 24 | | | ns |
| | | $V_{DD}$ = 2.0 V ±5% | 0.5T − 34 | | | ns |
| ASTB high-level width | $t_{WSTH}$ | $V_{DD}$ = 5.0 V ±10% | (0.5 + a) T − 17 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | (0.5 + a) T − 40 | | | ns |
| | | $V_{DD}$ = 2.0 V ±5% | (0.5 + a) T − 110 | | | ns |
| Address hold time (from $\overline{RD}$↑) | $t_{HRA}$ | $V_{DD}$ = 5.0 V ±10% | 0.5T − 14 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0.5T − 14 | | | ns |
| | | $V_{DD}$ = 2.0 V ±5% | 0.5T − 14 | | | ns |
| Delay time from address to $\overline{RD}$↓ | $t_{DAR}$ | $V_{DD}$ = 5.0 V ±10% | (1 + a) T − 24 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | (1 + a) T − 35 | | | ns |
| | | $V_{DD}$ = 2.0 V ±5% | (1 + a) T − 80 | | | ns |
| Address float time (from $\overline{RD}$↓) | $t_{FAR}$ | $V_{DD}$ = 5.0 V ±10% | | | 0 | ns |
| | | $V_{DD}$ = 3.0 V ±10% | | | 0 | ns |
| | | $V_{DD}$ = 2.0 V ±5% | | | 0 | ns |
| Data input time from address | $t_{DAID}$ | $V_{DD}$ = 5.0 V ±10% | | | (2.5 + a + n) T − 37 | ns |
| | | $V_{DD}$ = 3.0 V ±10% | | | (2.5 + a + n) T − 52 | ns |
| | | $V_{DD}$ = 2.0 V ±5% | | | (2.5 + a + n) T − 120 | ns |
| Data input time from ASTB↓ | $t_{DSTID}$ | $V_{DD}$ = 5.0 V ±10% | | | (2 + n) T − 35 | ns |
| | | $V_{DD}$ = 3.0 V ±10% | | | (2 + n) T − 50 | ns |
| | | $V_{DD}$ = 2.0 V ±5% | | | (2 + n) T − 80 | ns |
| Data input time from $\overline{RD}$↓ | $t_{DRID}$ | $V_{DD}$ = 5.0 V ±10% | | | (1.5 + n) T − 40 | ns |
| | | $V_{DD}$ = 3.0 V ±10% | | | (1.5 + n) T − 50 | ns |
| | | $V_{DD}$ = 2.0 V ±5% | | | (1.5 + n) T − 90 | ns |

**Remark** T: $t_{CYK}$ = 1/fxx (fxx: Main system clock frequency)
a: 1 (during address wait), otherwise, 0
n: Number of wait states (n ≥ 0)

### (1) Read/write operation (2/3)

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Delay time from ASTB↓ to $\overline{RD}$↓ | $t_{DSTR}$ | $V_{DD}$ = 5.0 V ±10% | 0.5T – 9 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0.5T – 9 | | | ns |
| | | $V_{DD}$ = 2.0 V ±5% | 0.5T – 20 | | | ns |
| Data hold time (from $\overline{RD}$↑) | $t_{HRID}$ | $V_{DD}$ = 5.0 V ±10% | 0 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0 | | | ns |
| | | $V_{DD}$ = 2.0 V ±5% | 0 | | | ns |
| Address active time from $\overline{RD}$↑ | $t_{DRA}$ | $V_{DD}$ = 5.0 V ±10% | 0.5T – 2 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0.5T – 12 | | | ns |
| | | $V_{DD}$ = 2.0 V ±5% | 0.5T – 35 | | | ns |
| Delay time from $\overline{RD}$↑ to ASTB↑ | $t_{DRST}$ | $V_{DD}$ = 5.0 V ±10% | 0.5T – 9 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0.5T – 9 | | | ns |
| | | $V_{DD}$ = 2.0 V ±5% | 0.5T – 40 | | | ns |
| $\overline{RD}$ low-level width | $t_{WRL}$ | $V_{DD}$ = 5.0 V ±10% | (1.5 + n) T – 25 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | (1.5 + n) T – 30 | | | ns |
| | | $V_{DD}$ = 2.0 V ±5% | (1.5 + n) T – 25 | | | ns |
| Address active time (from $\overline{WR}$↑) | $t_{DWA}$ | $V_{DD}$ = 5.0 V ±10% | 0.5T – 2 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0.5T – 12 | | | ns |
| | | $V_{DD}$ = 2.0 V ±5% | 0.5T – 35 | | | ns |
| Delay time from address to $\overline{WR}$↓ | $t_{DAW}$ | $V_{DD}$ = 5.0 V ±10% | (1 + a) T – 24 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | (1 + a) T – 34 | | | ns |
| | | $V_{DD}$ = 2.0 V ±5% | (1 + a) T – 70 | | | ns |
| Address hold time from $\overline{WR}$↑ | $t_{HWA}$ | $V_{DD}$ = 5.0 V ±10% | 0.5T – 14 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0.5T – 14 | | | ns |
| | | $V_{DD}$ = 2.0 V ±5% | 0.5T – 14 | | | ns |
| Delay time from ASTB↓ to data output | $t_{DSTOD}$ | $V_{DD}$ = 5.0 V ±10% | | | 0.5T + 15 | ns |
| | | $V_{DD}$ = 3.0 V ±10% | | | 0.5T + 30 | ns |
| | | $V_{DD}$ = 2.0 V ±5% | | | 0.5T + 240 | ns |
| Delay time from $\overline{WR}$↓ to data output | $t_{DWOD}$ | $V_{DD}$ = 5.0 V ±10% | | | 0.5T – 30 | ns |
| | | $V_{DD}$ = 3.0 V ±10% | | | 0.5T – 30 | ns |
| | | $V_{DD}$ = 2.0 V ±5% | | | 0.5T – 30 | ns |
| Delay time from ASTB↓ to $\overline{WR}$↓ | $t_{DSTW}$ | $V_{DD}$ = 5.0 V ±10% | 0.5T – 9 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0.5T – 9 | | | ns |
| | | $V_{DD}$ = 2.0 V ±5% | 0.5T – 20 | | | ns |
| Data setup time (to $\overline{WR}$↑) | $t_{SODWR}$ | $V_{DD}$ = 5.0 V ±10% | (1.5 + n) T – 20 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | (1.5 + n) T – 25 | | | ns |
| | | $V_{DD}$ = 2.0 V ±5% | (1.5 + n) T – 70 | | | ns |

**Remark** T: $t_{CYK}$ = 1/fxx (fxx: Main system clock frequency)
a: 1 (during address wait), otherwise, 0
n: Number of wait states (n ≥ 0)

### (1) Read/write operation (3/3)

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Data hold time (from $\overline{WR}\uparrow$) | $t_{HWOD}$ | $V_{DD}$ = 5.0 V ±10% | 0.5T – 14 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0.5T – 14 | | | ns |
| | | $V_{DD}$ = 2.0 V ±5% | 0.5T – 50 | | | ns |
| Delay time from $\overline{WR}\uparrow$ to ASTB$\uparrow$ | $t_{DWST}$ | $V_{DD}$ = 5.0 V ±10% | 0.5T – 9 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0.5T – 9 | | | ns |
| | | $V_{DD}$ = 2.0 V ±5% | 0.5T – 30 | | | ns |
| $\overline{WR}$ low-level width | $t_{WWL}$ | $V_{DD}$ = 5.0 V ±10% | (1.5 + n) T – 25 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | (1.5 + n) T – 30 | | | ns |
| | | $V_{DD}$ = 2.0 V ±5% | (1.5 + n) T – 30 | | | ns |
| Delay time from address to EXA$\downarrow$ | $t_{ADEXD}$ | $V_{DD}$ = 5.0 V ±10% | 0 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0 | | | ns |
| | | $V_{DD}$ = 2.0 V ±5% | 0 | | | ns |
| Delay time from EXA$\downarrow$ to ASTB$\downarrow$ | $t_{EXTAH}$ | $V_{DD}$ = 5.0 V ±10% | 0.5T – 20 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0.5T – 30 | | | ns |
| | | $V_{DD}$ = 2.0 V ±5% | 0.5T – 40 | | | ns |
| Delay time from $\overline{RD}\uparrow$ to EXA$\uparrow$ | $t_{EXRDS}$ | $V_{DD}$ = 5.0 V ±10% | 0 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0 | | | ns |
| | | $V_{DD}$ = 2.0 V ±5% | 0 | | | ns |
| Delay time from $\overline{WR}\uparrow$ to EXA$\uparrow$ | $t_{EXWDS}$ | $V_{DD}$ = 5.0 V ±10% | T | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | T | | | ns |
| | | $V_{DD}$ = 2.0 V ±5% | T | | | ns |
| Delay time from EXA$\uparrow$ to ASTB$\uparrow$ | $t_{EXADR}$ | $V_{DD}$ = 5.0 V ±10% | 0.5T | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0.5T | | | ns |
| | | $V_{DD}$ = 2.0 V ±5% | 0.5T | | | ns |

**Remark** T: $t_{CYK}$ = 1/fxx (fxx: Main system clock frequency)
n: Number of wait states ($n \geq 0$)

### (2) External wait timing (1/2)

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Input time from address to $\overline{WAIT}\downarrow$ | $t_{DAWT}$ | $V_{DD}$ = 5.0 V ±10% | | | (2 + a) T – 40 | ns |
| | | $V_{DD}$ = 3.0 V ±10% | | | (2 + a) T – 60 | ns |
| | | $V_{DD}$ = 2.0 V ±5% | | | (2 + a) T – 300 | ns |
| Input time from ASTB↓ to $\overline{WAIT}\downarrow$ | $t_{DSTWT}$ | $V_{DD}$ = 5.0 V ±10% | | | 1.5T – 40 | ns |
| | | $V_{DD}$ = 3.0 V ±10% | | | 1.5T – 60 | ns |
| | | $V_{DD}$ = 2.0 V ±5% | | | 1.5T – 260 | ns |
| Hold time from ASTB↓ to $\overline{WAIT}$ | $t_{HSTWT}$ | $V_{DD}$ = 5.0 V ±10% | (0.5 + n) T + 5 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | (0.5 + n) T + 10 | | | ns |
| | | $V_{DD}$ = 2.0 V ±5% | (0.5 + n) T + 30 | | | ns |
| Delay time from ASTB↓ to $\overline{WAIT}\uparrow$ | $t_{DSTWTH}$ | $V_{DD}$ = 5.0 V ±10% | | | (1.5 + n) T – 40 | ns |
| | | $V_{DD}$ = 3.0 V ±10% | | | (1.5 + n) T – 60 | ns |
| | | $V_{DD}$ = 2.0 V ±5% | | | (1.5 + n) T – 90 | ns |
| Input time from $\overline{RD}\downarrow$ to $\overline{WAIT}\downarrow$ | $t_{DRWTL}$ | $V_{DD}$ = 5.0 V ±10% | | | T – 40 | ns |
| | | $V_{DD}$ = 3.0 V ±10% | | | T – 60 | ns |
| | | $V_{DD}$ = 2.0 V ±5% | | | T – 70 | ns |
| Hold time from $\overline{RD}\downarrow$ to $\overline{WAIT}$ | $t_{HRWT}$ | $V_{DD}$ = 5.0 V ±10% | nT + 5 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | nT + 10 | | | ns |
| | | $V_{DD}$ = 2.0 V ±5% | nT + 30 | | | ns |
| Delay time from $\overline{RD}\downarrow$ to $\overline{WAIT}\uparrow$ | $t_{DRWTH}$ | $V_{DD}$ = 5.0 V ±10% | | | (1 + n) T – 40 | ns |
| | | $V_{DD}$ = 3.0 V ±10% | | | (1 + n) T – 60 | ns |
| | | $V_{DD}$ = 2.0 V ±5% | | | (1 + n) T – 90 | ns |
| Data input time from $\overline{WAIT}\uparrow$ | $t_{DWTID}$ | $V_{DD}$ = 5.0 V ±10% | | | 0.5T – 5 | ns |
| | | $V_{DD}$ = 3.0 V ±10% | | | 0.5T – 10 | ns |
| | | $V_{DD}$ = 2.0 V ±5% | | | 0.5T – 30 | ns |
| Delay time from $\overline{WAIT}\uparrow$ to $\overline{RD}\uparrow$ | $t_{DWTR}$ | $V_{DD}$ = 5.0 V ±10% | 0.5T | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0.5T | | | ns |
| | | $V_{DD}$ = 2.0 V ±5% | 0.5T + 5 | | | ns |
| Delay time from $\overline{WAIT}\uparrow$ to $\overline{WR}\uparrow$ | $t_{DWTW}$ | $V_{DD}$ = 5.0 V ±10% | 0.5T | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | 0.5T | | | ns |
| | | $V_{DD}$ = 2.0 V ±5% | 0.5T + 5 | | | ns |
| Input time from $\overline{WR}\downarrow$ to $\overline{WAIT}\downarrow$ | $t_{DWWTL}$ | $V_{DD}$ = 5.0 V ±10% | | | T – 40 | ns |
| | | $V_{DD}$ = 3.0 V ±10% | | | T – 60 | ns |
| | | $V_{DD}$ = 2.0 V ±5% | | | T – 90 | ns |

**Remark** T: $t_{CYK}$ = 1/$f_{XX}$ ($f_{XX}$: Main system clock frequency)
a: 1 (during address wait), otherwise, 0
n: Number of wait states (n ≥ 0)

### (2) External wait timing (2/2)

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Hold time from $\overline{WR}\downarrow$ to $\overline{WAIT}$ | $t_{HWWT}$ | $V_{DD}$ = 5.0 V ±10% | nT + 5 | | | ns |
| | | $V_{DD}$ = 3.0 V ±10% | nT + 10 | | | ns |
| | | $V_{DD}$ = 2.0 V ±5% | nT + 30 | | | ns |
| Delay time from $\overline{WR}\downarrow$ to $\overline{WAIT}\uparrow$ | $t_{DWWTH}$ | $V_{DD}$ = 5.0 V ±10% | | | (1 + n) T − 40 | ns |
| | | $V_{DD}$ = 3.0 V ±10% | | | (1 + n) T − 60 | ns |
| | | $V_{DD}$ = 2.0 V ±5% | | | (1 + n) T − 90 | ns |

**Remark** T: $t_{CYK}$ = 1/fxx (fxx: Main system clock frequency)
n: Number of wait states ($n \geq 0$)

### (3) Serial Operation ($T_A$ = −40 to +85°C, $V_{DD}$ = $AV_{DD}$ = 1.9 to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0 V) (1/2)

#### (a) 3-wire serial I/O mode ($\overline{SCK}$: Internal clock output)

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{SCK}$ cycle time | $t_{KCY1}$ | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | 640 | | | ns |
| | | 2.7 V ≤ $V_{DD}$ < 4.5 V | 1,280 | | | ns |
| | | 2.0 V ≤ $V_{DD}$ < 2.7 V | 2,560 | | | ns |
| | | 1.9 V ≤ $V_{DD}$ < 2.0 V | 4,000 | | | ns |
| $\overline{SCK}$ high-/low-level width | $t_{KH1}$, $t_{KL1}$ | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | 270 | | | ns |
| | | 2.7 V ≤ $V_{DD}$ < 4.5 V | 590 | | | ns |
| | | 2.0 V ≤ $V_{DD}$ < 2.7 V | 1,180 | | | ns |
| | | 1.8 V ≤ $V_{DD}$ < 2.0 V | 1,900 | | | ns |
| SI setup time (to $\overline{SCK}$↑) | $t_{SIK1}$ | 2.7 V ≤ $V_{DD}$ ≤ 5.5 V | 10 | | | ns |
| | | 1.9 V ≤ $V_{DD}$ < 2.7 V | 30 | | | ns |
| SI hold time (from $\overline{SCK}$↑) | $t_{HIK1}$ | | 40 | | | ns |
| SO output delay time (from SCK↓) | $t_{DSO1}$ | | | | 30 | ns |
| SO output hold time (from $\overline{SCK}$↑) | $t_{HSO1}$ | | $0.5t_{KCY1}$ – 50 | | | ns |

#### (b) 3-wire serial I/O mode ($\overline{SCK}$: External clock input)

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{SCK}$ cycle time | $t_{KCY2}$ | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | 640 | | | ns |
| | | 2.7 V ≤ $V_{DD}$ < 4.5 V | 1,280 | | | ns |
| | | 2.0 V ≤ $V_{DD}$ < 2.7 V | 2,560 | | | ns |
| | | 1.9 V ≤ $V_{DD}$ < 2.0 V | 4,000 | | | ns |
| $\overline{SCK}$ high-/low-level width | $t_{KH2}$, $t_{KL2}$ | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | 320 | | | ns |
| | | 2.7 V ≤ $V_{DD}$ < 4.5 V | 640 | | | ns |
| | | 2.0 V ≤ $V_{DD}$ < 2.7 V | 1,280 | | | ns |
| | | 1.9 V ≤ $V_{DD}$ < 2.0 V | 2,000 | | | ns |
| SI setup time (to $\overline{SCK}$↑) | $t_{SIK2}$ | 2.7 V ≤ $V_{DD}$ ≤ 5.5 V | 10 | | | ns |
| | | 1.9 V ≤ $V_{DD}$ < 2.7 V | 30 | | | ns |
| SI hold time (from $\overline{SCK}$↑) | $t_{HIK2}$ | | 40 | | | ns |
| SO output delay time (from $\overline{SCK}$↓) | $t_{DSO2}$ | | | | 30 | ns |
| SO output hold time (from $\overline{SCK}$↑) | $t_{HSO2}$ | | $0.5t_{KCY2}$ – 50 | | | ns |

#### (c) UART mode

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| ASCK cycle time | $t_{KCY3}$ | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | 417 | | | ns |
| | | 2.7 V ≤ $V_{DD}$ < 4.5 V | 833 | | | ns |
| | | 1.9 V ≤ $V_{DD}$ < 2.7 V | 1,667 | | | ns |
| ASCK high-/low-level width | $t_{KH3}$, $t_{KL3}$ | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | 208 | | | ns |
| | | 2.7 V ≤ $V_{DD}$ < 4.5 V | 416 | | | ns |
| | | 1.9 V ≤ $V_{DD}$ < 2.7 V | 833 | | | ns |

### (3) Serial Operation ($T_A$ = −40 to +85°C, $V_{DD}$ = $AV_{DD}$ = 1.9 to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0 V) (2/2)

#### (d) $I^2C$ bus mode ($\mu$PD78F4225Y only)

| Parameter | | Symbol | Standard Mode | | High-Speed Mode | | Unit |
|---|---|---|---|---|---|---|---|
| | | | MIN. | MAX. | MIN. | MAX. | |
| SCL0 clock frequency | | $f_{CLK}$ | 0 | 100 | 0 | 400 | kHz |
| Bus free time (between stop and start conditions) | | $t_{BUF}$ | 4.7 | – | 1.3 | – | μs |
| Hold time[Note1] | | $t_{HD:STA}$ | 4.0 | – | 0.6 | – | μs |
| Low-level width of SCL0 clock | | $t_{LOW}$ | 4.7 | – | 1.3 | – | μs |
| High-level width of SCL0 clock | | $t_{HIGH}$ | 4.0 | – | 0.6 | – | μs |
| Setup time of start/restart conditions | | $t_{SU:STA}$ | 4.7 | – | 0.6 | – | μs |
| Data hold time | When using CBUS-compatible master | $t_{HD:DAT}$ | 5.0 | – | – | – | μs |
| | When using $I^2C$ bus | | 0[Note 2] | – | 0[Note 2] | 0.9[Note 3] | μs |
| Data setup time | | $t_{SU:DAT}$ | 250 | – | 100[Note 4] | – | ns |
| Rise time of SDA0 and SCL0 signals | | $t_R$ | – | 1,000 | 20 + 0.1Cb[Note 5] | 300 | ns |
| Fall time of SDA0 and SCL0 signals | | $t_F$ | – | 300 | 20 + 0.1Cb[Note 5] | 300 | ns |
| Setup time of stop condition | | $t_{SU:STO}$ | 4.0 | – | 0.6 | – | μs |
| Pulse width of spike restricted by input filter | | $t_{SP}$ | – | – | 0 | 50 | ns |
| Load capacitance of each bus line | | Cb | – | 400 | – | 400 | pF |

**Notes 1.** For the start condition, the first clock pulse is generated after the hold time.

**2.** To fill the undefined area of the SCL0 falling edge, it is necessary for the device to provide an internal SDA0 signal (on $V_{IHmin.}$) with at least 300 ns of hold time.

**3.** If the device does not extend the SCL0 signal low-level hold time ($t_{LOW}$), only the maximum data hold time $t_{HD:DAT}$ needs to be satisfied.

**4.** The high-speed mode $I^2C$ bus can be used in a standard mode $I^2C$ bus system. In this case, the conditions described below must be satisfied.

- If the device does not extend the SCL0 signal low-level hold time
  $t_{SU:DAT} \geq 250$ ns
- If the device extends the SCL0 signal low-level hold time
  Be sure to transmit the data bit to the SDA0 line before the SCL0 line is released ($t_{Rmax.} + t_{SU:DAT}$ = 1,000 + 250 = 1,250 ns by standard mode $I^2C$ bus specification)

**5.** Cb: Total capacitance per bus line (unit: pF)

**(4) Clock Output Operation ($T_A$ = −40 to +85°C, $V_{DD}$ = $AV_{DD}$ = 1.9 to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0 V)**

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| PCL cycle time | $t_{CYCL}$ | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V, nT | 80 | | 31,250 | ns |
| PCL high-/low-level width | $t_{CLL}$, $t_{CLH}$ | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V, 0.5T − 10 | 30 | | 15,615 | ns |
| PCL rise/fall time | $t_{CLR}$, $t_{CLF}$ | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | | | 5 | ns |
| | | 2.7 V ≤ $V_{DD}$ < 4.5 V | | | 10 | ns |
| | | 1.9 V ≤ $V_{DD}$ < 2.7 V | | | 20 | ns |

**Remark** T: $t_{CYK}$ = 1/$f_{XX}$ ($f_{XX}$: Main system clock frequency)

n: Division ratio set by software in the CPU

- When using the main system clock: n = 1, 2, 4, 8, 16, 32, 64, 128
- When using the subsystem clock: n = 1

**(5) Other Operations ($T_A$ = −40 to +85°C, $V_{DD}$ = $AV_{DD}$ = 1.9 to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0 V)**

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| NMI high-/low-level width | $t_{WNIL}$, $t_{WNIH}$ | | 10 | | | μs |
| Interrupt input high-/low-level width | $t_{WITL}$, $t_{WITH}$ | INTP0 to INTP5 | 100 | | | ns |
| $\overline{\text{RESET}}$ high-/low-level width | $t_{WRSL}$, $t_{WRSH}$ | | 10 | | | μs |

**A/D Converter Characteristics ($T_A$ = −40 to +85°C, $V_{DD}$ = $AV_{DD}$ = 1.9 to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0 V)**

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Resolution | | | 8 | 8 | 8 | bit |
| Overall error[Notes 1, 2] | | 6.25 MHz < $f_{XX}$ ≤ 12.5 MHz, 4.5 V ≤ $V_{DD}$ ≤ 5.5 V, $AV_{DD}$ = $V_{DD}$ | | | ±1.2 | %FSR |
| | | 3.125 MHz < $f_{XX}$ ≤ 6.25 MHz, 2.7 V ≤ $V_{DD}$ ≤ 5.5 V, $AV_{DD}$ = $V_{DD}$ | | | ±1.2 | %FSR |
| | | 2 MHz < $f_{XX}$ ≤ 3.125 MHz, 2.0 V ≤ $V_{DD}$ ≤ 5.5 V, $AV_{DD}$ = $V_{DD}$ | | | ±1.6 | %FSR |
| | | $f_{XX}$ = 2 MHz, 1.9 V ≤ $V_{DD}$ ≤ 5.5 V $AV_{DD}$ = $V_{DD}$ | | | ±1.6 | %FSR |
| Conversion time | $t_{CONV}$ | | 14 | | 144 | μs |
| Sampling time | $t_{SAMP}$ | | 24/$f_{XX}$ | | | μs |
| Analog input voltage | $V_{IAN}$ | | $AV_{SS}$ | | $AV_{DD}$ | V |
| Reference voltage | $AV_{DD}$ | | $V_{DD}$ | $V_{DD}$ | $V_{DD}$ | V |
| Resistance between $AV_{DD}$ and $AV_{SS}$ | $R_{AVREF0}$ | A/D conversion is not performed | | 40 | | kΩ |

**Notes 1.** Excludes quantization error (±0.2%FSR).

**2.** This value is indicated as a ratio to the full-scale value (%FSR).

**Remark** $f_{XX}$: Main system clock frequency

**D/A Converter Characteristics ($T_A$ = −40 to +85°C, $V_{DD}$ = $AV_{DD}$ = 1.9 to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0 V)**

| Parameter | Symbol | Conditions | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Resolution | | | | 8 | 8 | 8 | bit |
| Overall error[Notes 1, 2] | | 2.0 V ≤ $V_{DD}$ ≤ 5.5 V, $AV_{DD}$ = $V_{DD}$ R = 10 MΩ, 2.0 V ≤ $AV_{REF1}$ ≤ $AV_{DD}$ | | | | ±0.6 | %FSR |
| | | 1.9 V ≤ $V_{DD}$ ≤ 2.0 V, $AV_{DD}$ = $V_{DD}$ R = 10 MΩ, 1.9 V ≤ $AV_{REF1}$ ≤ $AV_{DD}$ | | | | ±1.2 | %FSR |
| Settling time | | Load conditions: C = 30 pF | 4.5 V ≤ $AV_{REF1}$ ≤ 5.5 V | | | 10 | μs |
| | | | 2.7 V ≤ $AV_{REF1}$ < 4.5 V | | | 15 | μs |
| | | | 1.9 V ≤ $AV_{REF1}$ < 2.7 V | | | 20 | μs |
| Output resistance | $R_O$ | DACS0, 1 = 55H | | | 8 | | kΩ |
| Reference voltage | $AV_{REF1}$ | | | 1.9 | | $V_{DD}$ | V |
| $AV_{REF1}$ current | $AI_{REF1}$ | For only 1 channel | | | | 2.5 | mA |

**Notes 1.** Excludes quantization error (±0.2%FSR).

**2.** This value is indicated as a ratio to the full-scale value (%FSR).

**Data Retention Characteristics ($T_A$ = −40 to +85°C, $V_{DD}$ = $AV_{DD}$ = 1.9 to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0 V)**

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Data retention voltage | $V_{DDDR}$ | STOP mode | 1.9 | | 5.5 | V |
| Data retention current | $I_{DDDR}$ | $V_{DDDR}$ = 5.0 V ±10% | | 10 | 50 | μA |
| | | $V_{DDDR}$ = 2.0 V ±5% | | 2 | 10 | μA |
| $V_{DD}$ rise time | $t_{RVD}$ | | 200 | | | μs |
| $V_{DD}$ fall time | $t_{FVD}$ | | 200 | | | μs |
| $V_{DD}$ hold time (from STOP mode setting) | $t_{HVD}$ | | 0 | | | ms |
| STOP release signal input time | $t_{DREL}$ | | 0 | | | ms |
| Oscillation stabilization wait time | $t_{WAIT}$ | Crystal resonator | 30 | | | ms |
| | | Ceramic resonator | 5 | | | ms |
| Input voltage, low | $V_{IL}$ | $\overline{\text{RESET}}$, P00/INTP0 to P05/INTP5 | 0 | | $0.1V_{DDDR}$ | V |
| Input voltage, high | $V_{IH}$ | | $0.9V_{DDDR}$ | | $V_{DDDR}$ | V |

**Flash Memory Programming Characteristics**

**($T_A$ = 10 to 40°C, $V_{DD}$ = $AV_{DD}$ = 1.9 to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0 V, $V_{PP}$ = 9.7 to 10.3 V) (1/2)**

**(1) Basic characteristics**

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Operating frequency | $f_{XX}$ | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | 2 | | 12.5 | MHz |
| | | 2.7 V ≤ $V_{DD}$ < 4.5 V | 2 | | 6.25 | MHz |
| | | 2.0 V ≤ $V_{DD}$ < 2.7 V | 2 | | 3.125 | MHz |
| | | 1.9 V ≤ $V_{DD}$ < 2.0 V | 2 | 2 | 2 | MHz |
| Oscillation frequency[Note 1] | $f_X$ | 4.5 V ≤ $V_{DD}$ ≤ 5.5 V | 4 | | 25 | MHz |
| | | 2.7 V ≤ $V_{DD}$ < 4.5 V | 4 | | 12.5 | MHz |
| | | 2.0 V ≤ $V_{DD}$ < 2.7 V | 4 | | 6.25 | MHz |
| | | 1.9 V ≤ $V_{DD}$ < 2.0 V | 4 | 4 | 4 | MHz |
| Supply voltage | $V_{DD}$ | | 1.9 | | 5.5 | V |
| | $V_{PPL}$ | When detecting $V_{PP}$ low level | 0 | | $0.2V_{DD}$ | V |
| | $V_{PP}$ | When detecting $V_{PP}$ high level | $0.9V_{DD}$ | | $1.1V_{DD}$ | V |
| | $V_{PPH}$ | When detecting $V_{PP}$ high voltage | 9.7 | 10 | 10.3 | V |
| Write time | $C_{WRT}$ | | 20[Note 2] | | | times |
| Operating temperature | $T_A$ | | –40 | | 85 | °C |
| Storage temperature | $T_{stg}$ | | –65 | | 125 | °C |
| Programming temperature | $T_{PRG}$ | | 10 | | 40 | °C |

**Notes 1.** When rewriting without using handshake mode

**2.** Operation cannot be guaranteed when the number of rewrites exceeds 20.
In the case of K standard products, operation cannot be guaranteed when the number of rewrites exceeds 5.

**Cautions 1. If writing is not successful in the initial write operation, execute the program command again, and then execute the verify command to confirm that the write operation has been completed normally (K standard).**

**2. Handshake mode is supported by products other than those with the K standard.**

**Remarks 1.** The fifth letter from the left in the lot number indicates the standard of the product.

**2.** After executing the program command, execute the verify command to confirm that the write operation has been completed normally.

**3.** Handshake mode is the CSI write mode that uses P24. Handshake mode can be used with the PG-FR3 and FL-PR3.

**4.** The I standard only applies to ES (engineering sample) products. Because these products are engineering samples, their operation cannot be guaranteed.

**Flash Memory Programming Characteristics**

**($T_A$ = 10 to 40°C, $V_{DD}$ = $AV_{DD}$ = 1.9 to 5.5 V, $V_{SS}$ = $AV_{SS}$ = 0 V, $V_{PP}$ = 9.7 to 10.3 V) (2/2)**

**(2) Write erase characteristics**

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $V_{PP}$ supply voltage | $V_{PP2}$ | During flash memory programming | 9.7 | 10.0 | 10.3 | V |
| $V_{DD}$ supply current | $I_{DD}$ | When $V_{PP}$ = $V_{PP2}$, $f_{XX}$ = 12.5 MHz | | | 40 | mA |
| $V_{PP}$ supply current | $I_{PP}$ | When $V_{PP}$ = $V_{PP2}$ | | | 100 | mA |
| Step erase time | $T_{er}$ | **Note 1** | | 0.2 | | s |
| Overall erase time per area | $T_{era}$ | When step erase time = 0.2 s**Note 2** | | | 20 | s/area |
| Write-back time | $T_{wb}$ | **Note 3** | | 50 | | ms |
| Number of write-backs per write-back command | $C_{wb}$ | When write-back time = 50 ms**Note 4** | | | 60 | times/ write-back command |
| Number of erase/ write-backs | $C_{erwb}$ | | | | 16 | times |
| Step write time | $T_{wr}$ | **Note 5** | | 50 | | μs |
| Overall write time per word | $T_{wrw}$ | When step write time = 50 μs (1 word = 1 byte)**Note 6** | 50 | | 500 | μs/ word |
| Number of rewrites per area | $C_{erwr}$ | 1 erase + 1 write after erase = 1 rewrite**Note 7** | | 20 | | times/ area |

**Notes 1.** The recommended setting value for the step erase time is 0.2 s.

**2.** The prewrite time before erasure and the erase verify time (write-back time) is not included.

**3.** The recommended setting value for the write-back time is 50 ms.

**4.** Write-back is executed once by the issuance of the write-back command. Therefore, the retry times must be the maximum value minus the number of commands issued.

**5.** The recommended step write time setting value is 50 μs.

**6.** The actual write time per word is 100 μs longer. The internal verify time during or after a write is not included.

**7.** When a product is first written after shipment, "erase → write" and "write only" are both taken as one rewrite.

Example: P: Write, E: Erase

Shipped product → P → E → P→ E → P: 3 rewrites

Shipped product → E → P → E→ P → E → P: 3 rewrites

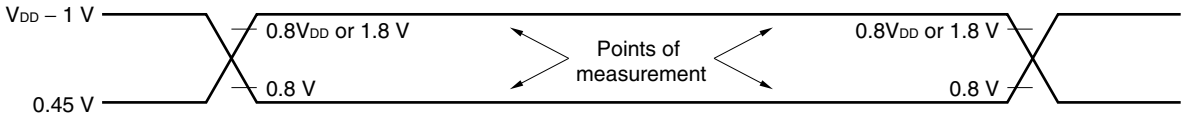
**Remarks 1.** The range of the operating clock during flash memory programming is the same as the range during normal operation.

**2.** When using the PG-FP3, the time parameters that need to be downloaded from the parameter files for write/erase are automatically set. Unless otherwise directed, do not change the set values.

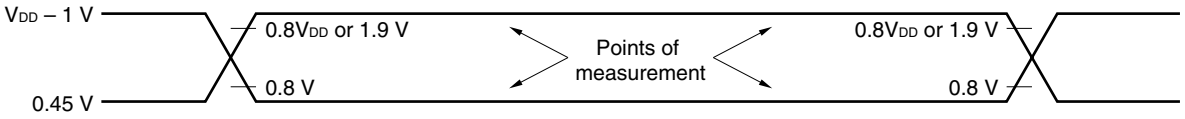
## 29.3 Timing Charts

### AC Timing Measurement Points

**(1) μPD784224, 784225, 784224Y, 784225Y**

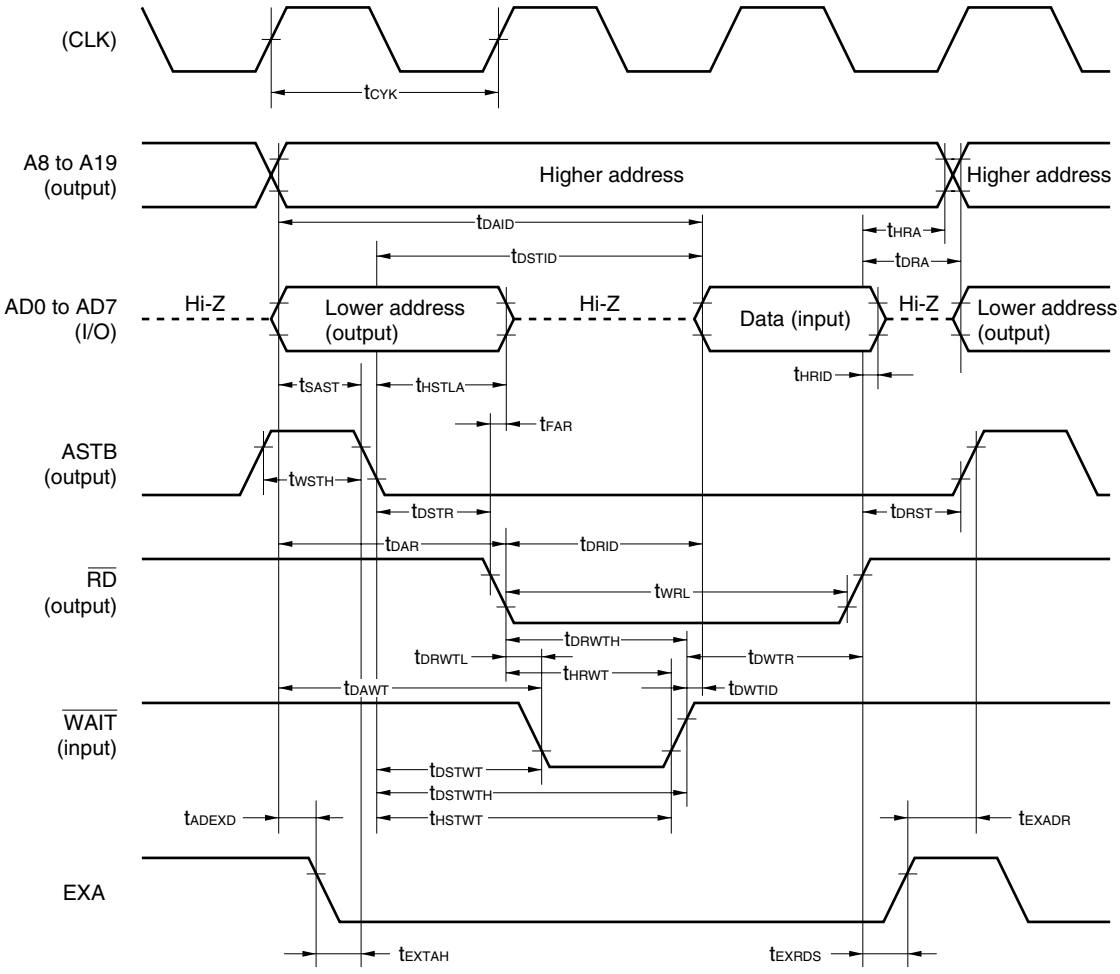



**(2) μPD78F4225, 78F4225Y**

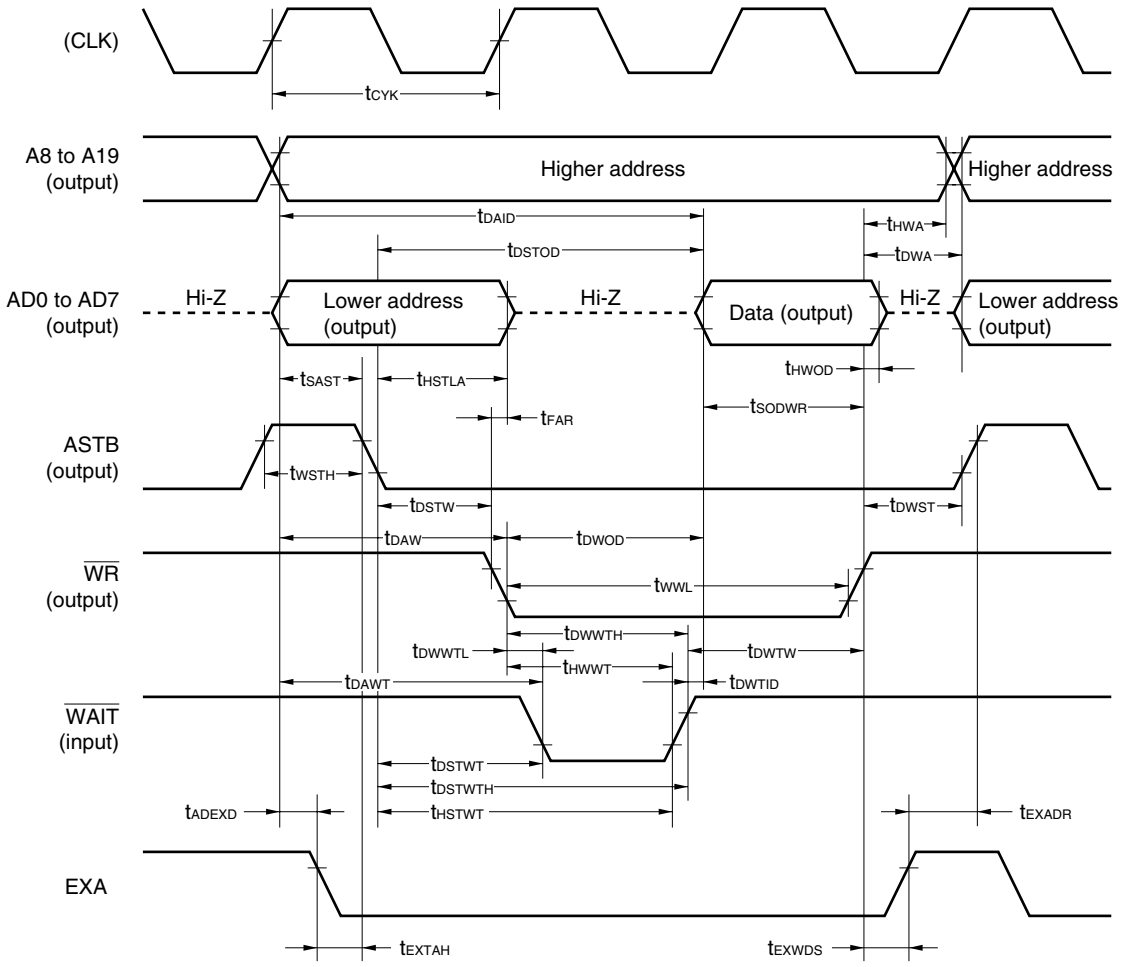


## Timing Waveforms

### (1) Read operation

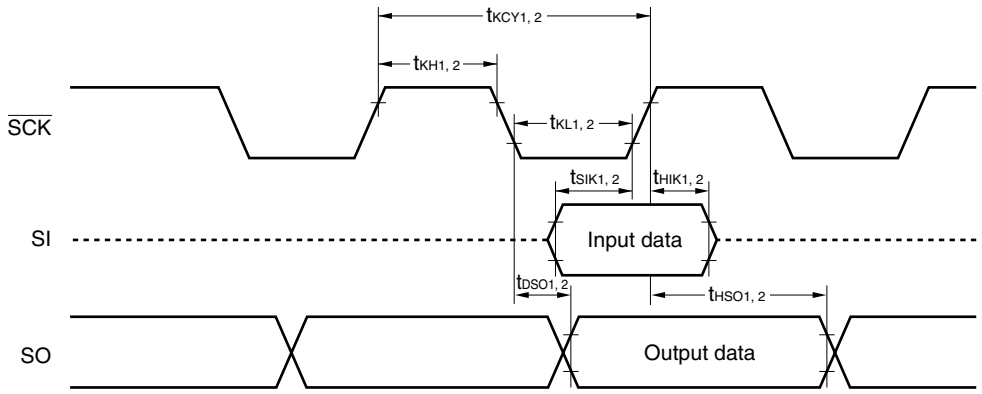


### (2) Write operation

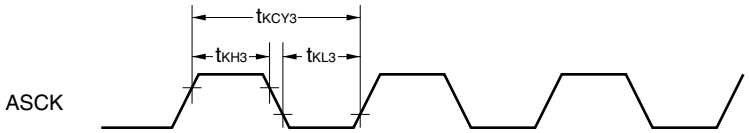


## Serial Operation

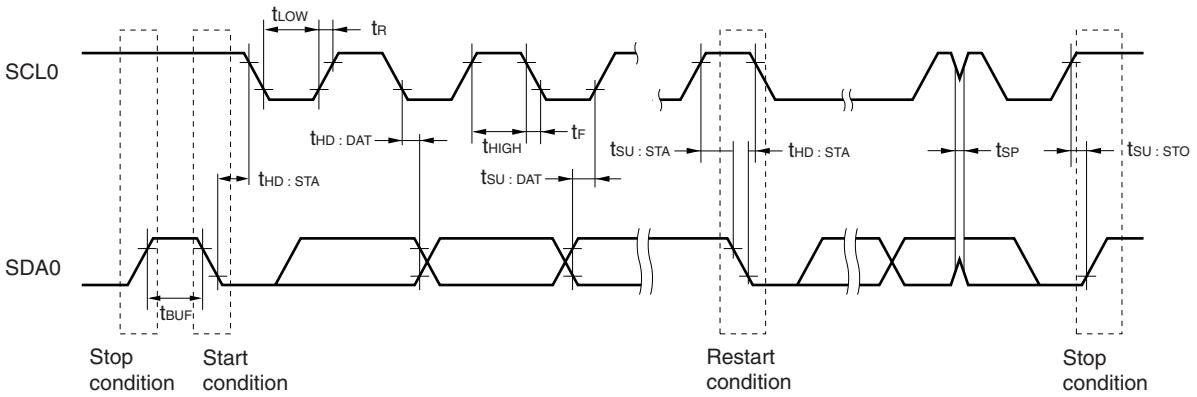
### (1) 3-wire serial I/O mode



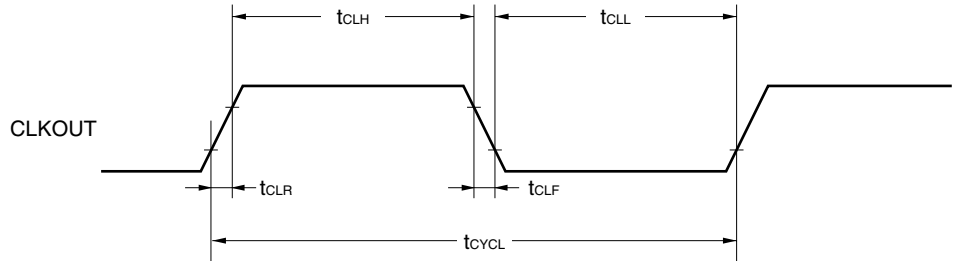

### (2) UART mode



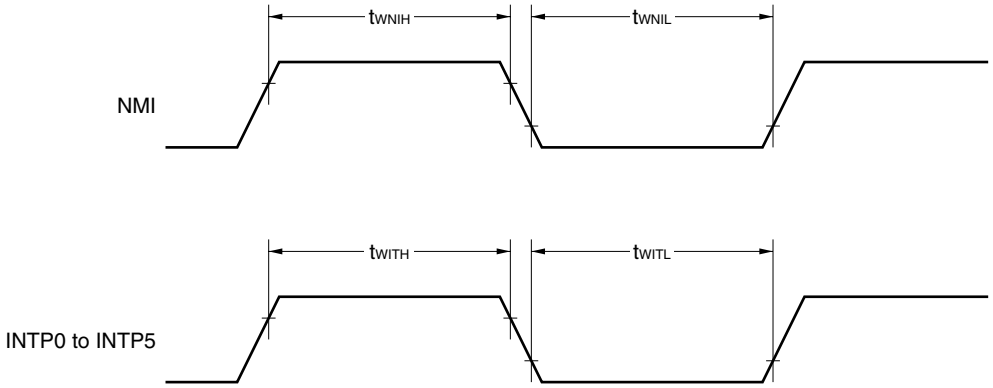

### (3) I²C bus mode (μPD784224Y, 784225Y, and 78F4255Y only)



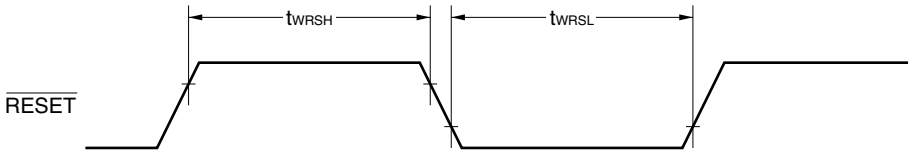
### Clock Output Timing



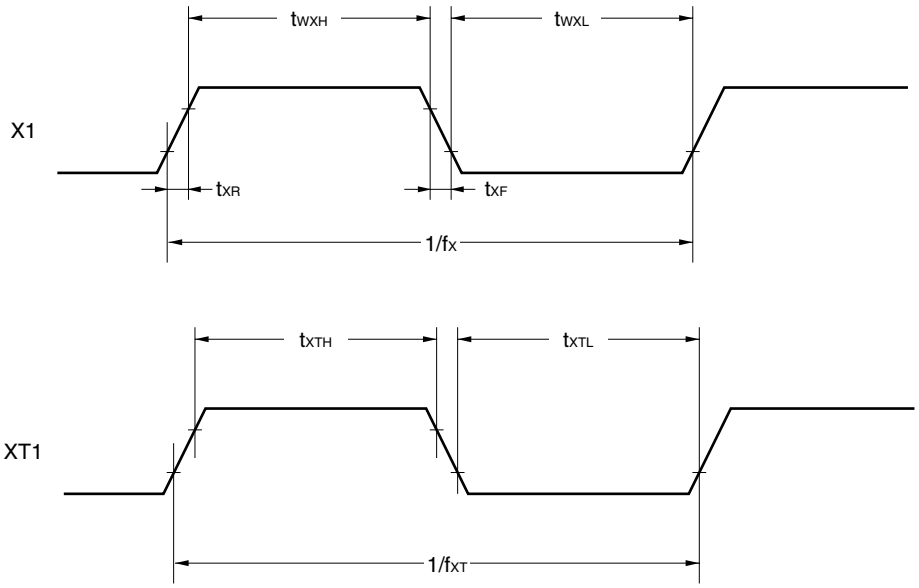

### Interrupt Input Timing



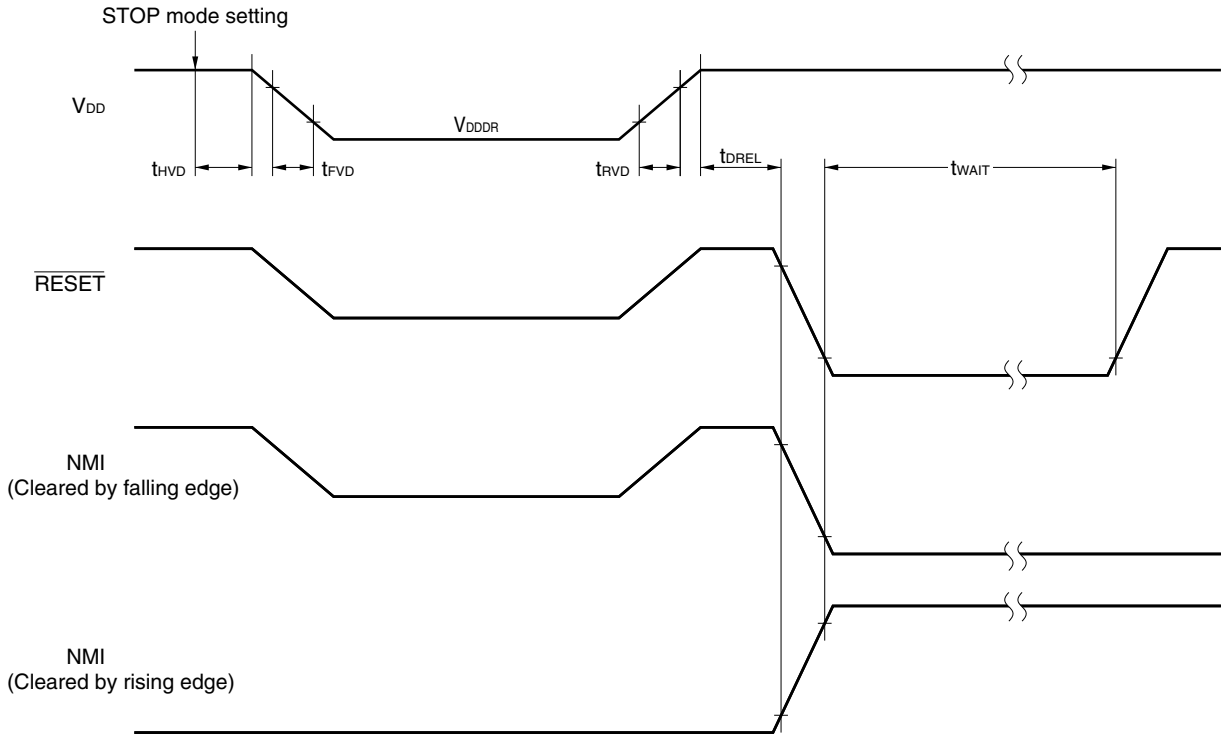

### Reset Input Timing



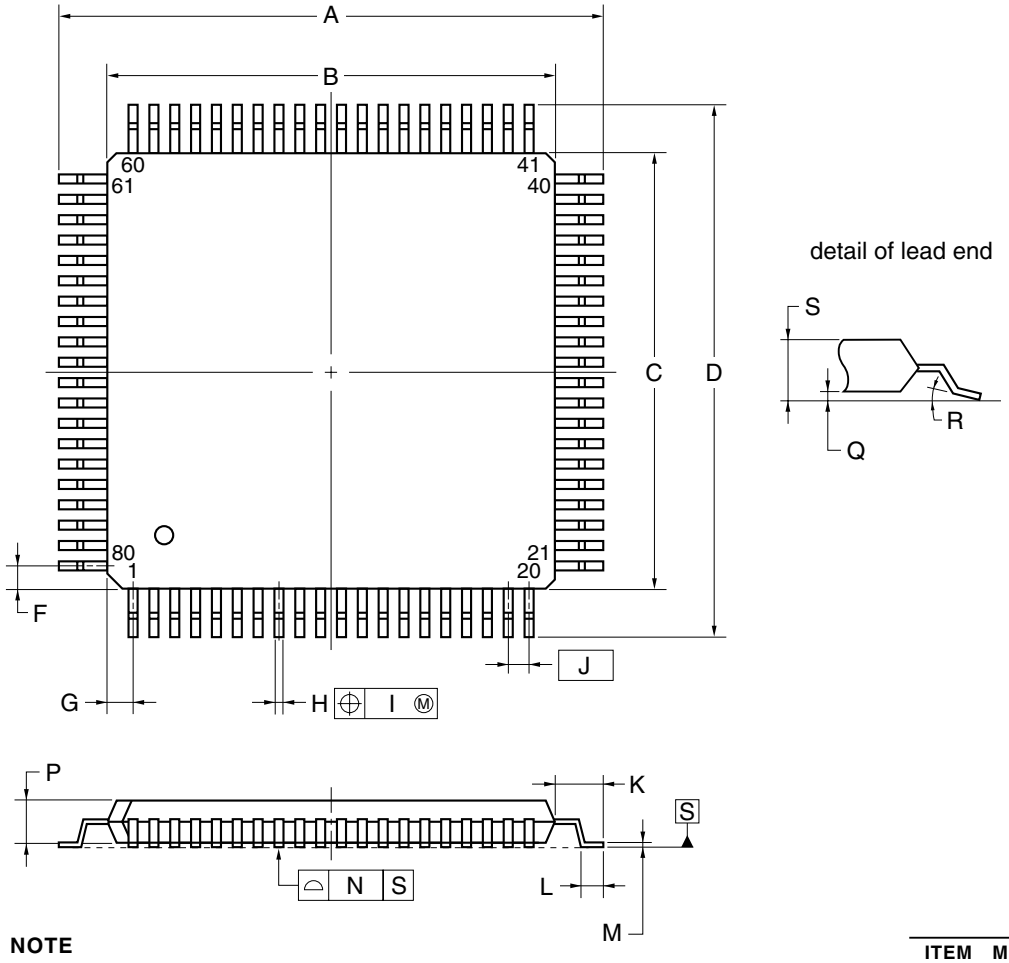
### Clock Timing




### Data Retention Characteristics



## 80-PIN PLASTIC QFP (14x14)



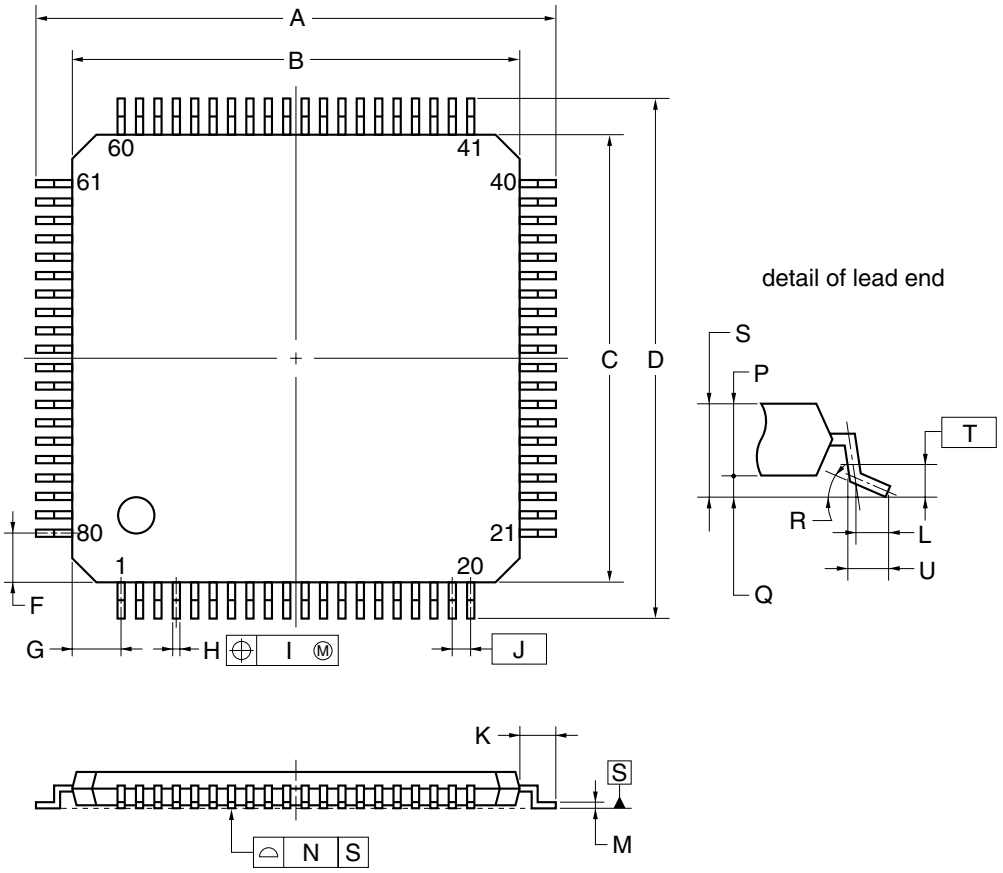

**NOTE**

Each lead centerline is located within 0.13 mm of its true position (T.P.) at maximum material condition.

| ITEM | MILLIMETERS |
|---|---|
| A | 17.20±0.20 |
| B | 14.00±0.20 |
| C | 14.00±0.20 |
| D | 17.20±0.20 |
| F | 0.825 |
| G | 0.825 |
| H | 0.32±0.06 |
| I | 0.13 |
| J | 0.65 (T.P.) |
| K | 1.60±0.20 |
| L | 0.80±0.20 |
| M | $0.17^{+0.03}_{-0.07}$ |
| N | 0.10 |
| P | 1.40±0.10 |
| Q | 0.125±0.075 |
| R | $3^{\circ}{}^{+7^{\circ}}_{-3^{\circ}}$ |
| S | 1.70 MAX. |

**P80GC-65-8BT-1**

## 80-PIN PLASTIC TQFP (FINE PITCH) (12x12)




**NOTE**

Each lead centerline is located within 0.08 mm of its true position (T.P.) at maximum material condition.

| ITEM | MILLIMETERS |
|---|---|
| A | 14.0±0.2 |
| B | 12.0±0.2 |
| C | 12.0±0.2 |
| D | 14.0±0.2 |
| F | 1.25 |
| G | 1.25 |
| H | 0.22±0.05 |
| I | 0.08 |
| J | 0.5 (T.P.) |
| K | 1.0±0.2 |
| L | 0.5 |
| M | 0.145±0.05 |
| N | 0.08 |
| P | 1.0 |
| Q | 0.1±0.05 |
| R | $3^{\circ}{}^{+4^{\circ}}_{-3^{\circ}}$ |
| S | 1.1±0.1 |
| T | 0.25 |
| U | 0.6±0.15 |

**P80GK-50-9EU-1**

★ # CHAPTER 31 RECOMMENDED SOLDERING CONDITIONS

The μPD784225, 784225Y Subseries should be soldered and mounted under the following recommended conditions.

For details of the recommended soldering conditions, refer to the document **Semiconductor Device Mounting Technology Manual (C10535E)**.

For soldering methods and conditions other than those recommended below, contact an NEC sales representative.

**Table 31-1. Soldering Conditions for Surface Mount Type (1/2)**

**(1) μPD784224GK-×××-9EU: 80-pin plastic TQFP (fine pitch) (12 × 12)**
**μPD784224YGK-×××-9EU: 80-pin plastic TQFP (fine pitch) (12 × 12)**
**μPD784225GK-×××-9EU: 80-pin plastic TQFP (fine pitch) (12 × 12)**
**μPD784225YGK-×××-9EU: 80-pin plastic TQFP (fine pitch) (12 × 12)**
**μPD78F4225GK-9EU: 80-pin plastic TQFP (fine pitch) (12 × 12)**
**μPD78F4225YGK-9EU: 80-pin plastic TQFP (fine pitch) (12 × 12)**

| Soldering Method | Soldering Conditions | Recommended Condition Symbol |
|---|---|---|
| Infrared reflow | Package peak temperature: 235°C, Time: 30 seconds max. (at 210°C or higher), Count: Two times or less, Exposure limit: 3 days[Note] (after that, prebake at 125°C for 10 hours) | IR35-103-2 |
| VPS | Package peak temperature: 215°C, Time: 40 seconds max. (at 200°C or higher), Count: Two times or less, Exposure limit: 3 days[Note] (after that, prebake at 125°C for 10 hours) | VP15-103-2 |
| Partial heating | Pin temperature: 300°C max., Time: 3 seconds max. (per pin row) | — |

**Note** After opening the dry pack, store it at 25°C or less and 65%RH or less for the allowable storage period.

**Caution Do not use different soldering methods together (except for partial heating).**

**(2) μPD784225GC-×××-8BT: 80-pin plastic QFP (14 × 14)**

| Soldering Method | Soldering Conditions | Recommended Condition Symbol |
|---|---|---|
| Infrared reflow | Package peak temperature: 235°C, Time: 30 seconds max. (at 210°C or higher), Count: Two times or less | IR35-00-2 |
| VPS | Package peak temperature: 215°C, Time: 40 seconds max. (at 200°C or higher), Count: Two times or less | VP15-00-2 |
| Wave soldering | Solder bath temperature: 260°C max., Time: 10 seconds max., Count: Once, Preheating temperature: 120°C max. (package surface temperature) | WS60-00-1 |
| Partial heating | Pin temperature: 300°C max., Time: 3 seconds max. (per pin row) | — |

**Caution Do not use different soldering methods together (except for partial heating).**

**Table 31-1. Soldering Conditions for Surface Mount Type (2/2)**

**(3) μPD784224GC-×××-8BT: 80-pin plastic QFP (14 × 14)**
**μPD784224YGC-×××-8BT: 80-pin plastic QFP (14 × 14)**
**μPD784225YGC-×××-8BT: 80-pin plastic QFP (14 × 14)**
**μPD78F4225GC-8BT: 80-pin plastic QFP (14 × 14)**
**μPD78F4225YGC-8BT: 80-pin plastic QFP (14 × 14)**

| Soldering Method | Soldering Conditions | Recommended Condition Symbol |
|---|---|---|
| Infrared reflow | Package peak temperature: 235°C, Time: 30 seconds max. (at 210°C or higher), Count: Two times or less, Exposure limit: 7 days[Note] (after that, prebake at 125°C for 10 hours) | IR35-107-2 |
| VPS | Package peak temperature: 215°C, Time: 40 seconds max. (at 200°C or higher), Count: Two times or less, Exposure limit: 7 days[Note] (after that, prebake at 125°C for 10 hours) | VP15-107-2 |
| Wave soldering | Solder bath temperature: 260°C max., Time: 10 seconds max., Count: Once, Preheating temperature: 120°C max. (package surface temperature), Exposure limit: 7 days[Note] (after that prepake at 125°C for 10 hours) | WS60-107-1 |
| Partial heating | Pin temperature: 300°C max., Time: 3 seconds max. (per pin row) | — |

**Caution Do not use different soldering methods together (except for partial heating).**

# APPENDIX A MAJOR DIFFERENCES BETWEEN THE μPD784225, 784225Y SUBSERIES, μPD784216A SUBSERIES AND μPD780058A SUBSERIES

| Item \ Series Name | | μPD784225 Subseries | μPD784216A Subseries | μPD780058 Subseries |
|---|---|---|---|---|
| CPU | | 16-bit CPU | | 8-bit CPU |
| Minimum instruction execution time | When the main system clock is selected | 160 ns (@ 12.5 MHz operation) | | 400 ns (@ 5.0 MHz operation) |
| | When the subsystem clock is selected | 61 μs (@ 32.768 kHz operation | | 122 μs (@ 32.768 kHz operation) |
| Memory space | | 1 MB | | 64 KB |
| I/O ports | Total | 67 | 86 | 68 |
| | CMOS inputs | 8 | 8 | 2 |
| | CMOS I/O | 59 | 72 | 62 |
| | N-ch open-drain I/O | — | 6 | 4 |
| Pins with added functions[Note] | Pins with pull-up resistors | 57 | 70 | 66 (62 for flash memory versions) |
| | LED direct drive outputs | 16 | 22 | 12 |
| | Middle voltage pins | — | 6 | 4 |
| Timer/counters | | • 16-bit timer/event counter × 1 unit<br>• 8-bit timer/event counter × 4 units<br>• 8-bit timer × 2 units | • 16-bit timer/event counter × 1 unit<br>• 8-bit timer/event counter × 6 units | • 16-bit timer/event counter × 1 unit<br>• 8-bit timer/event counter × 2 units |
| Serial interfaces | | • UART/IOE (3-wire serial I/O) × 2 channels<br>• CSI (3-wire serial I/O) × 1 channel | | • UART (time division transmission function)/ IOE (3-wire serial I/O) × 2 channels<br>• CSI (3-wire serial I/ O, 2-wire serial I/O, SBI) × 1 channel<br>• CSI (3-wire serial I/O with automatic communication function) × 1 channel |
| Interrupts | NMI pin | Yes | | No |
| | Macro service | Yes | | No |
| | Context switching | Yes | | No |
| | Programmable priority | 4 levels | | 2 levels |
| Standby function | | • HALT/STOP/IDLE mode<br>• In low power consumption mode: HALT or IDLE mode | | HALT/STOP mode |
| ROM correction | | Yes | No | Yes |
| Package | | • 80-pin plastic QFP (14 × 14)<br>• 80-pin plastic TQFP (fine pitch) (12 × 12) | • 100-pin plastic QFP (fine pitch) (14 × 14)<br>• 100-pin plastic QFP (14 × 20) | • 80-pin plastic QFP (14 × 14)<br>• 80-pin plastic TQFP (fine pitch) (12 × 12) |

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

# APPENDIX B DEVELOPMENT TOOLS

The following development tools are available for system development using the μPD784225 Subseries. Figure B-1 shows the development tools.

- For PC98-NX series
  Unless otherwise specified, products supported by IBM PC/AT™ and compatible machines can be used for the PC98-NX series. When using the PC98-NX series, refer to the explanation of IBM PC/AT and compatible machines.
- For Windows
  Unless otherwise specified, “Windows” indicates the following OSs.
  - Windows 3.1
  - Windows 95, 98, 2000
  - Windows NT™ Ver. 4.0

**Figure B-1. Development Tool Configuration (1/2)**

**(1) When using in-circuit emulator IE-78K4-NS**

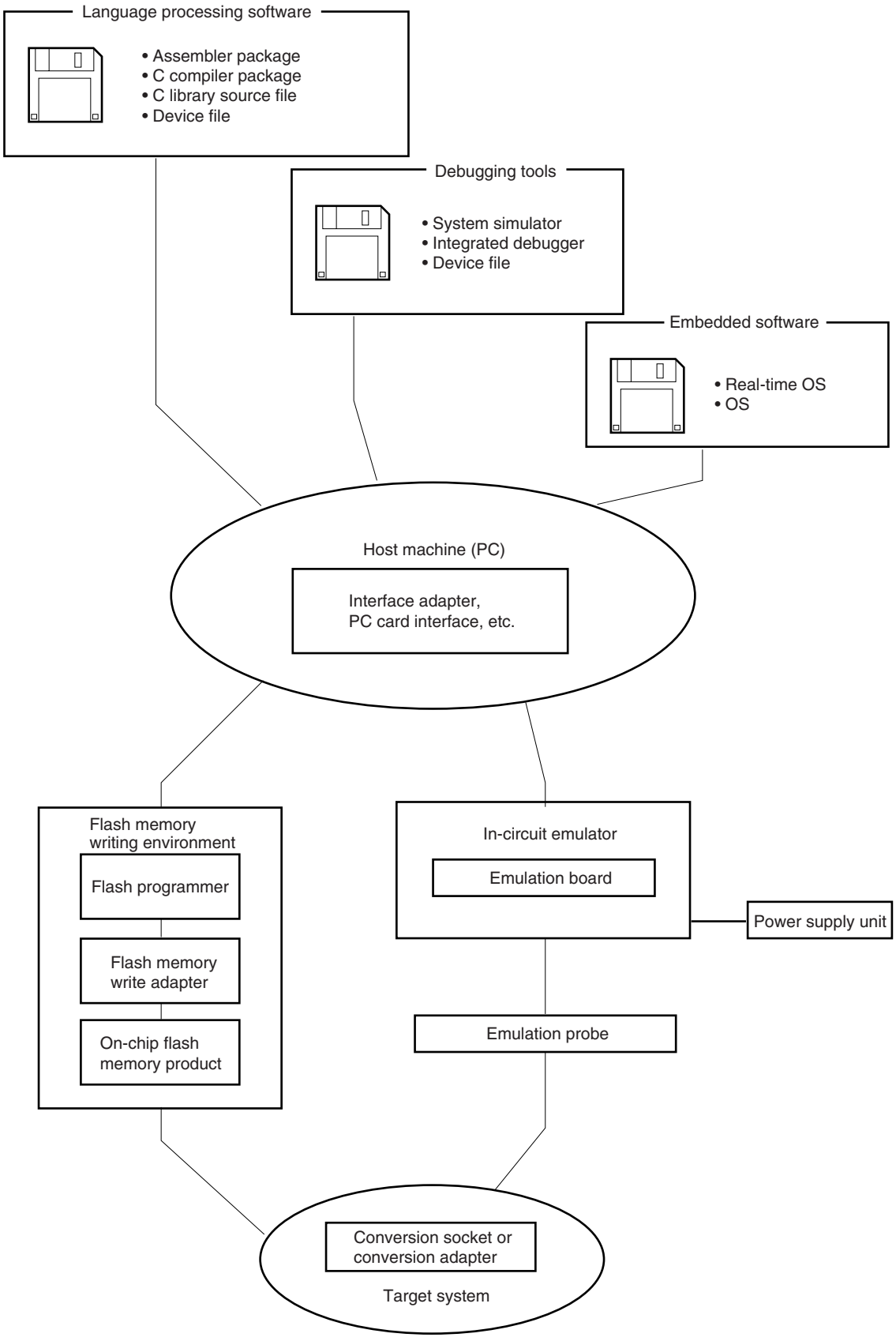
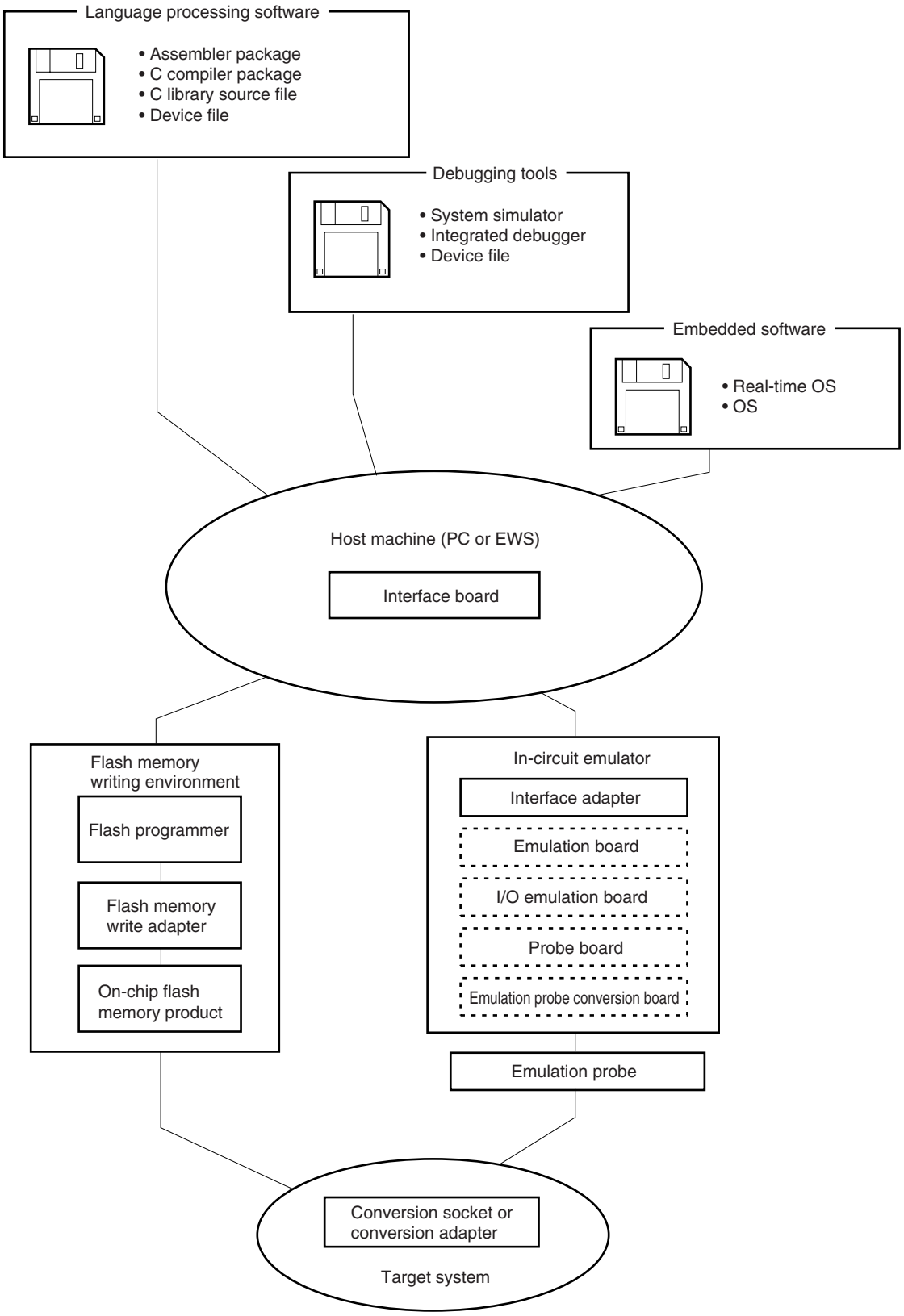


**Figure B-1. Development Tool Configuration (2/2)**

**(2) When using in-circuit emulator IE-784000-R**




**Remark** Parts enclosed by broken lines vary depending on the product. Refer to **B.3.1 Hardware**.

## B.1 Language Processing Software

| | | |
|---|---|---|
| ★ | SP78K4 78K/IV Series software package | Development tools (software) common to the 78K/IV Series are combined in this package. |
| | | Part number: μSxxxxSP78K4 |
| | RA78K4 Assembler package | Program that converts a program written in mnemonic to an executable microcontroller object code.<br>In addition, this assembler package has functions to create symbol tables and optimize branch instructions, etc. automatically.<br>Use this in combination with the device file (DF784225) sold separately.<br>**<Caution on using in PC environment>**<br>Although the assembler package is a DOS-based application, it can be used in the Windows environment by using the Project Manager (included in the assembler package) on Windows. |
| | | Part number: μSxxxxRA78K4 |
| | CC78K4 C compiler package | Program that converts a program written in C language to an executable microcontroller object code.<br>Use this in combination with the assembler package and device file sold separately.<br>**<Caution on using in PC environment>**<br>Although the C compiler package is a DOS-based application, it can be used in the Windows environment by using the Project Manager (included in the assembler package) on Windows. |
| | | Part number: μSxxxxCC78K4 |
| | DF784225[Note] Device file | File containing device-specific information.<br>Use this in combination with the tools sold separately (RA78K4, CC78K4, SM78K4, ID78K4-NS, ID78K4).<br>The supported OS and host machine differ depending on the tool combinations. |
| | | Part number: μSxxxxDF784225 |
| | CC78K4-L C library source file | Function source file configuring the object library included in the C compiler package.<br>This is required when changing the object library included in the C compiler package to accord with the user's specifications.<br>Because this is a source file, the operating environment does not depend on the OS. |
| | | Part number: μSxxxxCC78K4-L |

**Note** The DF784225 can be used commonly for all the RA78K4, CC78K4, SM78K4, ID78K4-NS, and ID78K4.

★ **Remark** The ××××part number differs depending on the host machine and operating system used.

μS××××SP78K4

| ×××× | Host Machine | OS | Supply Medium |
|---|---|---|---|
| AB17 | PC-9800 series, IBM PC/AT compatibles | Japanese Windows | CD-ROM |
| BB17 | | English Windows | |

μS××××RA78K4
μS××××CC78K4

| ×××× | Host Machine | OS | Supply Medium |
|---|---|---|---|
| AB13 | PC-9800 series, IBM PC/AT compatibles | Japanese Windows | 3.5-inch 2HD FD |
| BB13 | | English Windows | |
| AB17 | | Japanese Windows | CD-ROM |
| BB17 | | English Windows | |
| 3P17 | HP9000 series 700™ | HP-UX™ (Rel. 10.10) | |
| 3K17 | SPARCstation™ | SunOS™ (Rel. 4.1.4), Solaris™ (Rel. 2.5.1) | |

μS××××DF784225
μS××××CC78K4-L

| ×××× | Host Machine | OS | Supply Medium |
|---|---|---|---|
| AB13 | PC-9800 series, IBM PC/AT compatibles | Japanese Windows | 3.5-inch 2HD FD |
| BB13 | | English Windows | |
| 3P16 | HP9000 series 700 | HP-UX (Rel. 10.10) | DAT |
| 3K13 | SPARCstation | SunOS (Rel. 4.1.4) | 3.5-inch 2HD FD |
| 3K15 | | Solaris (Rel. 2.5.1) | 1/4-inch CGMT |

## B.2 Flash Memory Writing Tools

| | |
|---|---|
| Flashpro III (Part No.:FL-PR3, PG-FP3)<br>Flash programmer | Dedicated flash programmer for microcontrollers with on-chip flash memory. |
| FA-80GC-8BT<br>FA-80GK-9EU<br>Flash memory writing adapter | Adapter for flash memory writing. Used with the Flashpro III connected.<br>• FA-80GC-8BT: For 80-pin plastic QFP (GC-8BT type)<br>• FA-80GK-9EU: For 80-pin plastic TQFP (GK-9EU type) |

**Remark** The FL-PR3, FA-80GC-8BT, and FA-80GK-9EU are products made by Naito Densei Machida Mfg.Co., Ltd.
For further information, contact Naito Densei Machida Mfg. Co., Ltd. (TEL: +81-45-475-4191)

## B.3 Debugging Tools

### B.3.1 Hardware (1/2)

#### (1) When using in-circuit emulator IE-78K4-NS

| | |
|---|---|
| IE-78K4-NS<br>In-circuit emulator | In-circuit emulator used to debug hardware and software when developing application systems using the 78K/IV Series. Supports the integrated debugger (ID78K4-NS). Use in combination with an interface adapter to connect to the power supply unit, emulation probe, and host machine. |
| IE-70000-MC-PS-B<br>Power supply unit | Adapter to supply power from a socket of AC 100 V to 240 V |
| IE-70000-98-IF-C<br>Interface adapter | Interface adapter required when a PC-9800 series PC (except notebook type) is used as the host machine for the IE-78K4-NS (C bus supported) |
| IE-70000-CD-IF-A<br>PC card Interface | PC card and interface cable required when a notebook PC is used as the host machine for the IE-78K4-NS (PCMCIA socket supported) |
| IE-70000-PC-IF-C<br>Interface adapter | Interface adapter required when using an IBM PC/AT compatible as the host machine for the IE-78K4-NS (ISA bus supported) |
| IE-70000-PCI-IF-A<br>Interface adapter | Interface adapter required when using a PC that incorporates PCI bus as the host machine for the IE-78K4-NS |
| IE-784225-NS-EM1<br>Emulation board | Board to emulate the peripheral hardware specific to device. Used in combination with an in-circuit emulator. |
| NP-80GK<br>Emulation probe | Probe used to connect the in-circuit emulator and the target system. This is for an 80 pin plastic TQFP (fine pitch) (GK-9EU type). |
| TGK-080SDW<br>Conversion adapter<br>(refer to **Figure B-4**) | Conversion adapter to connect the NP-80GK and a target system board on which an 80-pin plastic TQFP (fine pitch) (GK-9EU type) can be mounted |
| NP-80GC<br>Emulation probe | Probe used to connect the in-circuit emulator and the target system. This is for an 80 pin plastic QFP (GC-8BT type). |
| EV-9200GC-80<br>Conversion socket (refer to **Figures B-2** and **B-3**) | Conversion socket to connect the NP-80GC and a target system board on which an 80-pin plastic QFP (GC-8BT type) can be mounted |

**Remarks 1.** The NP-80GK and NP-80GC are products made by Naito Densei Machida Mfg.Co., Ltd.
For further information, contact Naito Densei Machida Mfg. Co., Ltd. (TEL: +81-45-475-4191)

**2.** The TGK-080SDW is a product made by Tokyo Eletech Corporation.
For further information, contact Daimaru Kogyo, Ltd.
Tokyo Electronics Department (TEL: +81-3-3820-7112)
Osaka Electronics Department (TEL: +81-6-6244-6672)

**3.** The EV-9200GC-80 is sold in sets of 5.

**4.** The TGK-080SDW is sold individually.

### B.3.1 Hardware (2/2)

#### (2) When using in-circuit emulator IE-784000-R

| | |
|---|---|
| IE-784000-R<br>In-circuit emulator | The IE-784000-R is an in-circuit emulator common to the 78K/IV Series, and is used in combination with IE-784000-R-EM and IE-784225-NS-EM1, which are sold separately. This in-circuit emulator debugs the connected host machine. An integrated debugger (ID78K4) and device file (sold separately) are required to enable debugging in C language and structured assembly language at the source program level. More efficient debugging and program verification is possible with functions such as C0 coverage. Connect to a host machine via Ethernet™ or a dedicated bus. An interface adapter (sold separately) is required for connection. |
| IE-70000-98-IF-C<br>Interface adapter | Interface adapter required when a PC-9800 series (except notebook type PC) is used as the host machine for the IE-784000-R (C bus supported) |
| IE-70000-PC-IF-C<br>Interface adapter | Interface adapter required when using an IBM PC/AT compatible as the host machine (ISA bus supported) |
| IE-78000-R-SV3<br>Interface adapter | Interface adapter and cable required when an EWS is used as the host machine for the IE-784000-R. Connect to a board inside the IE-784000-R.<br>Note that 10Base-5 is supported as the Ethernet. A commercial conversion adapter is required for other systems. |
| IE-784000-R-EM | Emulation board common to 78K/IV Series |
| IE-784225-NS-EM1<br>Emulation board | Board to emulate peripheral hardware specific to device |
| IE-78K4-R-EX2<br>Emulation probe conversion board | Conversion board for 80-pin packages required when using the IE-784225-NS-EM1 on IE-784000-R |
| EP-78054GK-R<br>Emulation probe | Probe to connect the in-circuit emulator and the target system. For 80-pin plastic TQFP (fine pitch)(GK-9EU type). |
| TGK-080SDW<br>Conversion adapter<br>(refer to **Figure B-4**) | Conversion adapter to connect the EP-78054GK-R and a target system board on which an 80-pin plastic TQFP (fine pitch)(GK-9EU type) can be mounted |
| EP-78230GC-R<br>Emulation probe | Probe to connect the in-circuit emulator and the target system. For 80-pin plastic QFP (GC-8BT type). |
| EV-9200GC-80<br>Conversion socket<br>(refer to **Figures B-2** and **B-3**) | Conversion socket to connect the EP-78230GC-R and a target system board on which an 80-pin plastic QFP (GC-8BT type) can be mounted |

**Remarks 1.** The TGK-080SDW is a product made by Tokyo Eletech Corporation.
For further information, contact Daimaru Kogyo, Ltd.
Tokyo Electronics Department (TEL: +81-3-3820-7112)
Osaka Electronics Department (TEL: +81-6-6244-6672)

**2.** The EV-9200GC-80 is sold in sets of 5.

**3.** The TGK-080SDW is sold individually.

### B.3.2 Software

| | |
|---|---|
| SM78K4<br>System simulator | This enables debugging at the C source level or assembler level while simulating operation of the target system on the host machine.<br>The SM78K4 operates on Windows.<br>By using the SM78K4, logic verification and performance verification can be performed separately to hardware development without using an in-circuit emulator, thus improving development efficiency and software quality.<br>Use the SM78K4 in combination with the device file (DF784225) sold separately. |
| | Part number: μS××××SM78K4 |

**Remark** The ×××× part number differs depending on the host machine and operating system used.

μS××××SM78K4

★

| ×××× | Host Machine | OS | Supply Medium |
|---|---|---|---|
| AB13 | IBM PC/AT compatible | Japanese Windows | 3.5-inch 2HD FD |
| BB13 | | English Windows | |
| AB17 | | Japanese Windows | CD-ROM |
| BB17 | | English Windows | |

| | |
|---|---|
| ID78K4-NS<br>Integrated debugger (supporting in-circuit emulator IE-78K4-NS) | Windows and OSF/Motif™ are employed as the GUI for PC and EWS respectively providing users with their unique look and operability. In addition, the enhanced C language supported debug function enables the result of a trace to be displayed at the C language level using the window integration function in which the source program, disassemble display, and memory display are linked to the result of trace. Moreover, the efficiency of debugging programs that use a real-time OS can be raised by installing function expansion modules such as task debuggers and system performance analyzers.<br>Control program to debug the 78K/IV Series.<br>Use these integrated debuggers in combination with the device file (DF784225) sold separately. |
| ID78K4<br>Integrated debugger (supporting in-circuit emulator IE-784000-R) | |
| | Part number: μS××××ID78K4-NS, μS××××ID78K4 |

**Remark** The ×××× part number differs depending on the host machine and operating system used.

μS××××SM78K4-NS

★ μS××××ID78K4

★

| ×××× | Host Machine | OS | Supply Medium |
|---|---|---|---|
| AB13 | IBM PC/AT compatible | Japanese Windows | 3.5-inch 2HD FD |
| BB13 | | English Windows | |
| AB17 | | Japanese Windows | CD-ROM |
| BB17 | | English Windows | |

## ★ B.4 Cautions on Designing Target System

The connection condition diagrams for the emulation probe, conversion socket, and conversion adapter are shown below. Design the system considering the shape of components, etc. to be mounted on the target system in accordance with this configuration.

**Figure B-2. Distance Between In-Circuit Emulator and Conversion Socket**

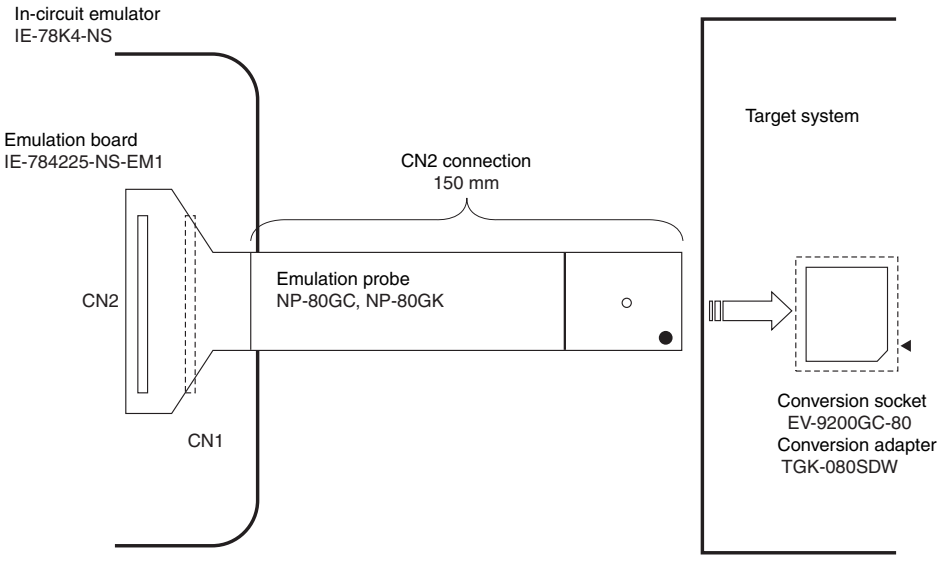
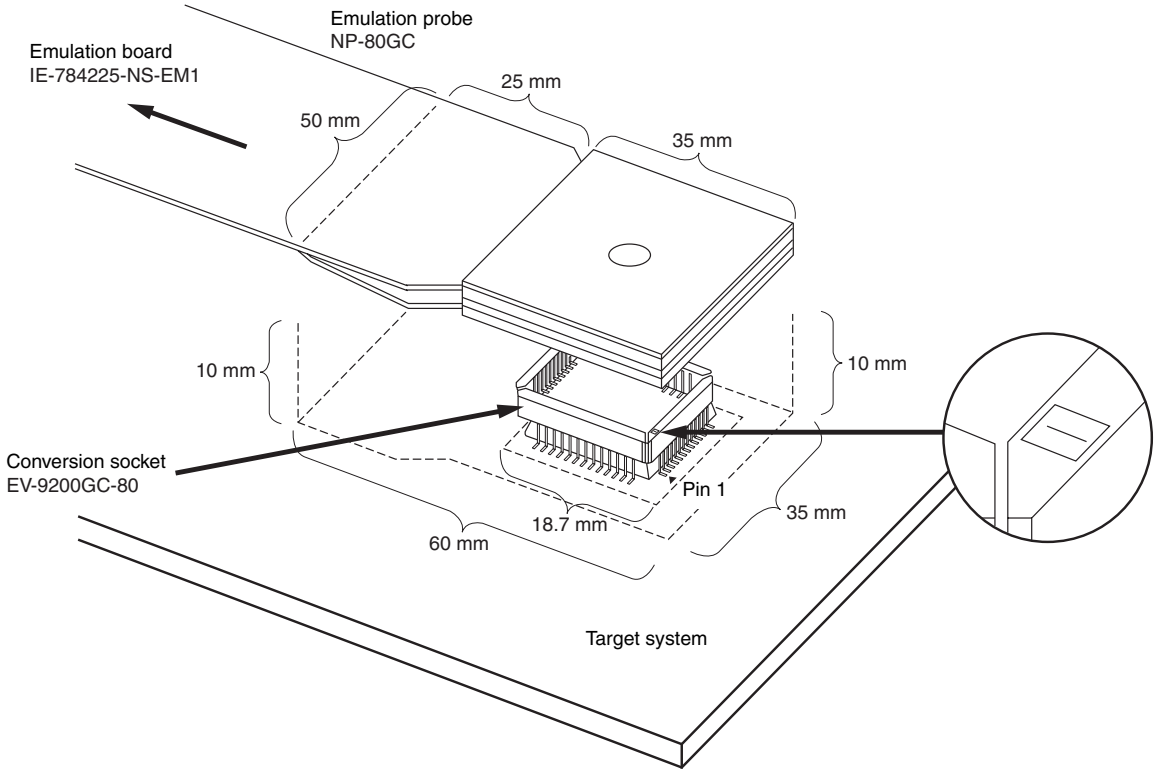
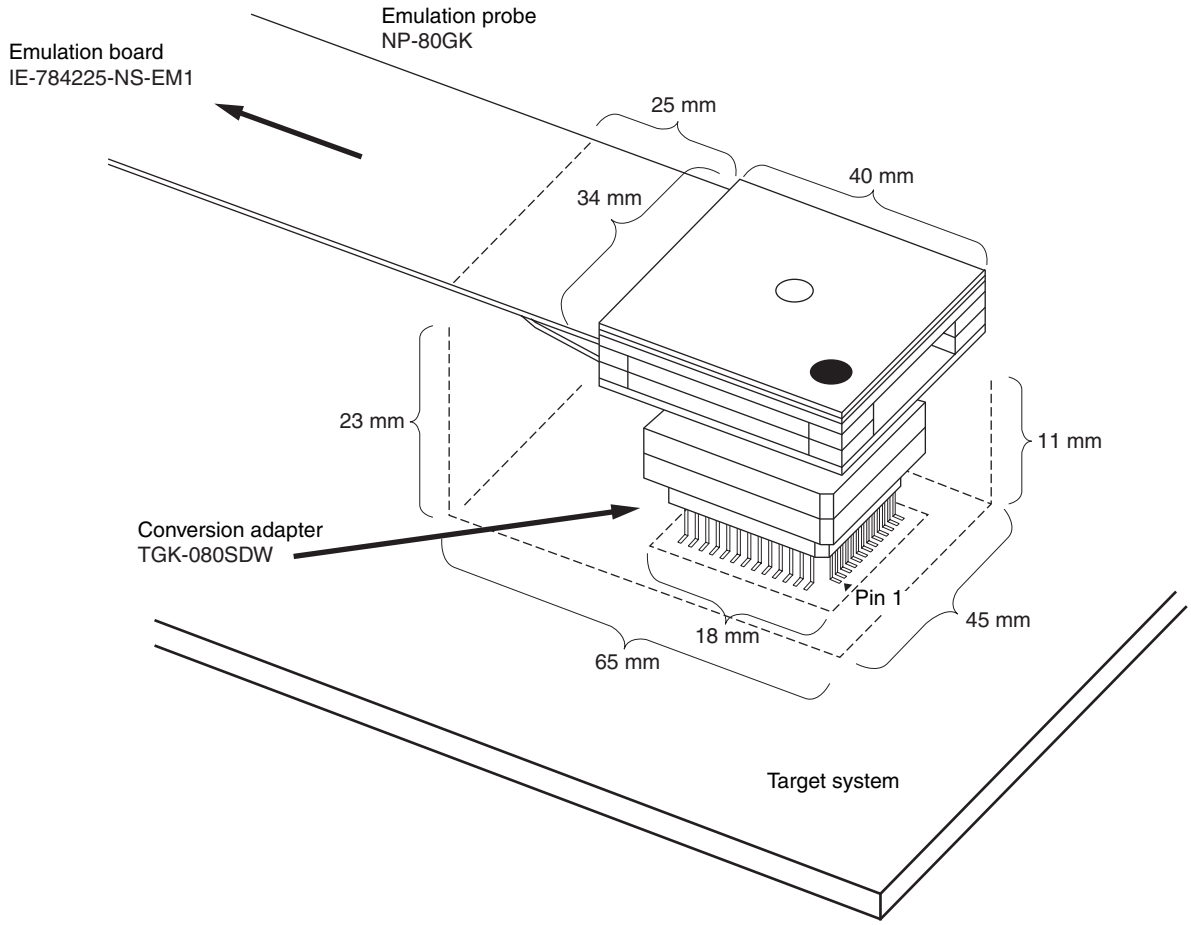



**Figure B-3. Target System Connection Conditions (1)**




**Remark** NP-80GC is a product made by Naito Densei Machida Mfg. Co., Ltd.

**Figure B-4. Target System Connection Conditions (2)**

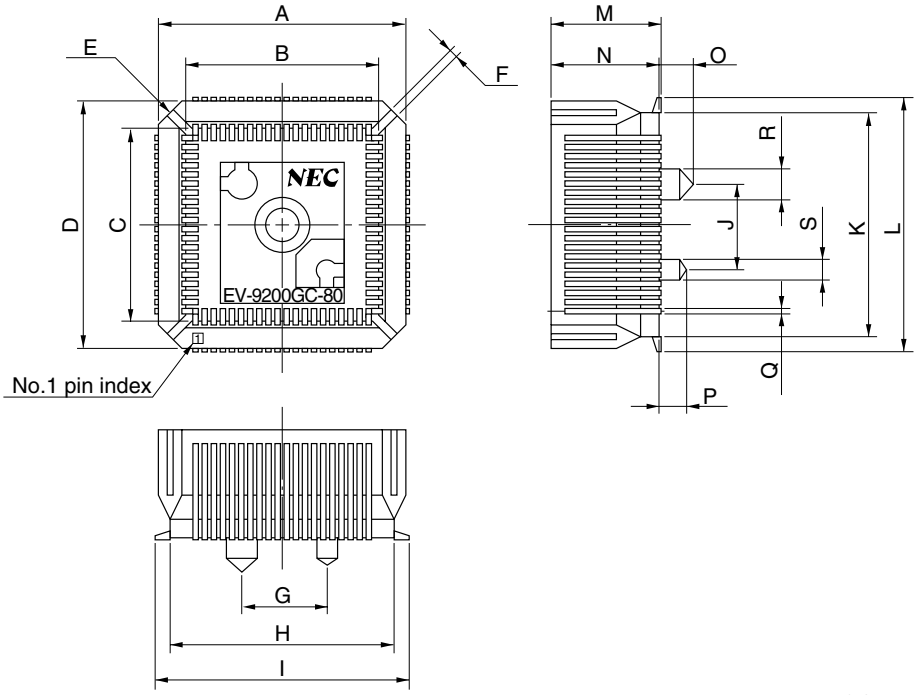



**Remark** NP-80GK is a product made by Naito Densei Machida Mfg. Co., Ltd.
TGK-080SDW is a product made by TOKYO ELETECH CORPORATION.

## B.5 Conversion Socket (EV-9200GC-80) and Conversion Adapter (TGK-080SDW)

### (1) The package drawing of the conversion socket (EV-9200GC-80) and recommended board installation pattern

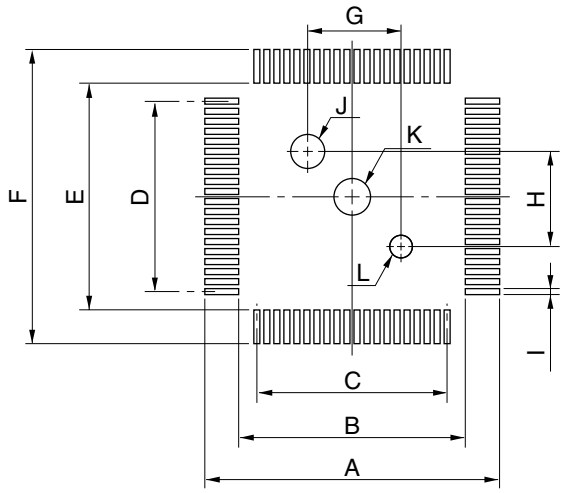
**Figure B-5. Package Drawing of EV-9200GC-80 (Reference) (Unit: mm)**




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 | $\phi$2.3 | $\phi$0.091 |
| S | $\phi$1.5 | $\phi$0.059 |

**Figure B-6. Recommended Board Installation Pattern of EV-9200GC-80 (Reference) (Unit: mm)**




EV-9200GC-80-P1E

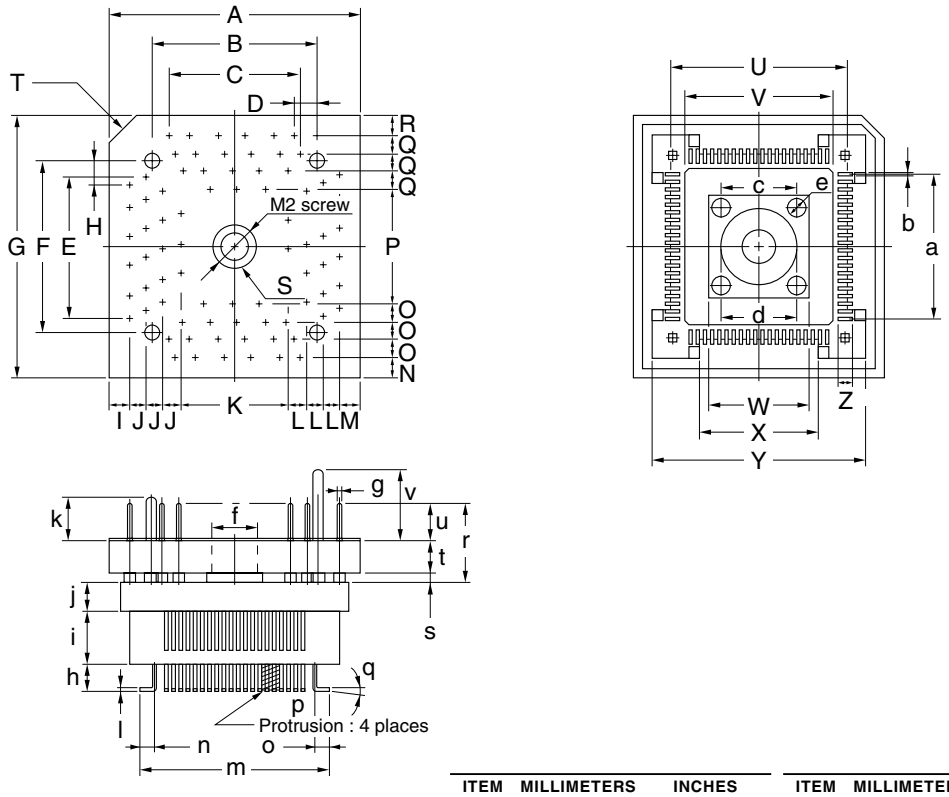
| ITEM | MILLIMETERS | INCHES |
|---|---|---|
| A | 19.7 | 0.776 |
| B | 15.0 | 0.591 |
| C | $0.65\pm0.02\times19=12.35\pm0.05$ | $0.026^{+0.001}_{-0.002}\times0.748=0.486^{+0.003}_{-0.002}$ |
| D | $0.65\pm0.02\times19=12.35\pm0.05$ | $0.026^{+0.001}_{-0.002}\times0.748=0.486^{+0.003}_{-0.002}$ |
| E | 15.0 | 0.591 |
| F | 19.7 | 0.776 |
| G | $6.0\pm0.05$ | $0.236^{+0.003}_{-0.002}$ |
| H | $6.0\pm0.05$ | $0.236^{+0.003}_{-0.002}$ |
| I | $0.35\pm0.02$ | $0.014^{+0.001}_{-0.001}$ |
| J | $\phi2.36\pm0.03$ | $\phi0.093^{+0.001}_{-0.002}$ |
| K | $\phi2.3$ | $\phi0.091$ |
| L | $\phi1.57\pm0.03$ | $\phi0.062^{+0.001}_{-0.002}$ |

**Caution Dimensions of mount pad for EV-9200 and that for target device (QFP) may be different in some parts. For the recommended mount pad dimensions for QFP, refer to "SEMICONDUCTOR DEVICE MOUNTING TECHNOLOGY MANUAL" (C10535E).**

**(2) Package drawing of the conversion adapter (TGK-080SDW)**

Combined with the emulation probe and mounted on the board.

**Figure B-7. TGK-080SDW Package Drawing (Reference) (Unit: mm)**




| ITEM | MILLIMETERS | INCHES |
|---|---|---|
| A | 18.0 | 0.709 |
| B | 11.77 | 0.463 |
| C | 0.5x19=9.5 | 0.020x0.748=0.374 |
| D | 0.5 | 0.020 |
| E | 0.5x19=9.5 | 0.020x0.748=0.374 |
| F | 11.77 | 0.463 |
| G | 18.0 | 0.709 |
| H | 0.5 | 0.020 |
| I | 1.58 | 0.062 |
| J | 1.2 | 0.047 |
| K | 7.64 | 0.301 |
| L | 1.2 | 0.047 |
| M | 1.58 | 0.062 |
| N | 1.58 | 0.062 |
| O | 1.2 | 0.047 |
| P | 7.64 | 0.301 |
| Q | 1.2 | 0.047 |
| R | 1.58 | 0.062 |
| S | $\phi$3.55 | $\phi$0.140 |
| T | C 2.0 | C 0.079 |
| U | 12.31 | 0.485 |
| V | 10.17 | 0.400 |
| W | 6.8 | 0.268 |
| X | 8.24 | 0.324 |
| Y | 14.8 | 0.583 |
| Z | 1.4±0.2 | 0.055±0.008 |

| ITEM | MILLIMETERS | INCHES |
|---|---|---|
| a | 0.5x19=9.5±0.10 | 0.020x0.748=0.374±0.004 |
| b | 0.25 | 0.010 |
| c | $\phi$5.3 | $\phi$0.209 |
| d | $\phi$5.3 | $\phi$0.209 |
| e | $\phi$1.3 | $\phi$0.051 |
| f | $\phi$3.55 | $\phi$0.140 |
| g | $\phi$0.3 | $\phi$0.012 |
| h | 1.85±0.2 | 0.073±0.008 |
| i | 3.5 | 0.138 |
| j | 2.0 | 0.079 |
| k | 3.0 | 0.118 |
| l | 0.25 | 0.010 |
| m | 14.0 | 0.551 |
| n | 1.4±0.2 | 0.055±0.008 |
| o | 1.4±0.2 | 0.055±0.008 |
| p | h=1.8 $\phi$1.3 | h=0.071 $\phi$0.051 |
| q | 0~5° | 0.000~0.197° |
| r | 5.9 | 0.232 |
| s | 0.8 | 0.031 |
| t | 2.4 | 0.094 |
| u | 2.7 | 0.106 |
| v | 3.9 | 0.154 |

**TGK-080SDW-G1E**

**Note** Made by TOKYO ELETECH Corp.

★

# APPENDIX C EMBEDDED SOFTWARE

The following embedded software is available for more efficient program development or maintenance of the μPD784225 Subseries.

## Real-Time Operating System

| | |
|---|---|
| RX78K4 real-time OS | This is a real-time OS complying with the μITRON specification. The RX78K4 nucleus and tools to create multiple information tables (configurator) have been added.<br>Use the RX78K4 in combination with the assembler package (RA78K4) and device file (DF784225) (sold separately).<br>**<Caution on using in PC environment>**<br>This real-time OS is a DOS-based application. With Windows, use the RX78K/IV at the DOS prompt. |
| | Part number: μS××××RX78K4-ΔΔΔΔ |

**Caution When purchasing the RX78K4, fill out the purchase application and sign the license agreement.**

**Remark** The ×××× and ΔΔΔΔ part numbers vary depending on the host machine and operating system used.

μS××××RX78K4-ΔΔΔΔ

| ΔΔΔΔ | Product Overview | Maximum Number Used During Production |
|---|---|---|
| 001 | Evaluation object | Do not use in mass-produced products. |
| 100K | Production object | 100,000 |
| 001M | | 1,000,000 |
| 010M | | 10,000,000 |
| S01 | Source program | Source program for the production object |

| ×××× | Host Machine | OS | Supply Medium |
|---|---|---|---|
| AA13 | PC-9800 series | Japanese Windows[Note] | 3.5-inch 2HD FD |
| AB13 | IBM PC/AT and compatibles | Japanese Windows[Note] | 3.5-inch 2HC FD |
| BB13 | | English Windows[Note] | |
| 3P16 | HP9000 series 700 | HP-UX (Rel.10.10) | DAT (DDS) |
| 3K13 | SPARCstation | SunOS (Rel.4.1.4) | 3.5-inch 2HC FD |
| 3K15 | | Solaris (Rel.2.5.1) | 1/4-inch CGMT |

**Note** Also operates in DOS environment.

# APPENDIX D REGISTER INDEX

## D.1 Register Index

## D.2 Register Index (Alphabetical Order)

**[W]**

# APPENDIX E REVISION HISTORY

(1/4)

| Edition | Contents | Applied to: |
|---|---|---|
| 2nd edition | Modification of power supply voltage range (only for μPD78F4225, 78F4225Y)<br>Before change: $V_{DD}$ 1.8 to 5.5 V → After change: $V_{DD}$ 1.9 to 5.5 V<br>Modification of package<br>Before change: GK-BE9 type → After change: GK-9EU type<br>Modification of I/O circuit<br>Addition of programmable wait control register 2 (PWC2) | Throughout |
| | Modification of **Table 2-1 Types of Pin I/O Circuits and Recommended Connection of Unused Pins** | **CHAPTER 2 PIN FUNCTIONS** |
| | Modification of stop condition for main system clock oscillator<br>Modification of **Figure 4-1 Block Diagram of Clock Generator**<br>Addition of caution about CST bit to **Figure 4-4 Format of Clock Status Register (PCS)**<br>Modification of CPU clock speed in **4.5 Clock Generator Operations** | **CHAPTER 4 CLOCK GENERATOR** |
| | Modification of block diagram of ports 0 to 13<br>Addition of caution to **Figure 5-21 Format of Pull-up Resistor Option Register** | **CHAPTER 5 PORT FUNCTIONS** |
| | Modification of **Figure 6-1 Block Diagram of Real-Time Output Port**<br>Addition of caution to **Figure 6-4 Format of Real-Time Output Port Control Register (RTPC)**<br>Modification of **6.5 Using This Function** | **CHAPTER 6 REAL-TIME OUTPUT FUNCTIONS** |
| | Modification of **Table 8-3 Valid Edge of TI01 Pin and Capture Trigger of CR00**<br>Addition of **Table 8-4 Valid Edge of TI00 Pin and Capture Trigger of CR01**<br>Modification of **Figure 8-4 Format of 16-Bit Timer Output Control Register (TOC0)**<br>Modification of description in **(1) Pulse width measurement with free-running counter and one capture register**<br>Modification of description in **(2) Measurement of two pulse widths with free-running counter**<br>Addition of caution to **Figure 8-15 Timing of Pulse Width Measurement with Free-Running Counter (with Both Edges Specified)**<br>Addition of caution to **Figure 8-17 Timing of Pulse Width Measurement with Free-Running Counter and Two Capture Registers (with Rising Edge Specified)**<br>Addition of caution to **Figure 8-19 Timing of Pulse Width Measurement by Restarting (with Rising Edge Specified)**<br>Modification of description in **8.4.4 Operation as external event counter**<br>Modification of **Figure 8-26 Timing of One-Shot Pulse Output Operation by Software Trigger** | **CHAPTER 8 16-BIT TIMER/ EVENT COUNTER** |

(2/4)

| Edition | Contents | Applied to: |
|---|---|---|
| 2nd edition | Modification of **Figure 9-1 Block Diagram of 8-Bit Timer/Event Counters 1 and 2**<br>Modification of caution in **(1) 8-bit timer counters 1 and 2 (TM1, TM2)**<br>Modification of caution in **(2) 8-bit compare registers 10 and 20 (CR10, CR20)**<br>Modification of **Figure 9-2 Format of 8-Bit Timer Mode Control Register 1 (TMC1)**<br>Modification of **Figure 9-3 Format of 8-Bit Timer Mode Control Register 2 (TMC2)**<br>Modification of caution in **Figure 9-4 Format of Prescaler Mode Register 1 (PRM1)**<br>Modification of caution in **Figure 9-5 Format of Prescaler Mode Register 2 (PRM2)**<br>Modification of setting method in **9.4.4 Operation as 8-bit PWM output**<br>Modification of **Figure 9-8 Timing of PWM Output** | **CHAPTER 9 8-BIT TIMER/ EVENT COUNTERS 1, 2** |
| | Deletion of timer output from **Table 10-1 Configuration of 8-Bit Timers 5 and 6**<br>Modification of **Figure 10-1 Block Diagram of 8-Bit Timers 5 and 6**<br>Modification of caution in **(1) 8-bit timer counters 5 and 6 (TM5, TM6)**<br>Modification of caution in **(2) 8-bit compare registers 50 and 60 (CR50, CR60)**<br>Modification of description in **(1) 8-bit timer mode control registers 5 and 6 (TMC5, TMC6)**<br>Modification of **Figure 10-2 Format of 8-Bit Timer Mode Control Register 5 (TMC5)**<br>Modification of **Figure 10-3 Format of 8-Bit Timer Mode Control Register 6 (TMC6)**<br>Modification of **Figure 10-4 Format of Prescaler Mode Register 5 (PRM5)**<br>Modification of **Figure 10-5 Format of Prescaler Mode Register 6 (PRM6)**<br>Modification of **Figure 10-6 Timing of Interval Timer Operation**<br>Modification of caution in **10.4.2 Operation as interval timer (16-bit operation)**<br>Modification of **Figure 10-8 Cascade Connection Mode with 16-Bit Resolution** | **CHAPTER 10 8-BIT TIMERS 5, 6** |
| | Modification of **Table 11-1 Interval Time of Interval Timer**<br>Modification of **Figure 11-1 Watch Timer Block Diagram**<br>Modification of **Figure 11-2 Format of Watch Timer Mode Control Register (WTM)**<br>Modification and addition of caution to **Figure 11-3 Operation Timing of Watch Timer/Interval Timer** | **CHAPTER 11 WATCH TIMER** |
| | Modification of **Figure 12-1 Watchdog Timer Block Diagram**<br>Modification of description in **12.3.2 Interrupt priority order** | **CHAPTER 12 WATCHDOG TIMER** |
| | Modification of **Figure 13-2 Format of A/D Converter Mode Register (ADM)**<br>Addition of **13.5 Reading the A/D Converter Characteristics Table**<br>Modification of description in **13.6 Cautions** | **CHAPTER 13 A/D CONVERTER** |
| | Modification of **Figure 16-2 Block Diagram in Asynchronous Serial Interface Mode**<br>Modification of **Table 16-3 Relationship Between 5-Bit Counter Source Clock and m Value**<br>Modification of **Table 16-4 Relationship Between Main System Clock and Baud Rate**<br>Modification of **Figure 16-11 Block Diagram in 3-Wire Serial I/O Mode** | **CHAPTER 16 ASYNCHRONOUS SERIAL INTERFACE/ 3-WIRE SERIAL I/O** |

(3/4)

| Edition | Contents | Applied to: |
|---|---|---|
| 2nd edition | Modification of **Figure 17-1 Block Diagram of Clocked Serial Interface (in 3-Wire Serial I/O Mode)** | **CHAPTER 17 3-WIRE SERIAL I/O MODE** |
| | Modification of **Figure 18-3 Format of $I^2C$ Bus Control Register (IICC0)**<br>Addition of note about bit 3 (TRC0) to **Figure 18-4 Format of $I^2C$ Bus Status Register (IICS0)**<br>Modification of **Figure 18-5 Format of Prescaler Mode Register (SPRM0) for Serial Clock**<br>Modification of description about interrupt request timing of master operation and slave operation in **18.5.7 $I^2C$ interrupt request (INTIIC0)**<br>Modification of value in **Table 18-5 Wait Times** | **CHAPTER 18 $I^2C$ BUS MODE ($\mu$PD784225Y SUBSERIES ONLY)** |
| | Modification of **Figure 21-2 Block Diagram of P00 to P05** | **CHAPTER 21 EDGE DETECTION FUNCTION** |
| | Addition of remark 3 to **Table 22-2 Interrupt Request Sources**<br>Modification of **Figure 22-33 Data Transfer Control Timing**<br>Modification of **Figure 22-36 Automatic Addition Control + Ring Control Block Diagram 1 (When Output Timing Varies with 1-2-Phase Excitation)**<br>Modification of **Figure 22-37 Automatic Addition Control + Ring Control Timing Diagram 1 (When Output Timing Varies with 1-2-Phase Excitation)**<br>Modification of **Figure 22-38 Automatic Addition Control + Ring Control Block Diagram 2 (1-2-Phase Excitation Constant-Velocity Operation)**<br>Modification of **Figure 22-39 Automatic Addition Control + Ring Control Timing Diagram 2 (1-2-Phase Excitation Constant-Velocity Operation)** | **CHAPTER 22 INTERRUPT FUNCTIONS** |
| | **23.2 Control Registers**<br>Modification of **Figure 23-1 Format of Memory Expansion Mode Register (MM)**<br>Addition of **(3) Programmable wait control register 2 (PWC2)** | **CHAPTER 23 LOCAL BUS INTERFACE FUNCTONS** |
| | Modification of **Figure 24-1 Standby Function State Transition**<br>Modification of **Figure 24-4 Format of Oscillation Stabilization Time Specification Register (OSTS)**<br>Modification of **Table 24-2 Operating States in HALT Mode**<br>Modification of **Figure 24-5 Operations After Releasing HALT Mode**<br>Modification of caution in **24.4.1 Settings and operating states of STOP mode**<br>Modification of **Table 24-5 Operating States in STOP Mode**<br>Modification of **Figure 24-6 Operations After Releasing STOP Mode**<br>Modification of **Table 24-7 Operating States in IDLE Mode**<br>Modification of **Figure 24-9 Operations After Releasing IDLE Mode**<br>Modification of description in **(iii) Releasing the HALT mode by $\overline{\text{RESET}}$ input**<br>Modification of description in **(iii) Releasing the IDLE mode by $\overline{\text{RESET}}$ input** | **CHAPTER 24 STANDBY FUNCTION** |
| | Modification of **Figure 25-2 Receiving Reset Signal** | **CHAPTER 25 RESET FUNCTION** |
| | Modification of **Table 26-1 Differences Between 78K/IV ROM Correction and 78K/0 ROM Correction**<br>Modification of example of four pointer settings in **26.5 Conditions for Executing ROM Correction** | **CHAPTER 26 ROM CORREC-TION** |
| | Modification of **Figure 27-1 Format of Internal Memory Size Switching Register (IMS)**<br>Addition of dedicated flash programmer (Flashpro III) | **CHAPTER 27 $\mu$PD78F4225 AND $\mu$PD78F4225Y PROGRAMMING** |

(4/4)

| Edition | Contents | Applied to: |
|---|---|---|
| 2nd edition | Modified throughout | **APPENDIX B DEVELOPMENT TOOLS** |
| | Modified throughout | **APPENDIX C EMBEDDED SOFTWARE** |
| 3rd edition | The following products have been developed.<br>• μPD78F4225GC-8BT, 78F4225GK-9EU, 78F4225YGC-8BT, 78F4225YGK-9EU | Throughout |
| | • Addition of **Caution** related to operation in one-shot pulse output mode | **CHAPTER 8 16-BIT TIMER/EVENT COUNTER** |
| | • Change of watch timer interrupt interval from 0.5 second to "$2^{14}/f_W$ or $2^5/f_W$" | **CHAPTER 11 WATCH TIMER** |
| | Modification of **Figure 18-17 Communication Reservation Procedure** | **CHAPTER 18 $I^2C$ BUS MODE (μPD784225Y SUBSERIES ONLY)** |
| | Modification of **Figure 23-8 Read Modify Write Timing for External Memory in External Memory Expansion Mode** | **CHAPTER 23 LOCAL BUS INTERFACE FUNCTIONS** |
| | **27.2 Flash Memory Overwriting**<br>• Deletion of self overwrite mode<br>**27.3 On-Board Overwrite Mode**<br>• Deletion of dedicated flash programmer (Flashpro II)<br>**Modification of Table 27-3 Communication Modes**<br>**Modification of Figure 27-2 Format of Communication Mode Selection**<br>**Modification of Table 27-4 Major Functions of On-Board Overwrite Mode**<br>• Deletion of batch erase, batch blank check, and batch verify<br>• Change of "block" to "area" | **CHAPTER 27 μPD78F4225 AND μPD78F4225Y PROGRAMMING** |
| 4th | Change of **78K/IV SERIES LINEUP**<br>Modification of minimum instruction execution time in **1.5 Function List** | **CHAPTER 1 OVERVIEW** |
| | Modification of **Cautions** in **Figure 13-2 Format of A/D Converter Mode Register (ADM)** | **CHAPTER 13 A/D CONVERTER** |
| | Addition of **Table 16-2 Serial Interface Operation Mode Settings** | **CHAPTER 16 ASYNCHRONOUS SERIAL INTERFACE/ 3-WIRE SERIAL I/O** |
| | Addition of **Table 17-2 Serial Interface Operation Mode Settings** | **CHAPTER 17 3-WIRE SERIAL I/O MODE** |
| | Addition of reserved words in **Figure 22-21 Format of Macro Service Control Word** | **CHAPTER 22 INTERRUPT FUNCTIONS** |
| | Modification of **Table 23-3 Settings of Program Wait Control Register 2 (PWC2)** | **CHAPTER 23 LOCAL BUS INTERFACE FUNCTIONS** |
| | Modification of **Figure 24-1 Standby Function State Transition** | **CHAPTER 24 STANDBY FUNCTION** |
| | Addition of chapter | **CHAPTER 29 ELECTRICAL SPECIFICATIONS** |
| | Addition of chapter | **CHAPTER 30 PACKAGE DRAWINGS** |
| | Addition of chapter | **CHAPTER 31 RECOMMENDED SOLDERING CONDITIONS** |
| | Addition of description on SP78K4 and change of **Remark** in **B.1 Language Processing Software**<br>Change of **Remark** in **B.3.2 Software**<br>Addition of **B.4 Cautions on Designing Target System** | **APPENDIX B DEVELOPMENT TOOLS** |
| | Deletion of MX78K4 description | **APPENDIX C EMBEDDED SOFTWARE** |

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
Market Communication Dept.
Fax: +49-211-6503-274

**South America**
NEC do Brasil S.A.
Fax: +55-11-6462-6829

**Hong Kong, Philippines, Oceania**
NEC Electronics Hong Kong Ltd.
Fax: +852-2886-9022/9044

**Korea**
NEC Electronics Hong Kong Ltd.
Seoul Branch
Fax: +82-2-528-4411

**P.R. China**
NEC Electronics Shanghai, Ltd.
Fax: +86-21-6841-1137

**Taiwan**
NEC Electronics Taiwan Ltd.
Fax: +886-2-2719-5951

**Asian Nations except Philippines**
NEC Electronics Singapore Pte. Ltd.
Fax: +65-250-3583

**Japan**
NEC Semiconductor Technical Hotline
Fax: +81- 44-435-9608

I would like to report the following error/make the following suggestion:

Document title: _____

Document number: _____ Page number: _____

If possible, please fax the referenced page or drawing.

| Document Rating | Excellent | Good | Acceptable | Poor |
|---|---|---|---|---|
| Clarity | ❑ | ❑ | ❑ | ❑ |
| Technical Accuracy | ❑ | ❑ | ❑ | ❑ |
| Organization | ❑ | ❑ | ❑ | ❑ |