NEC

Preliminary User's Manual

# V850ES/KF1+

## 32-Bit Single-Chip Microcontrollers

## Hardware

---

**μPD703308**
**μPD703308Y**
**μPD70F3306**
**μPD70F3306Y**
**μPD70F3308**
**μPD70F3308Y**

Document No. U16895EJ1V0UD00 (1st edition)
Date Published June 2004 N CP(K)

**[MEMO]**

## NOTES FOR CMOS DEVICES

**① VOLTAGE APPLICATION WAVEFORM AT INPUT PIN**

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

**② HANDLING OF UNUSED INPUT PINS**

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

**③ PRECAUTION AGAINST ESD**

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

**④ STATUS BEFORE INITIALIZATION**

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

Purchase of NEC Electronics $I^2C$ 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.

Caution: $\mu$PD70F3306, 70F3306Y, 70F3308, and 70F3308Y use SuperFlash® technology licensed from Silicon Storage Technology, Inc.

**Solaris and SunOS are trademarks of Sun Microsystems, Inc.**
**SuperFlash is a registered trademark of Silicon Storage Technology, Inc. in several countries including the United States and Japan.**

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

- **The information contained in this document is being issued in advance of the production cycle for the product. The parameters for the product may change before final production or NEC Electronics Corporation, at its own discretion, may withdraw the product prior to its production.**
- **Not all products and/or types are available in every country. Please check with an NEC Electronics sales representative for availability and additional information.**
- No part of this document may be copied or reproduced in any form or by any means without the prior written consent of NEC Electronics. NEC Electronics assumes no responsibility for any errors that may appear in this document.
- NEC Electronics does not assume any liability for infringement of patents, copyrights or other intellectual property rights of third parties by or arising from the use of NEC Electronics products listed in this document or any other liability arising from the use of such products. No license, express, implied or otherwise, is granted under any patents, copyrights or other intellectual property rights of NEC Electronics or others.
- Descriptions of circuits, software and other related information in this document are provided for illustrative purposes in semiconductor product operation and application examples. The incorporation of these circuits, software and information in the design of a customer's equipment shall be done under the full responsibility of the customer. NEC Electronics assumes no responsibility for any losses incurred by customers or third parties arising from the use of these circuits, software and information.
- While NEC Electronics endeavors to enhance the quality, reliability and safety of NEC Electronics products, customers agree and acknowledge that the possibility of defects thereof cannot be eliminated entirely. To minimize risks of damage to property or injury (including death) to persons arising from defects in NEC Electronics products, customers must incorporate sufficient safety measures in their design, such as redundancy, fire-containment and anti-failure features.
- NEC Electronics products are classified into the following three quality grades: "Standard", "Special" and "Specific". The "Specific" quality grade applies only to NEC Electronics products developed based on a customer-designated "quality assurance program" for a specific application. The recommended applications of an NEC Electronics products depend on its quality grade, as indicated below. Customers must check the quality grade of each NEC Electronics product before using it in a particular application.

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

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

  "Specific": Aircraft, aerospace equipment, submersible repeaters, nuclear reactor control systems, life support systems and medical equipment for life support, etc.

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

(Note)
(1) "NEC Electronics" as used in this statement means NEC Electronics Corporation and also includes its majority-owned subsidiaries.
(2) "NEC Electronics products" means any product developed or manufactured by or for NEC Electronics (as defined above).

M5D 02. 11-1

# Regional Information

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

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

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

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

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

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

- **Sucursal en España**
  Madrid, Spain
  Tel: 091-504 27 87
- **Succursale Française**
  Vélizy-Villacoublay, France
  Tel: 01-30-67 58 00
- **Filiale Italiana**
  Milano, Italy
  Tel: 02-66 75 41
- **Branch The Netherlands**
  Eindhoven, The Netherlands
  Tel: 040-244 58 45
- **Tyskland Filial**
  Taeby, Sweden
  Tel: 08-63 80 820
- **United Kingdom Branch**
  Milton Keynes, UK
  Tel: 01908-691-133

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

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

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

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

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

J04.1

# PREFACE

**Readers**

This manual is intended for users who wish to understand the functions of the V850ES/KF1+ and design application systems using these products.

**Purpose**

This manual is intended to give users an understanding of the hardware functions of the V850ES/KF1+ shown in the **Organization** below.

**Organization**

This manual is divided into two parts: Hardware (this manual) and Architecture (**V850ES Architecture User's Manual**).

| Hardware | Architecture |
|---|---|
| • Pin functions | • Data types |
| • CPU function | • Register set |
| • On-chip peripheral functions | • Instruction format and instruction set |
| • Flash memory programming | • Interrupts and exceptions |
| • Electrical specifications (target) | • Pipeline operation |

**How to Read This Manual**

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

To find the details of a register where the name is known
→ Refer to **APPENDIX C REGISTER INDEX**.

To understand the details of an instruction function
→ Refer to the **V850ES Architecture User's Manual**.

Register format
→ The name of the bit whose number is in angle brackets (<>) in the figure of the register format of each register is defined as a reserved word in the device file.

To understand the overall functions of the V850ES/KF1+
→ Read this manual according to the **CONTENTS**.

To know the electrical specifications of the V850ES/KF1+
→ Refer to **CHAPTER 30 ELECTRICAL SPECIFICATIONS (TARGET)**.

The "yyy bit of the xxx register" is described as the "xxx.yyy bit" in this manual. Note with caution that if "xxx.yyy" is described as is in a program, however, the compiler/assembler cannot recognize it correctly.

**Conventions**

| | | |
|---|---|---|
| Data significance: | Higher digits on the left and lower digits on the right | |
| Active low representation: | $\overline{xxx}$ (overscore over pin or signal name) | |
| Memory map address: | Higher addresses on the top and lower addresses on the bottom | |
| **Note**: | Footnote for item marked with **Note** in the text | |
| **Caution**: | Information requiring particular attention | |
| **Remark**: | Supplementary information | |
| Numeric representation: | Binary | ... xxxx or xxxxB |
| | Decimal | ... xxxx |
| | Hexadecimal | ... xxxxH |

Prefix indicating power of 2 (address space, memory capacity):

| | |
|---|---|
| K (kilo): | $2^{10} = 1,024$ |
| M (mega): | $2^{20} = 1,024^2$ |
| G (giga): | $2^{30} = 1,024^3$ |

**Related Documents**

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

**Documents related to V850ES/KF1+**

| Document Name | Document No. |
|---|---|
| V850ES Architecture User's Manual | U15943E |
| V850ES/Kx1, V850ES/Kx1+ On-chip Debug User's Manual | U16972E |
| V850ES/KF1+ Hardware User's Manual | This manual |

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

| Document Name | | Document No. |
|---|---|---|
| CA850 Ver. 2.50 C Compiler Package | Operation | U16053E |
| | C Language | U16054E |
| | Assembly Language | U16042E |
| PM plus Ver. 5.10 | | U16569E |
| ID850QB Ver. 2.80 Integrated Debugger | Operation | U16973E |
| SM plus Ver. 1.00 System Simulator | Operation | U16906E |
| | User Open Interface Specifications | U16907E |
| RX850 Ver. 3.13 or Later Real-Time OS | Basics | U13430E |
| | Installation | U13410E |
| | Technical | U13431E |
| RX850 Pro Ver. 3.15 Real-Time OS | Basics | U13773E |
| | Installation | U13774E |
| | Technical | U13772E |
| RD850 Ver. 3.01 Task Debugger | | U13737E |
| RD850 Pro Ver. 3.01 Task Debugger | | U13916E |
| AZ850 Ver. 3.20 System Performance Analyzer | | U14410E |
| PG-FP4 Flash Memory Programmer | | U15260E |

**CONTENTS**

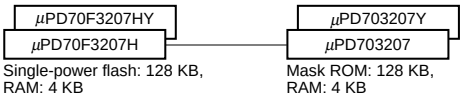
# CHAPTER 1 INTRODUCTION

## 1.1 K1 Family Product Lineup

### 1.1.1 V850ES/Kx1+, V850ES/Kx1 products lineup

- 64-pin plastic LQFP (10 × 10 mm, 0.5 mm pitch)
- 64-pin plastic TQFP (12 × 12 mm, 0.65 mm pitch)

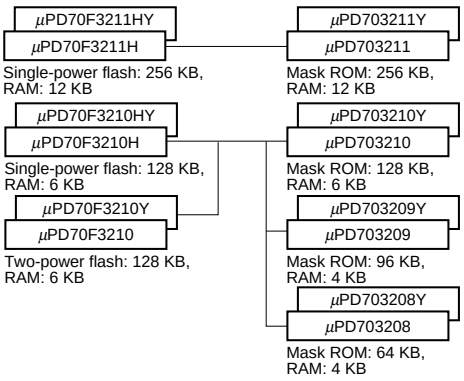
V850ES/KE1



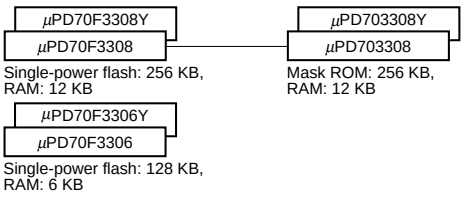

V850ES/KE1+


- 80-pin plastic TQFP (12 × 12 mm, 0.5 mm pitch)
- 80-pin plastic QFP (14 × 14 mm, 0.65 mm pitch)

V850ES/KF1

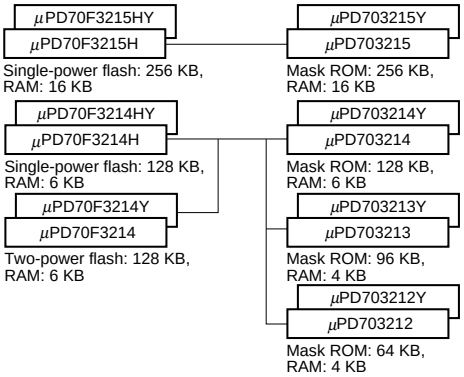



V850ES/KF1+

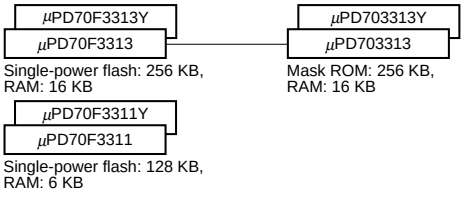



- 100-pin plastic LQFP (14 × 14 mm, 0.5 mm pitch)
- 100-pin plastic QFP (14 × 20 mm, 0.65 mm pitch)

V850ES/KG1

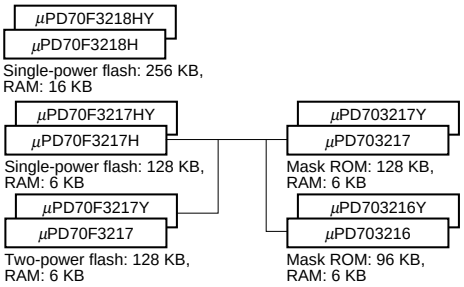



V850ES/KG1+

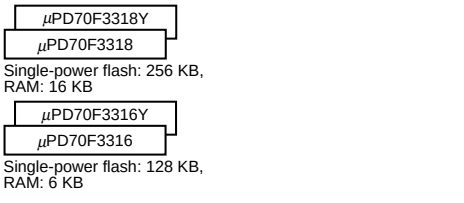



- 144-pin plastic LQFP (20 × 20 mm, 0.5 mm pitch)

V850ES/KJ1




V850ES/KJ1+



The function list of the V850ES/Kx1+ is shown below.

| Product Name | | V850ES/KE1+ | | V850ES/KF1+ | | | V850ES/KG1+ | | | V850ES/KJ1+ | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Number of pins | | 64 pins | | 80 pins | | | 100 pins | | | 144 pins | |
| Internal memory (KB) | Mask ROM | 128 | – | – | 256 | – | – | 256 | – | – | – |
| | Flash memory | – | 128 | 128 | – | 256 | 128 | – | 256 | 128 | 256 |
| | RAM | 4 | | 6 | 12 | | 6 | 16 | | 6 | 16 |
| Supply voltage | | 2.7 to 5.5 V | | | | | | | | | |
| Minimum instruction execution time | | 50 ns @20 MHz | | | | | | | | | |
| Clock | X1 input | 2 to 10 MHz | | | | | | | | | |
| | Subclock | 32.768 kHz | | | | | | | | | |
| | Ring-OSC | 240 kHz (TYP.) | | | | | | | | | |
| Port | CMOS input | 8 | | 8 | | | 8 | | | 16 | |
| | CMOS I/O | 43 | | 59 | | | 76 | | | 112 | |
| | N-ch open-drain I/O | 2 | | 2 | | | 4 | | | 6 | |
| Timer | 16-bit (TMP) | 1 ch | | 1 ch | | | 1 ch | | | 1 ch | |
| | 16-bit (TM0) | 1 ch | | 2 ch | | | 4 ch | | | 6 ch | |
| | 8-bit (TM5) | 2 ch | | 2 ch | | | 2 ch | | | 2 ch | |
| | 8-bit (TMH) | 2 ch | | 2 ch | | | 2 ch | | | 2 ch | |
| | Interval timer | 1 ch | | 1 ch | | | 1 ch | | | 1 ch | |
| | Watch | 1 ch | | 1 ch | | | 1 ch | | | 1 ch | |
| | WDT1 | 1 ch | | 1 ch | | | 1 ch | | | 1 ch | |
| | WDT2 | 1 ch | | 1 ch | | | 1 ch | | | 1 ch | |
| RTO | | 6 bits × 1 ch | | 6 bits × 1 ch | | | 6 bits × 1 ch | | | 6 bits × 2 ch | |
| Serial interface | CSI | 2 ch | | 2 ch | | | 2 ch | | | 3 ch | |
| | Automatic transmit/receive 3-wire CSI | – | | 1 ch | | | 2 ch | | | 2 ch | |
| | UART | 1 ch | | 1 ch | | | 2 ch | | | 2 ch | |
| | UART supporting LIN-bus | 1 ch | | 1 ch | | | 1 ch | | | 1 ch | |
| | $I^2C$[Note] | 1 ch | | 1 ch | | | 1 ch | | | 2 ch | |
| External bus | Address space | – | | 128 KB | | | 3 MB | | | 15 MB | |
| | Address bus | – | | 16 bits | | | 22 bits | | | 24 bits | |
| | Mode | – | | Multiplex only | | | Multiplex/separate | | | | |
| DMA controller | | – | | – | | | 4 ch | | | 4 ch | |
| 10-bit A/D converter | | 8 ch | | 8 ch | | | 8 ch | | | 16 ch | |
| 8-bit D/A converter | | – | | – | | | 2 ch | | | 2 ch | |
| Interrupt | External | 9 | | 9 | | | 9 | | | 9 | |
| | Internal | 27 | | 30 | | | 42 | | | 48 | |
| Key return input | | 8 ch | | 8 ch | | | 8 ch | | | 8 ch | |
| Reset | $\overline{\text{RESET}}$ pin | Provided | | | | | | | | | |
| | POC | 2.7 V or less fixed | | | | | | | | | |
| | LVI | 3.1 V/3.3 V ±0.15 V or 3.5 V/3.7 V/3.9 V/4.1 V/4.3 V ±0.2 V (selectable by software) | | | | | | | | | |
| | Clock monitor | Provided (monitor by Ring-OSC) | | | | | | | | | |
| | WDT1 | Provided | | | | | | | | | |
| | WDT2 | Provided | | | | | | | | | |
| ROM correction | | 4 | | | | | | | | None | |
| Regulator | | None | | Provided | | | | | | | |
| Standby function | | HALT/IDLE/STOP/sub-IDLE mode | | | | | | | | | |
| Operating ambient temperature | | $T_A$ = –40 to +85°C | | | | | | | | | |

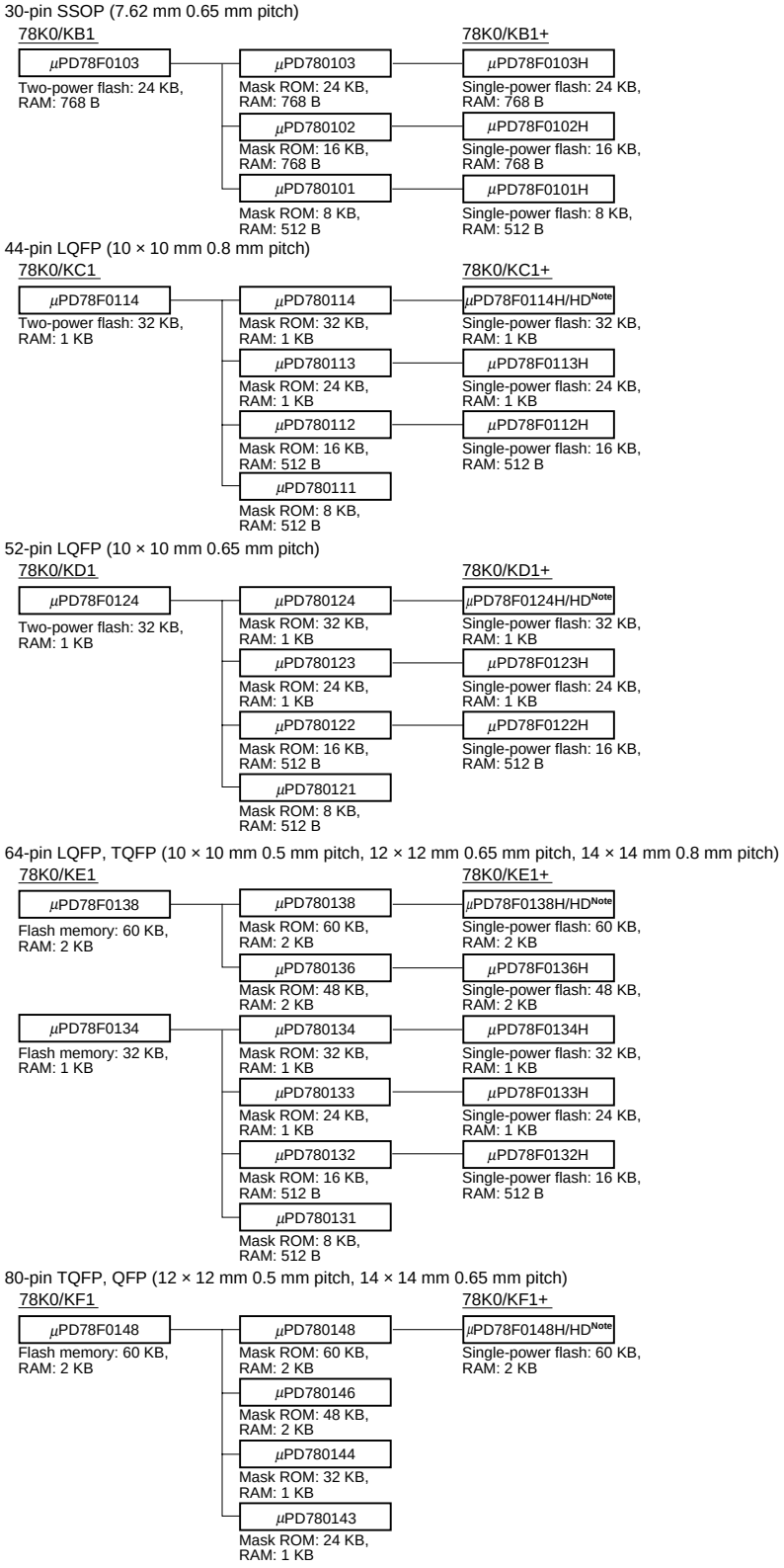
**Note** Only in products with an $I^2C$ bus (Y products). For the product name, refer to each user's manual.

The function list of the V850ES/Kx1 is shown below.

| Product Name | | V850ES/KE1 | | V850ES/KF1 | | | | | V850ES/KG1 | | | | | V850ES/KJ1 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Number of pins | | 64 pins | | 80 pins | | | | | 100 pins | | | | | 144 pins | | |
| Internal memory (KB) | Mask ROM | 128 | – | 64/96 | 128 | – | 256 | – | 64/96 | 128 | – | 256 | – | 96/128 | – | – |
| | Flash memory | – | 128 | – | – | 128 | – | 256 | – | – | 128 | – | 256 | – | 128 | 256 |
| | RAM | 4 | | 4 | 6 | | 12 | | 4 | 6 | | 16 | | 6 | | 16 |
| Supply voltage | | 2.7 to 5.5 V | | | | | | | | | | | | | | |
| Minimum instruction execution time | | 50 ns @20 MHz | | | | | | | | | | | | | | |
| Clock | X1 input | 2 to 10 MHz | | | | | | | | | | | | | | |
| | Subclock | 32.768 kHz | | | | | | | | | | | | | | |
| | Ring-OSC | – | | | | | | | | | | | | | | |
| Port | CMOS input | 8 | | 8 | | | | | 8 | | | | | 16 | | |
| | CMOS I/O | 43 | | 59 | | | | | 76 | | | | | 112 | | |
| | N-ch open-drain I/O | 2 | | 2 | | | | | 4 | | | | | 6 | | |
| Timer | 16-bit (TMP) | 1 ch | | – | | | 1 ch | | – | | | 1 ch | | – | | 1 ch |
| | 16-bit (TM0) | 1 ch | | 2 ch | | | | | 4 ch | | | | | 6 ch | | |
| | 8-bit (TM5) | 2 ch | | 2 ch | | | | | 2 ch | | | | | 2 ch | | |
| | 8-bit (TMH) | 2 ch | | 2 ch | | | | | 2 ch | | | | | 2 ch | | |
| | Interval timer | 1 ch | | 1 ch | | | | | 1 ch | | | | | 1 ch | | |
| | Watch | 1 ch | | 1 ch | | | | | 1 ch | | | | | 1 ch | | |
| | WDT1 | 1 ch | | 1 ch | | | | | 1 ch | | | | | 1 ch | | |
| | WDT2 | 1 ch | | 1 ch | | | | | 1 ch | | | | | 1 ch | | |
| RTO | | 6 bits × 1 ch | | 6 bits × 1 ch | | | | | 6 bits × 1 ch | | | | | 6 bits × 2 ch | | |
| Serial interface | CSI | 2 ch | | 2 ch | | | | | 2 ch | | | | | 3 ch | | |
| | Automatic transmit/receive 3-wire CSI | – | | 1 ch | | | | | 2 ch | | | | | 2 ch | | |
| | UART | 2 ch | | 2 ch | | | | | 2 ch | | | | | 3 ch | | |
| | UART supporting LIN-bus | – | | – | | | | | – | | | | | – | | |
| | $I^2C$[Note] | 1 ch | | 1 ch | | | | | 1 ch | | | | | 2 ch | | |
| External bus | Address space | – | | 128 KB | | | | | 3 MB | | | | | 15 MB | | |
| | Address bus | – | | 16 bits | | | | | 22 bits | | | | | 24 bits | | |
| | Mode | – | | Multiplex only | | | | | Multiplex/separate | | | | | | | |
| DMA controller | | – | | – | | | | | – | | | | | – | | |
| 10-bit A/D converter | | 8 ch | | 8 ch | | | | | 8 ch | | | | | 16 ch | | |
| 8-bit D/A converter | | – | | – | | | | | 2 ch | | | | | 2 ch | | |
| Interrupt | External | 8 | | 8 | | | | | 8 | | | | | 8 | | |
| | Internal | 26 | | 26 | | | 29 | | 31 | | | 34 | | 40 | | 43 |
| Key return input | | 8 ch | | 8 ch | | | | | 8 ch | | | | | 8 ch | | |
| Reset | $\overline{\text{RESET}}$ pin | Provided | | | | | | | | | | | | | | |
| | POC | None | | | | | | | | | | | | | | |
| | LVI | None | | | | | | | | | | | | | | |
| | Clock monitor | None | | | | | | | | | | | | | | |
| | WDT1 | Provided | | | | | | | | | | | | | | |
| | WDT2 | Provided | | | | | | | | | | | | | | |
| ROM correction | | 4 | | | | | | | | | | | | | | |
| Regulator | | None | | Provided | | | | | | | | | | | | |
| Standby function | | HALT/IDLE/STOP/sub-IDLE mode | | | | | | | | | | | | | | |
| Operating ambient temperature | | $T_A$ = –40 to +85°C | | | | | | | | | | | | | | |

**Note** Only in products with an $I^2C$ bus (Y products). For the product name, refer to each user's manual.

### 1.1.2 78K0/Kx1+, 78K0/Kx1 products lineup




**Note** Products with an on-chip debug function

The function list of the 78K0/Kx1+ is shown below.

| Item \ Product Name | | 78K0/KB1+ | | 78K0/KC1+ | | 78K0/KD1+ | | 78K0/KE1+ | | | 78K0/KF1+ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Number of pins | | 30 pins | | 44 pins | | 52 pins | | 64 pins | | | 80 pins |
| Internal memory (byte) | Flash memory | 8 K | 16 K/24 K | 16 K | 24 K/32 K | 16 K | 24 K/32 K | 16 K | 24 K/32 K | 48 K/60 K | 60 K |
| | RAM | 512 | 768 | 512 | 1 K | 512 | 1 K | 512 | 1 K | 2 K | 2 K |
| Supply voltage | | $V_{DD}$ = 2.7 to 5.5 V | | | | | | | | | |
| Minimum instruction execution time | | 0.125 μs (16 MHz, when $V_{DD}$ = 4.0 to 5.5 V)<br>0.24 μs (8.38 MHz, when $V_{DD}$ = 3.3 to 5.5 V)<br>0.4 μs (5 MHz, when $V_{DD}$ = 2.7 to 5.5 V) | | | | | | | | | |
| Clock | X1 input | 2 to 16 MHz | | | | | | | | | |
| | RC | 3 to 4 MHz ($V_{DD}$ = 2.7 to 5.5 V) | | | | – | | | | | |
| | Sub | – | | 32.768 kHz | | | | | | | |
| | Ring-OSC | 240 kHz (TYP.) | | | | | | | | | |
| Port | CMOS I/O | 17 | | 19 | | 26 | | 38 | | | 54 |
| | CMOS input | 4 | | 8 | | | | | | | |
| | CMOS output | 1 | | | | | | | | | |
| | N-ch open-drain I/O | – | | 4 | | | | | | | |
| Timer | 16-bit (TM0) | 1 ch | | | | 2 ch | | | | | |
| | 8-bit (TM5) | 2 ch | | | | | | | | | |
| | 8-bit (TMH) | 1 ch | | 2 ch | | | | | | | |
| | Watch | – | | 1 ch | | | | | | | |
| | WDT | 1 ch | | | | | | | | | |
| Serial interface | 3-wire CSI[Note] | 1 ch | | | | 2 ch | | | | | |
| | Automatic transmit/receive 3-wire CSI | – | | | | | | | | | 1 ch |
| | UART[Note] | – | 1 ch | | | | | | | | |
| | UART supporting LIN-bus | 1 ch | | | | | | | | | |
| 10-bit A/D converter | | 4 ch | | 8 ch | | | | | | | |
| Interrupt | External | 6 | | 7 | | 8 | | 9 | | | 9 |
| | Internal | 11 | 12 | 15 | | 15 | | 16 | 19 | | 20 |
| Key return input | | – | | 4 ch | | 8 ch | | | | | |
| Reset | $\overline{\text{RESET}}$ pin | Provided | | | | | | | | | |
| | POC | 2.1 V ±0.1 V (detection voltage fixed) | | | | | | | | | |
| | LVI | 2.35 V/2.6 V/2.85 V/3.1 V/3.3 V ±0.15 V/3.5 V/3.7 V/3.9 V/4.1 V/4.3 V ±0.2 V (selectable by software) | | | | | | | | | |
| | Clock monitor | Provided | | | | | | | | | |
| | WDT | Provided | | | | | | | | | |
| Clock output/buzzer output | | – | | | | Clock output only | | Provided | | | |
| External bus interface | | – | | | | | | | | | Provided |
| Multiplier/divider | | – | | | | | | 16 bits × 16 bits, 32 bits ÷ 16 bits | | | |
| ROM correction | | – | | | | | | | | Provided | – |
| Self programming function | | Provided | | | | | | | | | |
| On-chip debug function | | Function provided only in μPD78F0114HD, 78F0124HD, 78F0138HD, and 78F0148HD | | | | | | | | | |
| Standby function | | HALT/STOP mode | | | | | | | | | |
| Operating ambient temperature | | −40 to +85°C | | | | | | | | | |

**Note** If the pin is an alternate-function pin, either function is selected for use.

The function list of the 78K0/Kx1 is shown below.

| Item \ Product Name | | 78K0/KB1 | | | 78K0/KC1 | | | 78K0/KD1 | | | 78K0/KE1 | | | | | 78K0/KF1 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Number of pins | | 30 pins | | | 44 pins | | | 52 pins | | | 64 pins | | | | | 80 pins | | |
| Internal memory (byte) | Mask ROM | 8 K | 16 K/ 24 K | – | 8 K/ 16 K | 24 K/ 32 K | – | 8 K/ 16 K | 24 K/ 32 K | – | 8 K/ 16 K | 24 K/ 32 K | – | 48 K/ 60 K | – | 24 K/ 32 K | 48 K/ 60 K | – |
| | Flash memory | – | | 24 K | – | | 32 K | – | | 32 K | – | | 32 K | – | 60 K | – | | 60 K |
| | RAM | 512 | 768 | | 512 | 1 K | | 512 | 1 K | | 512 | 1 K | | 2 K | | 1 K | 2 K | |
| Supply voltage | | $V_{DD}$ = 2.7 to 5.5 V | | | | | | | | | | | | | | | | |
| Minimum instruction execution time | | 0.2 $\mu$s (10 MHz, when $V_{DD}$ = 4.0 to 5.5 V)<br>0.24 $\mu$s (8.38 MHz, when $V_{DD}$ = 3.3 to 5.5 V)<br>0.4 $\mu$s (5 MHz, when $V_{DD}$ = 2.7 to 5.5 V) | | | <REGC pin connected to $V_{DD}$><br>0.2 $\mu$s (10 MHz, when $V_{DD}$ = 4.0 to 5.5 V)<br>0.24 $\mu$s (8.38 MHz, when $V_{DD}$ = 3.3 to 5.5 V)<br>0.4 $\mu$s (5 MHz, when $V_{DD}$ = 2.7 to 5.5 V) | | | | | | | | | | | |
| Clock | X1 input | 2 to 10 MHz | | | | | | | | | | | | | | | | |
| | Sub | – | | | 32.768 kHz | | | | | | | | | | | | | |
| | RC | – | | | | | | | | | | | | | | | | |
| | Ring-OSC | 240 kHz (TYP.) | | | | | | | | | | | | | | | | |
| Port | CMOS I/O | 17 | | | 19 | | | 26 | | | 38 | | | | | 54 | | |
| | CMOS input | 4 | | | 8 | | | | | | | | | | | | | |
| | CMOS output | 1 | | | | | | | | | | | | | | | | |
| | N-ch open-drain I/O | – | | | 4 | | | | | | | | | | | | | |
| Timer | 16-bit (TM0) | 1 ch | | | | | | | | | 2 ch | | | | | 1 ch | 2 ch | |
| | 8-bit (TM5) | 1 ch | | | 2 ch | | | | | | | | | | | | | |
| | 8-bit (TMH) | 2 ch | | | | | | | | | | | | | | | | |
| | Watch | – | | | 1 ch | | | | | | | | | | | | | |
| | WDT | 1 ch | | | | | | | | | | | | | | | | |
| Serial interface | 3-wire CSI[Note] | 1 ch | | | | | | | | | 2 ch | | | | | 1 ch | 2 ch | |
| | Automatic transmit/ receive 3-wire CSI | – | | | | | | | | | | | | | | 1 ch | | |
| | UART[Note] | – | | | 1 ch | | | | | | | | | | | | | |
| | UART supporting LIN-bus | 1 ch | | | | | | | | | | | | | | | | |
| 10-bit A/D converter | | 4 ch | | | 8 ch | | | | | | | | | | | | | |
| Interrupt | External | 6 | | | 7 | | | 8 | | | 9 | | | | | 9 | | |
| | Internal | 11 | 12 | | 15 | | | | | | 16 | 19 | | | | 17 | 20 | |
| Key return input | | – | | | 4 ch | | | 8 ch | | | | | | | | | | |
| Reset | $\overline{\text{RESET}}$ pin | Provided | | | | | | | | | | | | | | | | |
| | POC | 2.85 V ±0.15 V/3.5 V ±0.20 V (selectable by a mask option) | | | | | | | | | | | | | | | | |
| | LVI | 3.1 V/3.3 V ±0.15 V/3.5 V/3.7 V/3.9 V/4.1 V/4.3 V ±0.2 V (selectable by software) | | | | | | | | | | | | | | | | |
| | Clock monitor | Provided | | | | | | | | | | | | | | | | |
| | WDT | Provided | | | | | | | | | | | | | | | | |
| Clock output/buzzer output | | – | | | | | | Clock output | | | Provided | | | | | | | |
| Multiplier/divider | | – | | | | | | | | | 16 bits × 16 bits, 32 bits ÷ 16 bits | | | | | | | |
| ROM correction | | – | | | | | | | | | | | | Provided | | – | | |
| Standby function | | HALT/STOP mode | | | | | | | | | | | | | | | | |
| Operating ambient temperature | | Standard products, special grade (A) products: –40 to +85°C<br>Special grade (A1) products: –40 to +110°C (mask ROM version), –40 to +105°C (flash memory version)<br>Special grade (A2) products: –40 to +125°C (mask ROM version) | | | | | | | | | | | | | | | | |

**Note** If the pin is an alternate-function pin, either function is selected for use.

## 1.2 Features

- Minimum instruction execution time: 50 ns (operation at main clock ($f_{XX}$) = 20 MHz)
- General-purpose registers: 32 bits × 32 registers
- CPU features: Signed multiplication (16 × 16 → 32): 1 to 2 clocks
  (Instructions without creating register hazards can be continuously executed in parallel)
  Saturated operations (overflow and underflow detection functions are included)
  32-bit shift instruction: 1 clock
  Bit manipulation instructions
  Load/store instructions with long/short format
- Memory space: 64 MB of linear address space
  Memory block division function: 64 KB, 64 KB (Total of 2 blocks)
  - Internal memory
    μPD703308, 703308Y (Mask ROM: 256 KB/RAM: 12 KB)
    μPD70F3306, 70F3306Y (Single-power flash memory: 128 KB/RAM: 6 KB)
    μPD70F3308, 70F3308Y (Single-power flash memory: 256 KB/RAM: 12 KB)
  - External bus interface
    Multiplex bus output
    8-/16-bit data bus sizing function
    Wait function
    - Programmable wait function
    - External wait function

    Idle state function
    Bus hold function
- Interrupts and exceptions
  Non-maskable interrupts: 3 sources
  Maskable interrupts: 35 sources (μPD703308, 70F3306, 70F3308)
  36 sources (μPD703308Y, 70F3306Y, 70F3308Y)
  Software exceptions: 32 sources
  Exception trap: 1 source
- I/O lines: Total: 67
- Key interrupt function
- Timer function
  16-bit timer/event counter P: 1 channel
  16-bit timer/event counter 0: 2 channels
  8-bit timer/event counter 5: 2 channels
  8-bit timer H: 2 channels
  8-bit interval timer BRG: 1 channel
  Watch timer/interval timer: 1 channel
  Watchdog timers
  Watchdog timer 1 (also usable as oscillation stabilization timer): 1 channel
  Watchdog timer 2: 1 channel

- Serial interface

  | | |
  |---|---|
  | Asynchronous serial interface (UART) (supporting LIN): | 1 channel |
  | Asynchronous serial interface (UART): | 1 channel |
  | 3-wire serial I/O (CSI0): | 2 channels |
  | 3-wire serial I/O (with automatic transmit/receive function) (CSIA): | 1 channel |
  | $I^2C$ bus interface ($I^2C$): | 1 channel |
  | ($\mu$PD703308Y, 70F3306Y, 70F3308Y) | |

- A/D converter: 10-bit resolution × 8 channels
- Real-time output port: 6 bits × 1 channel
- Standby functions: HALT/IDLE/STOP modes, subclock/sub-IDLE modes, ring clock operation/ring HALT modes
- ROM correction: 4 correction addresses specifiable
- Clock generator

  Main clock oscillation ($f_X$)/subclock oscillation ($f_{XT}$)/Ring-OSC ($f_R$)
  CPU clock ($f_{CPU}$) 7 steps ($f_{XX}$, $f_{XX}/2$, $f_{XX}/4$, $f_{XX}/8$, $f_{XX}/16$, $f_{XX}/32$, $f_{XT}$)
  Clock-through mode/PLL mode selectable
- Ring-OSC: 240 kHz (TYP.)
- Reset
  - Reset by $\overline{\text{RESET}}$ pin
  - Reset by overflow of watchdog timer 1 (WDTRES1)
  - Reset by overflow of watchdog timer 2 (WDTRES2)
  - Reset by low-voltage detector (LVIRES)
  - Reset by power-on-clear (POCRES)
  - Reset by clock monitor (CLMRES)
  - Reset output function (P00/TOH0 pin)
- Low-voltage detector (LVI)
- Power-on-clear (POC) circuit
- Clock monitor (CLM) circuit
- Package: 80-pin plastic TQFP (fine pitch) (12 × 12)
  80-pin plastic QFP (14 × 14)

## 1.3 Applications

- Automotive
  - System control of body electrical system (power windows, keyless entry reception, etc.)
  - Submicrocontroller of control system
- Home audio, car audio
- AV equipment
- PC peripheral devices (keyboards, etc.)
- Household appliances
  - Outdoor units of air conditioners
  - Microwave ovens, rice cookers
- Industrial devices
  - Pumps
  - Vending machines
  - FA

## 1.4 Ordering Information

| Part Number | Package | Quality Grade |
|---|---|---|
| μPD703308GK-×××-9EU | 80-pin plastic TQFP (fine pitch) (12 × 12) | Standard |
| μPD703308YGK-×××-9EU | 80-pin plastic TQFP (fine pitch) (12 × 12) | Standard |
| μPD703308GC-×××-8BT | 80-pin plastic QFP (14 × 14) | Standard |
| μPD703308YGC-×××-8BT | 80-pin plastic QFP (14 × 14) | Standard |
| μPD70F3306GK-9EU | 80-pin plastic TQFP (fine pitch) (12 × 12) | Standard |
| μPD70F3306YGK-9EU | 80-pin plastic TQFP (fine pitch) (12 × 12) | Standard |
| μPD70F3306GC-8BT | 80-pin plastic QFP (14 × 14) | Standard |
| μPD70F3306YGC-8BT | 80-pin plastic QFP (14 × 14) | Standard |
| μPD70F3308GK-9EU | 80-pin plastic TQFP (fine pitch) (12 × 12) | Standard |
| μPD70F3308YGK-9EU | 80-pin plastic TQFP (fine pitch) (12 × 12) | Standard |
| μPD70F3308GC-8BT | 80-pin plastic QFP (14 × 14) | Standard |
| μPD70F3308YGC-8BT | 80-pin plastic QFP (14 × 14) | Standard |

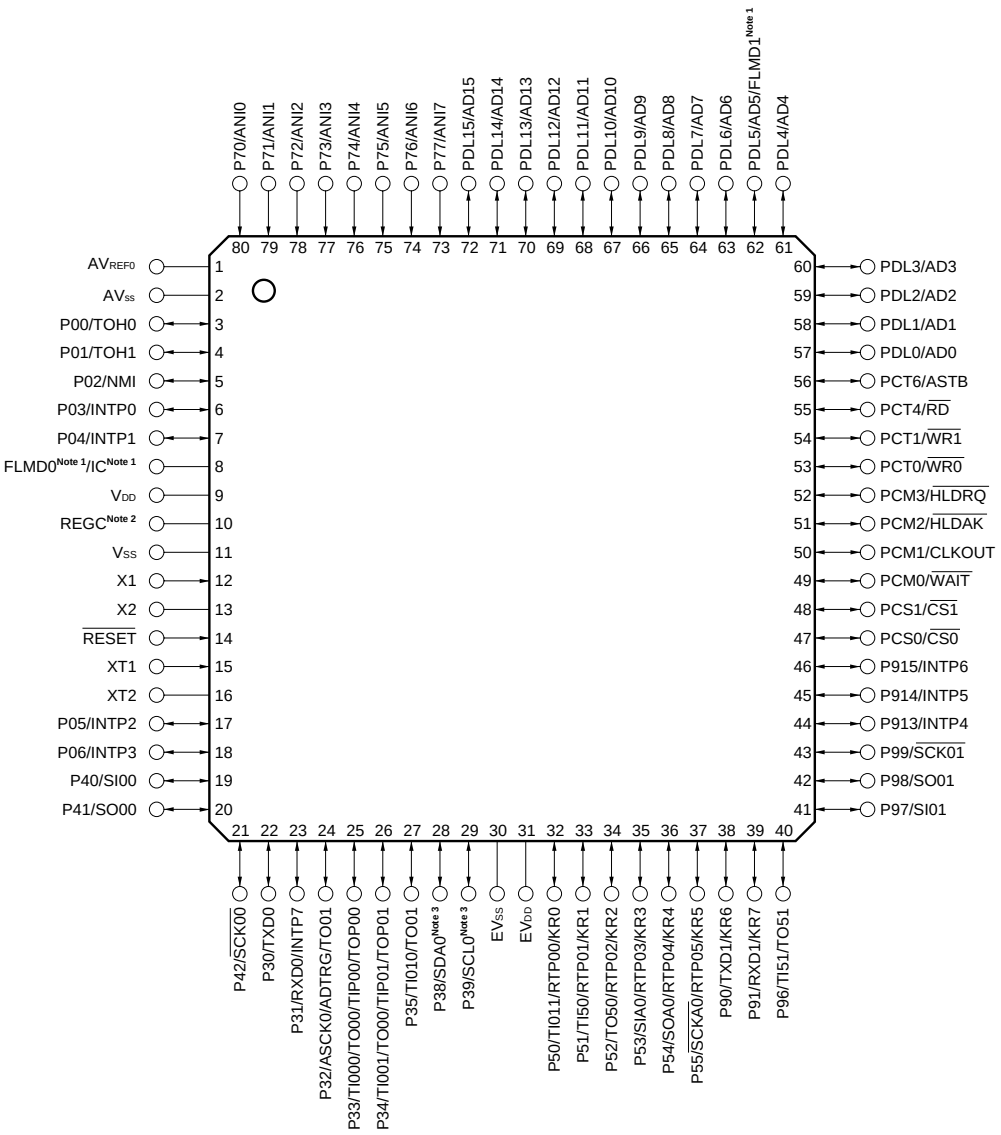
**Remark** ××× indicates ROM code suffix.

## 1.5 Pin Configuration (Top View)

80-pin plastic QFP (14 × 14)

80-pin plastic TQFP (fine pitch) (12 × 12)

| | | |
|---|---|---|
| μPD703308GC-×××-8BT | μPD70F3306GC-8BT | μPD70F3308GC-8BT |
| μPD703308YGC-×××-8BT | μPD70F3306YGC-8BT | μPD70F3308YGC-8BT |
| μPD703308GK-×××-9EU | μPD70F3306GK-9EU | μPD70F3308GK-9EU |
| μPD703308YGK-×××-9EU | μPD70F3306YGK-9EU | μPD70F3308YGK-9EU |




**Notes 1.** IC pin: Connect directly to $V_{SS}$ (μPD703308, 703308Y).

FLMD0 pin: Connect to $V_{SS}$ in normal operation mode (μPD70F3306, 70F3306Y, 70F3308, 70F3308Y).

FLMD1 pin: Used only in the μPD70F3306, 70F3306Y, 70F3308, and 70F3308Y.

**2.** When using a regulator, connect the REGC pin to $V_{SS}$ via a 10 μF capacitor.
When not using a regulator, connect the REGC pin directly to $V_{DD}$.

**3.** The SCL0 and SDA0 pins can be used only in the μPD703308Y, 70F3306Y, and 70F3308Y.

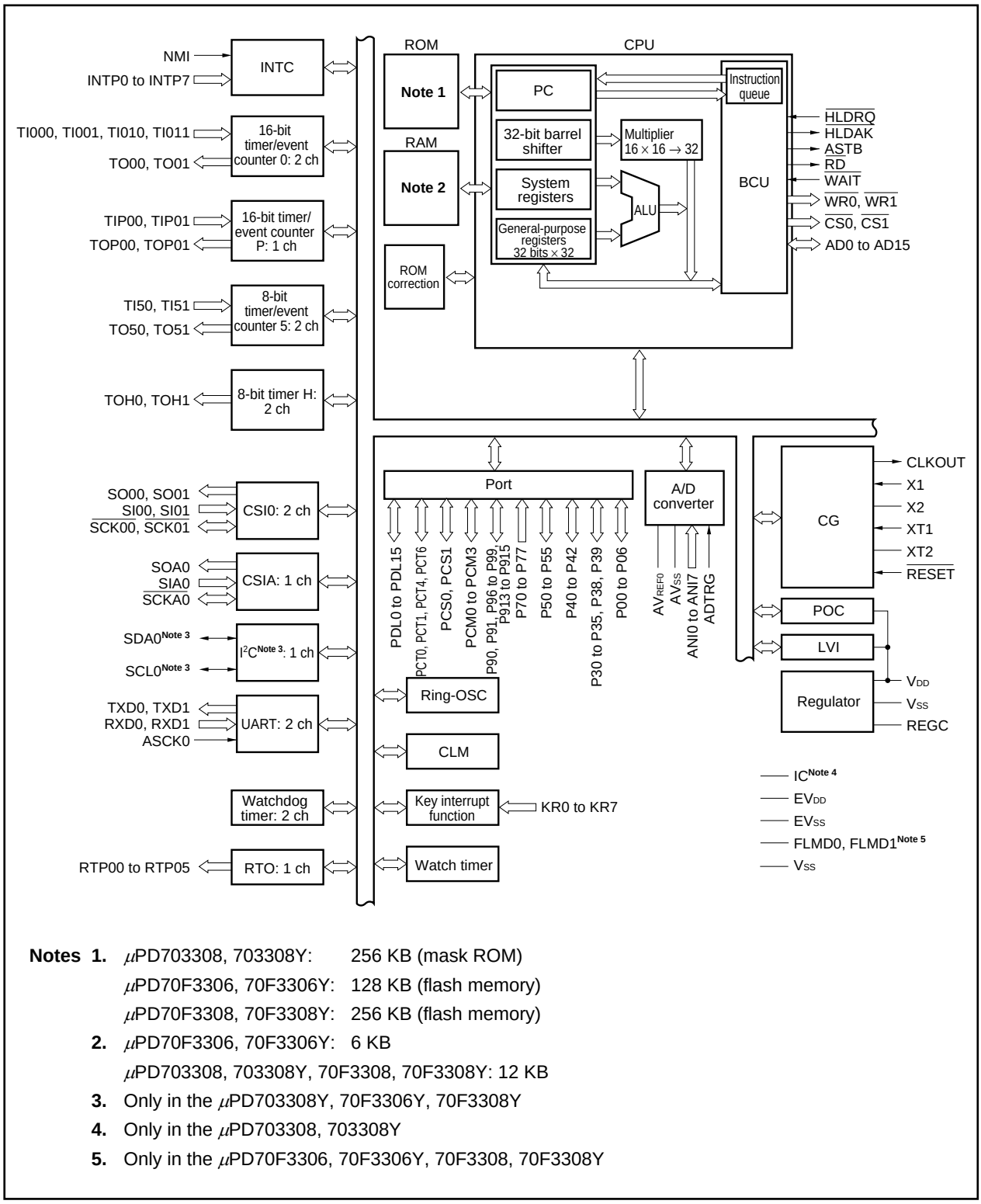
**Caution Make $EV_{DD}$ the same potential as $V_{DD}$.**

**Pin identification**

| | |
|---|---|
| AD0 to AD15: | Address/data bus |
| ADTRG: | A/D trigger input |
| ANI0 to ANI7: | Analog input |
| ASCK0: | Asynchronous serial clock |
| ASTB: | Address strobe |
| $AV_{REF0}$: | Analog reference voltage |
| $AV_{SS}$: | Ground for analog |
| CLKOUT: | Clock output |
| $\overline{CS0}$, $\overline{CS1}$: | Chip select |
| $EV_{DD}$: | Power supply for port |
| $EV_{SS}$: | Ground for port |
| FLMD0, FLMD1: | Flash programming mode |
| $\overline{HLDAK}$: | Hold acknowledge |
| $\overline{HLDRQ}$: | Hold request |
| IC: | Internally connected |
| INTP0 to INTP7: | External interrupt input |
| KR0 to KR7: | Key return |
| NMI: | Non-maskable interrupt request |
| P00 to P06: | Port 0 |
| P30 to P35, P38, P39: | Port 3 |
| P40 to P42: | Port 4 |
| P50 to P55: | Port 5 |
| P70 to P77: | Port 7 |
| P90, P91, P96 to P99, P913 to P915: | Port 9 |
| PCM0 to PCM3: | Port CM |
| PCS0, PCS1: | Port CS |
| PCT0, PCT1, PCT4, PCT6: | Port CT |
| PDL0 to PDL15: | Port DL |
| $\overline{RD}$: | Read strobe |
| REGC: | Regulator control |
| $\overline{RESET}$: | Reset |
| RTP00 to RTP05: | Real-time output port |
| RXD0, RXD1: | Receive data |
| $\overline{SCK00}$, $\overline{SCK01}$, $\overline{SCKA0}$: | Serial clock |
| SCL0: | Serial clock |
| SDA0: | Serial data |
| SI00, SI01, SIA0: | Serial input |
| SO00, SO01, SOA0: | Serial output |
| TI000, TI001, TI010, TI011, TI50, TI51, TIP00, TIP01: | Timer input |
| TO00, TO01, TO50, TO51, TOH0, TOH1, TOP00, TOP01: | Timer output |
| TXD0, TXD1: | Transmit data |
| $V_{DD}$: | Power supply |
| $V_{SS}$: | Ground |
| $\overline{WAIT}$: | Wait |
| $\overline{WR0}$: | Lower byte write strobe |
| $\overline{WR1}$: | Upper byte write strobe |
| X1, X2: | Crystal for main clock |
| XT1, XT2: | Crystal for subclock |

## 1.6 Function Block Configuration

### (1) Internal block diagram




**Notes 1.** $\mu$PD703308, 703308Y: 256 KB (mask ROM)
$\mu$PD70F3306, 70F3306Y: 128 KB (flash memory)
$\mu$PD70F3308, 70F3308Y: 256 KB (flash memory)

**2.** $\mu$PD70F3306, 70F3306Y: 6 KB
$\mu$PD703308, 703308Y, 70F3308, 70F3308Y: 12 KB

**3.** Only in the $\mu$PD703308Y, 70F3306Y, 70F3308Y

**4.** Only in the $\mu$PD703308, 703308Y

**5.** Only in the $\mu$PD70F3306, 70F3306Y, 70F3308, 70F3308Y

### (2) Internal units

#### (a) CPU

The CPU uses five-stage pipeline control to enable single-clock execution of address calculations, arithmetic logic operations, data transfers, and almost all other types of instruction processing.

Other dedicated on-chip hardware, such as a multiplier (16 bits × 16 bits → 32 bits) and a barrel shifter (32 bits) help accelerate complex processing.

#### (b) Bus control unit (BCU)

The BCU starts a required external bus cycle based on the physical address obtained by the CPU.

When an instruction is fetched from external memory area and the CPU does not send a bus cycle start request, the BCU generates a prefetch address and prefetches the instruction code. The prefetched instruction code is stored in an internal instruction queue.

#### (c) ROM

This consists of a 256 KB or 128 KB mask ROM or flash memory mapped to the address spaces from 0000000H to 003FFFFH or 0000000H to 001FFFFH, respectively.

ROM can be accessed by the CPU in one clock cycle during instruction fetch.

#### (d) RAM

This consists of a 12 KB or 6 KB RAM mapped to the address spaces from 3FFC000H to 3FFEFFFH or 3FFD800H to 3FFEFFFH.

RAM can be accessed by the CPU in one clock cycle during data access.

#### (e) Interrupt controller (INTC)

This controller handles hardware interrupt requests (NMI, INTP0 to INTP7) from on-chip peripheral hardware and external hardware. Eight levels of interrupt priorities can be specified for these interrupt requests, and multiplexed servicing control can be performed.

#### (f) Clock generator (CG)

A main clock oscillator and subclock oscillator are provided and generate the main clock oscillation frequency ($f_X$) and subclock frequency ($f_{XT}$), respectively.

There are two modes: In the clock-through mode, $f_X$ is used as the main clock frequency ($f_{XX}$) as is. In the PLL mode, $f_X$ is used multiplied by 4.

The CPU clock frequency ($f_{CPU}$) can be selected from among $f_{XX}$, $f_{XX}/2$, $f_{XX}/4$, $f_{XX}/8$, $f_{XX}/16$, $f_{XX}/32$, and $f_{XT}$.

#### (g) Timer/counter

Two 16-bit timer/event counter 0 channels, one 16-bit timer/event counter P channel, and two 8-bit timer/event counter 5 channels are incorporated, enabling measurement of pulse intervals and frequency as well as programmable pulse output.

Two 8-bit timer/event counter 5 channels can be connected in cascade to configure a 16-bit timer.

Two 8-bit timer H channels enabling programmable pulse output are provided on chip.

**(h) Watch timer**

This timer counts the reference time (0.5 seconds) for counting the clock from the subclock (32.768 kHz) or $f_{BRG}$ (32.768 kHz) from the clock generator. At the same time, the watch timer can be used as an interval timer.

**(i) Watchdog timer**

Two watchdog timer channels are provided on chip to detect program loops and system abnormalities. Watchdog timer 1 can be used as an interval timer. When used as a watchdog timer, it generates a non-maskable interrupt request signal (INTWDT1) or system reset signal (WDTRES1) after an overflow occurs. When used as an interval timer, it generates a maskable interrupt request signal (INTWDTM1) after an overflow occurs.

Watchdog timer 2 operates by default following reset release.

It generates a non-maskable interrupt request signal (INTWDT2) or system reset signal (WDTRES2) after an overflow occurs.

**(j) Serial interface (SIO)**

The V850ES/KF1+ includes four kinds of serial interfaces: an asynchronous serial interface (UARTn) (supporting 1-channel LIN), a clocked serial interface (CSI0n), a clocked serial interface with an automatic transmit/receive function (CSIA0), and an $I^2C$ bus interface ($I^2C0$), and can simultaneously use up to six channels.

For UARTn, data is transferred via the TXDn and RXDn pins.

For CSI0n, data is transferred via the SO0n, SI0n, and $\overline{SCK0n}$ pins.

For CSIA0, data is transferred via the SOA0, SIA0, and $\overline{SCKA0}$ pins.

For $I^2C0$, data is transferred via the SDA0 and SCL0 pins.

$I^2C0$ is provided only in the $\mu$PD703308Y, 70F3306Y, and 70F3308Y.

**Remark** n = 0, 1

**(k) A/D converter**

This high-speed, high-resolution 10-bit A/D converter includes 8 analog input pins. Conversion is performed using the successive approximation method.

**(l) ROM correction**

This function is used to replace part of a program in the mask ROM with that contained in the internal RAM. Up to four correction addresses can be specified.

**(m) Key interrupt function**

A key interrupt request signal (INTKR) can be generated by inputting a falling edge to the eight key input pins.

**(n) Real-time output function**

This function transfers 6-bit data set beforehand to output latches upon occurrence of a timer compare register match signal.

A 1-channel 6-bit data real-time output function is provided on chip.

**(o) Clock monitor**

The clock monitor samples the main clock ($f_X$) using the on-chip Ring-OSC clock ($f_R$), and generates a reset request signal when the oscillation of the main clock is stopped.

**(p) Low-voltage detector (LVI)**

The low-voltage detector compares the supply voltage ($V_{DD}$) and detection voltage ($V_{LVI}$), and generates an internal interrupt signal or internal reset signal when $V_{DD} < V_{LVI}$.

**(q) Power-on-clear (POC) circuit**

The power-on-clear circuit generates an internal reset signal at power on.

The power-on-clear circuit compares the supply voltage ($V_{DD}$) and detection voltage ($V_{POC}$), and generates an internal reset signal when $V_{DD} < V_{POC}$.

**(r) Ports**

As shown below, the following ports have general-purpose port functions and control pin functions.

| Port | I/O | Alternate Function |
|---|---|---|
| P0 | 7-bit I/O | NMI, external interrupt, timer output |
| P3 | 8-bit I/O | Serial interface, timer I/O, external interrupt, A/D converter trigger |
| P4 | 3-bit I/O | Serial interface |
| P5 | 6-bit I/O | Serial interface, timer I/O, key interrupt function, real-time output function |
| P7 | 8-bit input | A/D converter analog input |
| P9 | 9-bit I/O | Serial interface, timer I/O, external interrupt, key interrupt function |
| PCM | 4-bit I/O | External bus control signal |
| PCS | 2-bit I/O | Chip select output |
| PCT | 4-bit I/O | External bus control signal |
| PDL | 16-bit I/O | External address/data bus |

## 1.7 Overview of Functions

| Part Number | | μPD703308/ μPD703308Y | μPD70F3306/ μPD70F3306Y | μPD70F3308/ μPD70F3308Y |
|---|---|---|---|---|
| Internal memory | ROM | 256 KB | 128 KB (single-power flash memory) | 256 KB (single-power flash memory) |
| | High-speed RAM | 12 KB | 6 KB | 12 KB |
| Buffer RAM | | 32 bytes | | |
| Memory space | Logical space | 64 MB | | |
| | External memory area | 128 KB | | |
| External bus interface | | Address/data bus: 16<br>Multiplex bus mode | | |
| General-purpose registers | | 32 bits × 32 registers | | |
| Main clock (oscillation frequency) | | Ceramic/crystal/external clock | | |
| | When PLL not used | | 2 to 8 MHz[Note 1]: 2.7 to 5.5 V | |
| | When PLL used | REGC pin connected directly to $V_{DD}$ | 2 to 5 MHz: 4.5 to 5.5 V, 2 MHz: 2.7 to 5.5 V | |
| | | 10 μF capacitor connected to REGC pin | 2 MHz: 4.0 to 5.5 V | |
| Subclock (oscillation frequency) | | Crystal/external clock<br>(32.768 kHz) | | |
| Minimum instruction execution time | | 50 ns (When main clock operated at ($f_{XX}$) = 20 MHz) | | |
| DSP function | | 32 × 32 = 64: 200 to 250 ns (at 20 MHz)<br>32 × 32 + 32 = 32: 300 ns (at 20 MHz)<br>16 × 16 = 32: 50 to 100 ns (at 20 MHz)<br>16 × 16 + 32 = 32: 150 ns (at 20 MHz) | | |
| I/O ports | | 67<br>• Input: 8<br>• I/O: 59 (among these, N-ch open-drain output selectable: 6, fixed to N-ch open-drain output: 2) | | |
| Timer | | 16-bit timer/event counter P: 1 channel<br>16-bit timer/event counter 0: 2 channels<br>8-bit timer/event counter 5: 2 channels<br>(16-bit timer/event counter: usable as 1 channel)<br>8-bit timer H: 2 channels<br>Watchdog timer: 2 channels<br>Watch timer: 1 channel<br>8-bit interval timer: 1 channel | | |
| Real-time output port | | 4 bits × 1, 2 bits × 1, or 6 bits × 1 | | |
| A/D converter | | 10-bit resolution × 8 channels | | |
| Serial interface | | CSI: 2 channels<br>CSIA (with automatic transmit/receive function): 1 channel<br>UART (supporting LIN): 1 channel<br>UART: 1 channel<br>$I^2C$ bus: 1 channel[Note 2]<br>Dedicated baud rate generator: 2 channels | | |
| Interrupt sources | | External: 10 (10)[Note 3], internal: 30[Note 2]/29 | | |
| Power save function | | STOP/IDLE/HALT/sub-IDLE mode | | |
| Operating supply voltage | | 4.5 to 5.5 V (at 20 MHz)/2.7 to 5.5 V (at 8 MHz) | | |
| Package | | 80-pin plastic TQFP (fine pitch) (12 × 12 mm)<br>80-pin plastic QFP (14 × 14 mm) | | |

**Notes 1.** These values may change after evaluation.

**2.** Only in the μPD703308Y, 70F3306Y, 70F3308Y

**3.** The figure in parentheses indicates the number of external interrupts for which STOP mode can be released.

# CHAPTER 2 PIN FUNCTIONS

The names and functions of the pins of the V850ES/KF1+ are described below, divided into port pins and non-port pins.

The pin I/O buffer power supplies are divided into two systems; $AV_{REF0}$ and $EV_{DD}$. The relationship between these power supplies and the pins is shown below.

**Table 2-1. Pin I/O Buffer Power Supplies**

| Power Supply | Corresponding Pins |
|---|---|
| $AV_{REF0}$ | Port 7 |
| $EV_{DD}$ | $\overline{RESET}$, ports 0, 3 to 5, 9, CM, CS, CT, DL |

## 2.1 List of Pin Functions

### (1) Port pins

(1/3)

| Pin Name | Pin No. | I/O | Pull-up Resistor | Function | Alternate Function |
|---|---|---|---|---|---|
| P00 | 3 | I/O | Yes | Port 0<br>I/O port<br>Input/output can be specified in 1-bit units. | TOH0 |
| P01 | 4 | | | | TOH1 |
| P02 | 5 | | | | NMI |
| P03 | 6 | | | | INTP0 |
| P04 | 7 | | | | INTP1 |
| P05 | 17 | | | | INTP2 |
| P06 | 18 | | | | INTP3 |
| P30 | 25 | I/O | Yes | Port 3<br>I/O port<br>Input/output can be specified in 1-bit units.<br>P38 and P39 are fixed to N-ch open-drain output. | TXD0 |
| P31 | 26 | | | | RXD0/INTP7 |
| P32 | 27 | | | | ASCK0/ADTRG/TO01 |
| P33 | 28 | | | | TI000/TO00/TIP00/TOP00 |
| P34 | 29 | | | | TI001/TO00/TIP01/TOP01 |
| P35 | 30 | | | | TI010/TO01 |
| P38 | 35 | | No[Note 1] | | SDA0[Note 2] |
| P39 | 36 | | | | SCL0[Note 2] |
| P40 | 22 | I/O | Yes | Port 4<br>I/O port<br>Input/output can be specified in 1-bit units.<br>P41 and P42 can be specified as N-ch open-drain output in 1-bit units. | SI00 |
| P41 | 23 | | | | SO00 |
| P42 | 24 | | | | $\overline{SCK00}$ |

**Notes 1.** An on-chip pull-up resistor can be provided by a mask option (only in the μPD703308, 703308Y).

**2.** Only in the μPD703308Y, 70F3306Y, 70F3308Y

(2/3)

| Pin Name | Pin No. | I/O | Pull-up Resistor | Function | Alternate Function |
|---|---|---|---|---|---|
| P50 | 37 | I/O | Yes | Port 5<br>I/O port<br>Input/output can be specified in 1-bit units.<br>P54 and P55 can be specified as N-ch open-drain output in 1-bit units. | TI011/RTP00/KR0 |
| P51 | 38 | | | | TI50/RTP01/KR1 |
| P52 | 39 | | | | TO50/RTP02/KR2 |
| P53 | 40 | | | | SIA0/RTP03/KR3 |
| P54 | 41 | | | | SOA0/RTP04/KR4 |
| P55 | 42 | | | | $\overline{\text{SCKA0}}$/RTP05/KR5 |
| P70 | 80 | Input | No | Port 7<br>Input port | ANI0 |
| P71 | 79 | | | | ANI1 |
| P72 | 78 | | | | ANI2 |
| P73 | 77 | | | | ANI3 |
| P74 | 76 | | | | ANI4 |
| P75 | 75 | | | | ANI5 |
| P76 | 74 | | | | ANI6 |
| P77 | 73 | | | | ANI7 |
| P90 | 38 | I/O | Yes | Port 9<br>I/O port<br>Input/output can be specified in 1-bit units.<br>P98 and P99 can be specified as N-ch open-drain output in 1-bit units. | TXD1/KR6 |
| P91 | 39 | | | | RXD1/KR7 |
| P96 | 40 | | | | TI51/TO51 |
| P97 | 41 | | | | SI01 |
| P98 | 42 | | | | SO01 |
| P99 | 43 | | | | $\overline{\text{SCK01}}$ |
| P913 | 44 | | | | INTP4 |
| P914 | 45 | | | | INTP5 |
| P915 | 46 | | | | INTP6 |
| PCM0 | 49 | I/O | Yes | Port CM<br>I/O port<br>Input/output can be specified in 1-bit units. | $\overline{\text{WAIT}}$ |
| PCM1 | 50 | | | | CLKOUT |
| PCM2 | 51 | | | | $\overline{\text{HLDAK}}$ |
| PCM3 | 52 | | | | $\overline{\text{HLDRQ}}$ |
| PCS0 | 47 | I/O | Yes | Port CS<br>I/O port<br>Input/output can be specified in 1-bit units. | $\overline{\text{CS0}}$ |
| PCS1 | 48 | | | | $\overline{\text{CS1}}$ |
| PCT0 | 53 | I/O | Yes | Port CT<br>I/O port<br>Input/output can be specified in 1-bit units. | $\overline{\text{WR0}}$ |
| PCT1 | 54 | | | | $\overline{\text{WR1}}$ |
| PCT4 | 55 | | | | $\overline{\text{RD}}$ |
| PCT6 | 56 | | | | ASTB |

(3/3)

| Pin Name | Pin No. | I/O | Pull-up Resistor | Function | Alternate Function |
|---|---|---|---|---|---|
| PDL0 | 57 | I/O | Yes | Port DL<br>I/O port<br>Input/output can be specified in 1-bit units. | AD0 |
| PDL1 | 58 | | | | AD1 |
| PDL2 | 59 | | | | AD2 |
| PDL3 | 60 | | | | AD3 |
| PDL4 | 61 | | | | AD4 |
| PDL5 | 62 | | | | AD5/FLMD1[Note] |
| PDL6 | 63 | | | | AD6 |
| PDL7 | 64 | | | | AD7 |
| PDL8 | 65 | | | | AD8 |
| PDL9 | 66 | | | | AD9 |
| PDL10 | 67 | | | | AD10 |
| PDL11 | 68 | | | | AD11 |
| PDL12 | 69 | | | | AD12 |
| PDL13 | 70 | | | | AD13 |
| PDL14 | 71 | | | | AD14 |
| PDL15 | 72 | | | | AD15 |

**Note** Only in the μPD70F3306, 70F3306Y, 70F3308, 70F3308Y

### (2) Non-port pins

(1/3)

| Pin Name | Pin No. | I/O | Pull-up Resistor | Function | Alternate Function |
|---|---|---|---|---|---|
| AD0 | 57 | I/O | Yes | Address/data bus for external memory | PDL0 |
| AD1 | 58 | | | | PDL1 |
| AD2 | 59 | | | | PDL2 |
| AD3 | 60 | | | | PDL3 |
| AD4 | 61 | | | | PDL4 |
| AD5 | 62 | | | | PDL5/FLMD1[Note] |
| AD6 | 63 | | | | PDL6 |
| AD7 | 64 | | | | PDL7 |
| AD8 | 65 | | | | PDL8 |
| AD9 | 66 | | | | PDL9 |
| AD10 | 67 | | | | PDL10 |
| AD11 | 68 | | | | PDL11 |
| AD12 | 69 | | | | PDL12 |
| AD13 | 70 | | | | PDL13 |
| AD14 | 71 | | | | PDL14 |
| AD15 | 72 | | | | PDL15 |
| ADTRG | 24 | Input | Yes | A/D converter external trigger input | P32/ASCK0/TO01 |
| ANI0 | 80 | Input | No | Analog voltage input for A/D converter | P70 |
| ANI1 | 79 | | | | P71 |
| ANI2 | 78 | | | | P72 |
| ANI3 | 77 | | | | P73 |
| ANI4 | 76 | | | | P74 |
| ANI5 | 75 | | | | P75 |
| ANI6 | 74 | | | | P76 |
| ANI7 | 73 | | | | P77 |
| ASCK0 | 24 | Input | Yes | UART0 serial clock input | P32/ADTRG/TO01 |
| ASTB | 56 | Output | Yes | Address strobe signal output for external memory | PCT6 |
| $AV_{REF0}$ | 1 | – | – | Reference voltage for A/D converter and positive power supply for alternate-function ports | – |
| $AV_{SS}$ | 2 | – | – | Ground potential for A/D converter and alternate-function ports | – |
| CLKOUT | 50 | Output | Yes | Internal system clock output | PCM1 |
| $\overline{CS0}$ | 47 | Output | Yes | Chip select output | PCS0 |
| $\overline{CS1}$ | 48 | | | | PCS1 |
| $EV_{DD}$ | 31 | – | – | Positive power supply for external | – |
| $EV_{SS}$ | 30 | – | – | Ground potential for external | – |
| FLMD0[Note] | 8 | Input | No | Flash programming mode setting pin | – |
| FLMD1[Note] | 62 | | Yes | | PDL5/AD5 |
| $\overline{HLDAK}$ | 51 | Output | Yes | Bus hold acknowledge output | PCM2 |
| $\overline{HLDRQ}$ | 52 | Input | Yes | Bus hold request input | PCM3 |

**Note** Only in the μPD70F3306, 70F3306Y, 70F3308, 70F3308Y

(2/3)

| Pin Name | Pin No. | I/O | Pull-up Resistor | Function | Alternate Function |
|---|---|---|---|---|---|
| IC[Note 1] | 8 | – | – | Internally connected | – |
| INTP0 | 6 | Input | Yes | External interrupt request input (maskable, analog noise elimination) | P03 |
| INTP1 | 7 | | | | P04 |
| INTP2 | 17 | | | | P05 |
| INTP3 | 18 | | | External interrupt request input (maskable, digital + analog noise elimination) | P06 |
| INTP4 | 44 | | | External interrupt request input (maskable, analog noise elimination) | P913 |
| INTP5 | 45 | | | | P914 |
| INTP6 | 46 | | | | P915 |
| INTP7 | 23 | | | | P31/RXD0 |
| KR0 | 32 | Input | Yes | Key return input | P50/TI011/RTP00 |
| KR1 | 33 | | | | P51/TI50/RTP01 |
| KR2 | 34 | | | | P52/TO50/RTP02 |
| KR3 | 35 | | | | P53/SIA0/RTP03 |
| KR4 | 36 | | | | P54/SOA0/RTP04 |
| KR5 | 37 | | | | P55/$\overline{\text{SCKA0}}$/RTP05 |
| KR6 | 38 | | | | P90/TXD1 |
| KR7 | 39 | | | | P91/RXD1 |
| NMI | 5 | Input | Yes | External interrupt input (non-maskable, analog noise elimination) | P02 |
| $\overline{\text{RD}}$ | 55 | Output | Yes | Read strobe signal output for external memory | PCT4 |
| REGC | 10 | – | – | Connecting capacitor for regulator output stabilization | – |
| $\overline{\text{RESET}}$ | 14 | Input | – | System reset input | – |
| RTP00 | 32 | Output | Yes | Real-time output port | P50/TI011/KR0 |
| RTP01 | 33 | | | | P51/TI50/KR1 |
| RTP02 | 34 | | | | P52/TO50/KR2 |
| RTP03 | 35 | | | | P53/SIA0/KR3 |
| RTP04 | 36 | | | | P54/SOA0/KR4 |
| RTP05 | 37 | | | | P55/$\overline{\text{SCKA0}}$/KR5 |
| RXD0 | 23 | Input | Yes | Serial receive data input for UART0 | P31/INTP7 |
| RXD1 | 39 | | | Serial receive data input for UART1 | P91/KR7 |
| $\overline{\text{SCK00}}$ | 21 | I/O | Yes | Serial clock I/O for CSI00, CSI01, CSIA0 N-ch open-drain output can be specified in 1-bit units. | P42 |
| $\overline{\text{SCK01}}$ | 43 | | | | P99 |
| $\overline{\text{SCKA0}}$ | 37 | | | | P55/RTP05/KR5 |
| SCL0[Note 2] | 29 | I/O | No[Note 3] | Serial clock I/O for $I^2C0$ Fixed to N-ch open-drain output | P39 |
| SDA0[Note 2] | 28 | I/O | No[Note 3] | Serial transmit/receive data I/O for $I^2C0$ Fixed to N-ch open-drain output | P38 |

**Notes 1.** Only in the $\mu$PD703308, 703308Y

**2.** Only in the $\mu$PD703308Y, 70F3306Y, 70F3308Y

**3.** An on-chip pull-up resistor can be provided by a mask option (only in the $\mu$PD703308Y).

(3/3)

| Pin Name | Pin No. | I/O | Pull-up Resistor | Function | Alternate Function |
|---|---|---|---|---|---|
| SI00 | 19 | Input | Yes | Serial receive data input for CSI00 | P40 |
| SI01 | 41 | | | Serial receive data input for CSI01 | P97 |
| SIA0 | 35 | | | Serial receive data input for CSIA0 | P53/RTP03/KR3 |
| SO00 | 20 | Output | Yes | Serial transmit data output for CSI00, CSI01, CSIA0<br>N-ch open-drain output can be specified in 1-bit units. | P41 |
| SO01 | 42 | | | | P98 |
| SOA0 | 36 | | | | P54/RTP04/KR4 |
| TI000 | 25 | Input | Yes | Capture trigger input/external event input for TM00 | P33/TO00/TIP00/TOP00 |
| TI001 | 26 | | | Capture trigger input for TM00 | P34/TO00/TIP01/TOP01 |
| TI010 | 27 | | | Capture trigger input/external event input for TM01 | P35/TO01 |
| TI011 | 32 | | | Capture trigger input for TM01 | P50/RTP00/KR0 |
| TI50 | 33 | | | External event input for TM50 | P51/RTP01/KR1 |
| TI51 | 40 | | | External event input for TM51 | P96/TO51 |
| TIP00 | 25 | | | Capture trigger input/external event input for TMP0 | P33/TI000/TO00/TOP00 |
| TIP01 | 26 | | | Capture trigger input for TMP0 | P34/TI001/TO00/TOP01 |
| TO00 | 25 | Output | Yes | Timer output for TM00 | P33/TI000/TIP00/TOP00 |
| | 26 | | | | P34/TI001/TIP01/TOP01 |
| TO01 | 24 | | | Timer output for TM01 | P32/ASCK0/ADTRG |
| | 27 | | | | P35/TI010 |
| TO50 | 34 | | | Timer output for TM50 | P52/RTP02/KR2 |
| TO51 | 40 | | | Timer output for TM51 | P96/TI51 |
| TOH0 | 3 | | | Timer output for TMH0 | P00 |
| TOH1 | 4 | | | Timer output for TMH1 | P01 |
| TOP00 | 25 | | | Timer output for TMP0 | P33/TI000/TO00/TIP00 |
| TOP01 | 26 | | | | P34/TI001/TO00/TIP01 |
| TXD0 | 22 | Output | Yes | Serial transmit data output for UART0 | P30/TO02 |
| TXD1 | 38 | | | Serial transmit data output for UART1 | P90/KR6 |
| $V_{DD}$ | 9 | – | – | Positive power supply pin for internal | – |
| $V_{SS}$ | 11 | – | – | Ground potential for internal | – |
| $\overline{WAIT}$ | 49 | Input | No | External wait input | PCM0 |
| $\overline{WR0}$ | 53 | Output | No | Write strobe for external memory (lower 8 bits) | PCT0 |
| $\overline{WR1}$ | 54 | | | Write strobe for external memory (higher 8 bits) | PCT1 |
| X1 | 12 | Input | No | Connecting resonator for main clock | – |
| X2 | 13 | – | No | | – |
| XT1 | 15 | Input | No | Connecting resonator for subclock | – |
| XT2 | 16 | – | No | | – |

## 2.2 Pin Status

The address bus becomes undefined during accesses to the internal RAM and ROM. The data bus goes into the high-impedance state without data output. The external bus control signal becomes inactive.

During peripheral I/O access, the address bus outputs the addresses of the on-chip peripheral I/Os that are accessed. The data bus goes into the high-impedance state without data output. The external bus control signal becomes inactive.

**Table 2-2. Pin Operation Status in Operation Modes**

| Operating Status / Pin | Reset[Note 1] | HALT Mode | IDLE Mode/ STOP Mode | Idle State[Note 2] | Bus Hold |
|---|---|---|---|---|---|
| AD0 to AD15 (PDL0 to PDL15) | Hi-Z | Undefined | Hi-Z | Held | Hi-Z |
| $\overline{\text{WAIT}}$ (PCM0) | Hi-Z | – | – | – | – |
| CLKOUT (PCM1) | Hi-Z | Operating | L | Operating | Operating |
| $\overline{\text{CS0}}$, $\overline{\text{CS1}}$ (PCS0, PCS1) | Hi-Z | H | H | Held | Hi-Z |
| $\overline{\text{WR0}}$, $\overline{\text{WR1}}$ (PCT0, PCT1) | Hi-Z | H | H | H | Hi-Z |
| $\overline{\text{RD}}$ (PCT4) | Hi-Z | H | H | H | Hi-Z |
| ASTB (PCT6) | Hi-Z | H | H | H | Hi-Z |
| $\overline{\text{HLDAK}}$ (PCM2) | Hi-Z | Operating | H | H | L |
| $\overline{\text{HLDRQ}}$ (PCM3) | Hi-Z | Operating | – | – | Operating |

**Notes 1.** Since the bus control pin is also used as a port pin, it is initialized to the port mode (input) after reset.

**2.** The pin statuses in the idle state inserted after the T3 state are listed.

**Remark** Hi-Z: High impedance
H: High-level output
L: Low-level output
–: Input without sampling (input acknowledgment not possible)

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

(1/2)

| Pin | Alternate Function | Pin No. | I/O Circuit Type | Recommended Connection |
|---|---|---|---|---|
| P00 | TOH0 | 3 | 5-A | Input: Independently connect to $EV_{DD}$ or $EV_{SS}$ via a resistor.<br>Output: Leave open. |
| P01 | TOH1 | 4 | | |
| P02 | NMI | 5 | 5-W | |
| P03 to P06 | INTP0 to INTP3 | 6, 7, 17, 18 | | |
| P30 | TXD0 | 22 | 5-A | |
| P31 | RXD0/INTP7 | 23 | 5-W | |
| P32 | ASCK0/ADTRG/TO01 | 24 | | |
| P33 | TI000/TO00/TIP00/TOP00 | 25 | | |
| P34 | TI001/TO00/TIP01/TOP01 | 26 | | |
| P35 | TI010/TO01 | 27 | | |
| P38 | SDA0[Note] | 28 | 13-AE | |
| P39 | SCL0[Note] | 29 | | |
| P40 | SI00 | 19 | 5-W | |
| P41 | SO00 | 20 | 10-E | |
| P42 | $\overline{SCK00}$ | 21 | 10-F | |
| P50 | TI011/RTP00/KR0 | 32 | 8-A | |
| P51 | TI50/RTP01/KR1 | 33 | | |
| P52 | TO50/RTP02/KR2 | 34 | | |
| P53 | SIA0/RTP03/KR3 | 35 | | |
| P54 | SOA0/RTP04/KR4 | 36 | 10-A | |
| P55 | $\overline{SCKA0}$/RTP05/KR5 | 37 | | |
| P70 to P77 | ANI0 to ANI7 | 80 to 73 | 9-C | Connect to $AV_{REF0}$ or $AV_{SS}$. |
| P90 | TXD1/KR6 | 38 | 8-A | Input: Independently connect to $EV_{DD}$ or $EV_{SS}$ via a resistor.<br>Output: Leave open. |
| P91 | RXD1/KR7 | 39 | | |
| P96 | TI51/TO51 | 40 | 8-A | |
| P97 | SI01 | 41 | 5-W | |
| P98 | SO01 | 42 | 10-E | |
| P99 | $\overline{SCK01}$ | 43 | 10-F | |
| P913 to P915 | INTP4 to INTP6 | 44 to 46 | 5-W | |
| PCM0 | $\overline{WAIT}$ | 49 | 5-A | |
| PCM1 | CLKOUT | 50 | | |
| PCM2 | $\overline{HLDAK}$ | 51 | | |
| PCM3 | $\overline{HLDRQ}$ | 52 | | |
| PCS0, PCS1 | $\overline{CS0}$, $\overline{CS1}$ | 47, 48 | 5-A | |

**Note** Only in the μPD703308Y, 70F3306Y, 70F3308Y

(2/2)

| Pin | Alternate Function | Pin No. | I/O Circuit Type | Recommended Connection |
|---|---|---|---|---|
| PCT0 | $\overline{WR0}$ | 53 | 5-A | Input: Independently connect to $EV_{DD}$ or $EV_{SS}$ via a resistor.<br>Output: Leave open. |
| PCT1 | $\overline{WR1}$ | 54 | | |
| PCT4 | $\overline{RD}$ | 55 | | |
| PCT6 | ASTB | 56 | | |
| PDL0 to PDL4 | AD0 to AD4 | 57 to 61 | 5-A | |
| PDL5 | AD5/FLMD1[Note 1] | 62 | | |
| PDL6 to PDL15 | AD6 to AD15 | 63 to 72 | | |
| $AV_{REF0}$ | – | 1 | – | Directly connect to $V_{DD}$. |
| $AV_{SS}$ | – | 2 | – | – |
| $EV_{DD}$ | – | 31 | – | – |
| $EV_{SS}$ | – | 30 | – | – |
| IC[Note 2] | – | 8 | – | Directly connect to $EV_{SS}$ or $V_{SS}$ or pull down with a 10 kΩ resistor. |
| $\overline{RESET}$ | – | 14 | 2 | – |
| FLMD0[Note 1] | – | 8 | – | Directly connect to $EV_{SS}$ or $V_{SS}$ or pull down with a 10 kΩ resistor. |
| $V_{DD}$ | – | 9 | – | – |
| $V_{SS}$ | – | 11 | – | – |
| X1 | – | 12 | – | – |
| X2 | – | 13 | – | – |
| XT1 | – | 15 | 16 | Directly connect to $V_{SS}$[Note 3]. |
| XT2 | – | 16 | 16 | Leave open. |

**Notes 1.** Only in the μPD70F3306, 70F3306Y, 70F3308, 70F3308Y

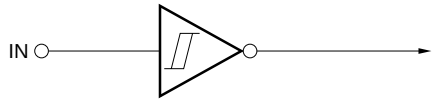
**2.** Only in the μPD703308, 703308Y

**3.** Be sure to set the PSMR.XTSTP bit to 1 when this pin is not used.

## 2.4 Pin I/O Circuits

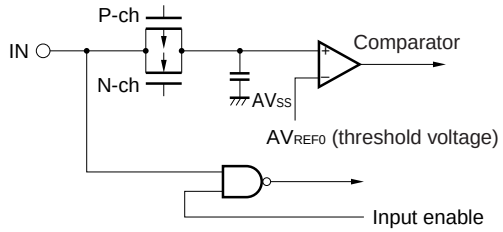
(1/2)

Type 2

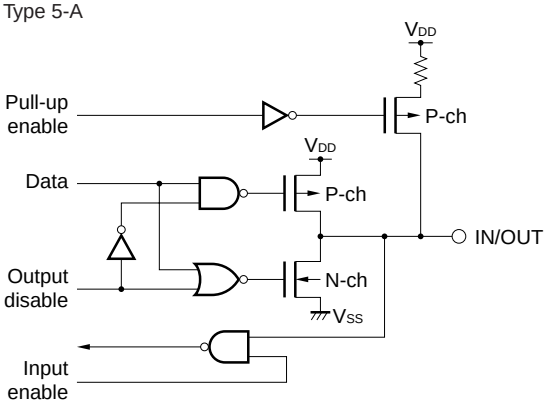



Schmitt-triggered input with hysteresis characteristics

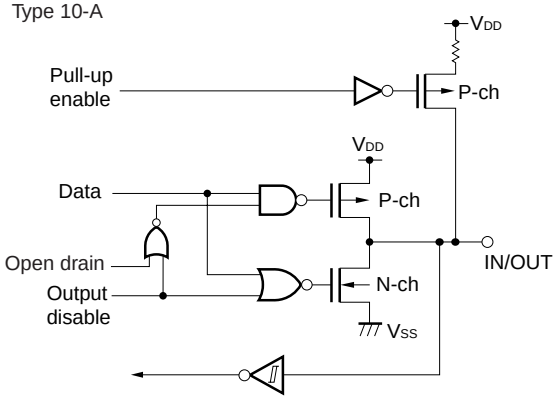
Type 9-C



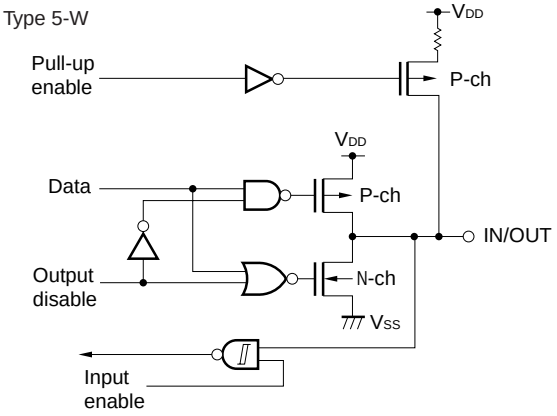

Type 5-A



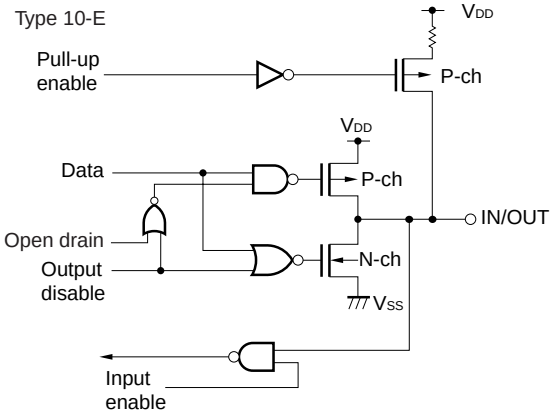

Type 10-A



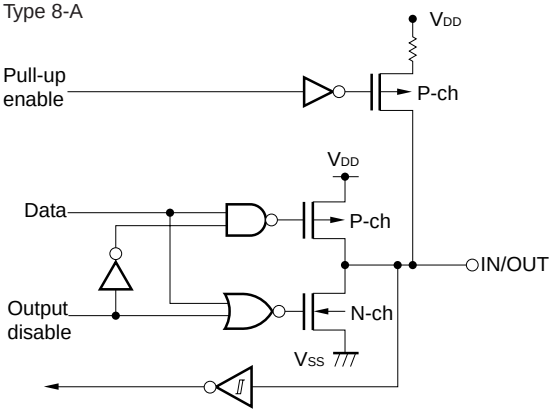

Type 5-W



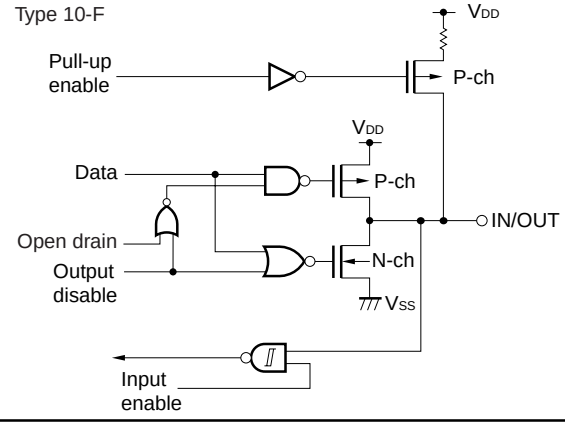

Type 10-E




Type 8-A

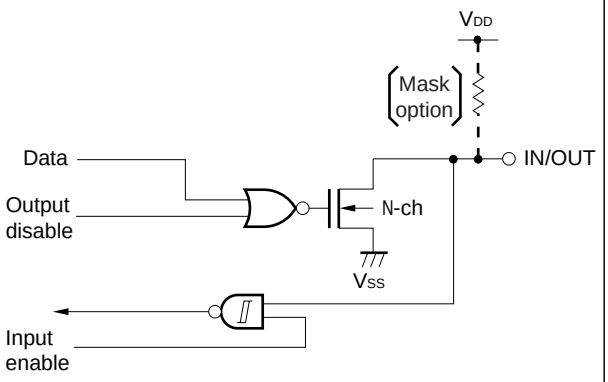



Type 10-F

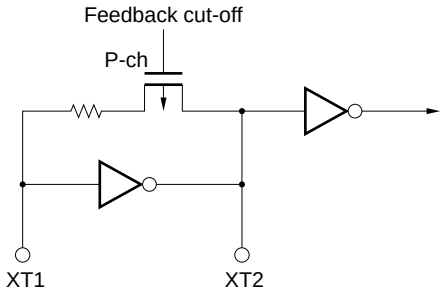


(2/2)

Type 13-AE




Type 16




**Remark** Read $V_{DD}$ as $EV_{DD}$. Also, read $V_{SS}$ as $EV_{SS}$.

# CHAPTER 3 CPU FUNCTIONS

The CPU of the V850ES/KF1+ is based on the RISC architecture and executes most instructions in one clock cycle by using 5-stage pipeline control.

## 3.1 Features

- Number of instructions: 83
- Minimum instruction execution time: 50.0 ns (@ 20 MHz operation: 4.5 to 5.5 V, not using regulator)
  125 ns[Note] (@ 8 MHz operation: 2.7 to 5.5 V, not using regulator)
- Memory space
  Program (physical address) space: 64 MB linear
  Data (logical address) space: 4 GB linear
  - Memory block division function: 64 KB, 64 KB/Total of 2 blocks
- General-purpose registers: 32 bits × 32
- Internal 32-bit architecture
- 5-stage pipeline control
- Multiply/divide instructions
- Saturated operation instructions
- 32-bit shift instruction: 1 clock
- Load/store instruction with long/short format
- Four types of bit manipulation instructions
  - SET1
  - CLR1
  - NOT1
  - TST1

**Note** This value may change after evaluation.

## 3.2 CPU Register Set

The CPU registers of the V850ES/KF1+ can be classified into two categories: a general-purpose program register set and a dedicated system register set. All the registers have 32-bit width.

For details, refer to the **V850ES Architecture User's Manual**.

**(1) Program register set**

| 31 | | 0 |
|---|---|---|
| r0 | (Zero register) | |
| r1 | (Assembler-reserved register) | |
| r2 | | |
| r3 | (Stack pointer (SP)) | |
| r4 | (Global pointer (GP)) | |
| r5 | (Text pointer (TP)) | |
| r6 | | |
| r7 | | |
| r8 | | |
| r9 | | |
| r10 | | |
| r11 | | |
| r12 | | |
| r13 | | |
| r14 | | |
| r15 | | |
| r16 | | |
| r17 | | |
| r18 | | |
| r19 | | |
| r20 | | |
| r21 | | |
| r22 | | |
| r23 | | |
| r24 | | |
| r25 | | |
| r26 | | |
| r27 | | |
| r28 | | |
| r29 | | |
| r30 | (Element pointer (EP)) | |
| r31 | (Link pointer (LP)) | |

| 31 | | 0 |
|---|---|---|
| PC | (Program counter) | |

**(2) System register set**

| 31 | | 0 |
|---|---|---|
| EIPC | (Interrupt status saving register) | |
| EIPSW | (Interrupt status saving register) | |
| FEPC | (NMI status saving register) | |
| FEPSW | (NMI status saving register) | |
| ECR | (Interrupt source register) | |
| PSW | (Program status word) | |
| CTPC | (CALLT execution status saving register) | |
| CTPSW | (CALLT execution status saving register) | |
| DBPC | (Exception/debug trap status saving register) | |
| DBPSW | (Exception/debug trap status saving register) | |
| CTBP | (CALLT base pointer) | |

### 3.2.1 Program register set

The program register set includes general-purpose registers and a program counter.

#### (1) General-purpose registers (r0 to r31)

Thirty-two general-purpose registers, r0 to r31, are available. All of these registers can be used as a data variable or address variable.

However, r0 and r30 are implicitly used by instructions and care must be exercised when using these registers. r0 always holds 0 and is used for operations that use 0 and offset 0 addressing. r30 is used as a base pointer when performing memory access with the SLD and SST instructions.

Also, r1, r3 to r5, and r31 are implicitly used by the assembler and C compiler. Therefore, before using these registers, their contents must be saved so that they are not lost, and they must be restored to the registers after the registers have been used. There are cases when r2 is used by the real-time OS. If r2 is not used by the real-time OS, r2 can be used as a variable register.

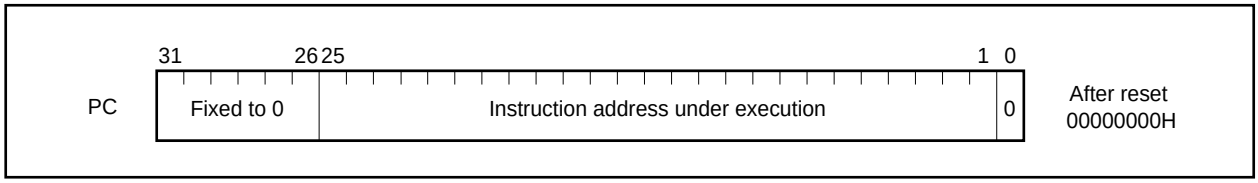
**Table 3-1. Program Registers**

| Name | Usage | Operation |
|---|---|---|
| r0 | Zero register | Always holds 0 |
| r1 | Assembler-reserved register | Working register for generating 32-bit immediate |
| r2 | Address/data variable register (when r2 is not used by the real-time OS to be used) | |
| r3 | Stack pointer | Used to generate stack frame when function is called |
| r4 | Global pointer | Used to access global variable in data area |
| r5 | Text pointer | Register to indicate the start of the text area (area for placing program code) |
| r6 to r29 | Address/data variable register | |
| r30 | Element pointer | Base pointer when memory is accessed |
| r31 | Link pointer | Used by compiler when calling function |
| PC | Program counter | Holds instruction address during program execution |

#### (2) Program counter (PC)

This register holds the address of the instruction under execution. The lower 26 bits of this register are valid, and bits 31 to 26 are fixed to 0. If a carry occurs from bit 25 to bit 26, it is ignored.

Bit 0 is fixed to 0, and branching to an odd address cannot be performed.



### 3.2.2 System register set

System registers control the status of the CPU and hold interrupt information.

Read from and write to system registers are performed by setting the system register numbers shown below with the system register load/store instructions (LDSR, STSR instructions).

**Table 3-2. System Register Numbers**

| System Register No. | System Register Name | Operand Specification Enabled | |
|---|---|---|---|
| | | LDSR Instruction | STSR Instruction |
| 0 | Interrupt status saving register (EIPC)[Note 1] | Yes | Yes |
| 1 | Interrupt status saving register (EIPSW)[Note 1] | Yes | Yes |
| 2 | NMI status saving register (FEPC)[Note 1] | Yes | Yes |
| 3 | NMI status saving register (FEPSW)[Note 1] | Yes | Yes |
| 4 | Interrupt source register (ECR) | No | Yes |
| 5 | Program status word (PSW) | Yes | Yes |
| 6 to 15 | Reserved numbers for future function expansion (The operation is not guaranteed if accessed.) | No | No |
| 16 | CALLT execution status saving register (CTPC) | Yes | Yes |
| 17 | CALLT execution status saving register (CTPSW) | Yes | Yes |
| 18 | Exception/debug trap status saving register (DBPC) | Yes[Note 2] | Yes |
| 19 | Exception/debug trap status saving register (DBPSW) | Yes[Note 2] | Yes |
| 20 | CALLT base pointer (CTBP) | Yes | Yes |
| 21 to 31 | Reserved numbers for future function expansion (The operation is not guaranteed if accessed.) | No | No |

**Notes 1.** Since only one set of these registers is available, the contents of this register must be saved by the program when multiple interrupt servicing is enabled.

**2.** Can be accessed only during the period from the DBTRAP instruction to the DBRET instruction.

**Caution Even if bit 0 of EIPC, FEPC, or CTPC is set (1) by the LDSR instruction, bit 0 is ignored during return with the RETI instruction following interrupt servicing (because bit 0 of PC is fixed to 0). When setting a value to EIPC, FEPC, and CTPC, set an even number (bit 0 = 0).**

**(1) Interrupt status saving registers (EIPC, EIPSW)**

There are two interrupt status saving registers, EIPC and EIPSW.

Upon occurrence of a software exception or a maskable interrupt, the contents of the program counter (PC) are saved to EIPC and the contents of the program status word (PSW) are saved to EIPSW (upon occurrence of a non-maskable interrupt (NMI), the contents are saved to the NMI status saving registers (FEPC, FEPSW)).

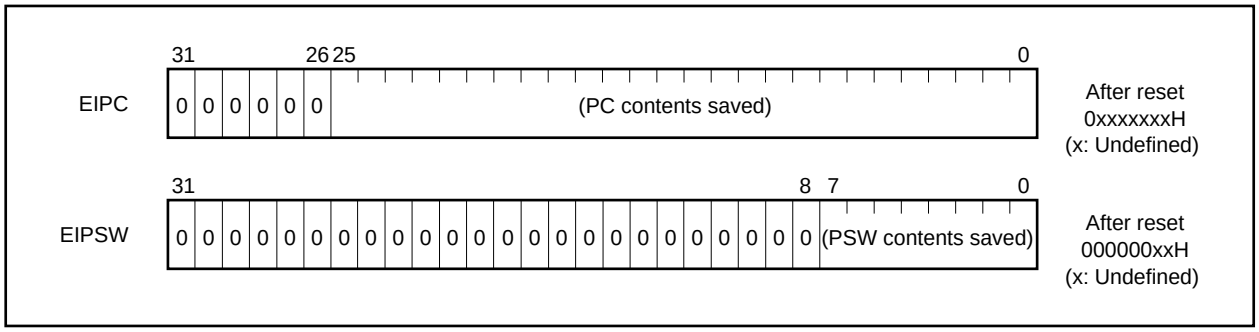
The address of the next instruction following the instruction executed when a software exception or maskable interrupt occurs is saved to EIPC, except for some instructions (refer to **19.9 Periods in Which Interrupts Are Not Acknowledged by CPU**).

The current PSW contents are saved to EIPSW.

Since there is only one set of interrupt status saving registers, the contents of these registers must be saved by the program when multiple interrupt servicing is enabled.

Bits 31 to 26 of EIPC and bits 31 to 8 of EIPSW are reserved (fixed to 0) for future function expansion.

When the RETI instruction is executed, the values in EIPC and EIPSW are restored to the PC and PSW, respectively.



**(2) NMI status saving registers (FEPC, FEPSW)**

There are two NMI status saving registers, FEPC and FEPSW.

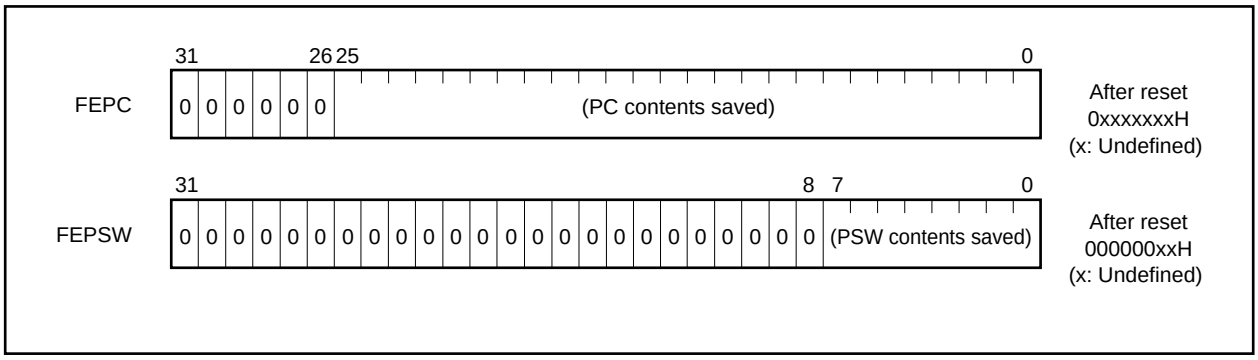
Upon occurrence of a non-maskable interrupt (NMI), the contents of the program counter (PC) are saved to FEPC and the contents of the program status word (PSW) are saved to FEPSW.

The address of the next instruction following the instruction executed when a non-maskable interrupt occurs is saved to FEPC, except for some instructions.

The current PSW contents are saved to FEPSW.

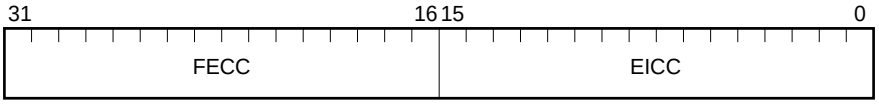
Since there is only one set of NMI status saving registers, the contents of these registers must be saved by the program when multiple interrupt servicing is performed.

Bits 31 to 26 of FEPC and bits 31 to 8 of FEPSW are reserved (fixed to 0) for future function expansion.




**(3) Interrupt source register (ECR)**

Upon occurrence of an interrupt or an exception, the interrupt source register (ECR) holds the source of an interrupt or an exception. The value held by ECR is the exception code coded for each interrupt source. This register is a read-only register, and thus data cannot be written to it using the LDSR instruction.




| Bit position | Bit name | Description |
|---|---|---|
| 31 to 16 | FECC | Non-maskable interrupt (NMI) exception code |
| 15 to 0 | EICC | Exception, maskable interrupt exception code |

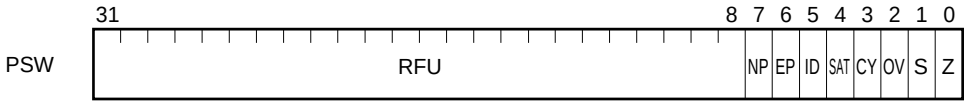
**(4) Program status word (PSW)**

The program status word (PSW) is a collection of flags that indicate the program status (instruction execution result) and the CPU status.

When the contents of this register are changed using the LDSR instruction, the new contents become valid immediately following completion of LDSR instruction execution. Interrupt request acknowledgment is held pending while a write to the PSW is being executed by the LDSR instruction.

Bits 31 to 8 are reserved (fixed to 0) for future function expansion.

(1/2)




| Bit position | Flag name | Description |
|---|---|---|
| 31 to 8 | RFU | Reserved field. Fixed to 0. |
| 7 | NP | Indicates that non-maskable interrupt (NMI) servicing is in progress. This flag is set to 1 when an NMI request is acknowledged, and disables multiple interrupts.<br>0: NMI servicing not in progress<br>1: NMI servicing in progress |
| 6 | EP | Indicates that exception processing is in progress. This flag is set to 1 when an exception occurs. Moreover, interrupt requests can be acknowledged even when this bit is set.<br>0: Exception processing not in progress<br>1: Exception processing in progress |
| 5 | ID | Indicates whether maskable interrupt request acknowledgment is enabled.<br>0: Interrupt enabled<br>1: Interrupt disabled |
| 4 | SAT[Note] | Indicates that the result of executing a saturated operation instruction has overflowed and that the calculation result is saturated. Since this is a cumulative flag, it is set to 1 when the result of a saturated operation instruction becomes saturated, and it is not cleared to 0 even if the operation results of successive instructions do not become saturated. This flag is neither set nor cleared when arithmetic operation instructions are executed.<br>0: Not saturated<br>1: Saturated |
| 3 | CY | Indicates whether carry or borrow occurred as the result of an operation.<br>0: No carry or borrow occurred<br>1: Carry or borrow occurred |
| 2 | OV[Note] | Indicates whether overflow occurred during an operation.<br>0: No overflow occurred<br>1: Overflow occurred. |
| 1 | S[Note] | Indicates whether the result of an operation is negative.<br>0: Operation result is positive or 0.<br>1: Operation result is negative. |
| 0 | Z | Indicates whether operation result is 0.<br>0: Operation result is not 0.<br>1: Operation result is 0. |

**Remark** **Note** is explained on the following page.

(2/2)

**Note** During saturated operation, the saturated operation results are determined by the contents of the OV flag and S flag. The SAT flag is set (to 1) only when the OV flag is set (to 1) during saturated operation.

| Operation result status | Flag status | | | Saturated operation result |
|---|---|---|---|---|
| | SAT | OV | S | |
| Maximum positive value exceeded | 1 | 1 | 0 | 7FFFFFFFH |
| Maximum negative value exceeded | 1 | 1 | 1 | 80000000H |
| Positive (maximum value not exceeded) | Holds value before operation | 0 | 0 | Actual operation result |
| Negative (maximum value not exceeded) | | | 1 | |

**(5) CALLT execution status saving registers (CTPC, CTPSW)**

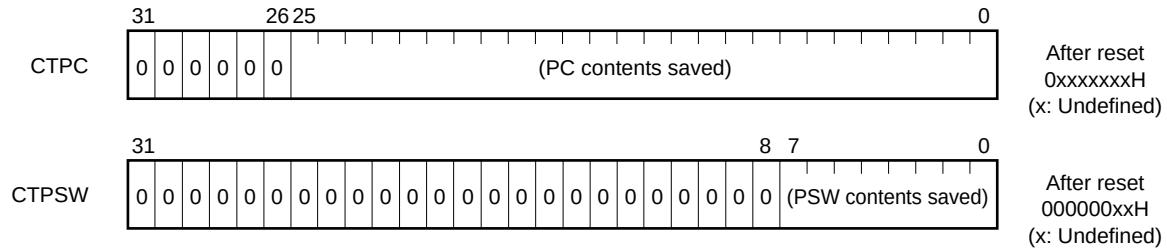
There are two CALLT execution status saving registers, CTPC and CTPSW.

When the CALLT instruction is executed, the contents of the program counter (PC) are saved to CTPC, and the program status word (PSW) contents are saved to CTPSW.

The contents saved to CTPC consist of the address of the next instruction after the CALLT instruction.

The current PSW contents are saved to CTPSW.

Bits 31 to 26 of CTPC and bits 31 to 8 of CTPSW are reserved (fixed to 0) for future function expansion.



### (6) Exception/debug trap status saving registers (DBPC, DBPSW)

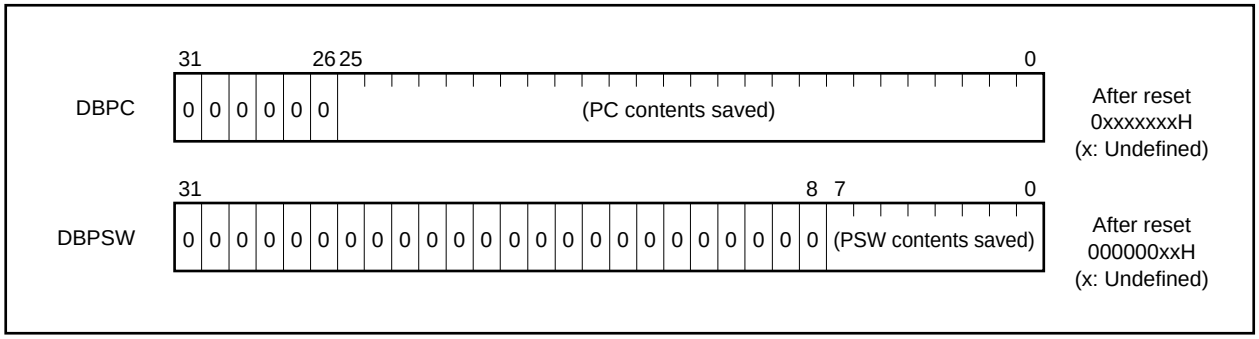
There are two exception/debug trap status saving registers, DBPC and DBPSW.

Upon occurrence of an exception trap or debug trap, the contents of the program counter (PC) are saved to DBPC, and the program status word (PSW) contents are saved to DBPSW.

The contents saved to DBPC consist of the address of the next instruction after the instruction executed when an exception trap or debug trap occurs.

The current PSW contents are saved to DBPSW.

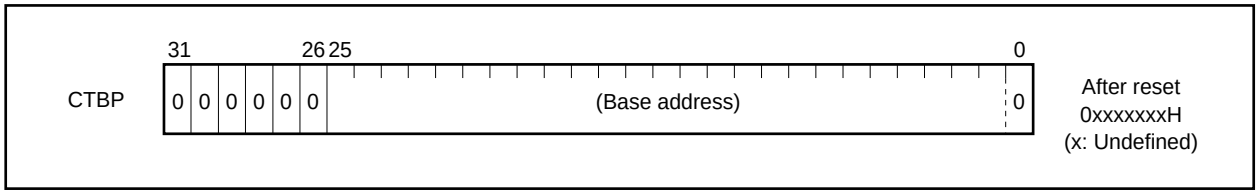
Bits 31 to 26 of DBPC and bits 31 to 8 of DBPSW are reserved (fixed to 0) for future function expansion.




### (7) CALLT base pointer (CTBP)

The CALLT base pointer (CTBP) is used to specify table addresses and generate target addresses (bit 0 is fixed to 0).

Bits 31 to 26 are reserved (fixed to 0) for future function expansion.



## 3.3 Operating Modes

The V850ES/KF1+ has the following operating modes.

**(1) Normal operating mode**

After the system has been released from the reset state, the pins related to the bus interface are set to the port mode, execution branches to the reset entry address of the internal ROM, and instruction processing is started.

**(2) Flash memory programming mode**

This mode is valid only in flash memory versions (μPD70F3306, 70F3306Y, 70F3308, and 70F3308Y).

When this mode is specified, the internal flash memory can be programmed by using a flash programmer.

**(a) Specifying operating mode**

The operating mode is specified according to the status (input level) of the FLMD0 and FLMD1 pins.

In the normal operating mode, input a low level to the FLMD0 pin during the reset period.

A high level is input to the FLMD0 pin by the flash programmer in the flash memory programming mode if a flash programmer is connected. In the self-programming mode, input a high level to this pin from an external circuit.

Fix the specification of these pins in the application system and do not change the setting of these pins during operation.

| FLMD0 | FLMD1 | Operating Mode |
|---|---|---|
| L | × | Normal operating mode |
| H | L | Flash memory programming mode |
| H | H | Setting prohibited |

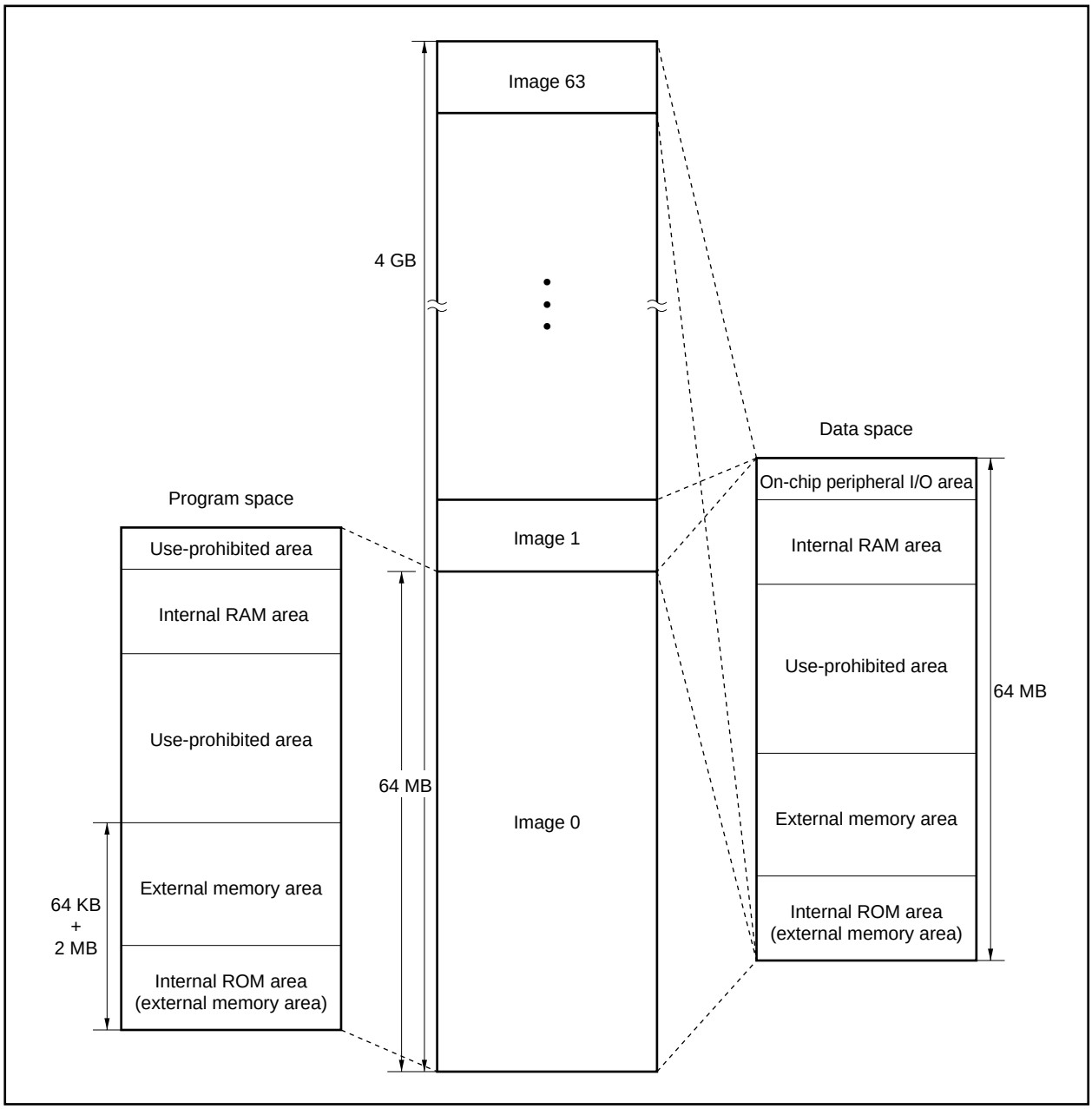
**Remark** H: High level
L: Low level
×: don't care

## 3.4 Address Space

### 3.4.1 CPU address space

Up to 64 KB + 2 MB of external memory area in a linear address space (program space) of up to 64 MB, internal ROM area, and internal RAM area are supported for instruction address addressing. During operand addressing (data access), up to 4 GB of linear address space (data space) is supported. However, the 4 GB address space is viewed as 64 images of a 64 MB physical address space. In other words, the same 64 MB physical address space is accessed regardless of the value of bits 31 to 26.

**Figure 3-1. Address Space Image**



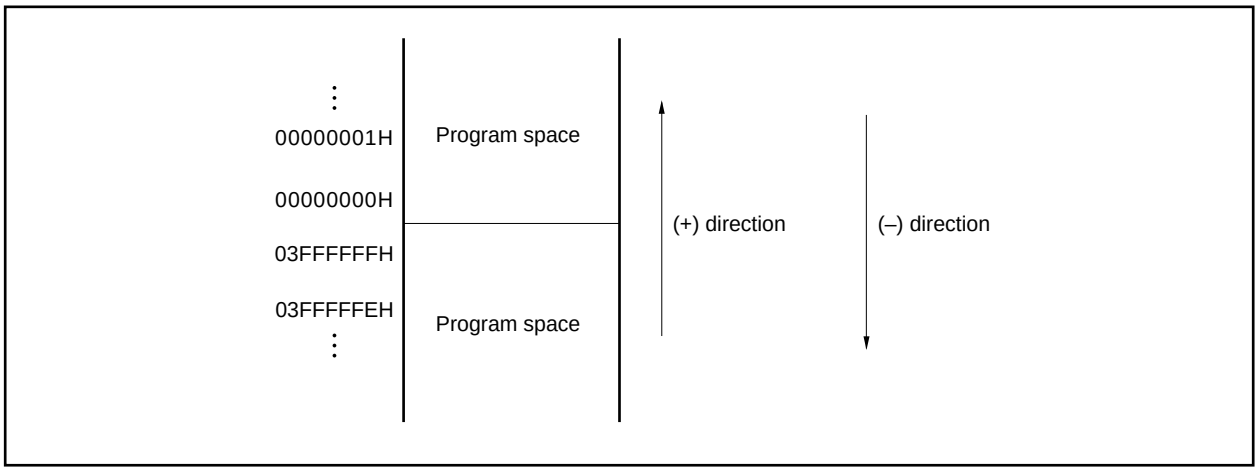
### 3.4.2 Wraparound of CPU address space

**(1) Program space**

Of the 32 bits of the program counter (PC), the higher 6 bits are fixed to 0 and only the lower 26 bits are valid. Even if a carry or borrow occurs from bit 25 to bit 26 as a result of branch address calculation, the higher 6 bits ignore this and remain 0.

Therefore, the lower-limit address of the program space, 00000000H, and the upper-limit address, 03FFFFFFH, are contiguous addresses, and the program space is wrapped around at the boundary of these addresses.

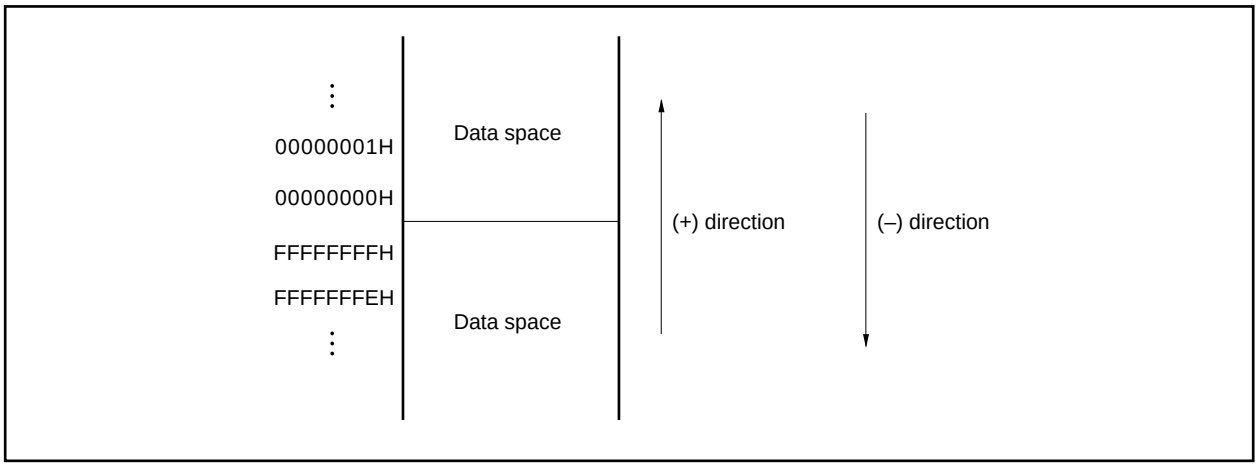
**Caution No instructions can be fetched from the 4 KB area of 03FFF000H to 03FFFFFFH because this area is an on-chip peripheral I/O area. Therefore, do not execute any branch operation instructions in which the destination address will reside in any part of this area.**




**(2) Data space**

The result of an operand address calculation that exceeds 32 bits is ignored.

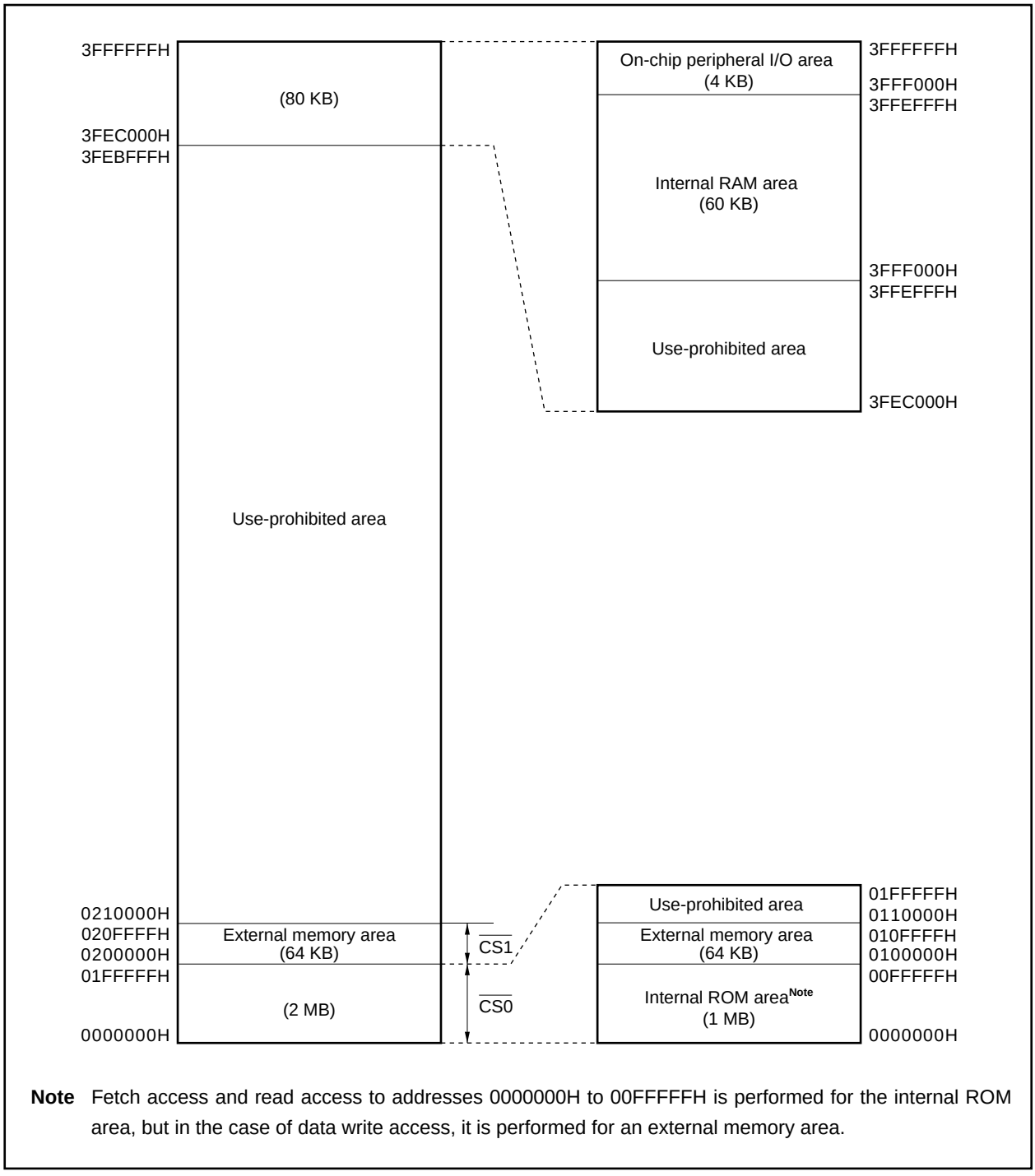
Therefore, the lower-limit address of the data space, address 00000000H, and the upper-limit address, FFFFFFFFH, are contiguous addresses, and the data space is wrapped around at the boundary of these addresses.



### 3.4.3 Memory map

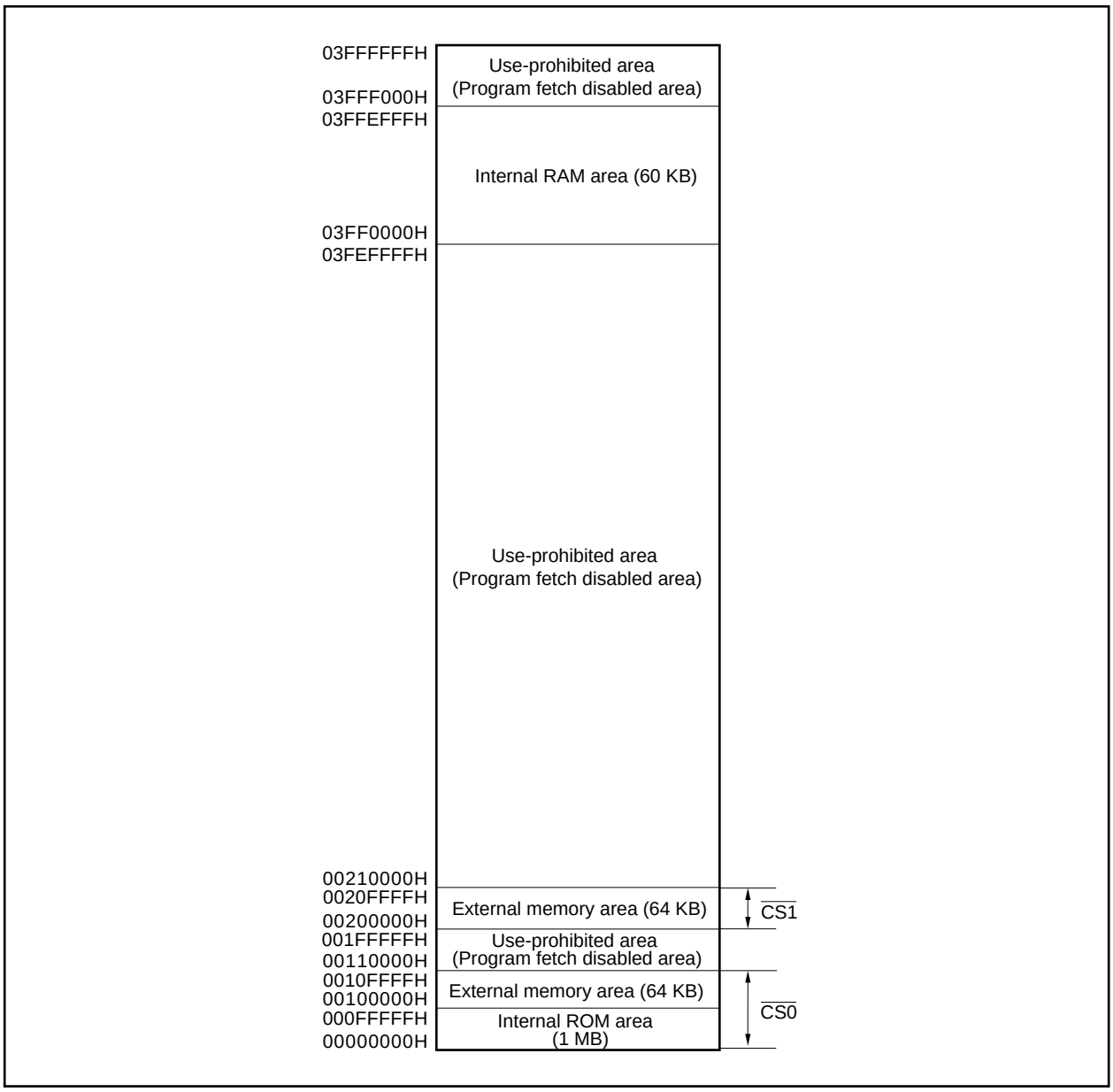
The V850ES/KF1+ has reserved areas as shown below.

**Figure 3-2. Data Memory Map (Physical Addresses)**




**Note** Fetch access and read access to addresses 0000000H to 00FFFFFH is performed for the internal ROM area, but in the case of data write access, it is performed for an external memory area.

**Figure 3-3. Program Memory Map**



### 3.4.4 Areas

**(1) Internal ROM area**

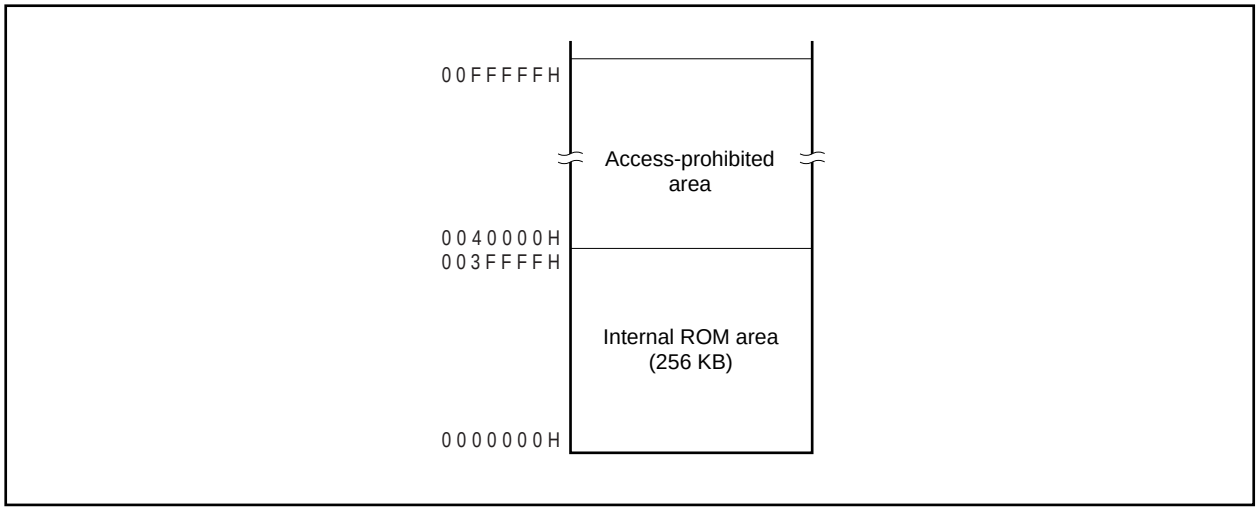
An area of 1 MB from 0000000H to 00FFFFFH is reserved for the internal ROM area.

**(a) Internal ROM (256 KB)**

A 256 KB area from 0000000H to 003FFFFH is provided in the following products.
Addresses 0040000H to 00FFFFFH are an access-prohibited area.

- μPD703308, 703308Y, 70F3308, 70F3308Y

**Figure 3-4. Internal ROM Area (256 KB)**

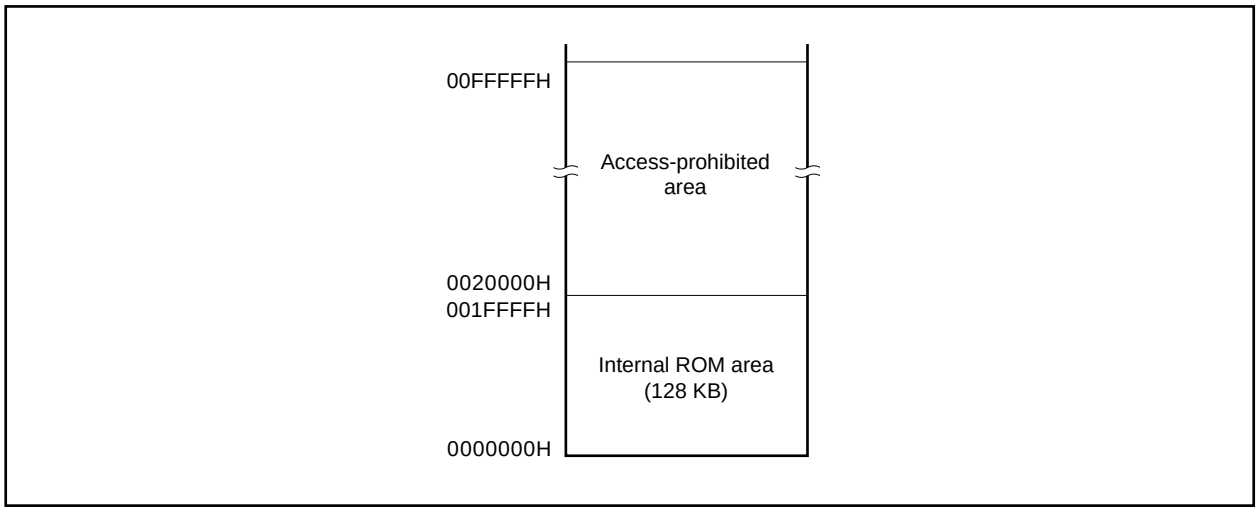



**(b) Internal ROM (128 KB)**

A 128 KB area from 0000000H to 001FFFFH is provided in the following products.
Addresses 0020000H to 00FFFFFH are an access-prohibited area.

- μPD70F3306, 70F3306Y

**Figure 3-5. Internal ROM Area (128 KB)**



**(2) Internal RAM area**

An area of 60 KB maximum from 3FF0000H to 3FFEFFFH is reserved for the internal RAM area.

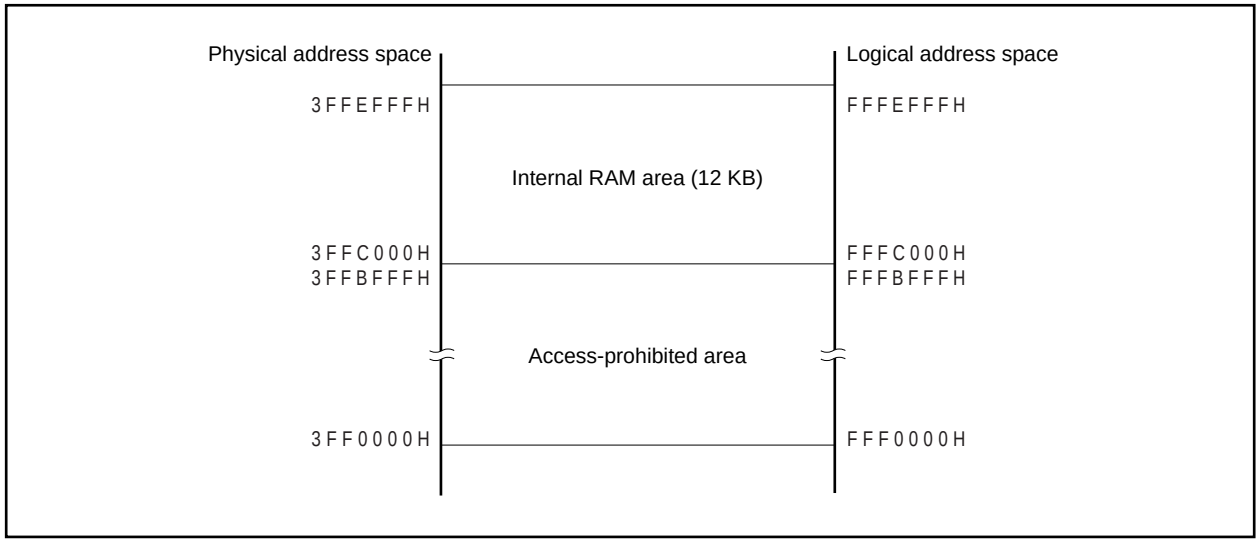
**(a) Internal RAM (12 KB)**

A 12 KB area from 3FFC000H to 3FFEFFFH is provided as physical internal RAM in the following products.

Addresses 3FF0000H to 3FFBFFFH are an access-prohibited area.

- $\mu$PD703308, 703308Y, 70F3308, 70F3308Y

**Figure 3-6. Internal RAM Area (12 KB)**

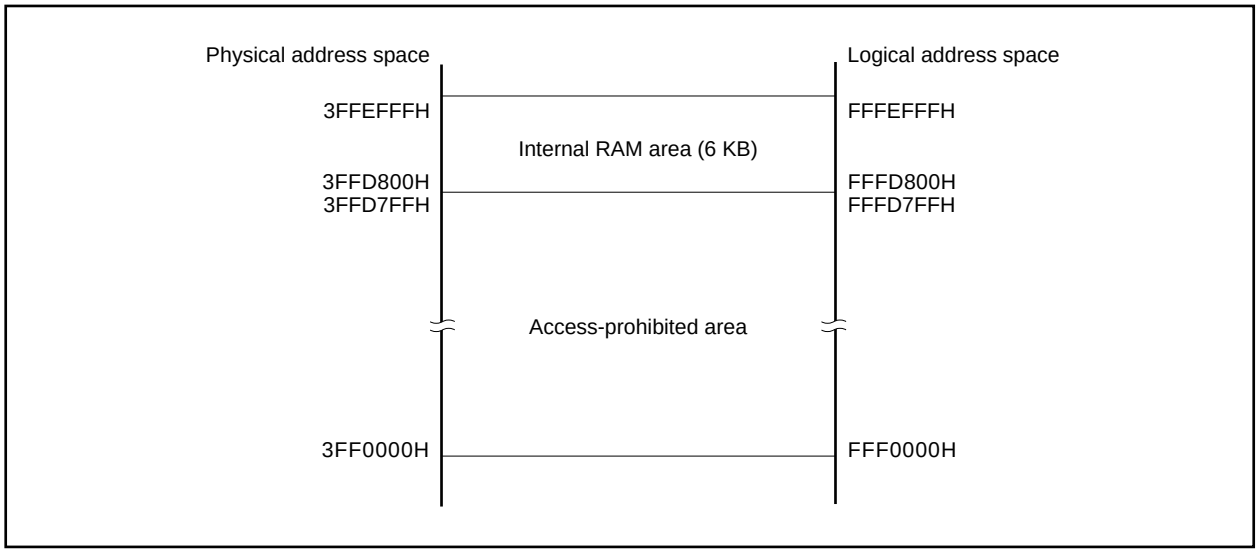


**(b) Internal RAM (6 KB)**

A 6 KB area from 3FFD800H to 3FFEFFFH is provided as physical internal RAM in the following products. Addresses 3FF0000H to 3FFD7FFH are an access-prohibited area.

- $\mu$PD70F3306, 70F3306Y

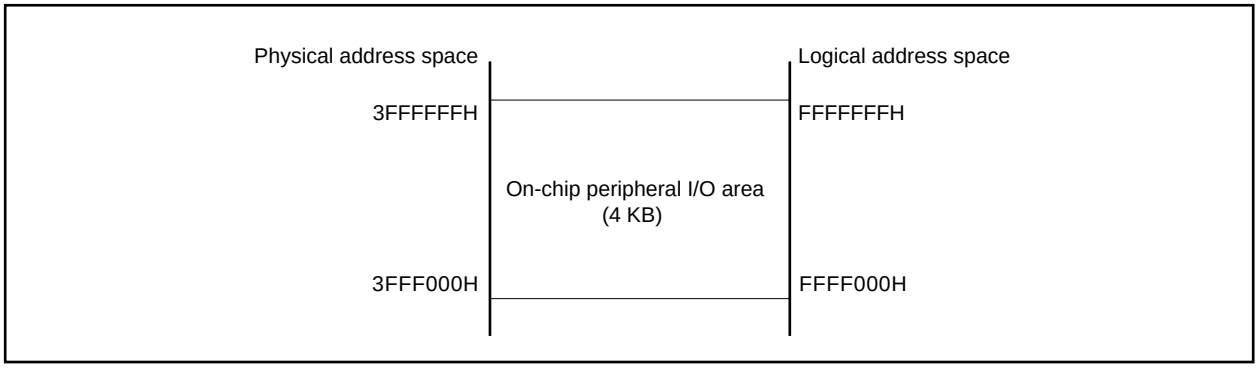
**Figure 3-7. Internal RAM Area (6 KB)**



**(3) On-chip peripheral I/O area**

A 4 KB area from 3FFF000H to 3FFFFFFH is reserved as the on-chip peripheral I/O area.

**Figure 3-8. On-Chip Peripheral I/O Area**




Peripheral I/O registers assigned with functions such as on-chip peripheral I/O operation mode specification and state monitoring are mapped to the on-chip peripheral I/O area. Program fetches are not allowed in this area.

**Cautions 1. If word access of a register is attempted, halfword access to the word area is performed twice, first for the lower bits, then for the higher bits, ignoring the lower 2 address bits.**

**2. If a register that can be accessed in byte units is accessed in halfword units, the higher 8 bits become undefined if the access is a read operation. If a write access is performed, only the data in the lower 8 bits is written to the register.**

**3. Addresses that are not defined as registers are reserved for future expansion. If these addresses are accessed, the operation is undefined and not guaranteed.**

**(4) External memory area**

128 KB (0100000H to 010FFFFH, 0200000H to 020FFFFH) are provided as the external memory area. For details, refer to **CHAPTER 5 BUS CONTROL FUNCTION**.

### 3.4.5 Recommended use of address space

The architecture of the V850ES/KF1+ requires that a register that serves as a pointer be secured for address generation when operand data in the data space is accessed. The address stored in this pointer ±32 KB can be directly accessed by an instruction for operand data. Because the number of general-purpose registers that can be used as a pointer is limited, however, by keeping the performance from dropping during address calculation when a pointer value is changed, as many general-purpose registers as possible can be secured for variables, and the program size can be reduced.

**(1) Program space**

Of the 32 bits of the PC (program counter), the higher 6 bits are fixed to 0, and only the lower 26 bits are valid. Regarding the program space, therefore, a 64 MB space of contiguous addresses starting from 00000000H unconditionally corresponds to the memory map.

To use the internal RAM area as the program space, access the following addresses.

| RAM Size | Access Address |
|---|---|
| 6 KB | 3FFD800H to 3FFEFFFH |
| 12 KB | 3FFC000H to 3FFEFFFH |

**(2) Data space**

With the V850ES/KF1+, it seems that there are sixty-four 64 MB physical address spaces on the 4 GB CPU address space. Therefore, the least significant bit (bit 25) of a 26-bit address is sign-extended to 32 bits and allocated as an address.

**(a) Application example of wraparound**

If R = r0 (zero register) is specified for the LD/ST disp16 [R] instruction, a range of addresses 00000000H ±32 KB can be addressed by sign-extended disp16. All the resources, including the internal hardware, can be addressed by one pointer.

The zero register (r0) is a register fixed to 0 by hardware, and practically eliminates the need for registers dedicated to pointers.

**Example**: μPD703308, 703308Y

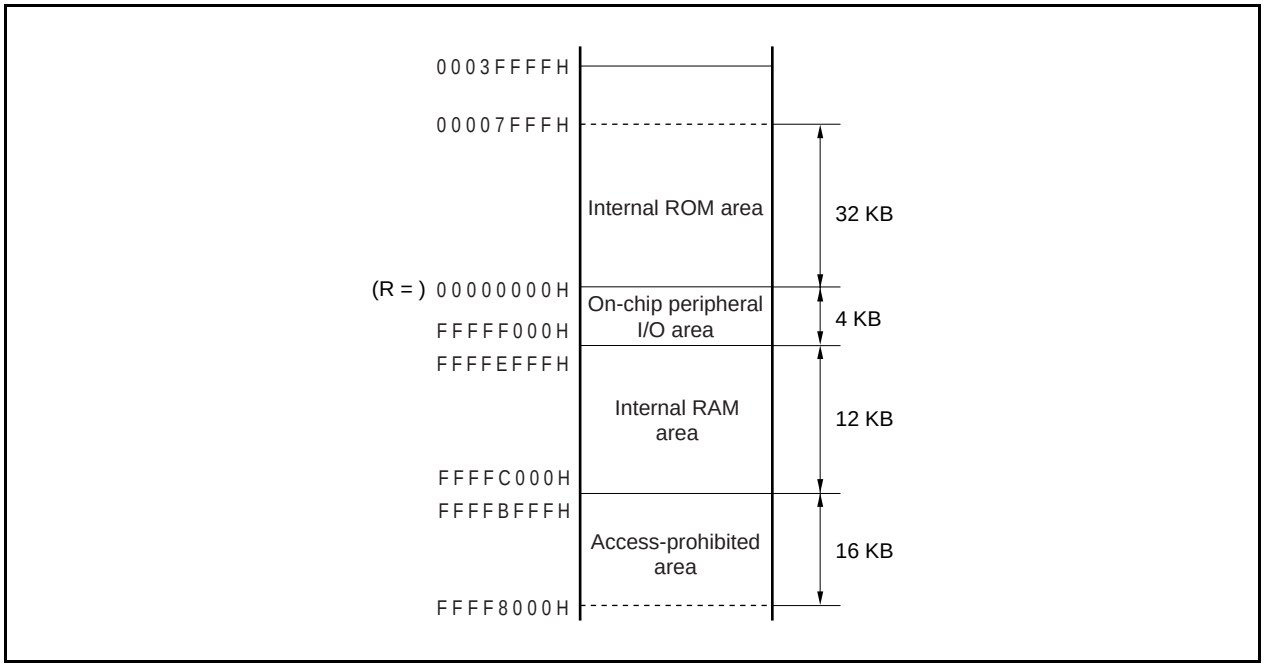
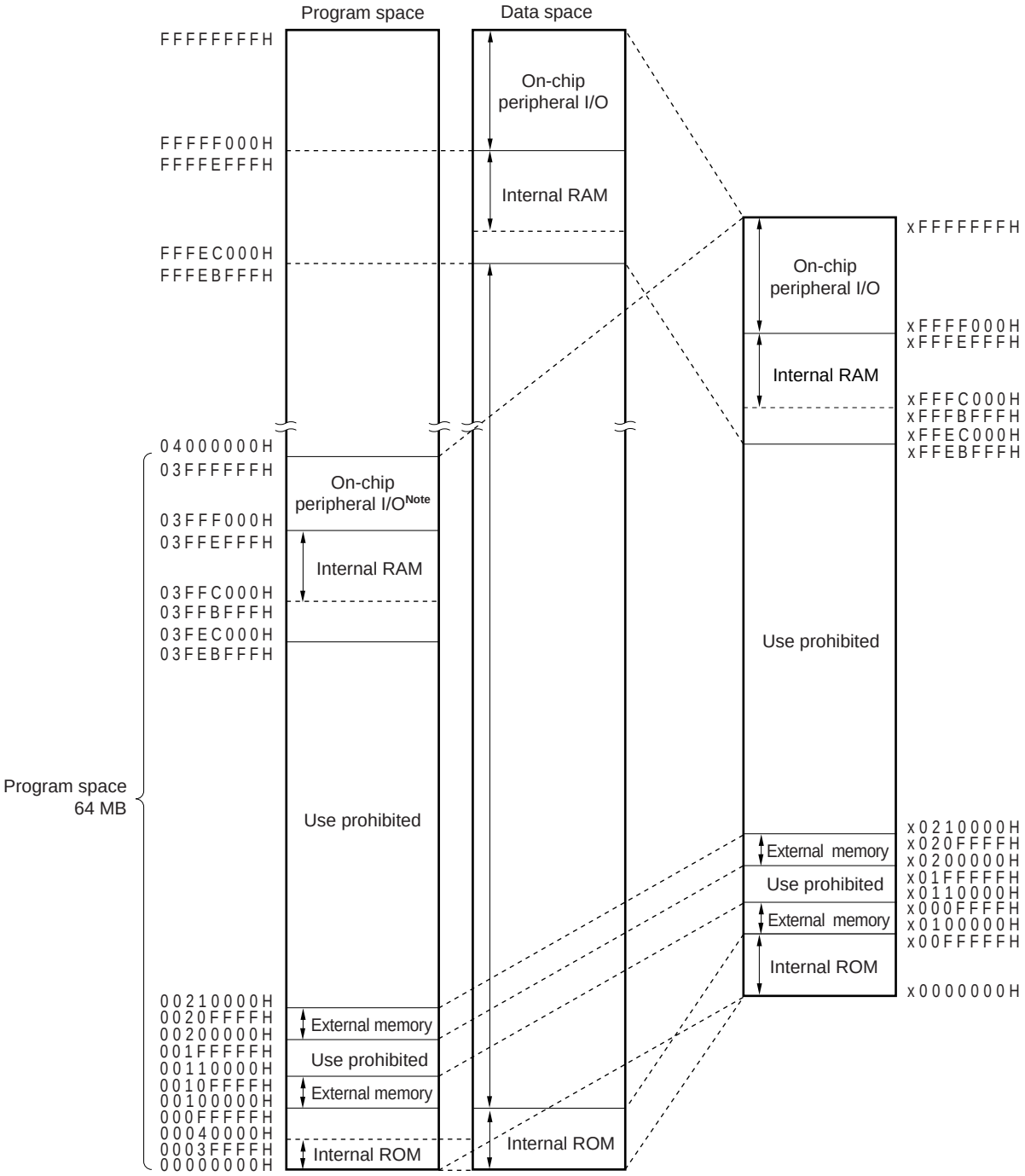


**Figure 3-9. Recommended Memory Map**




**Note** Access to this area is prohibited. To access the on-chip peripheral I/O in this area, specify addresses FFFF000H to FFFFFFFH.

**Remarks 1.** ↕ indicates the recommended area.

**2.** This figure is the recommended memory map of the μPD703308 and 703308Y.

### 3.4.6 Peripheral I/O registers

(1/8)

| Address | Function Register Name | Symbol | R/W | Operable Bit Unit | | | After Reset |
|---|---|---|---|---|---|---|---|
| | | | | 1 | 8 | 16 | |
| FFFFF004H | Port DL register | PDL | R/W | | | √ | 0000H[Note] |
| FFFFF004H | Port DL register L | PDLL | R/W | √ | √ | | 00H[Note] |
| FFFFF005H | Port DL register H | PDLH | R/W | √ | √ | | 00H[Note] |
| FFFFF008H | Port CS register | PCS | R/W | √ | √ | | 00H[Note] |
| FFFFF00AH | Port CT register | PCT | R/W | √ | √ | | 00H[Note] |
| FFFFF00CH | Port CM register | PCM | R/W | √ | √ | | 00H[Note] |
| FFFFF024H | Port DL mode register | PMDL | R/W | | | √ | FFFFH |
| FFFFF024H | Port DL mode register L | PMDLL | R/W | √ | √ | | FFH |
| FFFFF025H | Port DL mode register H | PMDLH | R/W | √ | √ | | FFH |
| FFFFF028H | Port CS mode register | PMCS | R/W | √ | √ | | FFH |
| FFFFF02AH | Port CT mode register | PMCT | R/W | √ | √ | | FFH |
| FFFFF02CH | Port CM mode register | PMCM | R/W | √ | √ | | FFH |
| FFFFF044H | Port DL mode control register | PMCDL | R/W | | | √ | 0000H |
| FFFFF044H | Port DL mode control register L | PMCDLL | R/W | √ | √ | | 00H |
| FFFFF045H | Port DL mode control register H | PMCDLH | R/W | √ | √ | | 00H |
| FFFFF048H | Port CS mode control register | PMCCS | R/W | √ | √ | | 00H |
| FFFFF04AH | Port CT mode control register | PMCCT | R/W | √ | √ | | 00H |
| FFFFF04CH | Port CM mode control register | PMCCM | R/W | √ | √ | | 00H |
| FFFFF066H | Bus size configuration register | BSC | R/W | | | √ | 5555H |
| FFFFF06EH | System wait control register | VSWC | R/W | √ | √ | | 77H |
| FFFFF100H | Interrupt mask register 0 | IMR0 | R/W | | | √ | FFFFH |
| FFFFF100H | Interrupt mask register 0L | IMR0L | R/W | √ | √ | | FFH |
| FFFFF101H | Interrupt mask register 0H | IMR0H | R/W | √ | √ | | FFH |
| FFFFF102H | Interrupt mask register 1 | IMR1 | R/W | | | √ | FFFFH |
| FFFFF102H | Interrupt mask register 1L | IMR1L | R/W | √ | √ | | FFH |
| FFFFF103H | Interrupt mask register 1H | IMR1H | R/W | √ | √ | | FFH |
| FFFFF106H | Interrupt mask register 3 | IMR3 | R/W | | | √ | FFFFH |
| FFFFF106H | Interrupt mask register 3L | IMR3L | R/W | √ | √ | | FFH |
| FFFFF110H | Interrupt control register | WDT1IC | R/W | √ | √ | | 47H |
| FFFFF112H | Interrupt control register | PIC0 | R/W | √ | √ | | 47H |
| FFFFF114H | Interrupt control register | PIC1 | R/W | √ | √ | | 47H |
| FFFFF116H | Interrupt control register | PIC2 | R/W | √ | √ | | 47H |
| FFFFF118H | Interrupt control register | PIC3 | R/W | √ | √ | | 47H |
| FFFFF11AH | Interrupt control register | PIC4 | R/W | √ | √ | | 47H |
| FFFFF11CH | Interrupt control register | PIC5 | R/W | √ | √ | | 47H |
| FFFFF11EH | Interrupt control register | PIC6 | R/W | √ | √ | | 47H |
| FFFFF120H | Interrupt control register | TM0IC00 | R/W | √ | √ | | 47H |
| FFFFF122H | Interrupt control register | TM0IC01 | R/W | √ | √ | | 47H |
| FFFFF124H | Interrupt control register | TM0IC10 | R/W | √ | √ | | 47H |
| FFFFF126H | Interrupt control register | TM0IC11 | R/W | √ | √ | | 47H |
| FFFFF128H | Interrupt control register | TM5IC0 | R/W | √ | √ | | 47H |

**Note** The output latch is 00H or 0000H. When input, the pin status is read.

(2/8)

| Address | Function Register Name | Symbol | R/W | Operable Bit Unit | | | After Reset |
|---|---|---|---|---|---|---|---|
| | | | | 1 | 8 | 16 | |
| FFFFF12AH | Interrupt control register | TM5IC1 | R/W | √ | √ | | 47H |
| FFFFF12CH | Interrupt control register | CSI0IC0 | R/W | √ | √ | | 47H |
| FFFFF12EH | Interrupt control register | CSI0IC1 | R/W | √ | √ | | 47H |
| FFFFF130H | Interrupt control register | SREIC0 | R/W | √ | √ | | 47H |
| FFFFF132H | Interrupt control register | SRIC0 | R/W | √ | √ | | 47H |
| FFFFF134H | Interrupt control register | STIC0 | R/W | √ | √ | | 47H |
| FFFFF136H | Interrupt control register | SREIC1 | R/W | √ | √ | | 47H |
| FFFFF138H | Interrupt control register | SRIC1 | R/W | √ | √ | | 47H |
| FFFFF13AH | Interrupt control register | STIC1 | R/W | √ | √ | | 47H |
| FFFFF13CH | Interrupt control register | TMHIC0 | R/W | √ | √ | | 47H |
| FFFFF13EH | Interrupt control register | TMHIC1 | R/W | √ | √ | | 47H |
| FFFFF140H | Interrupt control register | CSIAIC0 | R/W | √ | √ | | 47H |
| FFFFF142H | Interrupt control register | IICIC0[Note 1] | R/W | √ | √ | | 47H |
| FFFFF144H | Interrupt control register | ADIC | R/W | √ | √ | | 47H |
| FFFFF146H | Interrupt control register | KRIC | R/W | √ | √ | | 47H |
| FFFFF148H | Interrupt control register | WTIIC | R/W | √ | √ | | 47H |
| FFFFF14AH | Interrupt control register | WTIC | R/W | √ | √ | | 47H |
| FFFFF14CH | Interrupt control register | BRGIC | R/W | √ | √ | | 47H |
| FFFFF170H | Interrupt control register | LVIIC | R/W | √ | √ | | 47H |
| FFFFF172H | Interrupt control register | PIC7 | R/W | √ | √ | | 47H |
| FFFFF174H | Interrupt control register | TP0OVIC | R/W | √ | √ | | 47H |
| FFFFF176H | Interrupt control register | TP0CCIC0 | R/W | √ | √ | | 47H |
| FFFFF178H | Interrupt control register | TP0CCIC1 | R/W | √ | √ | | 47H |
| FFFFF1FAH | In-service priority register | ISPR | R | √ | √ | | 00H |
| FFFFF1FCH | Command register | PRCMD | W | | √ | | Undefined |
| FFFFF1FEH | Power save control register | PSC | R/W | √ | √ | | 00H |
| FFFFF200H | A/D converter mode register | ADM | R/W | √ | √ | | 00H |
| FFFFF201H | Analog input channel specification register | ADS | R/W | √ | √ | | 00H |
| FFFFF202H | Power fail comparison mode register | PFM | R/W | √ | √ | | 00H |
| FFFFF203H | Power fail comparison threshold register | PFT | R/W | | √ | | 00H |
| FFFFF204H | A/D conversion result register | ADCR | R | | | √ | Undefined |
| FFFFF205H | A/D conversion result register H | ADCRH | R | | √ | | Undefined |
| FFFFF300H | Key return mode register | KRM | R/W | √ | √ | | 00H |
| FFFFF308H | Selector operation control register 0 | SELCNT0 | R/W | √ | √ | | 00H |
| FFFFF30AH | Selector operation control register 1 | SELCNT1 | R/W | √ | √ | | 00H |
| FFFFF318H | Digital noise elimination control register | NFC | R/W | √ | √ | | 00H |
| FFFFF400H | Port 0 register | P0 | R/W | √ | √ | | 00H[Note 2] |
| FFFFF406H | Port 3 register | P3 | R/W | | | √ | 0000H[Note 2] |
| FFFFF406H | Port 3 register L | P3L | R/W | √ | √ | | 00H[Note 2] |
| FFFFF407H | Port 3 register H | P3H | R/W | √ | √ | | 00H[Note 2] |
| FFFFF408H | Port 4 register | P4 | R/W | √ | √ | | 00H[Note 2] |

**Notes 1.** Only in the μPD703308Y, 70F3306Y, 70F3308Y

**2.** The output latch is 00H or 0000H. When input, the pin status is read.

(3/8)

| Address | Function Register Name | Symbol | R/W | Operable Bit Unit | | | After Reset |
|---|---|---|---|---|---|---|---|
| | | | | 1 | 8 | 16 | |
| FFFFF40AH | Port 5 register | P5 | R/W | √ | √ | | 00H[Note] |
| FFFFF40EH | Port 7 register | P7 | R | | √ | | Undefined |
| FFFFF412H | Port 9 register | P9 | R/W | | | √ | 0000H[Note] |
| FFFFF412H | Port 9 register L | P9L | R/W | √ | √ | | 00H[Note] |
| FFFFF413H | Port 9 register H | P9H | R/W | √ | √ | | 00H[Note] |
| FFFFF420H | Port 0 mode register | PM0 | R/W | √ | √ | | FEH |
| FFFFF426H | Port 3 mode register | PM3 | R/W | | | √ | FFFFH |
| FFFFF426H | Port 3 mode register L | PM3L | R/W | √ | √ | | FFH |
| FFFFF427H | Port 3 mode register H | PM3H | R/W | √ | √ | | FFH |
| FFFFF428H | Port 4 mode register | PM4 | R/W | √ | √ | | FFH |
| FFFFF42AH | Port 5 mode register | PM5 | R/W | √ | √ | | FFH |
| FFFFF432H | Port 9 mode register | PM9 | R/W | | | √ | FFFFH |
| FFFFF432H | Port 9 mode register L | PM9L | R/W | √ | √ | | FFH |
| FFFFF433H | Port 9 mode register H | PM9H | R/W | √ | √ | | FFH |
| FFFFF440H | Port 0 mode control register | PMC0 | R/W | √ | √ | | 00H |
| FFFFF446H | Port 3 mode control register | PMC3 | R/W | | | √ | 0000H |
| FFFFF446H | Port 3 mode control register L | PMC3L | R/W | √ | √ | | 00H |
| FFFFF447H | Port 3 mode control register H | PMC3H | R/W | √ | √ | | 00H |
| FFFFF448H | Port 4 mode control register | PMC4 | R/W | √ | √ | | 00H |
| FFFFF44AH | Port 5 mode control register | PMC5 | R/W | √ | √ | | 00H |
| FFFFF452H | Port 9 mode control register | PMC9 | R/W | | | √ | 0000H |
| FFFFF452H | Port 9 mode control register L | PMC9L | R/W | √ | √ | | 00H |
| FFFFF453H | Port 9 mode control register H | PMC9H | R/W | √ | √ | | 00H |
| FFFFF466H | Port 3 function control register | PFC3 | R/W | √ | √ | | 00H |
| FFFFF46AH | Port 5 function control register | PFC5 | R/W | √ | √ | | 00H |
| FFFFF472H | Port 9 function control register | PFC9 | R/W | | | √ | 0000H |
| FFFFF472H | Port 9 function control register L | PFC9L | R/W | √ | √ | | 00H |
| FFFFF473H | Port 9 function control register H | PFC9H | R/W | √ | √ | | 00H |
| FFFFF484H | Data wait control register 0 | DWC0 | R/W | | | √ | 7777H |
| FFFFF488H | Address wait control register | AWC | R/W | | | √ | FFFFH |
| FFFFF48AH | Bus cycle control register | BCC | R/W | | | √ | AAAAH |
| FFFFF580H | 8-bit timer H mode register 0 | TMHMD0 | R/W | √ | √ | | 00H |
| FFFFF581H | 8-bit timer H carrier control register 0 | TMCYC0 | R/W | √ | √ | | 00H |
| FFFFF582H | 8-bit timer H compare register 00 | CMP00 | R/W | | √ | | 00H |
| FFFFF583H | 8-bit timer H compare register 01 | CMP01 | R/W | | √ | | 00H |
| FFFFF590H | 8-bit timer H mode register 1 | TMHMD1 | R/W | √ | √ | | 00H |
| FFFFF591H | 8-bit timer H carrier control register 1 | TMCYC1 | R/W | √ | √ | | 00H |
| FFFFF592H | 8-bit timer H compare register 10 | CMP10 | R/W | | √ | | 00H |
| FFFFF593H | 8-bit timer H compare register 11 | CMP11 | R/W | | √ | | 00H |
| FFFFF5A0H | TMP0 control register 0 | TP0CTL0 | R/W | √ | √ | | 00H |
| FFFFF5A1H | TMP0 control register 1 | TP0CTL1 | R/W | √ | √ | | 00H |
| FFFFF5A2H | TMP0 I/O control register 0 | TP0IOC0 | R/W | √ | √ | | 00H |

**Note** The output latch is 00H or 0000H. When input, the pin status is read.

(4/8)

| Address | Function Register Name | Symbol | R/W | Operable Bit Unit | | | After Reset |
|---|---|---|---|---|---|---|---|
| | | | | 1 | 8 | 16 | |
| FFFFF5A3H | TMP0 I/O control register 1 | TP0IOC1 | R/W | √ | √ | | 00H |
| FFFFF5A4H | TMP0 I/O control register 2 | TP0IOC2 | R/W | √ | √ | | 00H |
| FFFFF5A5H | TMP0 option register 0 | TP0OPT0 | R/W | √ | √ | | 00H |
| FFFFF5A6H | TMP0 capture/compare register 0 | TP0CCR0 | R/W | | | √ | 0000H |
| FFFFF5A8H | TMP0 capture/compare register 1 | TP0CCR1 | R/W | | | √ | 0000H |
| FFFFF5AAH | TMP0 counter read buffer register | TP0CNT | R | | | √ | 0000H |
| FFFFF5C0H | 16-bit timer counter 5 | TM5 | R | | | √ | 0000H |
| FFFFF5C0H | 8-bit timer counter 50 | TM50 | R | | √ | | 00H |
| FFFFF5C1H | 8-bit timer counter 51 | TM51 | R | | √ | | 00H |
| FFFFF5C2H | 16-bit timer compare register 5 | CR5 | R/W | | | √ | 0000H |
| FFFFF5C2H | 8-bit timer compare register 50 | CR50 | R/W | | √ | | 00H |
| FFFFF5C3H | 8-bit timer compare register 51 | CR51 | R/W | | √ | | 00H |
| FFFFF5C4H | Timer clock selection register 5 | TCL5 | R/W | | | √ | 0000H |
| FFFFF5C4H | Timer clock selection register 50 | TCL50 | R/W | | √ | | 00H |
| FFFFF5C5H | Timer clock selection register 51 | TCL51 | R/W | | √ | | 00H |
| FFFFF5C6H | 16-bit timer mode control register 5 | TMC5 | R/W | | | √ | 0000H |
| FFFFF5C6H | 8-bit timer mode control register 50 | TMC50 | R/W | √ | √ | | 00H |
| FFFFF5C7H | 8-bit timer mode control register 51 | TMC51 | R/W | √ | √ | | 00H |
| FFFFF600H | 16-bit timer counter 00 | TM00 | R | | | √ | 0000H |
| FFFFF602H | 16-bit timer capture/compare register 000 | CR000 | R/W | | | √ | 0000H |
| FFFFF604H | 16-bit timer capture/compare register 001 | CR001 | R/W | | | √ | 0000H |
| FFFFF606H | 16-bit timer mode control register 00 | TMC00 | R/W | √ | √ | | 00H |
| FFFFF607H | Prescaler mode register 00 | PRM00 | R/W | √ | √ | | 00H |
| FFFFF608H | Capture/compare control register 00 | CRC00 | R/W | √ | √ | | 00H |
| FFFFF609H | 16-bit timer output control register 00 | TOC00 | R/W | √ | √ | | 00H |
| FFFFF610H | 16-bit timer counter 01 | TM01 | R | | | √ | 0000H |
| FFFFF612H | 16-bit timer capture/compare register 010 | CR010 | R/W | | | √ | 0000H |
| FFFFF614H | 16-bit timer capture/compare register 011 | CR011 | R/W | | | √ | 0000H |
| FFFFF616H | 16-bit timer mode control register 01 | TMC01 | R/W | √ | √ | | 00H |
| FFFFF617H | Prescaler mode register 01 | PRM01 | R/W | √ | √ | | 00H |
| FFFFF618H | Capture/compare control register 01 | CRC01 | R/W | √ | √ | | 00H |
| FFFFF619H | 16-bit timer output control register 01 | TOC01 | R/W | √ | √ | | 00H |
| FFFFF680H | Watch timer operation mode register | WTM | R/W | √ | √ | | 00H |
| FFFFF6C0H | Oscillation stabilization time selection register | OSTS | R/W | | √ | | **Note** |
| FFFFF6C1H | Watchdog timer clock selection register | WDCS | R/W | | √ | | 00H |
| FFFFF6C2H | Watchdog timer mode register 1 | WDTM1 | R/W | √ | √ | | 00H |
| FFFFF6D0H | Watchdog timer mode register 2 | WDTM2 | R/W | | √ | | 67H |
| FFFFF6D1H | Watchdog timer enable register | WDTE | R/W | | √ | | 9AH |
| FFFFF6E0H | Real-time output buffer register L0 | RTBL0 | R/W | √ | √ | | 00H |
| FFFFF6E2H | Real-time output buffer register H0 | RTBH0 | R/W | √ | √ | | 00H |
| FFFFF6E4H | Real-time output port mode register 0 | RTPM0 | R/W | √ | √ | | 00H |

**Note** The value can be set to 00H or 01H by the option byte or a mask option setting.
For details, refer to **CHAPTER 28 MASK OPTION/OPTION BYTE.**

(5/8)

| Address | Function Register Name | Symbol | R/W | Operable Bit Unit | | | | After Reset |
|---|---|---|---|---|---|---|---|---|
| | | | | 1 | 8 | 16 | 32 | |
| FFFFF6E5H | Real-time output port control register 0 | RTPC0 | R/W | √ | √ | | | 00H |
| FFFFF706H | Port 3 function control expansion register | PFCE3 | R/W | √ | √ | | | 00H |
| FFFFF802H | System status register | SYS | R/W | √ | √ | | | 00H |
| FFFFF806H | PLL control register | PLLCTL | R/W | √ | √ | | | 01H |
| FFFFF80CH | Ring-OSC mode register | RCM | R/W | √ | √ | | | 00H |
| FFFFF820H | Power save mode register | PSMR | R/W | √ | √ | | | 00H |
| FFFFF828H | Processor clock control register | PCC | R/W | √ | √ | | | 03H |
| FFFFF82EH | CPU operation clock status register | CCLS | R | √ | √ | | | 00H |
| FFFFF840H | Correction address register 0 | CORAD0 | R/W | | | | √ | 00000000H |
| FFFFF840H | Correction address register 0L | CORAD0L | R/W | | | √ | | 0000H |
| FFFFF842H | Correction address register 0H | CORAD0H | R/W | | | √ | | 0000H |
| FFFFF844H | Correction address register 1 | CORAD1 | R/W | | | | √ | 00000000H |
| FFFFF844H | Correction address register 1L | CORAD1L | R/W | | | √ | | 0000H |
| FFFFF846H | Correction address register 1H | CORAD1H | R/W | | | √ | | 0000H |
| FFFFF848H | Correction address register 2 | CORAD2 | R/W | | | | √ | 00000000H |
| FFFFF848H | Correction address register 2L | CORAD2L | R/W | | | √ | | 0000H |
| FFFFF84AH | Correction address register 2H | CORAD2H | R/W | | | √ | | 0000H |
| FFFFF84CH | Correction address register 3 | CORAD3 | R/W | | | | √ | 00000000H |
| FFFFF84CH | Correction address register 3L | CORAD3L | R/W | | | √ | | 0000H |
| FFFFF84EH | Correction address register 3H | CORAD3H | R/W | | | √ | | 0000H |
| FFFFF860H | Reset noise elimination control register | RNZC | R/W | | √ | | | 00H |
| FFFFF870H | Clock monitor mode register | CLM | R/W | √ | √ | | | 00H |
| FFFFF880H | Correction control register | CORCN | R/W | √ | √ | | | 00H |
| FFFFF888H | Reset source flag register | RESF | R/W | √ | √ | | | **Note** |
| FFFFF890H | Low-voltage detection register | LVIM | R/W | √ | √ | | | 00H |
| FFFFF891H | Low-voltage detection level selection register | LVIS | R/W | | √ | | | 00H |
| FFFFF8B0H | Interval timer BRG mode register | PRSM | R/W | | √ | | | 00H |
| FFFFF8B1H | Interval timer BRG compare register | PRSCM | R/W | | √ | | | 00H |
| FFFFFA00H | Asynchronous serial interface mode register 0 | ASIM0 | R/W | √ | √ | | | 01H |
| FFFFFA02H | Receive buffer register 0 | RXB0 | R | | √ | | | FFH |
| FFFFFA03H | Asynchronous serial interface status register 0 | ASIS0 | R | | √ | | | 00H |
| FFFFFA04H | Transmit buffer register 0 | TXB0 | R/W | | √ | | | FFH |
| FFFFFA05H | Asynchronous serial interface transmit status register 0 | ASIF0 | R | √ | √ | | | 00H |
| FFFFFA06H | Clock select register 0 | CKSR0 | R/W | | √ | | | 00H |
| FFFFFA07H | Baud rate generator control register 0 | BRGC0 | R/W | | √ | | | FFH |
| FFFFFA08H | LIN operation control register 0 | ASICL0 | R/W | √ | √ | | | 16H |
| FFFFFA10H | Asynchronous serial interface mode register 1 | ASIM1 | R/W | √ | √ | | | 01H |
| FFFFFA12H | Receive buffer register 1 | RXB1 | R | | √ | | | FFH |
| FFFFFA13H | Asynchronous serial interface status register 1 | ASIS1 | R | | √ | | | 00H |
| FFFFFA14H | Transmit buffer register 1 | TXB1 | R/W | | √ | | | FFH |
| FFFFFA15H | Asynchronous serial interface transmit status register 1 | ASIF1 | R | √ | √ | | | 00H |
| FFFFFA16H | Clock select register 1 | CKSR1 | R/W | | √ | | | 00H |

**Note** The value varies depending on the reset source (refer to **22.3 (1) Reset source flag register (RESF)**).

(6/8)

| Address | Function Register Name | Symbol | R/W | Operable Bit Unit | | | After Reset |
|---|---|---|---|---|---|---|---|
| | | | | 1 | 8 | 16 | |
| FFFFFA17H | Baud rate generator control register 1 | BRGC1 | R/W | | √ | | FFH |
| FFFFFB00H | TIP00 noise elimination control register | P0NFC | R/W | √ | √ | | 00H |
| FFFFFB04H | TIP01 noise elimination control register | P1NFC | R/W | √ | √ | | 00H |
| FFFFFC00H | External interrupt falling edge specification register 0 | INTF0 | R/W | √ | √ | | 00H |
| FFFFFC06H | External interrupt falling edge specification register 3 | INTF3 | R/W | √ | √ | | 00H |
| FFFFFC13H | External interrupt falling edge specification register 9H | INTF9H | R/W | √ | √ | | 00H |
| FFFFFC20H | External interrupt rising edge specification register 0 | INTR0 | R/W | √ | √ | | 00H |
| FFFFFC26H | External interrupt rising edge specification register 3 | INTR3 | R/W | √ | √ | | 00H |
| FFFFFC33H | External interrupt rising edge specification register 9H | INTR9H | R/W | √ | √ | | 00H |
| FFFFFC40H | Pull-up resistor option register 0 | PU0 | R/W | √ | √ | | 00H |
| FFFFFC46H | Pull-up resistor option register 3 | PU3 | R/W | √ | √ | | 00H |
| FFFFFC48H | Pull-up resistor option register 4 | PU4 | R/W | √ | √ | | 00H |
| FFFFFC4AH | Pull-up resistor option register 5 | PU5 | R/W | √ | √ | | 00H |
| FFFFFC52H | Pull-up resistor option register 9 | PU9 | R/W | | | √ | 0000H |
| FFFFFC52H | Pull-up resistor option register 9L | PU9L | R/W | √ | √ | | 00H |
| FFFFFC53H | Pull-up resistor option register 9H | PU9H | R/W | √ | √ | | 00H |
| FFFFFC67H | Port 3 function register H | PF3H | R/W | √ | √ | | 00H |
| FFFFFC68H | Port 4 function register | PF4 | R/W | √ | √ | | 00H |
| FFFFFC6AH | Port 5 function register | PF5 | R/W | √ | √ | | 00H |
| FFFFFC73H | Port 9 function register H | PF9H | R/W | √ | √ | | 00H |
| FFFFFD00H | Clocked serial interface mode register 00 | CSIM00 | R/W | √ | √ | | 00H |
| FFFFFD01H | Clocked serial interface clock selection register 0 | CSIC0 | R/W | √ | √ | | 00H |
| FFFFFD02H | Clocked serial interface receive buffer register 0 | SIRB0 | R | | | √ | 0000H |
| FFFFFD02H | Clocked serial interface receive buffer register 0L | SIRB0L | R | | √ | | 00H |
| FFFFFD04H | Clocked serial interface transmit buffer register 0 | SOTB0 | R/W | | | √ | 0000H |
| FFFFFD04H | Clocked serial interface transmit buffer register 0L | SOTB0L | R/W | | √ | | 00H |
| FFFFFD06H | Clocked serial interface read-only receive buffer register 0 | SIRBE0 | R | | | √ | 0000H |
| FFFFFD06H | Clocked serial interface read-only receive buffer register 0L | SIRBE0L | R | | √ | | 00H |
| FFFFFD08H | Clocked serial interface initial transmit buffer register 0 | SOTBF0 | R/W | | | √ | 0000H |
| FFFFFD08H | Clocked serial interface initial transmit buffer register 0L | SOTBF0L | R/W | | √ | | 00H |
| FFFFFD0AH | Serial I/O shift register 0 | SIO00 | R/W | | | √ | 00H |
| FFFFFD0AH | Serial I/O shift register 0L | SIO00L | R/W | | √ | | 0000H |
| FFFFFD10H | Clocked serial interface mode register 01 | CSIM01 | R/W | √ | √ | | 00H |
| FFFFFD11H | Clocked serial interface clock selection register 1 | CSIC1 | R/W | √ | √ | | 00H |
| FFFFFD12H | Clocked serial interface receive buffer register 1 | SIRB1 | R | | | √ | 0000H |
| FFFFFD12H | Clocked serial interface receive buffer register 1L | SIRB1L | R | | √ | | 00H |
| FFFFFD14H | Clocked serial interface transmit buffer register 1 | SOTB1 | R/W | | | √ | 0000H |
| FFFFFD14H | Clocked serial interface transmit buffer register 1L | SOTB1L | R/W | | √ | | 00H |
| FFFFFD16H | Clocked serial interface read-only receive buffer register 1 | SIRBE1 | R | | | √ | 0000H |
| FFFFFD16H | Clocked serial interface read-only receive buffer register 1L | SIRBE1L | R | | √ | | 00H |
| FFFFFD18H | Clocked serial interface initial transmit buffer register 1 | SOTBF1 | R/W | | | √ | 0000H |
| FFFFFD18H | Clocked serial interface initial transmit buffer register 1L | SOTBF1L | R/W | | √ | | 00H |

(7/8)

| Address | Function Register Name | Symbol | R/W | Operable Bit Unit | | | After Reset |
|---|---|---|---|---|---|---|---|
| | | | | 1 | 8 | 16 | |
| FFFFFD1AH | Serial I/O shift register 1 | SIO01 | R/W | | | √ | 00H |
| FFFFFD1AH | Serial I/O shift register 1L | SIO01L | R/W | | √ | | 0000H |
| FFFFFD40H | Serial operation mode specification register 0 | CSIMA0 | R/W | √ | √ | | 00H |
| FFFFFD41H | Serial status register 0 | CSIS0 | R/W | √ | √ | | 00H |
| FFFFFD42H | Serial trigger register 0 | CSIT0 | R/W | √ | √ | | 00H |
| FFFFFD43H | Divisor selection register 0 | BRGCA0 | R/W | | √ | | 03H |
| FFFFFD44H | Automatic data transfer address point specification register 0 | ADTP0 | R/W | | √ | | 00H |
| FFFFFD45H | Automatic data transfer interval specification register 0 | ADTI0 | R/W | √ | √ | | 00H |
| FFFFFD46H | Serial I/O shift register A0 | SIOA0 | R/W | | √ | | 00H |
| FFFFFD47H | Automatic data transfer address count register 0 | ADTC0 | R | | √ | | 00H |
| FFFFFD80H | IIC shift register 0 | IIC0Note | R/W | | √ | | 00H |
| FFFFFD82H | IIC control register 0 | IICC0Note | R/W | √ | √ | | 00H |
| FFFFFD83H | Slave address register 0 | SVA0Note | R/W | | √ | | 00H |
| FFFFFD84H | IIC clock selection register 0 | IICCL0Note | R/W | √ | √ | | 00H |
| FFFFFD85H | IIC function expansion register 0 | IICX0Note | R/W | √ | √ | | 00H |
| FFFFFD86H | IIC status register 0 | IICS0Note | R | √ | √ | | 00H |
| FFFFFD8AH | IIC flag register 0 | IICF0Note | R/W | √ | √ | | 00H |
| FFFFFE00H | CSIA0 buffer RAM 0 | CSIA0B0 | R/W | | | √ | Undefined |
| FFFFFE00H | CSIA0 buffer RAM 0L | CSIA0B0L | R/W | | √ | | Undefined |
| FFFFFE01H | CSIA0 buffer RAM 0H | CSIA0B0H | R/W | | √ | | Undefined |
| FFFFFE02H | CSIA0 buffer RAM 1 | CSIA0B1 | R/W | | | √ | Undefined |
| FFFFFE02H | CSIA0 buffer RAM 1L | CSIA0B1L | R/W | | √ | | Undefined |
| FFFFFE03H | CSIA0 buffer RAM 1H | CSIA0B1H | R/W | | √ | | Undefined |
| FFFFFE04H | CSIA0 buffer RAM 2 | CSIA0B2 | R/W | | | √ | Undefined |
| FFFFFE04H | CSIA0 buffer RAM 2L | CSIA0B2L | R/W | | √ | | Undefined |
| FFFFFE05H | CSIA0 buffer RAM 2H | CSIA0B2H | R/W | | √ | | Undefined |
| FFFFFE06H | CSIA0 buffer RAM 3 | CSIA0B3 | R/W | | | √ | Undefined |
| FFFFFE06H | CSIA0 buffer RAM 3L | CSIA0B3L | R/W | | √ | | Undefined |
| FFFFFE07H | CSIA0 buffer RAM 3H | CSIA0B3H | R/W | | √ | | Undefined |
| FFFFFE08H | CSIA0 buffer RAM 4 | CSIA0B4 | R/W | | | √ | Undefined |
| FFFFFE08H | CSIA0 buffer RAM 4L | CSIA0B4L | R/W | | √ | | Undefined |
| FFFFFE09H | CSIA0 buffer RAM 4H | CSIA0B4H | R/W | | √ | | Undefined |
| FFFFFE0AH | CSIA0 buffer RAM 5 | CSIA0B5 | R/W | | | √ | Undefined |
| FFFFFE0AH | CSIA0 buffer RAM 5L | CSIA0B5L | R/W | | √ | | Undefined |
| FFFFFE0BH | CSIA0 buffer RAM 5H | CSIA0B5H | R/W | | √ | | Undefined |
| FFFFFE0CH | CSIA0 buffer RAM 6 | CSIA0B6 | R/W | | | √ | Undefined |
| FFFFFE0CH | CSIA0 buffer RAM 6L | CSIA0B6L | R/W | | √ | | Undefined |
| FFFFFE0DH | CSIA0 buffer RAM 6H | CSIA0B6H | R/W | | √ | | Undefined |
| FFFFFE0EH | CSIA0 buffer RAM 7 | CSIA0B7 | R/W | | | √ | Undefined |
| FFFFFE0EH | CSIA0 buffer RAM 7L | CSIA0B7L | R/W | | √ | | Undefined |
| FFFFFE0FH | CSIA0 buffer RAM 7H | CSIA0B7H | R/W | | √ | | Undefined |

**Note** Only in the μPD703308Y, 70F3306Y, 70F3308Y

(8/8)

| Address | Function Register Name | Symbol | R/W | Operable Bit Unit | | | After Reset |
|---|---|---|---|---|---|---|---|
| | | | | 1 | 8 | 16 | |
| FFFFFE10H | CSIA0 buffer RAM 8 | CSIA0B8 | R/W | | | √ | Undefined |
| FFFFFE10H | CSIA0 buffer RAM 8L | CSIA0B8L | R/W | | √ | | Undefined |
| FFFFFE11H | CSIA0 buffer RAM 8H | CSIA0B8H | R/W | | √ | | Undefined |
| FFFFFE12H | CSIA0 buffer RAM 9 | CSIA0B9 | R/W | | | √ | Undefined |
| FFFFFE12H | CSIA0 buffer RAM 9L | CSIA0B9L | R/W | | √ | | Undefined |
| FFFFFE13H | CSIA0 buffer RAM 9H | CSIA0B9H | R/W | | √ | | Undefined |
| FFFFFE14H | CSIA0 buffer RAM A | CSIA0BA | R/W | | | √ | Undefined |
| FFFFFE14H | CSIA0 buffer RAM AL | CSIA0BAL | R/W | | √ | | Undefined |
| FFFFFE15H | CSIA0 buffer RAM AH | CSIA0BAH | R/W | | √ | | Undefined |
| FFFFFE16H | CSIA0 buffer RAM B | CSIA0BB | R/W | | | √ | Undefined |
| FFFFFE16H | CSIA0 buffer RAM BL | CSIA0BBL | R/W | | √ | | Undefined |
| FFFFFE17H | CSIA0 buffer RAM BH | CSIA0BBH | R/W | | √ | | Undefined |
| FFFFFE18H | CSIA0 buffer RAM C | CSIA0BC | R/W | | | √ | Undefined |
| FFFFFE18H | CSIA0 buffer RAM CL | CSIA0BCL | R/W | | √ | | Undefined |
| FFFFFE19H | CSIA0 buffer RAM CH | CSIA0BCH | R/W | | √ | | Undefined |
| FFFFFE1AH | CSIA0 buffer RAM D | CSIA0BD | R/W | | | √ | Undefined |
| FFFFFE1AH | CSIA0 buffer RAM DL | CSIA0BDL | R/W | | √ | | Undefined |
| FFFFFE1BH | CSIA0 buffer RAM DH | CSIA0BDH | R/W | | √ | | Undefined |
| FFFFFE1CH | CSIA0 buffer RAM E | CSIA0BE | R/W | | | √ | Undefined |
| FFFFFE1CH | CSIA0 buffer RAM EL | CSIA0BEL | R/W | | √ | | Undefined |
| FFFFFE1DH | CSIA0 buffer RAM EH | CSIA0BEH | R/W | | √ | | Undefined |
| FFFFFE1EH | CSIA0 buffer RAM F | CSIA0BF | R/W | | | √ | Undefined |
| FFFFFE1EH | CSIA0 buffer RAM FL | CSIA0BFL | R/W | | √ | | Undefined |
| FFFFFE1FH | CSIA0 buffer RAM FH | CSIA0BFH | R/W | | √ | | Undefined |
| FFFFFF44H | Pull-up resistor option register DL | PUDL | R/W | | | √ | 0000H |
| FFFFFF44H | Pull-up resistor option register DLL | PUDLL | R/W | √ | √ | | 00H |
| FFFFFF45H | Pull-up resistor option register DLH | PUDLH | R/W | √ | √ | | 00H |
| FFFFFF48H | Pull-up resistor option register CS | PUCS | R/W | √ | √ | | 00H |
| FFFFFF4AH | Pull-up resistor option register CT | PUCT | R/W | √ | √ | | 00H |
| FFFFFF4CH | Pull-up resistor option register CM | PUCM | R/W | √ | √ | | 00H |

### 3.4.7 Special registers

Special registers are registers that prevent invalid data from being written when an inadvertent program loop occurs.

The V850ES/KF1+ has the following six special registers.

- Power save control register (PSC)
- Processor clock control register (PCC)
- Watchdog timer mode register (WDTM1)
- Clock monitor mode register (CLM)
- Reset source flag register (RESF)
- Low-voltage detection register (LVIM)

Moreover, there is also the PRCMD register, which is a protection register for write operations to the special registers that prevents the application system from unexpectedly stopping due to an inadvertent program loop. Write access to the special registers is performed with a special sequence and illegal store operations are notified to the SYS register.

#### (1) Setting data to special registers

Setting data to a special register is done in the following sequence.

<1> Prepare the data to be set to the special register in a general-purpose register.

<2> Write the data prepared in step <1> to the PRCMD register.

<3> Write the setting data to the special register (using following instructions).

- Store instruction (ST/SST instruction)
- Bit manipulation instruction (SET1/CLR1/NOT1 instruction)

<4> to <8> Insert NOP instructions (5 instructions)[Note].

**Note** When switching to the IDLE mode or the STOP mode (PSC.STP bit = 1), 5 NOP instructions must be inserted immediately after switching is performed.

**[Description Example] When using PSC register (standby mode setting)**

```
     ST.B r11, PSMR[r0]   ; PSMR register setting (IDLE, STOP mode setting)
<1>  MOV 0x02, r10
<2>  ST.B r10, PRCMD[r0]  ; PRCMD register write
<3>  ST.B r10, PSC[r0]    ; PSC register setting
<4>  NOP(Note)            ; Dummy instruction
<5>  NOP(Note)            ; Dummy instruction
<6>  NOP(Note)            ; Dummy instruction
<7>  NOP(Note)            ; Dummy instruction
<8>  NOP(Note)            ; Dummy instruction
     (next instruction)
```

No special sequence is required to read special registers.

**Note** When switching to the IDLE mode or the STOP mode (PSC.STP bit = 1), 5 NOP instructions must be inserted immediately after switching is performed.

**Cautions 1. Interrupts are not acknowledged for the store instruction for the PRCMD register. This is because continuous execution of store instructions by the program in steps <2> and <3> above is assumed. If another instruction is placed between step <2> and <3>, the above sequence may not be realized when an interrupt is acknowledged for that instruction, which may cause malfunction.**

**2. The data written to the PRCMD register is dummy data, but use the same register as the general-purpose register used for setting data to the special register (step <3>) when writing to the PRCMD register (step <2>). The same applies to when using a general-purpose register for addressing.**

**(2) Command register (PRCMD)**

The PRCMD register is an 8-bit register used to prevent data from being written to registers that may have a large influence on the system, possibly causing the application system to unexpectedly stop, when an inadvertent program loop occurs. Only the first write operation to the special register following the execution of a previously executed write operation to the PRCMD register, is valid.

As a result, register values can be overwritten only using a preset sequence, preventing invalid write operations.

This register can only be written in 8-bit units (if it is read, an undefined value is returned).

After reset: Undefined W Address: FFFFF1FCH

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PRCMD | REG7 | REG6 | REG5 | REG4 | REG3 | REG2 | REG1 | REG0 |

**(3) System status register (SYS)**

This register is allocated with status flags showing the operating state of the entire system.

This register can be read or written in 8-bit or 1-bit units.

After reset: 00H R/W Address: FFFFF802H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | <0> |
|---|---|---|---|---|---|---|---|---|
| SYS | 0 | 0 | 0 | 0 | 0 | 0 | 0 | PRERR |

| PRERR | Detection of protection error |
|---|---|
| 0 | Protection error has not occurred |
| 1 | Protection error has occurred |

The operation conditions of the PRERR flag are described below.

**(a) Set conditions (PRERR = 1)**

(i) When a write operation to the special register takes place without write operation being performed to the PRCMD register (when step <3> is performed without performing step <2> as described in **3.4.7 (1) Setting data to special registers**).

(ii) When a write operation (including bit manipulation instruction) to an on-chip peripheral I/O register other than a special register is performed following write to the PRCMD register (when <3> in **3.4.7 (1) Setting data to special registers** is not a special register).

**Remark** Regarding the special registers other than the WDTM register (PCC and PSC registers), even if on-chip peripheral I/O register read (except bit manipulation instruction) (internal RAM access, etc.) is performed in between write to the PRCMD register and write to a special register, the PRERR flag is not set and setting data can be written to the special register.

**(b) Clear conditions (PRERR = 0)**

(i) When 0 is written to the PRERR flag

(ii) When system reset is performed

**Cautions 1. If 0 is written to the PRERR bit of the SYS register that is not a special register immediately following write to the PRCMD register, the PRERR bit becomes 0 (write priority).**

**2. If data is written to the PRCMD register that is not a special register immediately following write to the PRCMD register, the PRERR bit becomes 1.**

### 3.4.8 Cautions

**(1) Wait when accessing register**

Be sure to set the following register before using the V850ES/KF1+.

- System wait control register (VSWC)

After setting the VSWC register, set the other registers as required.

When using an external bus, set the VSWC register and then set the various pins to the control mode by setting the port-related registers.

**(a) System wait control register (VSWC)**

The VSWC register controls the bus access wait time for the on-chip peripheral I/O registers.

Access to the on-chip peripheral I/O register lasts 3 clocks (during no wait), but in the V850ES/KF1+, waits are required according to the internal system clock frequency. Set the values shown below to the VSWC register according to the internal system clock frequency that is used.

This register can be read or written in 8-bit units (Address: FFFFF06EH, After reset: 77H).

| Operation Conditions | Internal System Clock Frequency ($f_{CLK}$) | VSWC Register Setting |
|---|---|---|
| REGC = $V_{DD}$ = 5 V ±10% | 32 kHz ≤ $f_{CLK}$ < 16.6 MHz | 00H |
| | 16.6 MHz ≤ $f_{CLK}$ ≤ 20 MHz | 01H |
| REGC = $V_{DD}$ = 4.0 to 5.5 V | 32 kHz ≤ $f_{CPU}$ < 16 MHz | 00H |
| REGC = Capacity, $V_{DD}$ = 4.0 to 5.5 V | 32 kHz ≤ $f_{CLK}$ < 8 MHz | 00H |
| REGC = $V_{DD}$ = 2.7 to 4.0 V | 32 kHz ≤ $f_{CLK}$ ≤ 8 MHz | 00H |

**(b) Access to special on-chip peripheral I/O register**

This product has two types of internal system buses.

One type is for the CPU bus and the other is for the peripheral bus to interface with low-speed peripheral hardware.

Since the CPU bus clock and peripheral bus clock are asynchronous, if a conflict occurs during access between the CPU and peripheral hardware, illegal data may be passed unexpectedly. Therefore, when accessing peripheral hardware that may cause a conflict, the number of access cycles is changed so that the data is received/passed correctly in the CPU. As a result, the CPU does not shift to the next instruction processing and enters the wait status. When this wait status occurs, the number of execution clocks of the instruction is increased by the number of wait clocks.

Note this with caution when performing real-time processing.

When accessing a special on-chip peripheral I/O register, additional waits may be required further to the waits set by the VSWC register.

The access conditions at that time and the method to calculate the number of waits to be inserted (number of CPU clocks) are shown below.

| Peripheral Function | Register Name | Access | k |
|---|---|---|---|
| Watchdog timer 1 (WDT1) | WDTM1 | Write | 1 to 5 |
| | <Calculation of number of waits><br>{(1/$f_X$) × 2/((2 + m)/$f_{CPU}$)} + 1<br>$f_X$: Main clock oscillation frequency | | |
| Watchdog timer 2 (WDT2) | WDTM2 | Write | 3 (fixed) |
| 16-bit timer/event counter P0 (TMP0) | TP0CCR0, TP0CCR1, TP0CNT | Read | 1 |
| | <Calculation of number of waits><br>{(1/$f_{XX}$)/((2 + m)/$f_{CPU}$)} + 1 | | |
| | TP0CCR0, TP0CCR1 | Write | 0 to 2 |
| | <Calculation of number of waits><br>{(1/$f_{XX}$) × 5/((2 + m)/$f_{CPU}$)}<br>A wait occurs when performing continuous write to same register | | |
| 16-bit timer/event counters 00, 01 (TM00, TM01) | TMC00, TMC01 | Read-modify-write | 1 (fixed)<br>A wait occurs during write |
| Clocked serial interface 0 with automatic transmit/receive function (CSIA0) | CSIA0B0 to CSIA0BF | Write[Note 1] | 0 to 18 (when performing continuous write via write instruction) |
| | <Calculation of number of waits><br>{(1/$f_{SCKA}$) × 5 – (4 + m)/$f_{CPU}$)}/{((2 + m)/$f_{CPU}$)}<br>However, 1 wait if $f_{CPU}$ = $f_{XX}$ if the CSIS0.CKSA01 and CSIS0.CKSA00 bits are 00.<br>$f_{SCKA}$: CSIA selection clock frequency | | |
| | CSIA0B0 to CSIA0BF | Write[Note 1] | 0 to 20 (when conflict occurs between write instruction and write via receive operation) |
| | <Calculation of number of waits><br>{((1/$f_{SCKA}$) × 5)/((2 + m)/$f_{CPU}$)}<br>$f_{SCKA}$: CSIA selection clock frequency | | |
| $I^2C0$[Note 2] | IICS0 | Read | 1 (fixed) |
| Asynchronous serial interfaces 0, 1 (UART0, UART1) | ASIS0, ASIS1 | Read | 1 (fixed) |
| Real-time output function 0 (RTO0) | RTBL0, RTBH0 | Write (when RTPC0.RTPOE0 bit = 0) | 1 |
| A/D converter | ADM, ADS, PFM, PFT | Write | 1 to 2 |
| | ADCR, ADCRH | Read | 1 to 2 |
| | <Calculation of maximum number of waits><br>{(1/$f_{XX}$) × 2/[(2 + m)/$f_{CPU}$]} + 1 | | |

Number of waits to be added = (2 + m) × k [clocks]

**Notes 1.** If fetched from the internal RAM, the number of waits is as shown above.
If fetched from the external memory, the number of waits may be fewer than the number shown above.
The effect of the external memory access cycle differs depending on the wait settings, etc. However, the number of waits above is the maximum value.

**2.** $I^2C0$ is available only in the μPD703308Y, 70F3306Y, and 70F3308Y.

**Caution When the CPU operates on the subclock and no clock is input to the X1 pin, do not access a register in which a wait occurs using an access method that causes a wait. If a wait occurs, it can only be released by a reset.**

**Remark** In the calculation for the number of waits:

$f_{CPU}$: CPU clock frequency

m: Set value of bits 2 to 0 of the VSWC register

$f_{CLK}$: Internal system clock

When $f_{CLK} < 16.6$ MHz: m = 0

When $f_{CLK} \geq 16.6$ MHz: m = 1

The digits below the decimal point are truncated if less than $(1/f_{CPU})/(2 + m)$ or rounded up if larger than $(1/f_{CPU})/(2 + m)$ when multiplied by $(1/f_{CPU})$.

### (2) Restriction on conflict between sld instruction and interrupt request

#### (a) Description

If a conflict occurs between the decode operation of an instruction in <2> immediately before the sld instruction following an instruction in <1> and an interrupt request before the instruction in <1> is complete, the execution result of the instruction in <1> may not be stored in a register.

Instruction <1>

- ld instruction: ld.b, ld.h, ld.w, ld.bu, ld.hu
- sld instruction: sld.b, sld.h, sld.w, sld.bu, sld.hu
- Multiplication instruction: mul, mulh, mulhi, mulu

Instruction <2>

| | | | |
|---|---|---|---|
| mov reg1, reg2 | not reg1, reg2 | satsubr reg1, reg2 | satsub reg1, reg2 |
| satadd reg1, reg2 | satadd imm5, reg2 | or reg1, reg2 | xor reg1, reg2 |
| and reg1, reg2 | tst reg1, reg2 | subr reg1, reg2 | sub reg1, reg2 |
| add reg1, reg2 | add imm5, reg2 | cmp reg1, reg2 | cmp imm5, reg2 |
| mulh reg1, reg2 | shr imm5, reg2 | sar imm5, reg2 | shl imm5, reg2 |

<Example>

<i> ld.w [r11], r10
•
•
•
<ii> mov r10, r28
<iii> sld.w 0x28, r10

If the decode operation of the mov instruction <ii> immediately before the sld instruction <iii> and an interrupt request conflict before execution of the ld instruction <i> is complete, the execution result of instruction <i> may not be stored in a register.

#### (b) Countermeasure

When executing the sld instruction immediately after instruction <ii>, avoid the above operation using either of the following methods.

- Insert a nop instruction immediately before the sld instruction.
- Do not use the same register as the sld instruction destination register in the above instruction <ii> executed immediately before the sld instruction.

# CHAPTER 4 PORT FUNCTIONS

## 4.1 Features

- Input-only ports: 8 pins
- I/O ports: 59 pins
  - Fixed to N-ch open-drain output: 2
  - Switchable to N-ch open-drain output: 6
- Input/output can be specified in 1-bit units

## 4.2 Basic Port Configuration

The V850ES/KF1+ incorporates a total of 67 I/O port pins consisting of ports 0, 3 to 5, 7, 9, CM, CS, CT, and DL (including 8 input-only port pins). The port configuration is shown below.

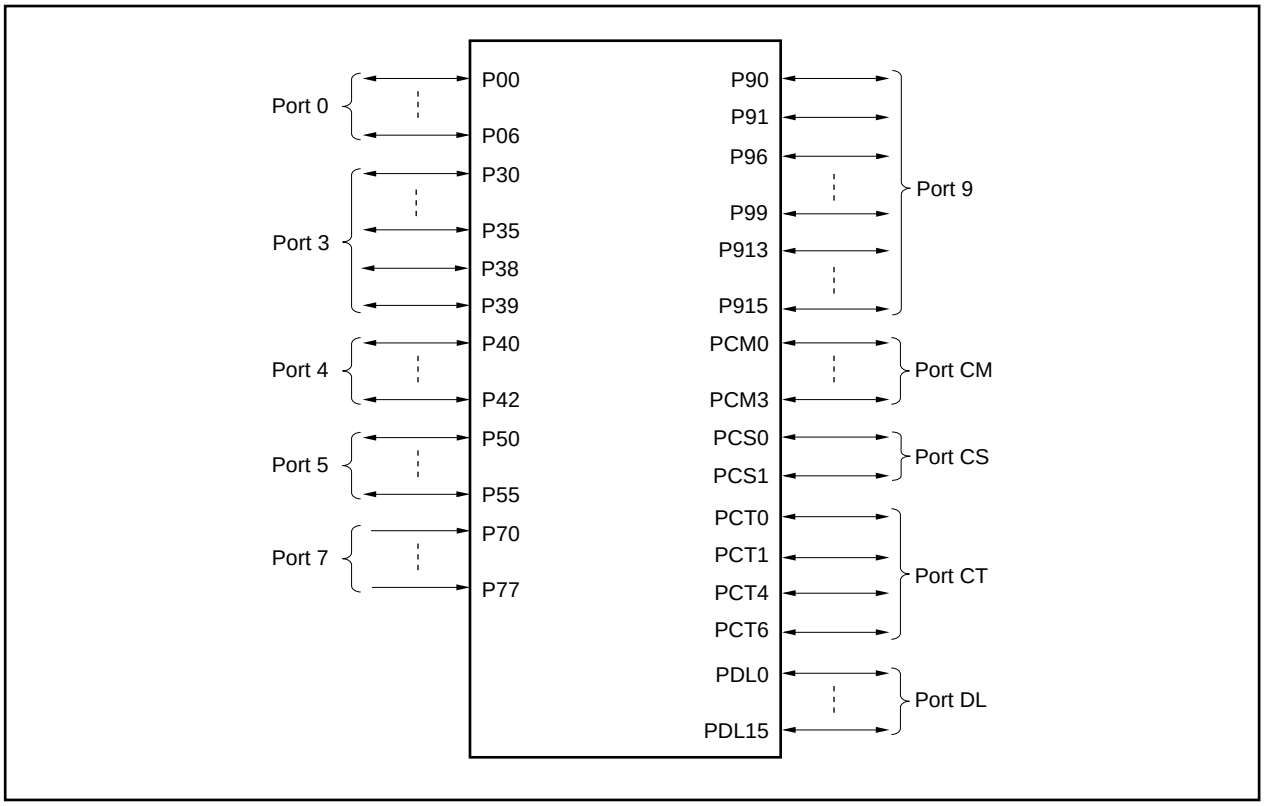



**Table 4-1. Pin I/O Buffer Power Supplies of V850ES/KF1+**

| Power Supply | Corresponding Pins |
|---|---|
| $AV_{REF0}$ | Port 7 |
| $EV_{DD}$ | $\overline{RESET}$, ports 0, 3 to 5, 9, CM, CS, CT, DL |

## 4.3 Port Configuration

**Table 4-2. Port Configuration**

| Item | Configuration |
|---|---|
| Control registers | Port n register (Pn: n = 0, 3 to 5, 7, 9, CM, CS, CT, DL)<br>Port n mode register (PMn: n = 0, 3 to 5, 9, CM, CS, CT, DL)<br>Port n mode control register (PMCn: n = 0, 3 to 5, 9, CM, CS, CT, DL)<br>Port n function control register (PFCn: n = 3, 5, 9)<br>Port n function register (PFn: n = 3 to 5, 9)<br>Port 3 function control expansion register (PFCE3)<br>Pull-up resistor option register (PUn: n = 0, 3 to 5, 9, CM, CS, CT, DL) |
| Ports | Input only: 8<br>I/O: 59 |
| Pull-up resistors | Software control: 48 |

**(1) Port n register (Pn)**

Data I/O with external devices is performed by writing to and reading from the Pn register. The Pn register is configured of a port latch that retains the output data and a circuit that reads the pin status.

Each bit of the Pn register corresponds to one pin of port n and can be read or written in 1-bit units.

After reset: 00H[Note] (output latch) R/W

| | 7 | 6 | 5 | 7 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Pn | Pn7 | Pn6 | Pn5 | Pn4 | Pn3 | Pn2 | Pn1 | Pn0 |

| Pnm | Control of output data (in output mode) |
|---|---|
| 0 | 0 is output |
| 1 | 1 is output |

**Note** Input-only port pins are undefined.

Writing to and reading from the Pn register is executed as follows independent of the setting of the PMCn register.

**Table 4-3. Reading to/Writing from Pn Register**

| Setting of PMn Register | Writing to Pn Register | Reading from Pn Register |
|---|---|---|
| Output mode<br>(PMnm bit = 0) | Write to the output latch[Note].<br>In the port mode (PMCnm bit = 0), the contents of the output latch are output from the pin. | The value of the output latch is read. |
| Input mode<br>(PMnm bit = 1) | Write to the output latch.<br>The status of the pin is not affected[Note]. | The pin status is read. |

**Note** The value written to the output latch is retained until a value is next written to the output latch.

**(2) Port n mode register (PMn)**

PMn specifies the input mode/output mode of the port.

Each bit of the PMn register corresponds to one pin of port n and can be specified in 1-bit units.

After reset: FFH R/W

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PMn | PMn7 | PMn6 | PMn5 | PMn4 | PMn3 | PMn2 | PMn1 | PMn0 |

| PMnm | Control of I/O mode |
|---|---|
| 0 | Output mode |
| 1 | Input mode |

**(3) Port n mode control register (PMCn)**

PMCn specifies the port mode/alternate function.

Each bit of the PMCn register corresponds to one pin of port n and can be specified in 1-bit units.

After reset: 00H R/W

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PMCn | PMCn7 | PMCn6 | PMCn5 | PMCn4 | PMCn3 | PMCn2 | PMCn1 | PMCn0 |

| PMCnm | Specification of operation mode |
|---|---|
| 0 | Port mode |
| 1 | Alternate function mode |

**(4) Port n function control register (PFCn)**

PFCn is a register that specifies the alternate function to be used when one pin has two or more alternate functions.

Each bit of the PFCn register corresponds to one pin of port n and can be specified in 1-bit units.

After reset: 00H R/W

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PFCn | PFCn7 | PFCn6 | PFCn5 | PFCn4 | PFCn3 | PFCn2 | PFCn1 | PFCn0 |

| PFCnm | Specification of alternate function |
|---|---|
| 0 | Alternate function 1 |
| 1 | Alternate function 2 |

**(5) Port n function control expansion register (PFCEn)**

PFCEn is a register that specifies the alternate function to be used when one pin has three or more alternate functions.

Each bit of the PFCEn register corresponds to one pin of port n and can be specified in 1-bit units.

After reset: 00H R/W

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PFCEn | PFCEn7 | PFCEn6 | PFCEn5 | PFCEn4 | PFCEn3 | PFCEn2 | PFCEn1 | PFCEn0 |
| PFCn | PFCn7 | PFCn6 | PFCn5 | PFCn4 | PFCn3 | PFCn2 | PFCn1 | PFCn0 |

| PFCEnm | PFCnm | Specification of alternate function |
|---|---|---|
| 0 | 0 | Alternate function 1 |
| 0 | 1 | Alternate function 2 |
| 1 | 0 | Alternate function 3 |
| 1 | 1 | Alternate function 4 |

**(6) Port n function register (PFn)**

PFn is a register that specifies normal output/N-ch open-drain output.

Each bit of the PFn register corresponds to one pin of port n and can be specified in 1-bit units.

After reset: 00H R/W

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PFn | PFn7 | PFn6 | PFn5 | PFn4 | PFn3 | PFn2 | PFn1 | PFn0 |

| PFnm[Note] | Control of normal output/N-ch open-drain output |
|---|---|
| 0 | Normal output (CMOS output) |
| 1 | N-ch open-drain output |

**Note** The PFnm bit is valid only when the PMn.PMnm bit is 0 (output mode) regardless of the setting of the PMCn register. When the PMnm bit is 1 (input mode), the set value in the PFn register is invalid.

**Example** <1> When the value of the PFn register is valid
PFnm bit = 1 … N-ch open-drain output is specified.
PMnm bit = 0 … Output mode is specified.
PMCnm bit = 0 or 1

<2> When the value of the PFn register is invalid
PFnm bit = 0 … N-ch open-drain output is specified.
PMnm bit = 1 … Input mode is specified.
PMCnm bit = 0 or 1

**(7) Pull-up resistor option register (PUn)**

PUn is a register that specifies the connection of an on-chip pull-up resistor.

Each bit of the PUn register corresponds to one pin of port n and can be specified in 1-bit units.

After reset: 00H R/W

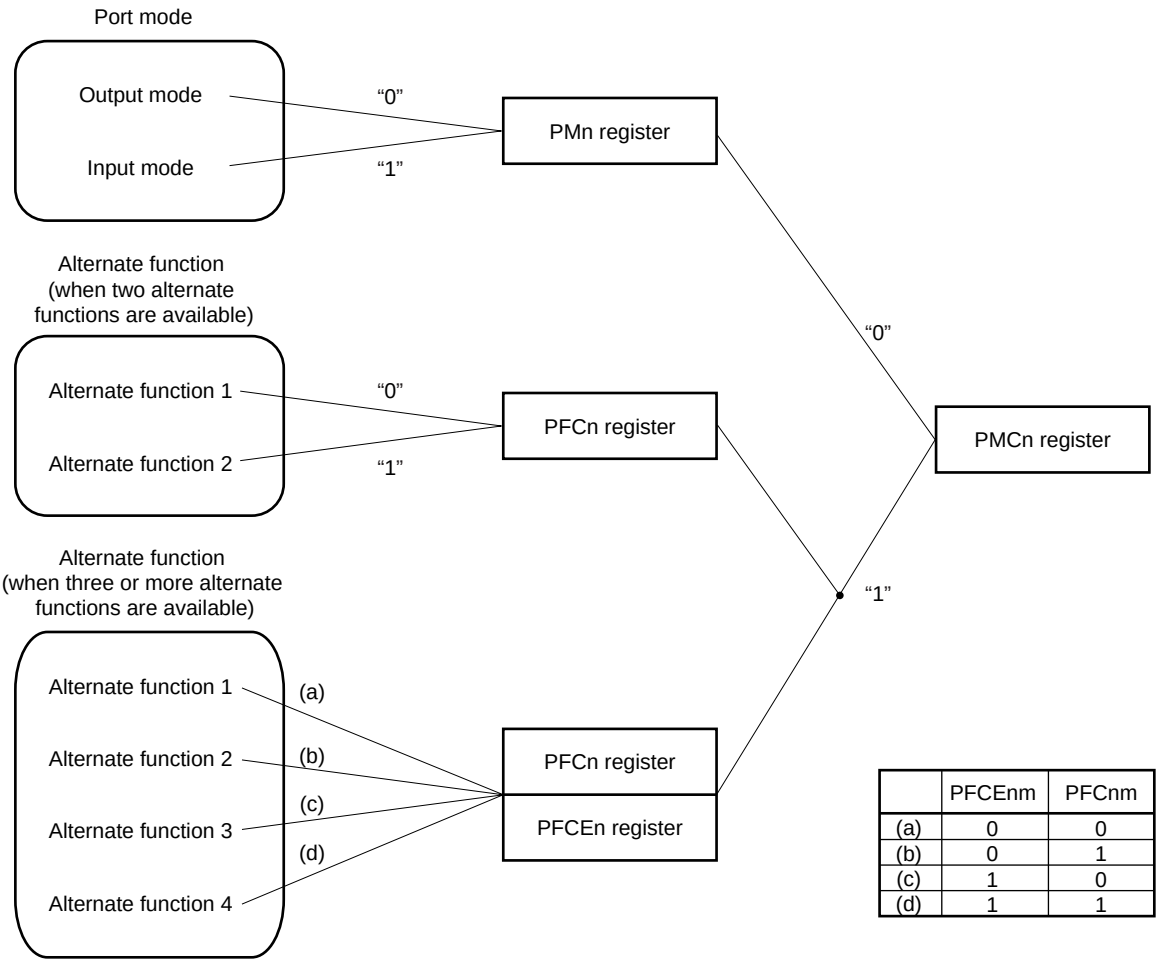
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PUn | PUn7 | PUn6 | PUn5 | PUn4 | PUn3 | PUn2 | PUn1 | PUn0 |

| PUnm | Control of on-chip pull-up resistor connection |
|---|---|
| 0 | Not connected |
| 1 | Connected |

**(8) Port settings**

Set the ports as follows.

**Figure 4-1. Register Settings and Pin Functions**




| | PFCEnm | PFCnm |
|---|---|---|
| (a) | 0 | 0 |
| (b) | 0 | 1 |
| (c) | 1 | 0 |
| (d) | 1 | 1 |

**Remark** Switch to the alternate function using the following procedure.

<1> Set the PFCn and PFCEn registers.
<2> Set the PMCn register.
<3> Set the INTRn or INTFn register (to specify an external interrupt pin).

If the PMCn register is set first, an unintended function may be set while the PFCn and PFCEn registers are being set.

### 4.3.1 Port 0

Port 0 is a 7-bit I/O port for which I/O settings can be controlled in 1-bit units.

Port 0 includes the following alternate functions.

**Table 4-4. Alternate-Function Pins of Port 0**

| Pin No. | Pin Name | Alternate Function | I/O | PULL[Note 1] | Remark | Block Type |
|---|---|---|---|---|---|---|
| 3 | P00[Note 2] | TOH0 | Output | Yes | – | D0-U |
| 4 | P01 | TOH1 | Output | | | D0-U |
| 5 | P02 | NMI | Input | | Analog noise elimination | D1-SUIL |
| 6 | P03 | INTP0 | Input | | | D1-SUIL |
| 7 | P04 | INTP1 | Input | | | D1-SUIL |
| 17 | P05 | INTP2 | Input | | | D1-SUIL |
| 18 | P06 | INTP3 | Input | | Analog/digital noise elimination | D1-SUIL |

**Notes 1.** Software pull-up function

**2.** Only the P00 pin outputs a low level after reset (other port pins are in input mode). Therefore, the low-level output from the P00 pin after reset can be used as a dummy reset signal from the CPU.

**Caution P02 to P06 have hysteresis characteristics when the alternate function is input, but not in the port mode.**

#### (1) Port 0 register (P0)

After reset: 00H (output latch) R/W Address: FFFFF400H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| P0 | 0 | P06 | P05 | P04 | P03 | P02 | P01 | P00 |

| P0n | Control of output data (in output mode) (n = 0 to 6) |
|---|---|
| 0 | 0 is output |
| 1 | 1 is output |

#### (2) Port 0 mode register (PM0)

After reset: FEH R/W Address: FFFFF420H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PM0 | 1 | PM06 | PM05 | PM04 | PM03 | PM02 | PM01 | PM00 |

| PM0n | Control of I/O mode (n = 0 to 6) |
|---|---|
| 0 | Output mode |
| 1 | Input mode |

### (3) Port 0 mode control register (PMC0)

After reset: 00H R/W Address: FFFFF440H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PMC0 | 0 | PMC06 | PMC05 | PMC04 | PMC03 | PMC02 | PMC01 | PMC00 |

| PMC06 | Specification of P06 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | INTP3 input |

| PMC05 | Specification of P05 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | INTP2 input |

| PMC04 | Specification of P04 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | INTP1 input |

| PMC03 | Specification of P03 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | INTP0 input |

| PMC02 | Specification of P02 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | NMI input |

| PMC01 | Specification of P01 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | TOH1 output |

| PMC00 | Specification of P00 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | TOH0 output |

### (4) Pull-up resistor option register 0 (PU0)

After reset: 00H R/W Address: FFFFFC40H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PU0 | 0 | PU06 | PU05 | PU04 | PU03 | PU02 | PU01 | PU00 |

| PU0n | Control of on-chip pull-up resistor connection (n = 0 to 6) |
|---|---|
| 0 | Not connected |
| 1 | Connected |

### 4.3.2 Port 3

Port 3 is an 8-bit I/O port for which I/O settings can be controlled in 1-bit units.

Port 3 includes the following alternate functions.

**Table 4-5. Alternate-Function Pins of Port 3**

| Pin No. | Pin Name | Alternate Function | I/O | PULL[Note 1] | Remark | Block Type |
|---|---|---|---|---|---|---|
| 22 | P30 | TXD0 | Output | Yes | – | D0-U |
| 23 | P31 | RXD0/INTP7 | Input | | | D1-SUIHL |
| 24 | P32 | ASCK0/ADTRG/TO01 | I/O | | | E10-SUL |
| 25 | P33 | TI000/TO00/TIP00/TOP00 | I/O | | | G1010-SUL |
| 26 | P34 | TI001/TO00/TIP01/TOP01 | I/O | | | G1010-SUL |
| 27 | P35 | TI010/TO01 | I/O | | | E10-SUL |
| 28 | P38 | SDA0[Note 3] | I/O | No[Note 2] | N-ch open-drain output | D2-SNMUFH |
| 29 | P39 | SCL0[Note 3] | I/O | | | D2-SNMUFH |

**Notes 1.** Software pull-up function

**2.** An on-chip pull-up resistor can be provided by a mask option (only in the μPD703308, 703308Y).

**3.** Only in the μPD703308Y, 70F3306Y, 70F3308Y

**Caution P31 to P35, P38, and P39 have hysteresis characteristics when the alternate function is input, but not in the port mode.**

### (1) Port 3 register (P3)

After reset: 00H (output latch) R/W Address: P3 FFFFF406H,
P3L FFFFF406H, P3H FFFFF407H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| P3 (P3H[Note]) | 0 | 0 | 0 | 0 | 0 | 0 | P39 | P38 |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| (P3L) | 0 | 0 | P35 | P34 | P33 | P32 | P31 | P30 |

| P3n | Control of output data (in output mode) (n = 0 to 5, 8, 9) |
|---|---|
| 0 | 0 is output |
| 1 | 1 is output |

**Note** When reading from or writing to bits 8 to 15 of the P3 register in 8-bit or 1-bit units, specify these bits as bits 0 to 7 of the P3H register.

**Remark** The P3 register can be read or written in 16-bit units.
However, when the higher 8 bits and the lower 8 bits of the P3 register are used as the P3H register and as the P3L register, respectively, this register can be read or written in 8-bit or 1-bit units.

### (2) Port 3 mode register (PM3)

After reset: FFFFH R/W Address: PM3 FFFFF426H,
PM3L FFFFF426H, PM3H FFFFF427H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| PM3 (PM3H[Note]) | 1 | 1 | 1 | 1 | 1 | 1 | PM39 | PM38 |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| (PM3L) | 1 | 1 | PM35 | PM34 | PM33 | PM32 | PM31 | PM30 |

| PM3n | Control of I/O mode (n = 0 to 5, 8, 9) |
|---|---|
| 0 | Output mode |
| 1 | Input mode |

**Note** When reading from or writing to bits 8 to 15 of the PM3 register in 8-bit or 1-bit units, specify these bits as bits 0 to 7 of the PM3H register.

**Remark** The PM3 register can be read or written in 16-bit units.
When the higher 8 bits and the lower 8 bits of the PM3 register are used as the PM3H register and as the PM3L register, respectively, this register can be read or written in 8-bit or 1-bit units.

### (3) Port 3 mode control register (PMC3)

After reset: 0000H R/W Address: PMC3 FFFFF446H,
PMC3L FFFFF446H, PMC3H FFFFF447H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| PMC3 (PMC3H[Note 1]) | 0 | 0 | 0 | 0 | 0 | 0 | PMC39[Note 2] | PMC38[Note 2] |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| (PMC3L) | 0 | 0 | PMC35 | PMC34 | PMC33 | PMC32 | PMC31 | PMC30 |

| PMC39 | Specification of P39 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | SCL0 I/O |

| PMC38 | Specification of P38 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | SDA0 I/O |

| PMC35 | Specification of P35 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | TI010 input/TO01 output |

| PMC34 | Specification of P34 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | TI001 input/TO00 output/TIP01 input/TOP01 output |

| PMC33 | Specification of P33 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | TI000 input/TO00 output/TIP00 input/TOP00 output |

| PMC32 | Specification of P32 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | ASCK0 input/ADTRG input/TO01 output |

| PMC31 | Specification of P31 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | RXD0 input/INTP7 input[Note 3] |

| PMC30 | Specification of P30 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | TXD0 output |

**Notes 1.** When reading from or writing to bits 8 to 15 of the PMC3 register in 8-bit or 1-bit units, specify these bits as bits 0 to 7 of the PMC3H register.

**2.** Valid only in the μPD703308Y, 70F3306Y, 70F3308Y. In all other products, set this bit to 0.

**3.** The INTP7 and RXD0 pins are alternate-function pins. When using the pin as the RXD0 pin, disable edge detection of the alternate-function INTP7 pin (clear the INTF3.INTF31 and INTR3.INTR31 bits to 0). When using the pin as the INTP7 pin, stop the UART0 receive operation (clear the ASIM0.RXE0 bit to 0).

**Remark** The PMC3 register can be read or written in 16-bit units.
When the higher 8 bits and the lower 8 bits of the PMC3 register are used as the PMC3H register and as the PMC3L register, respectively, this register can be read or written in 8-bit or 1-bit units.

**(4) Port 3 function register H (PF3H)**

After reset: 00H R/W Address: FFFFFC67H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PF3H | 0 | 0 | 0 | 0 | 0 | 0 | PF39 | PF38 |

| PF3n | Specification of normal port/alternate function (n = 8, 9) |
|---|---|
| 0 | When used as normal port (N-ch open-drain output) |
| 1 | When used as alternate-function (N-ch open-drain output) |

**Caution When using P38 and P39 as N-ch open-drain-output alternate-function pins, set in the following sequence.**
**Be sure to set the port latch to 1 before setting the pin to N-ch open-drain output.**
**P3n bit = 1 → PF3n bit = 1 → PMC3n bit = 1**

**(5) Port 3 function control register (PFC3)**

After reset: 00H R/W Address: FFFFF466H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PFC3 | 0 | 0 | PFC35 | PFC34 | PFC33 | PFC32 | 0 | 0 |

**Remark** For details of specification of alternate-function pins, refer to **4.3.2 (7) Specifying alternate-function pins of port 3**.

**(6) Port 3 function control expansion register (PFCE3)**

After reset: 00H R/W Address: FFFFF706H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PFCE3 | 0 | 0 | 0 | PFCE34 | PFCE33 | 0 | 0 | 0 |

**Remark** For details of specification of alternate-function pins, refer to **4.3.2 (7) Specifying alternate-function pins of port 3**.

**(7) Specifying alternate-function pins of port 3**

| PFC35 | Specification of Alternate-Function Pin of P35 Pin |
|---|---|
| 0 | TI010 input |
| 1 | TO01 output |

| PFCE34 | PFC34 | Specification of Alternate-Function Pin of P34 Pin |
|---|---|---|
| 0 | 0 | TI001 input |
| 0 | 1 | TO00 output |
| 1 | 0 | TIP01 input |
| 1 | 1 | TOP01 output |

| PFCE33 | PFC33 | Specification of Alternate-Function Pin of P33 Pin |
|---|---|---|
| 0 | 0 | TI000 input |
| 0 | 1 | TO00 output |
| 1 | 0 | TIP00 input |
| 1 | 1 | TOP00 output |

| PFC32 | Specification of Alternate-Function Pin of P32 Pin |
|---|---|
| 0 | ASCK0/ADTRG[Note] input |
| 1 | TO01 output |

**Note** The ASCK0 and ADTRG pins are alternate-function pins. When using the pin as the ASCK0 pin, disable the trigger input of the alternate-function ADTRG pin (clear the ADS.TRG bit to 0 or set the ADS.ADTMD bit to 1). When using the pin as the ADTRG pin, do not set the UART0 operation clock to external input (set the CKSR0.TPS03 to CKSR0.TPS00 bits to other than 1011).

**(8) Pull-up resistor option register 3 (PU3)**

After reset: 00H R/W Address: FFFFFC46H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PU3 | 0 | 0 | PU35 | PU34 | PU33 | PU32 | PU31 | PU30 |

| PU3n | Control of on-chip pull-up resistor connection (n = 0 to 5) |
|---|---|
| 0 | Not connected |
| 1 | Connected |

**Caution An on-chip pull-up resistor can be provided for P38 and P39 by a mask option (only in the μPD703308, 703308Y).**

### 4.3.3 Port 4

Port 4 is a 3-bit I/O port for which I/O settings can be controlled in 1-bit units.

Port 4 includes the following alternate functions.

**Table 4-6. Alternate-Function Pins of Port 4**

| Pin No. | Pin Name | Alternate Function | I/O | PULL[Note] | Remark | Block Type |
|---|---|---|---|---|---|---|
| 19 | P40 | SI00 | Input | Yes | – | D1-SUL |
| 20 | P41 | SO00 | Output | | N-ch open-drain output can be selected. | D0-UF |
| 21 | P42 | $\overline{\text{SCK00}}$ | I/O | | | D2-SUFL |

**Note** Software pull-up function

**Caution P40 and P42 have hysteresis characteristics when the alternate function is input, but not in the port mode.**

### (1) Port 4 register (P4)

After reset: 00H (output latch) R/W Address: FFFFF408H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| P4 | 0 | 0 | 0 | 0 | 0 | P42 | P41 | P40 |

| P4n | Control of output data (in output mode) (n = 0 to 2) |
|---|---|
| 0 | 0 is output |
| 1 | 1 is output |

### (2) Port 4 mode register (PM4)

After reset: FFH R/W Address: FFFFF428H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PM4 | 1 | 1 | 1 | 1 | 1 | PM42 | PM41 | PM40 |

| PM4n | Control of I/O mode (n = 0 to 2) |
|---|---|
| 0 | Output mode |
| 1 | Input mode |

### (3) Port 4 mode control register (PMC4)

After reset: 00H R/W Address: FFFFF448H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PMC4 | 0 | 0 | 0 | 0 | 0 | PMC42 | PMC41 | PMC40 |

| PMC42 | Specification of P42 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | $\overline{\text{SCK00}}$ I/O |

| PMC41 | Specification of P41 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | SO00 output |

| PMC40 | Specification of P40 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | SI00 input |

### (4) Port 4 function register (PF4)

After reset: 00H R/W Address: FFFFFC68H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PF4 | 0 | 0 | 0 | 0 | 0 | PF42 | PF41 | 0 |

| PF4n | Control of normal output/N-ch open-drain output (n = 1, 2) |
|---|---|
| 0 | Normal output |
| 1 | N-ch open-drain output |

**Caution When using P41 and P42 as N-ch open-drain-output alternate-function pins, set in the following sequence.**
**Be sure to set the port latch to 1 before setting the pin to N-ch open-drain output.**
**P4n bit = 1 → PF4n bit = 1 → PMC4n bit = 1**

### (5) Pull-up resistor option register 4 (PU4)

After reset: 00H R/W Address: FFFFFC48H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PU4 | 0 | 0 | 0 | 0 | 0 | PU42 | PU41 | PU40 |

| PU4n | Control of on-chip pull-up resistor connection (n = 0 to 2) |
|---|---|
| 0 | Not connected |
| 1 | Connected |

### 4.3.4 Port 5

Port 5 is a 6-bit I/O port for which I/O settings can be controlled in 1-bit units.

Port 5 includes the following alternate functions.

**Table 4-7. Alternate-Function Pins of Port 5**

| Pin No. | Pin Name | Alternate Function | I/O | PULL[Note] | Remark | Block Type |
|---|---|---|---|---|---|---|
| 32 | P50 | TI011/RTP00/KR0 | I/O | Yes | – | E10-SULT |
| 33 | P51 | TI50/RTP01/KR1 | I/O | | | E10-SULT |
| 34 | P52 | TO50/RTP02/KR2 | I/O | | | E00-SUT |
| 35 | P53 | SIA0/RTP03/KR3 | I/O | | | E10-SULT |
| 36 | P54 | SOA0/RTP04/KR4 | I/O | | N-ch open-drain output can be selected. | E00-SUFT |
| 37 | P55 | $\overline{\text{SCKA0}}$/RTP05/KR5 | I/O | | | E20-SUFLT |

**Note** Software pull-up function

#### (1) Port 5 register (P5)

After reset: 00H (output latch) R/W Address: FFFFF40AH

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| P5 | 0 | 0 | P55 | P54 | P53 | P52 | P51 | P50 |

| P5n | Control of output data (in output mode) (n = 0 to 5) |
|---|---|
| 0 | 0 is output |
| 1 | 1 is output |

#### (2) Port 5 mode register (PM5)

After reset: FFH R/W Address: FFFFF42AH

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PM5 | 1 | 1 | PM55 | PM54 | PM53 | PM52 | PM51 | PM50 |

| PM5n | Control of I/O mode (n = 0 to 5) |
|---|---|
| 0 | Output mode |
| 1 | Input mode |

### (3) Port 5 mode control register (PMC5)

After reset: 00H R/W Address: FFFFF44AH

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PMC5 | 0 | 0 | PMC55 | PMC54 | PMC53 | PMC52 | PMC51 | PMC50 |

| PMC55 | Specification of P55 pin operation mode |
|---|---|
| 0 | I/O port/KR5 input |
| 1 | $\overline{\text{SCKA0}}$ I/O/RTP05 output |

| PMC54 | Specification of P54 pin operation mode |
|---|---|
| 0 | I/O port/KR4 input |
| 1 | SOA0 output/RTP04 output |

| PMC53 | Specification of P53 pin operation mode |
|---|---|
| 0 | I/O port/KR3 input |
| 1 | SIA0 input/RTP03 output |

| PMC52 | Specification of P52 pin operation mode |
|---|---|
| 0 | I/O port/KR2 input |
| 1 | TO50 output/RTP02 output |

| PMC51 | Specification of P51 pin operation mode |
|---|---|
| 0 | I/O port/KR1 input |
| 1 | TI50 input/RTP01 output |

| PMC50 | Specification of P50 pin operation mode |
|---|---|
| 0 | I/O port/KR0 input |
| 1 | TI011 input/RTP00 output |

### (4) Port 5 function register 5 (PF5)

After reset: 00H R/W Address: FFFFFC6AH

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PF5 | 0 | 0 | PF55 | PF54 | 0 | 0 | 0 | 0 |

| PF5n | Control of normal output/N-ch open-drain output (n = 4, 5) |
|---|---|
| 0 | Normal output |
| 1 | N-ch open-drain output |

**Cautions 1. Always set bits 0 to 3, 6, and 7 of the PF5 register to 0.**

**2. When using P54 and P55 as N-ch open-drain-output alternate-function pins, set in the following sequence.**

**Be sure to set the port latch to 1 before setting the pin to N-ch open-drain output.**

**P5n bit = 1 → PF5n bit = 1 → PMC5n bit = 1**

### (5) Port 5 function control register (PFC5)

After reset: 00H R/W Address: FFFFF46AH

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PFC5 | 0 | 0 | PFC55 | PFC54 | PFC53 | PFC52 | PFC51 | PFC50 |

| PFC55 | Specification of alternate-function pin of P55 pin |
|---|---|
| 0 | $\overline{\text{SCKA0}}$ I/O |
| 1 | RTP05 output |

| PFC54 | Specification of alternate-function pin of P54 pin |
|---|---|
| 0 | SOA0 output |
| 1 | RTP04 output |

| PFC53 | Specification of alternate-function pin of P53 pin |
|---|---|
| 0 | SIA0 input |
| 1 | RTP03 output |

| PFC52 | Specification of alternate-function pin of P52 pin |
|---|---|
| 0 | TO50 output |
| 1 | RTP02 output |

| PFC51 | Specification of alternate-function pin of P51 pin |
|---|---|
| 0 | TI50 input |
| 1 | RTP01 output |

| PFC50 | Specification of alternate-function pin of P50 pin |
|---|---|
| 0 | TI011 input |
| 1 | RTP00 output |

### (6) Pull-up resistor option register 5 (PU5)

After reset: 00H R/W Address: FFFFFC4AH

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PU5 | 0 | 0 | PU55 | PU54 | PU53 | PU52 | PU51 | PU50 |

| PU5n | Control of on-chip pull-up resistor connection (n = 0 to 5) |
|---|---|
| 0 | Not connected |
| 1 | Connected |

### 4.3.5 Port 7

Port 7 is an 8-bit input-only port for which all the pins are fixed to input.

Port 7 includes the following alternate functions.

**Table 4-8. Alternate-Function Pins of Port 7**

| Pin No. | Pin Name | Alternate Function | I/O | PULL[Note] | Remark | Block Type |
|---|---|---|---|---|---|---|
| 80 | P70 | ANI0 | Input | No | – | A-A |
| 79 | P71 | ANI1 | Input | | | A-A |
| 78 | P72 | ANI2 | Input | | | A-A |
| 77 | P73 | ANI3 | Input | | | A-A |
| 76 | P74 | ANI4 | Input | | | A-A |
| 75 | P75 | ANI5 | Input | | | A-A |
| 74 | P76 | ANI6 | Input | | | A-A |
| 73 | P77 | ANI7 | Input | | | A-A |

**Note** Software pull-up function

#### (1) Port 7 register (P7)

After reset: Undefined R Address: FFFFF40EH

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| P7 | P77 | P76 | P75 | P74 | P73 | P72 | P71 | P70 |

| P7n | Input data read (n = 0 to 7) |
|---|---|
| 0 | Input low level |
| 1 | Input high level |

### 4.3.6 Port 9

Port 9 is a 9-bit I/O port for which I/O settings can be controlled in 1-bit units.

Port 9 includes the following alternate functions.

**Table 4-9. Alternate-Function Pins of Port 9**

| Pin No. | Pin Name | Alternate Function | I/O | PULL[Note] | Remark | Block Type |
|---|---|---|---|---|---|---|
| 38 | P90 | TXD1/KR6 | I/O | No | – | Ex0-SUT |
| 39 | P91 | RXD1/KR7 | Input | | | Ex1-SUHT |
| 40 | P96 | TI51/TO51 | I/O | | | Ex0-SUT |
| 41 | P97 | SI01 | Input | | | Ex1-SUL |
| 42 | P98 | SO01 | Output | | N-ch open-drain output can be specified. | Ex0-UF |
| 43 | P99 | $\overline{\text{SCK01}}$ | I/O | | | Ex2-SUFL |
| 44 | P913 | INTP4 | Input | | Analog noise elimination | Ex1-SUIL |
| 45 | P914 | INTP5 | Input | | | Ex1-SUIL |
| 46 | P915 | INTP6 | Input | | | Ex1-SUIL |

**Note** Software pull-up function

**Caution P97, P99, and P913 to P915 have hysteresis characteristics when the alternate function is input, but not in the port mode.**

### (1) Port 9 register (P9)

After reset: 00H (output latch) R/W Address: P9 FFFFF412H,
P9L FFFFF412H, P9H FFFFF413H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| P9 (P9H[Note]) | P915 | P914 | P913 | 0 | 0 | 0 | P99 | P98 |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| (P9L) | P97 | P96 | 0 | 0 | 0 | 0 | P91 | P90 |

| P9n | Control of output data (in output mode) (n = 0, 1, 6 to 9, 13 to 15) |
|---|---|
| 0 | 0 is output |
| 1 | 1 is output |

**Note** When reading from or writing to bits 8 to 15 of the P9 register in 8-bit or 1-bit units, specify these bits as bits 0 to 7 of the P9H register.

**Remark** The P9 register can be read or written in 16-bit units.
However, when the higher 8 bits and the lower 8 bits of the P9 register are used as the P9H register and as the P9L register, respectively, these registers can be read or written in 8-bit or 1-bit units.

### (2) Port 9 mode register (PM9)

After reset: FFFFH R/W Address: PM9 FFFFF432H,
PM9L FFFFF432H, PM9H FFFFF433H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| PM9 (PM9H[Note]) | PM915 | PM914 | PM913 | 1 | 1 | 1 | PM99 | PM98 |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| (PM9L) | PM97 | PM96 | 1 | 1 | 1 | 1 | PM91 | PM90 |

| PM9n | Control of I/O mode (n = 0, 1, 6 to 9, 13 to 15) |
|---|---|
| 0 | Output mode |
| 1 | Input mode |

**Note** When reading from or writing to bits 8 to 15 of the PM9 register in 8-bit or 1-bit units, specify these bits as bits 0 to 7 of the PM9H register.

**Remark** The PM9 register can be read or written in 16-bit units.
However, when the higher 8 bits and the lower 8 bits of the PM9 register are used as the PM9H register and as the PM9L register, respectively, this register can be read or written in 8-bit or 1-bit units.

### (3) Port 9 mode control register (PMC9)

After reset: 0000H R/W Address: PMC9 FFFFF452H,
PMC9L FFFFF452H, PMC9H FFFFF453H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| PMC9 (PMC9H[Note]) | PMC915 | PMC914 | PMC913 | 0 | 0 | 0 | PMC99 | PMC98 |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| (PMC9L) | PMC97 | PMC96 | 0 | 0 | 0 | 0 | PMC91 | PMC90 |

| PMC915 | Specification of P915 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | INTP6 input |

| PMC914 | Specification of P914 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | INTP5 input |

| PMC913 | Specification of P913 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | INTP4 input |

| PMC99 | Specification of P99 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | $\overline{\text{SCK01}}$ I/O |

| PMC98 | Specification of P98 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | SO01 output |

| PMC97 | Specification of P97 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | SI01 input |

| PMC96 | Specification of P96 pin operation mode |
|---|---|
| 0 | I/O port/TI51 input |
| 1 | TO51 output |

| PMC91 | Specification of P91 pin operation mode |
|---|---|
| 0 | I/O port/KR7 input |
| 1 | RXD1 input |

| PMC90 | Specification of P90 pin operation mode |
|---|---|
| 0 | I/O port/KR6 input |
| 1 | TXD1 output |

**Note** When reading from or writing to bits 8 to 15 of the PMC9 register in 8-bit or 1-bit units, specify these bits as bits 0 to 7 of the PMC9H register.

**Remark** The PMC9 register can be read or written in 16-bit units.
However, when the higher 8 bits and the lower 8 bits of the PMC9 register are used as the PMC9H register and as the PMC9L register, respectively, these registers can be read or written in 8-bit or 1-bit units.

### (4) Port 9 function register H (PF9H)

After reset: 00H R/W Address: FFFFFC73H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PF9H | 0 | 0 | 0 | 0 | 0 | 0 | PF99 | PF98 |

| PF9n | Control of normal output/N-ch open-drain output (n = 8, 9) |
|---|---|
| 0 | Normal output |
| 1 | N-ch open-drain output |

**Caution When using P98 and P99 as N-ch open-drain-output alternate-function pins, set in the following sequence.**

**Be sure to set the port latch to 1 before setting the pin to N-ch open-drain output.**

**P9n bit = 1 → PFC9n bit = 0/1 → PF9n bit = 1 → PMC9n bit = 1**

### (5) Port 9 function control register (PFC9)

**Caution When port 9 is specified as an alternate function by the PMC9.PMC9n bit with the PFC9n bit maintaining the initial value (0), output becomes undefined. Therefore, to specify port 9 as alternate function 2, set the PFC9n bit to 1 first and then set the PMC9n bit to 1 (n = 0, 1, 6 to 9, 13 to 15).**

After reset: 0000H R/W Address: PFC9 FFFFF472H,
PFC9L FFFFF472H, PFC9H FFFFF473H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| PFC9 (PFC9H[Note]) | PFC915 | PFC914 | PFC913 | 0 | 0 | 0 | PFC99 | PFC98 |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| (PFC9L) | PFC97 | PFC96 | 0 | 0 | 0 | 0 | PFC91 | PFC90 |

| PFC915 | Specification of alternate-function pin of P915 pin |
|---|---|
| 1 | INTP6 input |

| PFC914 | Specification of alternate-function pin of P914 pin |
|---|---|
| 1 | INTP5 input |

| PFC913 | Specification of alternate-function pin of P913 pin |
|---|---|
| 1 | INTP4 input |

| PFC99 | Specification of alternate-function pin of P99 pin |
|---|---|
| 1 | $\overline{\text{SCK01}}$ I/O |

| PFC98 | Specification of alternate-function pin of P98 pin |
|---|---|
| 1 | SO01 output |

| PFC97 | Specification of alternate-function pin of P97 pin |
|---|---|
| 1 | SI01 input |

| PFC96 | Specification of alternate-function pin of P96 pin |
|---|---|
| 1 | TO51 output |

| PFC91 | Specification of alternate-function pin of P91 pin |
|---|---|
| 1 | RXD1 input |

| PFC90 | Specification of alternate-function pin of P90 pin |
|---|---|
| 1 | TXD1 output |

**Note** When reading from or writing to bits 8 to 15 of the PFC9 register in 8-bit or 1-bit units, specify these bits as bits 0 to 7 of the PFC9H register.

**Remark** The PFC9 register can be read or written in 16-bit units.
However, when the higher 8 bits and the lower 8 bits of the PFC9 register are used as the PFC9H register and as the PFC9L register, respectively, these registers can be read or written in 8-bit or 1-bit units.

### (6) Pull-up resistor option register 9 (PU9)

After reset: 0000H R/W Address: PU9 FFFFFC52H,
PU9L FFFFFC52H, PU9H FFFFFC53H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| PU9 (PU9H[Note]) | PU915 | PU914 | PU913 | 0 | 0 | 0 | PU99 | PU98 |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| (PU9L) | PU97 | PU96 | 0 | 0 | 0 | 0 | PU91 | PU90 |

| PU9n | Control of on-chip pull-up resistor connection (n = 0, 1, 6 to 9, 13 to 15) |
|---|---|
| 0 | Not connected |
| 1 | Connected |

**Note** When reading from or writing to bits 8 to 15 of the PU9 register in 8-bit or 1-bit units, specify these bits as bits 0 to 7 of the PU9H register.

**Remark** The PU9 register can be read or written in 16-bit units.
However, when the higher 8 bits and the lower 8 bits of the PU9 register are used as the PU9H register and as the PU9L register, respectively, these registers can be read or written in 8-bit or 1-bit units.

### 4.3.7 Port CM

Port CM is a 4-bit I/O port for which I/O settings can be controlled in 1-bit units.

Port CM includes the following alternate functions.

**Table 4-10. Alternate-Function Pins of Port CM**

| Pin No. | Pin Name | Alternate Function | I/O | PULL[Note] | Remark | Block Type |
|---|---|---|---|---|---|---|
| 49 | PCM0 | $\overline{\text{WAIT}}$ | Input | Yes | – | D1-UH |
| 50 | PCM1 | CLKOUT | Output | | | D0-U |
| 51 | PCM2 | $\overline{\text{HLDAK}}$ | Output | | | D0-U |
| 52 | PCM3 | $\overline{\text{HLDRQ}}$ | Input | | | D1-UH |

**Note** Software pull-up function

#### (1) Port CM register (PCM)

After reset: 00H (output latch) R/W Address: FFFFF00CH

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PCM | 0 | 0 | 0 | 0 | PCM3 | PCM2 | PCM1 | PCM0 |

| PCMn | Control of output data (in output mode) (n = 0 to 3) |
|---|---|
| 0 | 0 is output |
| 1 | 1 is output |

#### (2) Port CM mode register (PMCM)

After reset: FFH R/W Address: FFFFF02CH

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PMCM | 1 | 1 | 1 | 1 | PMCM3 | PMCM2 | PMCM1 | PMCM0 |

| PMCMn | Control of I/O mode (n = 0 to 3) |
|---|---|
| 0 | Output mode |
| 1 | Input mode |

### (3) Port CM mode control register (PMCCM)

After reset: 00H R/W Address: FFFFF04CH

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PMCCM | 0 | 0 | 0 | 0 | PMCCM3 | PMCCM2 | PMCCM1 | PMCCM0 |

| PMCCM3 | Specification of PCM3 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | $\overline{\text{HLDRQ}}$ input |

| PMCCM2 | Specification of PCM2 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | $\overline{\text{HLDAK}}$ output |

| PMCCM1 | Specification of PCM1 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | CLKOUT output |

| PMCCM0 | Specification of PCM0 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | $\overline{\text{WAIT}}$ input |

### (4) Pull-up resistor option register CM (PUCM)

After reset: 00H R/W Address: FFFFFF4CH

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PUCM | 0 | 0 | 0 | 0 | PUCM3 | PUCM2 | PUCM1 | PUCM0 |

| PUCMn | Control of on-chip pull-up resistor connection (n = 0 to 3) |
|---|---|
| 0 | Not connected |
| 1 | Connected |

### 4.3.8 Port CS

Port CS is a 2-bit I/O port for which I/O settings can be controlled in 1-bit units.

Port CS includes the following alternate functions.

**Table 4-11. Alternate-Function Pins of Port CS**

| Pin No. | Pin Name | Alternate Function | I/O | PULL[Note] | Remark | Block Type |
|---|---|---|---|---|---|---|
| 47 | PCS0 | $\overline{CS0}$ | Output | Yes | – | D0-UZ |
| 48 | PCS1 | $\overline{CS1}$ | Output | | | D0-UZ |

**Note** Software pull-up function

#### (1) Port CS register (PCS)

After reset: 00H (output latch) R/W Address: FFFFF008H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PCS | 0 | 0 | 0 | 0 | 0 | 0 | PCS1 | PCS0 |

| PCSn | Control of output data (in output mode) (n = 0, 1) |
|---|---|
| 0 | 0 is output |
| 1 | 1 is output |

#### (2) Port CS mode register (PMCS)

After reset: FFH R/W Address: FFFFF028H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PMCS | 0 | 0 | 0 | 0 | 0 | 0 | PMCS1 | PMCS0 |

| PMCSn | Control of I/O mode (n = 0, 1) |
|---|---|
| 0 | Output mode |
| 1 | Input mode |

### (3) Port CS mode control register (PMCCS)

After reset: 00H R/W Address: FFFFF048H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PMCCS | 0 | 0 | 0 | 0 | 0 | 0 | PMCCS1 | PMCCS0 |

| PMCCSn | Specification of PCSn pin operation mode (n = 0, 1) |
|---|---|
| 0 | I/O port |
| 1 | $\overline{\text{CSn}}$ output |

### (4) Pull-up resistor option register CS (PUCS)

After reset: 00H R/W Address: FFFFFF48H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PUCS | 0 | 0 | 0 | 0 | 0 | 0 | PUCS1 | PUCS0 |

| PUCSn | Control of on-chip pull-up resistor connection (n = 0, 1) |
|---|---|
| 0 | Not connected |
| 1 | Connected |

### 4.3.9 Port CT

Port CT is a 4-bit I/O port for which I/O settings can be controlled in 1-bit units.

Port CT includes the following alternate functions.

**Table 4-12. Alternate-Function Pins of Port CT**

| Pin No. | Pin Name | Alternate Function | I/O | PULL[Note] | Remark | Block Type |
|---|---|---|---|---|---|---|
| 53 | PCT0 | $\overline{WR0}$ | Output | Yes | – | D0-UZ |
| 54 | PCT1 | $\overline{WR1}$ | Output | | | D0-UZ |
| 55 | PCT4 | $\overline{RD}$ | Output | | | D0-UZ |
| 56 | PCT6 | ASTB | Output | | | D0-UZ |

**Note** Software pull-up function

#### (1) Port CT register (PCT)

After reset: 00H (output latch) R/W Address: FFFFF00AH

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PCT | 0 | PCT6 | 0 | PCT4 | 0 | 0 | PCT1 | PCT0 |

| PCTn | Control of output data (in output mode) (n = 0, 1, 4, 6) |
|---|---|
| 0 | 0 is output |
| 1 | 1 is output |

#### (2) Port CT mode register (PMCT)

After reset: FFH R/W Address: FFFFF02AH

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PMCT | 0 | PMCT6 | 0 | PMCT4 | 0 | 0 | PMCT1 | PMCT0 |

| PMCTn | Control of I/O mode (n = 0, 1, 4, 6) |
|---|---|
| 0 | Output mode |
| 1 | Input mode |

### (3) Port CT mode control register (PMCCT)

After reset: 00H R/W Address: FFFFF04AH

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PMCCT | 0 | PMCCT6 | 0 | PMCCT4 | 0 | 0 | PMCCT1 | PMCCT0 |

| PMCCT6 | Specification of PCT6 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | ASTB output |

| PMCCT4 | Specification of PCT4 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | $\overline{RD}$ output |

| PMCCT1 | Specification of PCT1 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | $\overline{WR1}$ output |

| PMCCT0 | Specification of PCT0 pin operation mode |
|---|---|
| 0 | I/O port |
| 1 | $\overline{WR0}$ output |

### (4) Pull-up resistor option register CT (PUCT)

After reset: 00H R/W Address: FFFFFF4AH

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PUCT | 0 | PUCT6 | 0 | PUCT4 | 0 | 0 | PUCT1 | PUCT0 |

| PUCTn | Control of on-chip pull-up resistor connection (n = 0, 1, 4, 6) |
|---|---|
| 0 | Not connected |
| 1 | Connected |

### 4.3.10 Port DL

Port DL is a 16-bit I/O port for which I/O settings can be controlled in 1-bit units.

Port DL includes the following alternate functions.

**Table 4-13. Alternate-Function Pins of Port DL**

| Pin No. | Pin Name | Alternate Function | I/O | PULL[Note] | Remark | Block Type |
|---|---|---|---|---|---|---|
| 57 | PDL0 | AD0 | I/O | Yes | – | D2-ULZ |
| 58 | PDL1 | AD1 | I/O | | | D2-ULZ |
| 59 | PDL2 | AD2 | I/O | | | D2-ULZ |
| 60 | PDL3 | AD3 | I/O | | | D2-ULZ |
| 61 | PDL4 | AD4 | I/O | | | D2-ULZ |
| 62 | PDL5 | AD5 | I/O | | | D2-ULZ |
| 63 | PDL6 | AD6 | I/O | | | D2-ULZ |
| 64 | PDL7 | AD7 | I/O | | | D2-ULZ |
| 65 | PDL8 | AD8 | I/O | | | D2-ULZ |
| 66 | PDL9 | AD9 | I/O | | | D2-ULZ |
| 67 | PDL10 | AD10 | I/O | | | D2-ULZ |
| 68 | PDL11 | AD11 | I/O | | | D2-ULZ |
| 69 | PDL12 | AD12 | I/O | | | D2-ULZ |
| 70 | PDL13 | AD13 | I/O | | | D2-ULZ |
| 71 | PDL14 | AD14 | I/O | | | D2-ULZ |
| 72 | PDL15 | AD15 | I/O | | | D2-ULZ |

**Note** Software pull-up function

### (1) Port DL register (PDL)

After reset: 00H (output latch) R/W Address: PDL FFFFF004H,
PDLL FFFFF004H, PDLH FFFFF005H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| PDL (PDLH[Note]) | PDL15 | PDL14 | PDL13 | PDL12 | PDL11 | PDL10 | PDL9 | PDL8 |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| (PDLL) | PDL7 | PDL6 | PDL5 | PDL4 | PDL3 | PDL2 | PDL1 | PDL0 |

| PDLn | Control of output data (in output mode) (n = 0 to 15) |
|---|---|
| 0 | 0 is output |
| 1 | 1 is output |

**Note** When reading from or writing to bits 8 to 15 of the PDL register in 8-bit or 1-bit units, specify these bits as bits 0 to 7 of the PDLH register.

**Remark** The PDL register can be read or written in 16-bit units.
However, when the higher 8 bits and the lower 8 bits of the PDL register are used as the PDLH register and as the PDLL register, respectively, these registers can be read or written in 8-bit or 1-bit units.

### (2) Port DL mode register (PMDL)

After reset: FFFFH R/W Address: PMDL FFFFF024H,
PMDLL FFFFF024H, PMDLH FFFFF025H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| PMDL (PMDLH[Note]) | PMDL15 | PMDL14 | PMDL13 | PMDL12 | PMDL11 | PMDL10 | PMDL9 | PMDL8 |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| (PMDLL) | PMDL7 | PMDL6 | PMDL5 | PMDL4 | PMDL3 | PMDL2 | PMDL1 | PMDL0 |

| PMDLn | Control of I/O mode (n = 0 to 15) |
|---|---|
| 0 | Output mode |
| 1 | Input mode |

**Note** When reading from or writing to bits 8 to 15 of the PMDL register in 8-bit or 1-bit units, specify these bits as bits 0 to 7 of the PMDLH register.

**Remark** The PMDL register can be read or written in 16-bit units.
However, when the higher 8 bits and the lower 8 bits of the PMDL register are used as the PMDLH register and as the PMDLL register, respectively, these registers can be read or written in 8-bit or 1-bit units.

### (3) Port DL mode control register (PMCDL)

After reset: 0000H R/W Address: PMCDL FFFFF044H,
PMCDLL FFFFF044H, PMCDLH FFFFF045H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| PMCDL (PMCDLH[Note]) | PMCDL15 | PMCDL14 | PMCDL13 | PMCDL12 | PMCDL11 | PMCDL10 | PMCDL9 | PMCDL8 |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| (PMCDLL) | PMCDL7 | PMCDL6 | PMCDL5 | PMCDL4 | PMCDL3 | PMCDL2 | PMCDL1 | PMCDL0 |

| PMCDLn | Specification of PDLn pin operation mode (n = 0 to 15) |
|---|---|
| 0 | I/O port |
| 1 | ADn I/O (address/data bus I/O) |

**Note** When reading from or writing to bits 8 to 15 of the PMCDL register in 8-bit or 1-bit units, specify these bits as bits 0 to 7 of the PMCDLH register.

**Caution When specifying the port/alternate function for each bit, pay careful attention to the operation of the alternate functions.**

**Remark** The PMCDL register can be read or written in 16-bit units.
However, when the higher 8 bits and the lower 8 bits of the PMCDL register are used as the PMCDLH register and as the PMCDLL register, respectively, these registers can be read or written in 8-bit or 1-bit units.

### (4) Pull-up resistor option register DL (PUDL)

After reset: 0000H R/W Address: PUDL FFFFFF44H,
PUDLL FFFFFF44H, PUDLH FFFFFF45H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| PUDL (PUDLH[Note]) | PUDL15 | PUDL14 | PUDL13 | PUDL12 | PUDL11 | PUDL10 | PUDL9 | PUDL8 |

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| (PUDLL) | PUDL7 | PUDL6 | PUDL5 | PUDL4 | PUDL3 | PUDL2 | PUDL1 | PUDL0 |

| PUDLn | Control of on-chip pull-up resistor connection (n = 0 to 15) |
|---|---|
| 0 | Not connected |
| 1 | Connected |

**Note** When reading from or writing to bits 8 to 15 of the PUDL register in 8-bit or 1-bit units, specify these bits as bits 0 to 7 of the PUDLH register.

**Remark** The PUDL register can be read or written in 16-bit units.
However, when the higher 8 bits and the lower 8 bits of the PUDL register are used as the PUDLH register and as the PUDLL register, respectively, these registers can be read or written in 8-bit or 1-bit units.

## 4.4 Block Diagrams

**Figure 4-2. Block Diagram of Type A-A**

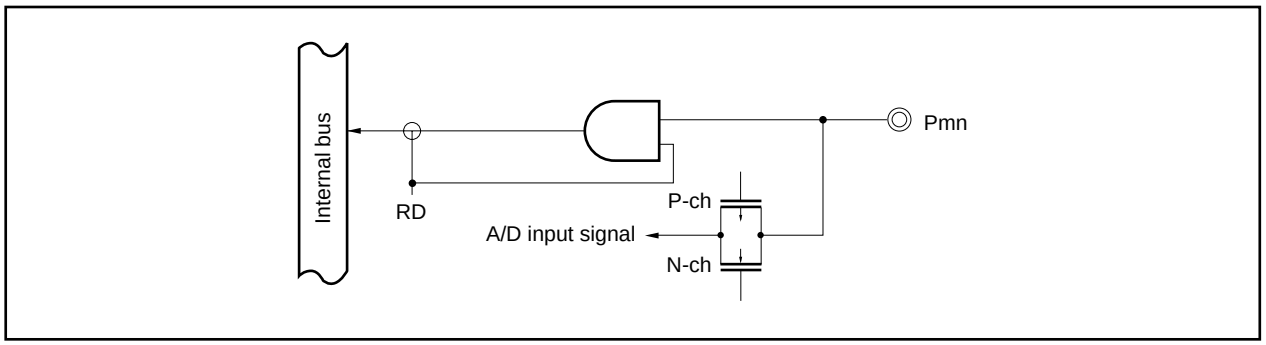


**Figure 4-3. Block Diagram of Type D0-U**

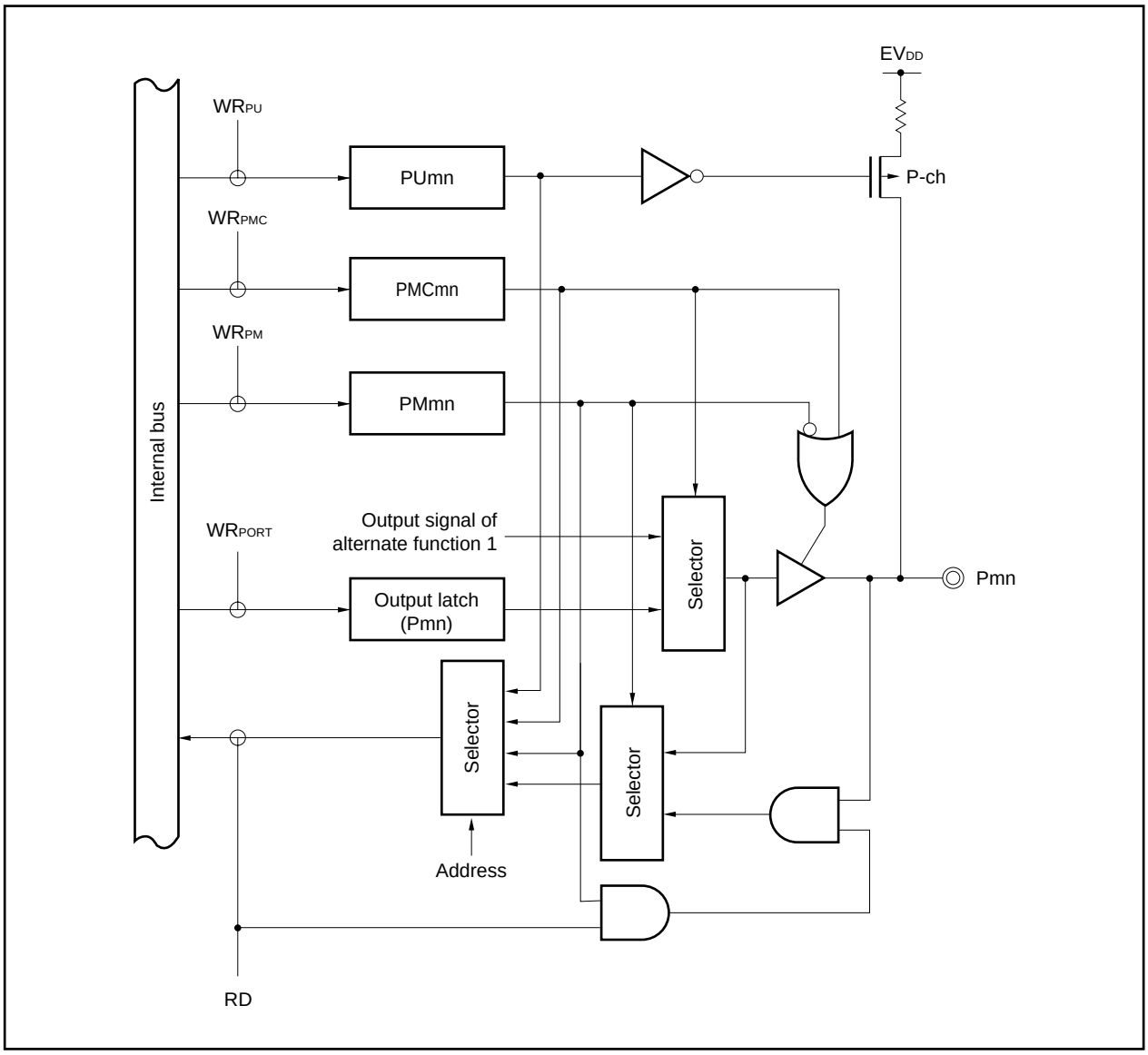


**Figure 4-4. Block Diagram of Type D0-UF**

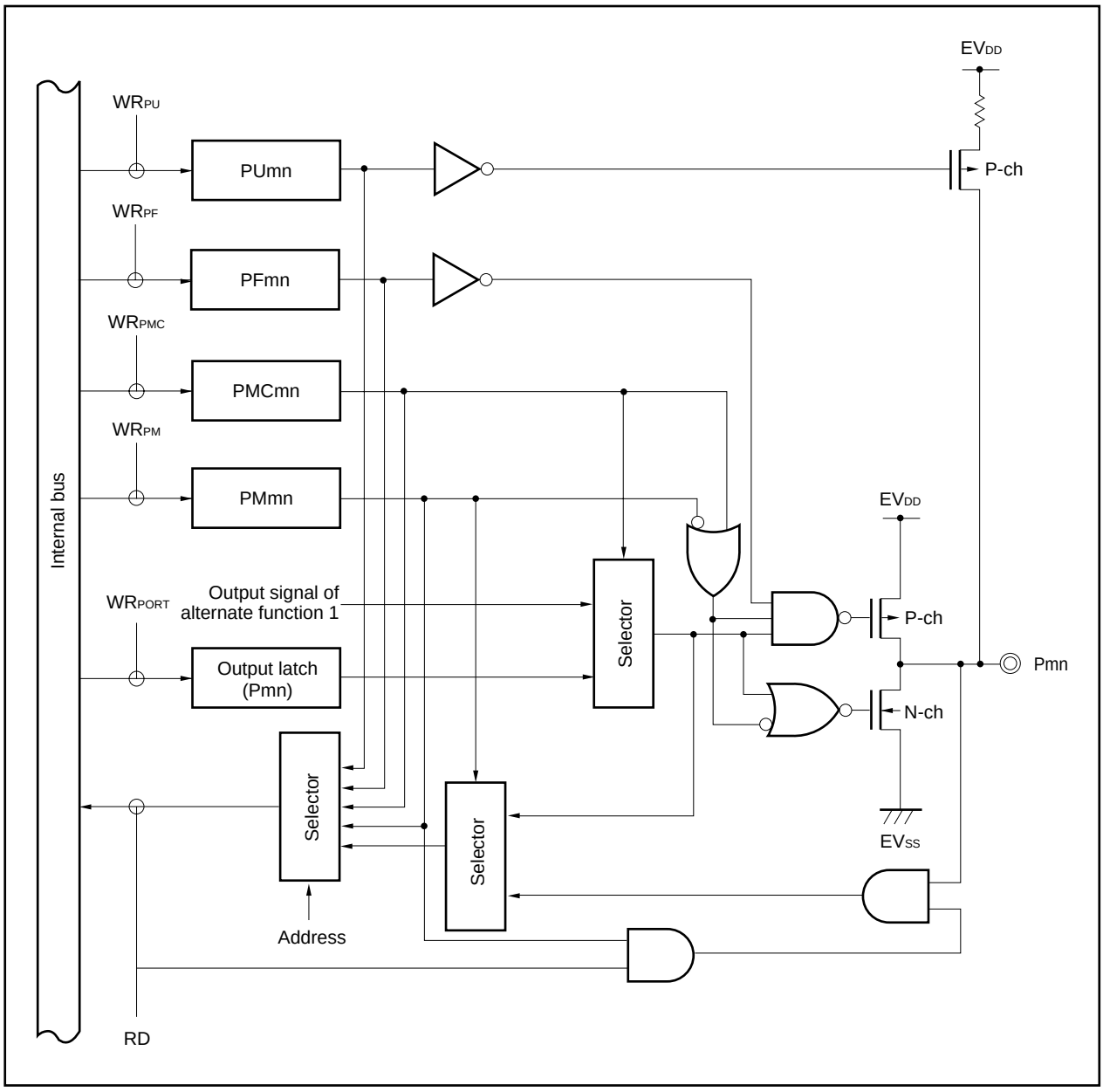
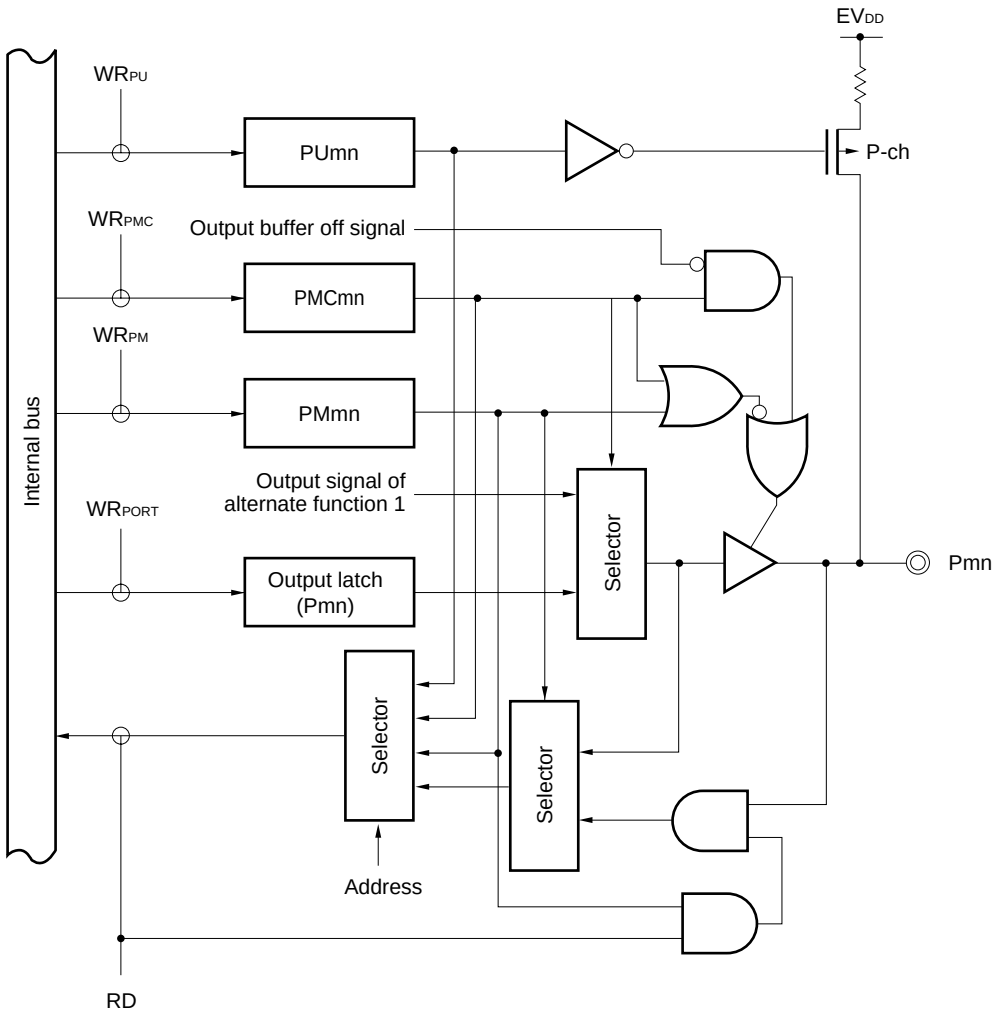
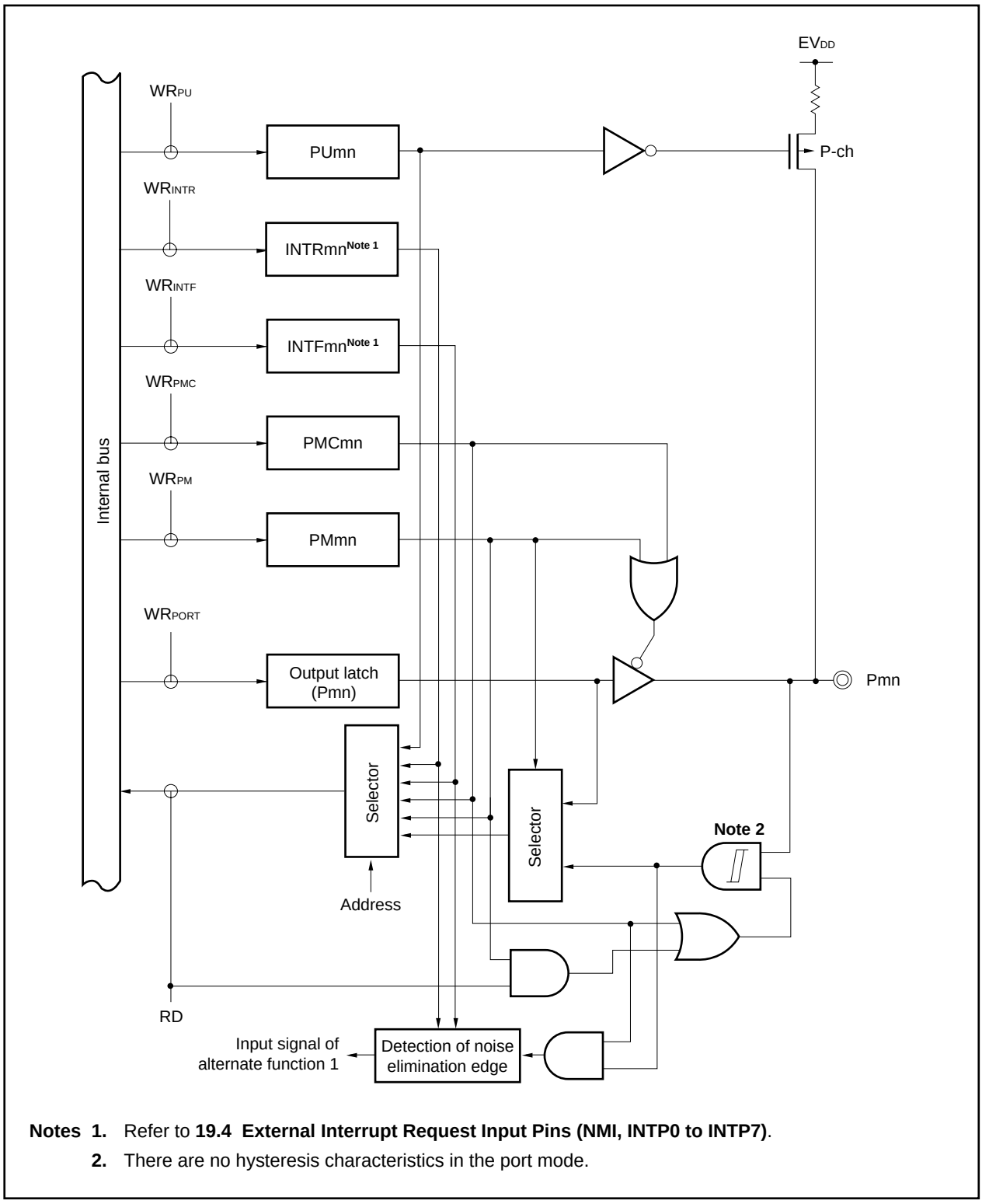


**Figure 4-5. Block Diagram of Type D0-UZ**




**Remark** Output buffer off signal: Signal that is asserted in the IDLE or STOP mode, or when the bus is held.

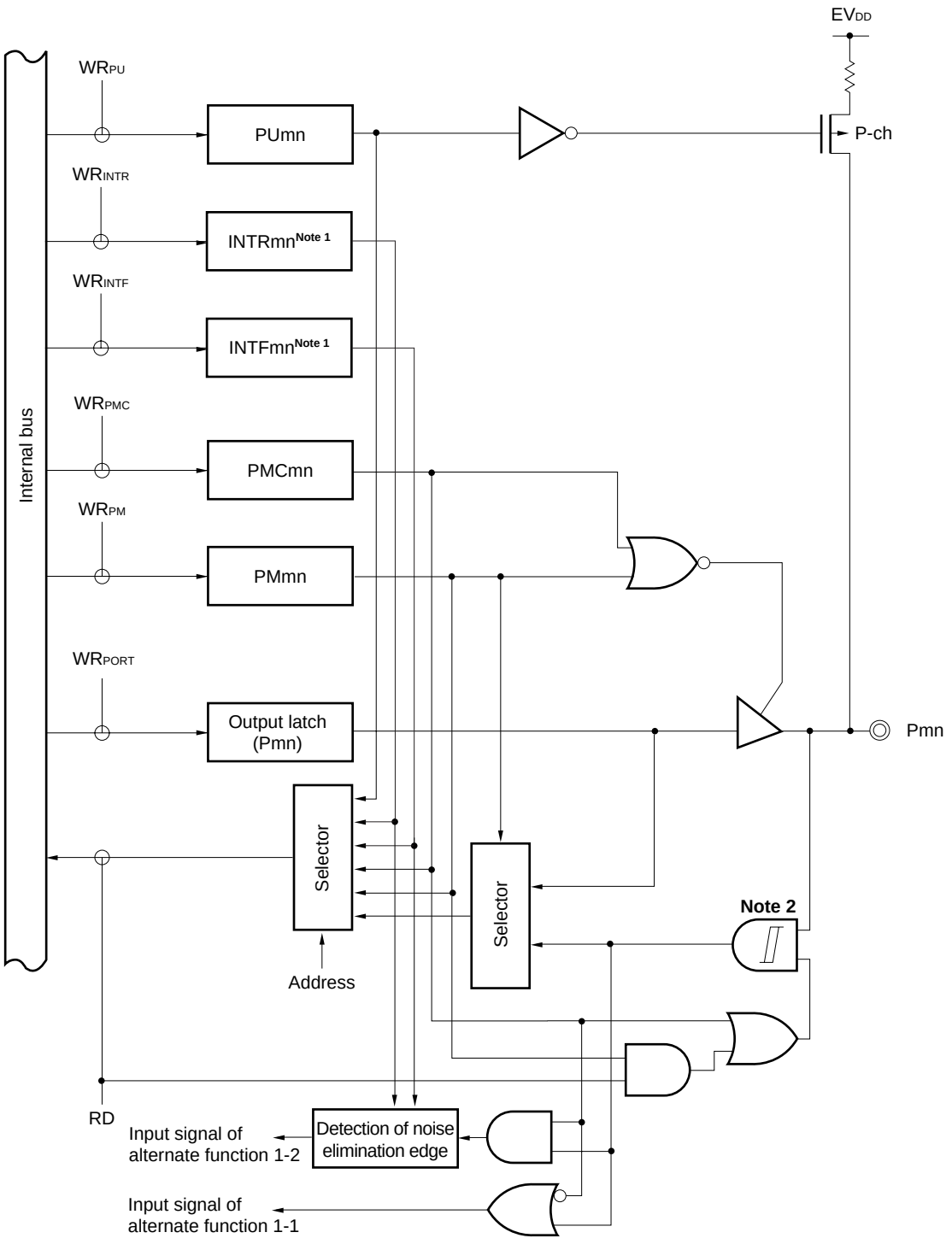
**Figure 4-6. Block Diagram of Type D1-SUIL**




**Notes 1.** Refer to **19.4 External Interrupt Request Input Pins (NMI, INTP0 to INTP7)**.

**2.** There are no hysteresis characteristics in the port mode.

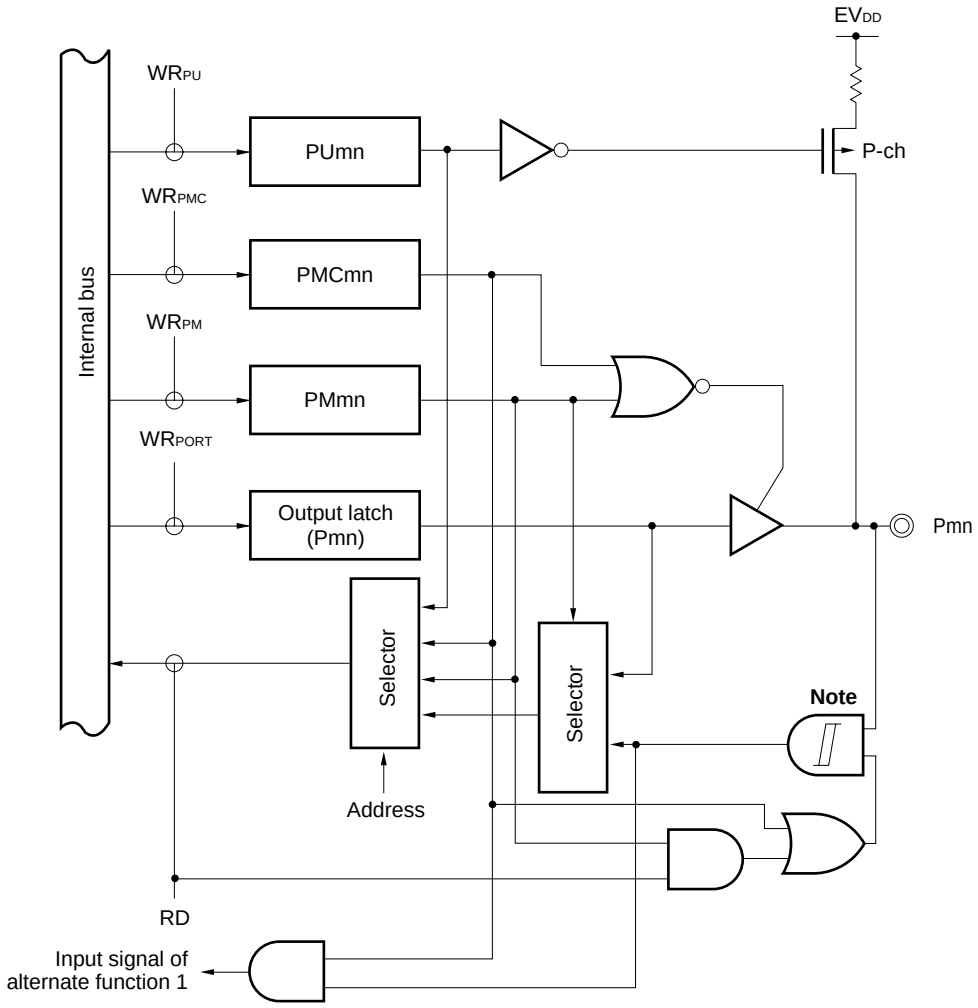
**Figure 4-7. Block Diagram of Type D1-SUIHL**




**Notes 1.** Refer to **19.4 External Interrupt Request Input Pins (NMI, INTP0 to INTP7)**.

**2.** There are no hysteresis characteristics in the port mode.

**Figure 4-8. Block Diagram of Type D1-SUL**




**Note** There are no hysteresis characteristics in the port mode.

**Figure 4-9. Block Diagram of Type D1-UH**

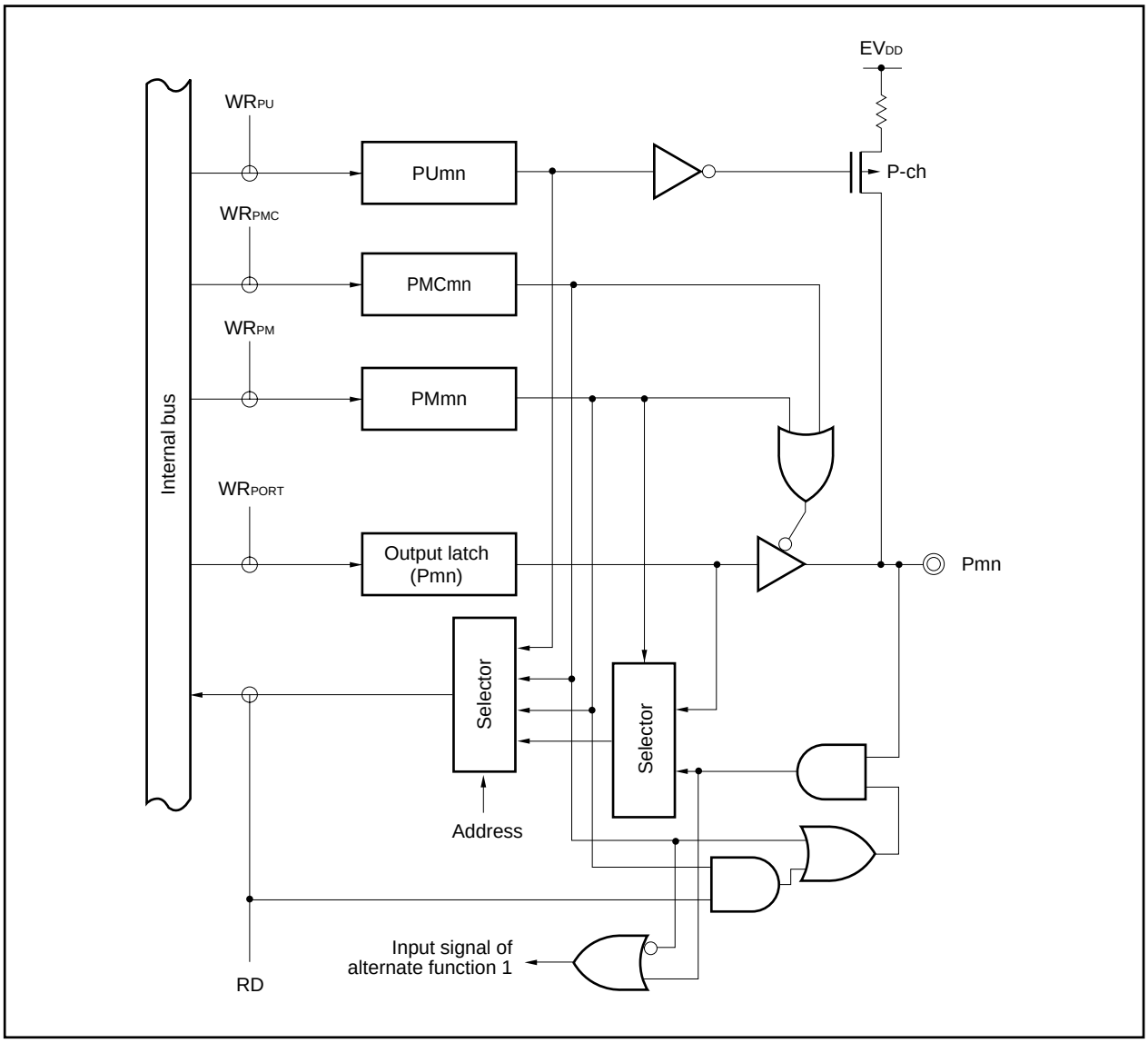
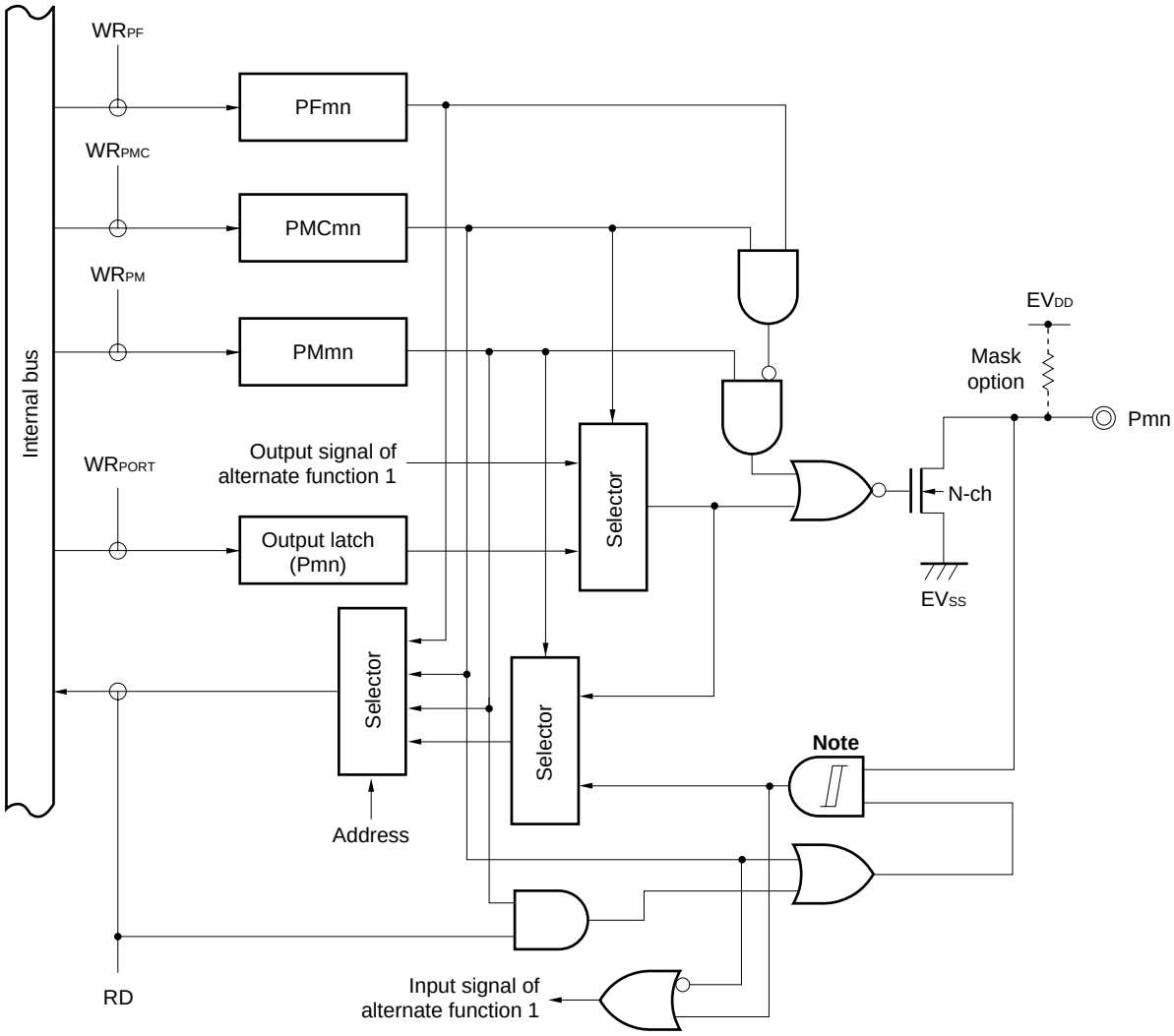
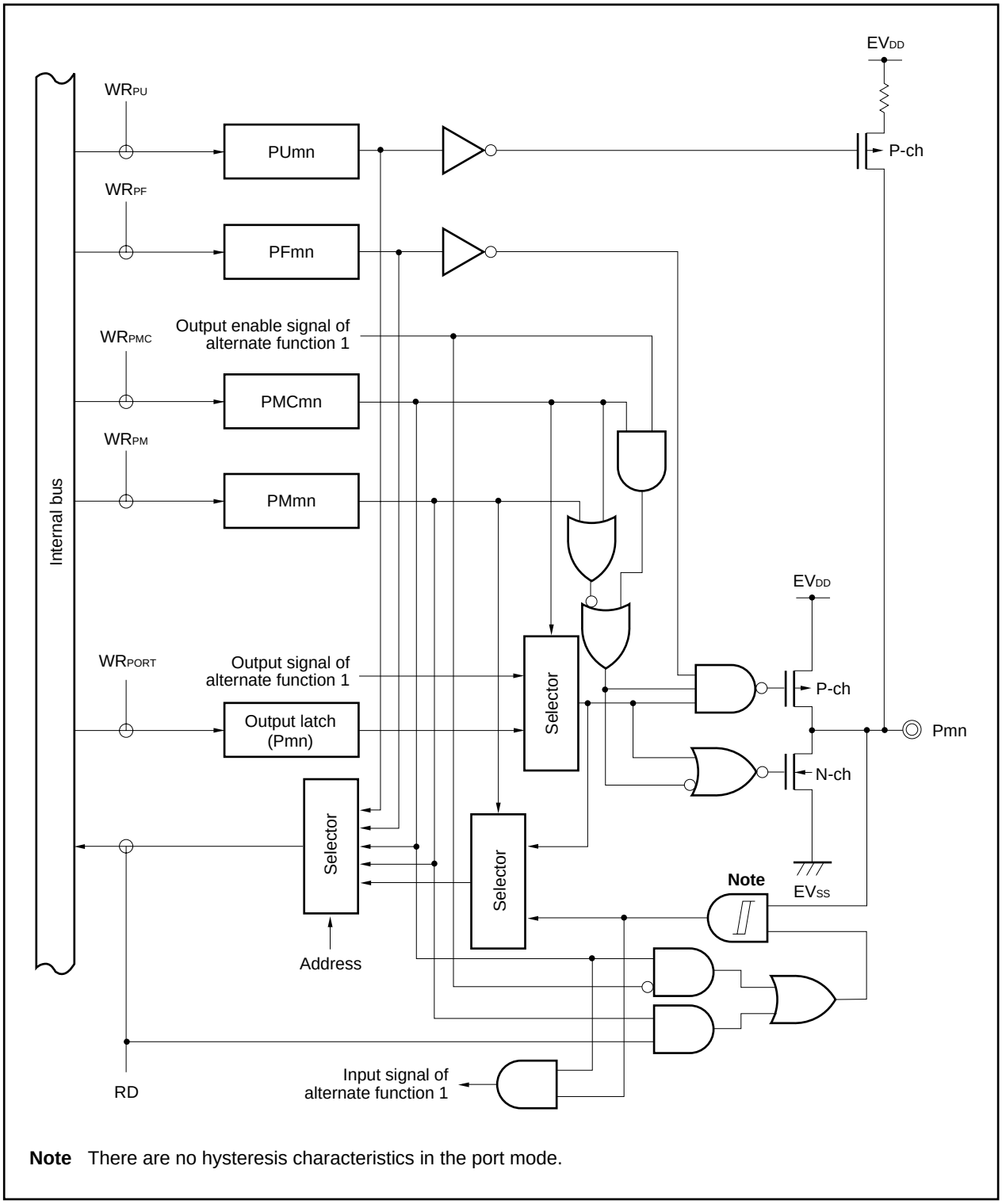


**Figure 4-10. Block Diagram of Type D2-SNMUFH**



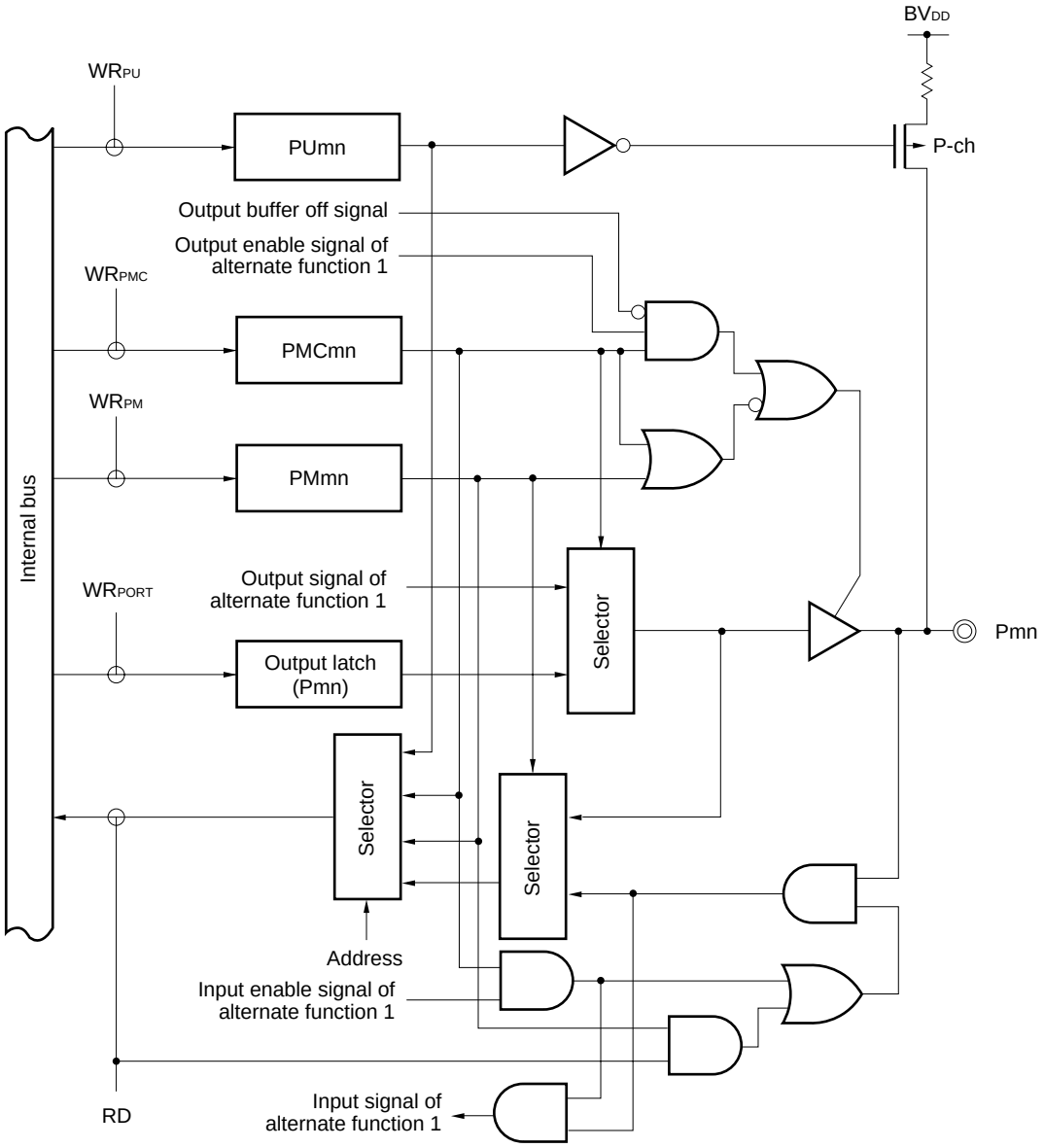

**Note** There are no hysteresis characteristics in the port mode.

**Figure 4-11. Block Diagram of Type D2-SUFL**




**Note** There are no hysteresis characteristics in the port mode.

**Figure 4-12. Block Diagram of Type D2-ULZ**




**Remark** Output buffer off signal: Signal that is asserted in the IDLE or STOP mode, or when the bus is held.

**Figure 4-13. Block Diagram of Type E00-SUFT**

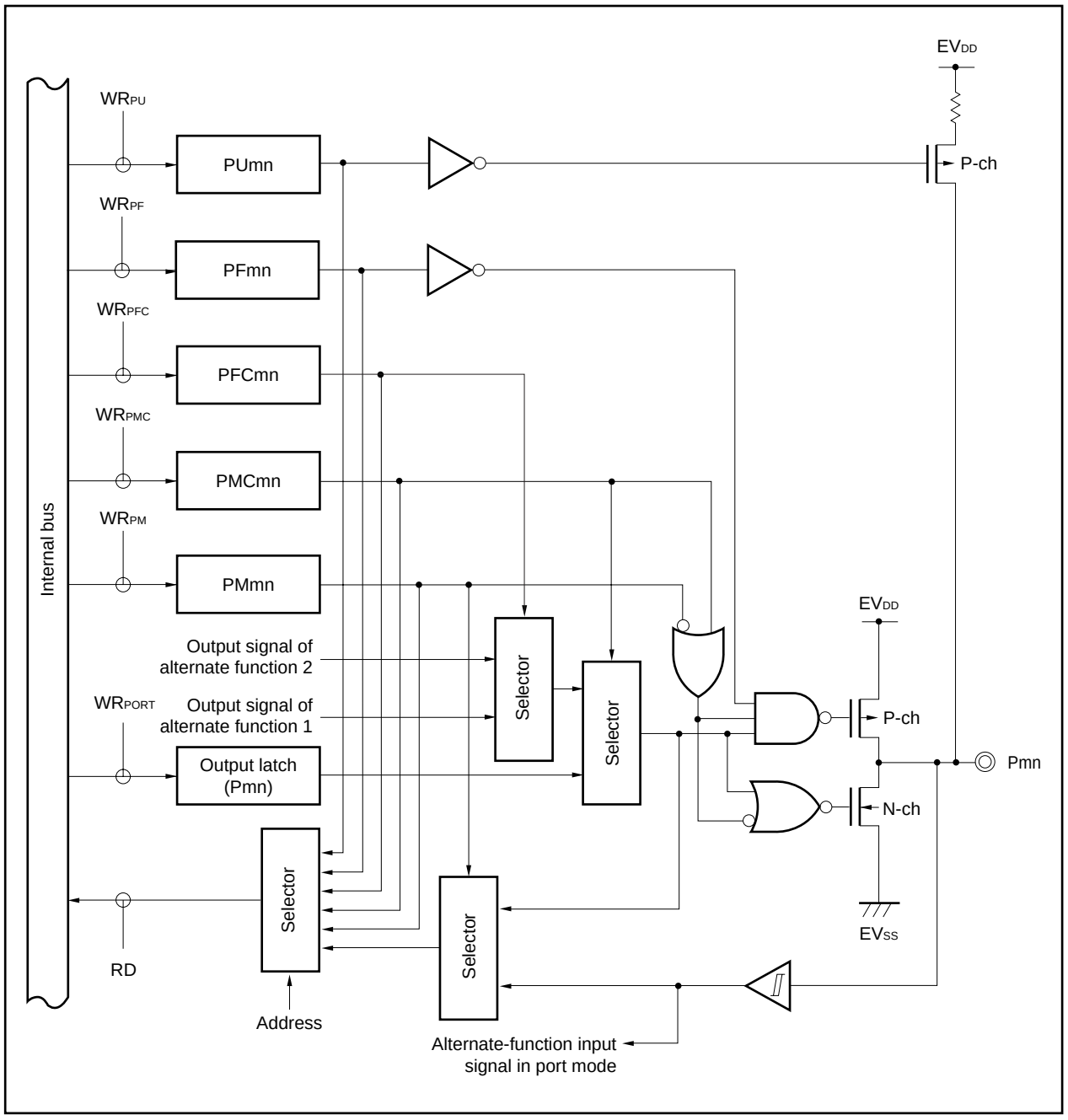


**Figure 4-14. Block Diagram of Type E00-SUT**

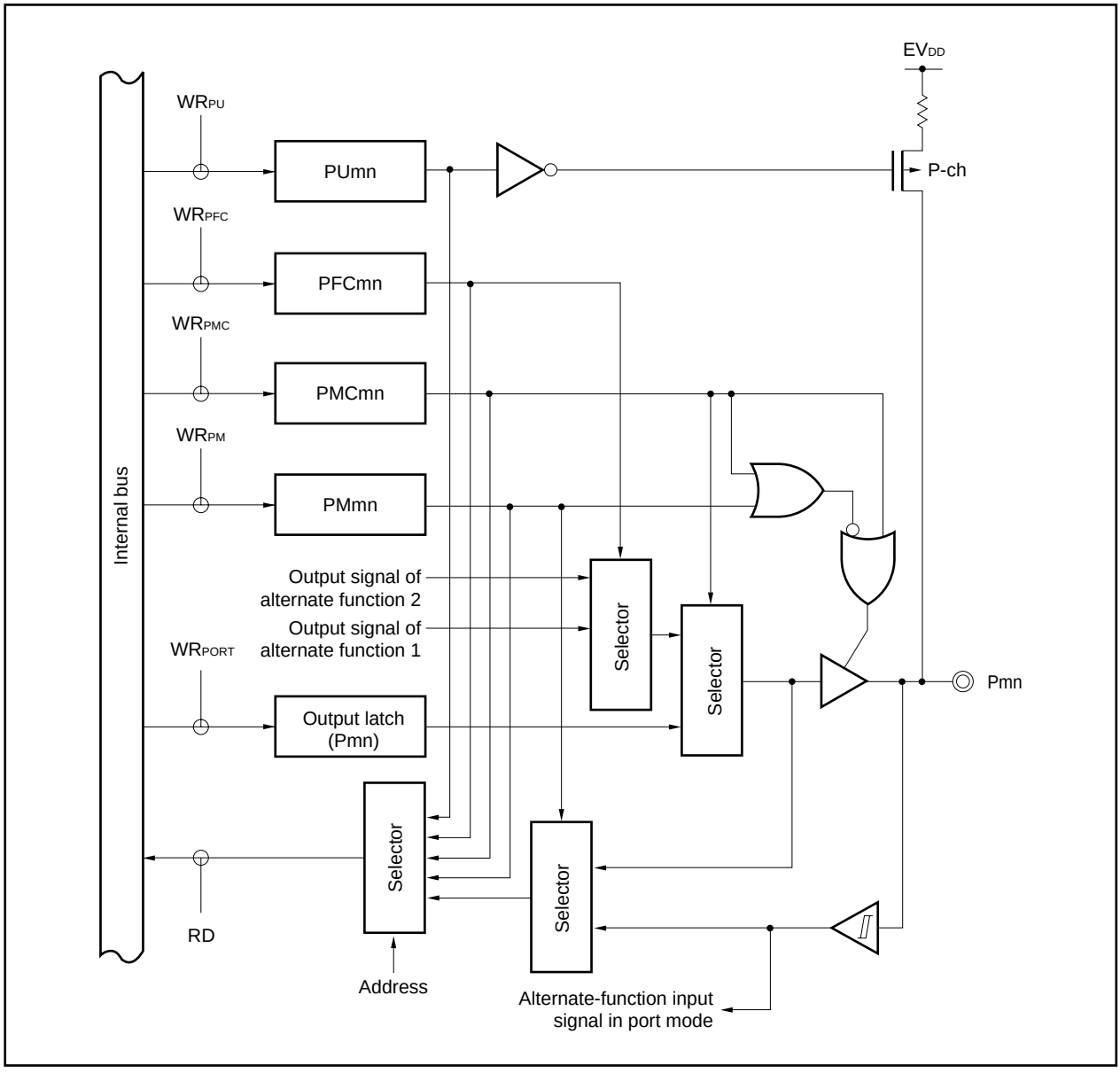
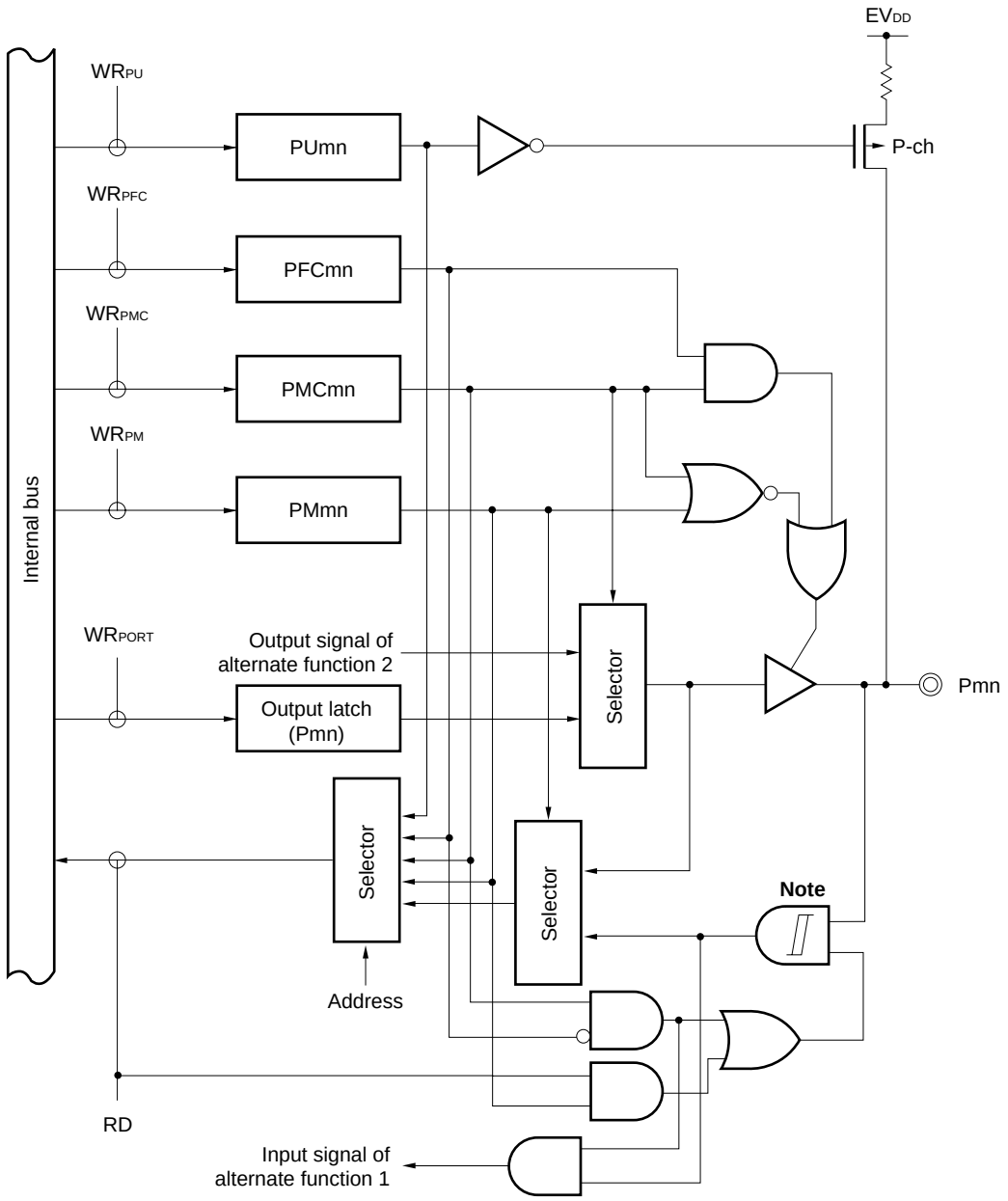


**Figure 4-15. Block Diagram of Type E10-SUL**




**Note** There are no hysteresis characteristics in the port mode.

**Figure 4-16. Block Diagram of Type E10-SULT**

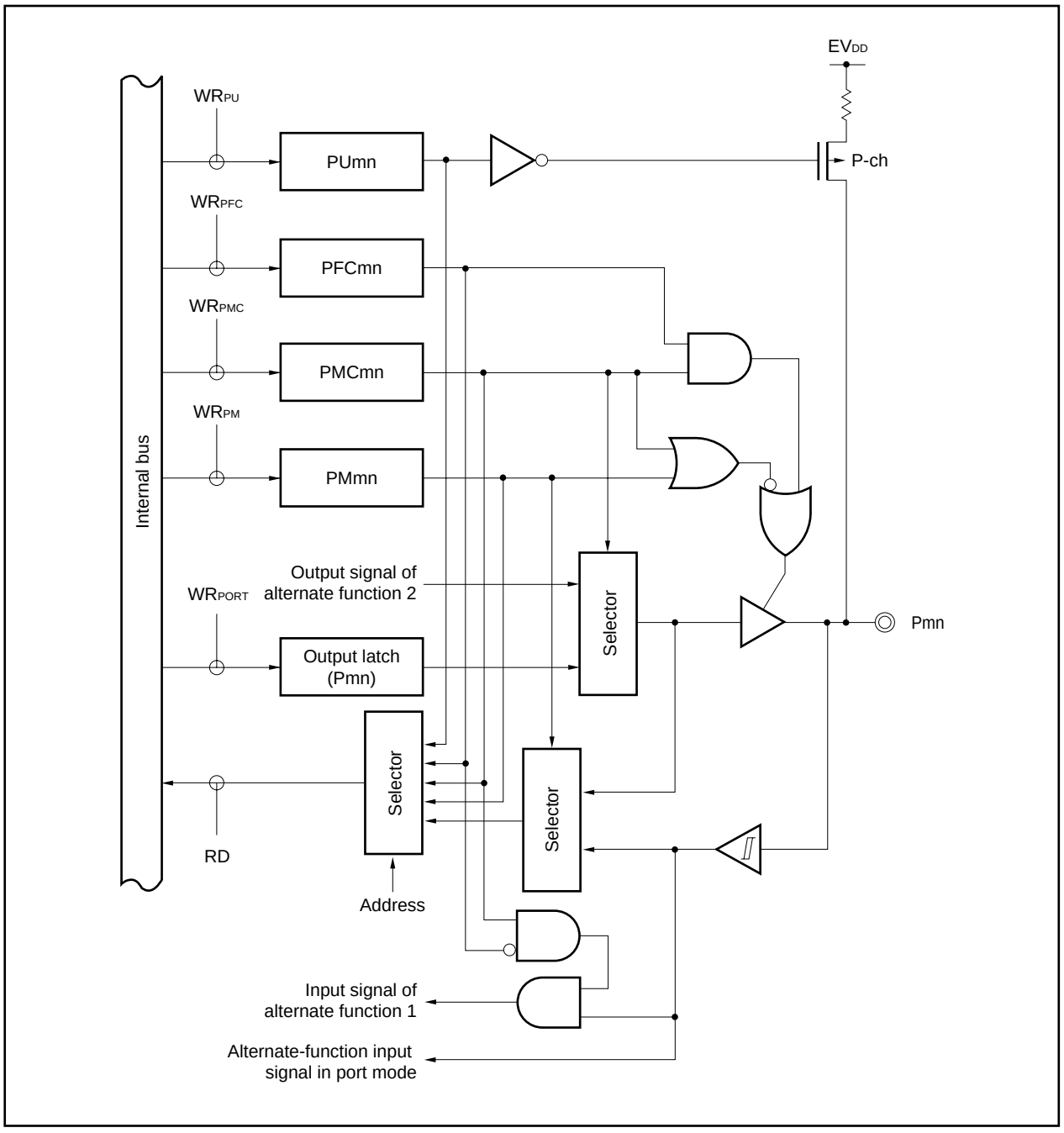


**Figure 4-17. Block Diagram of Type E20-SUFLT**

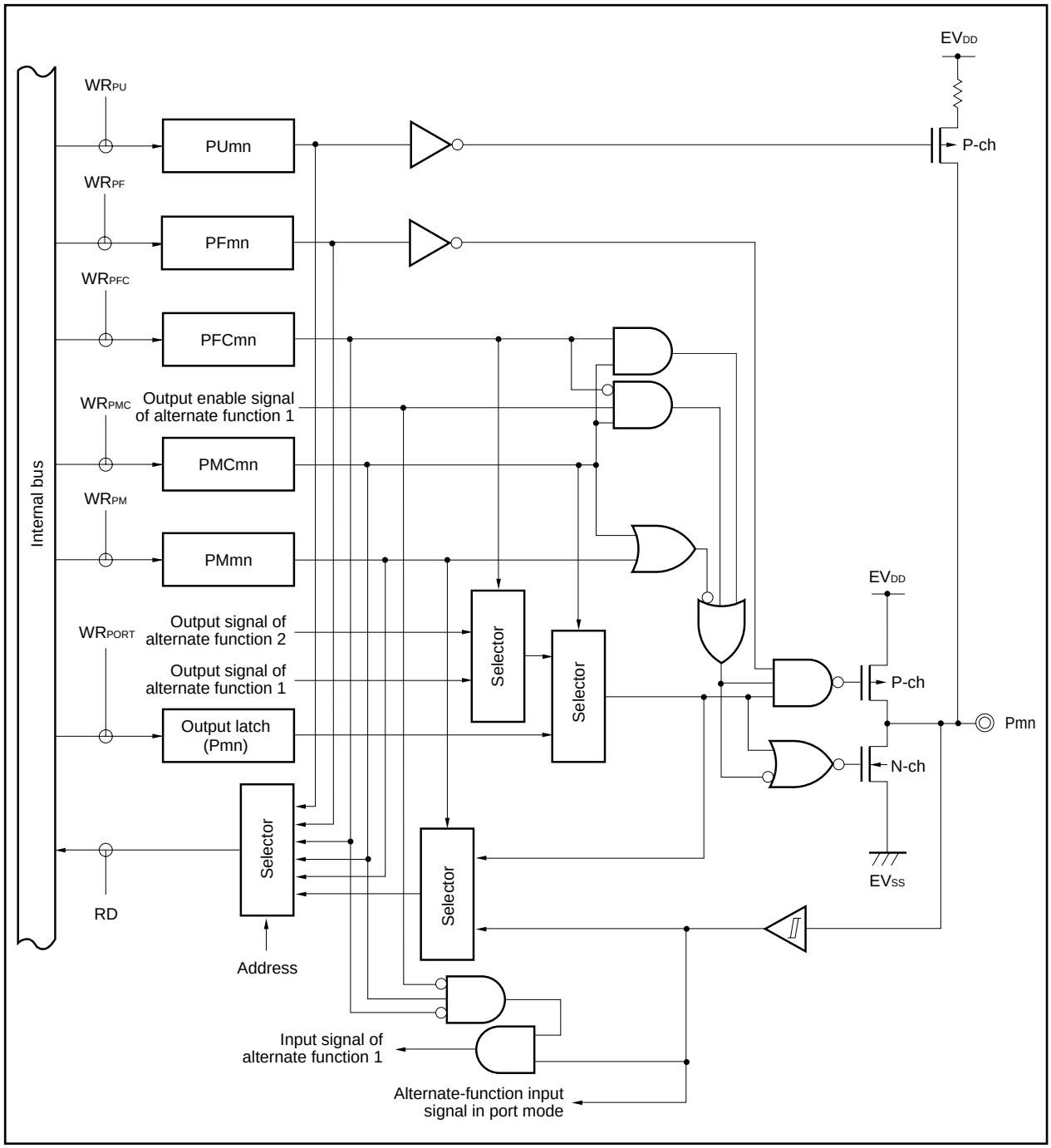


**Figure 4-18. Block Diagram of Type Ex0-SUT**

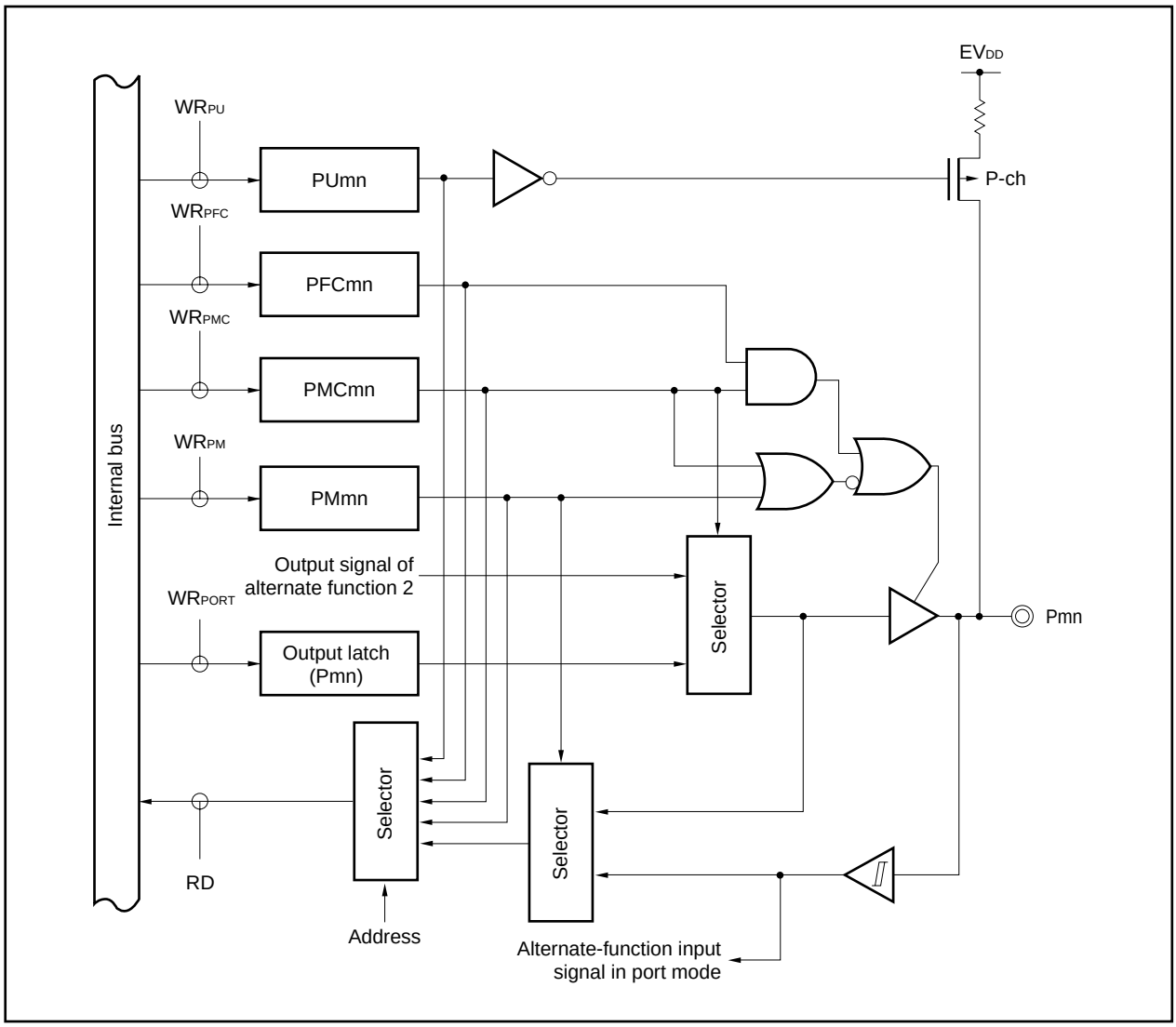


**Figure 4-19. Block Diagram of Type Ex0-UF**

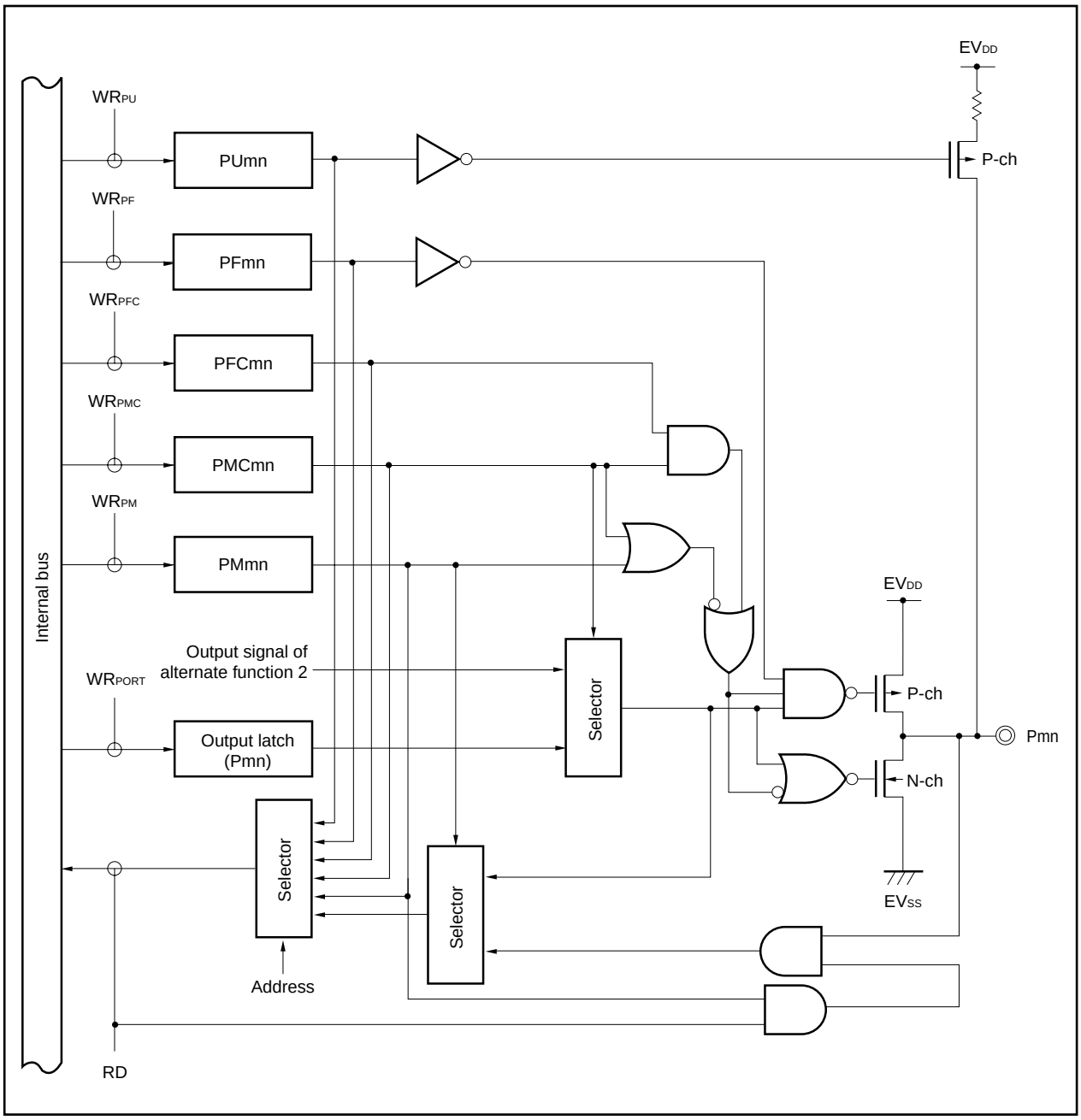


**Figure 4-20. Block Diagram of Type Ex1-SUHT**

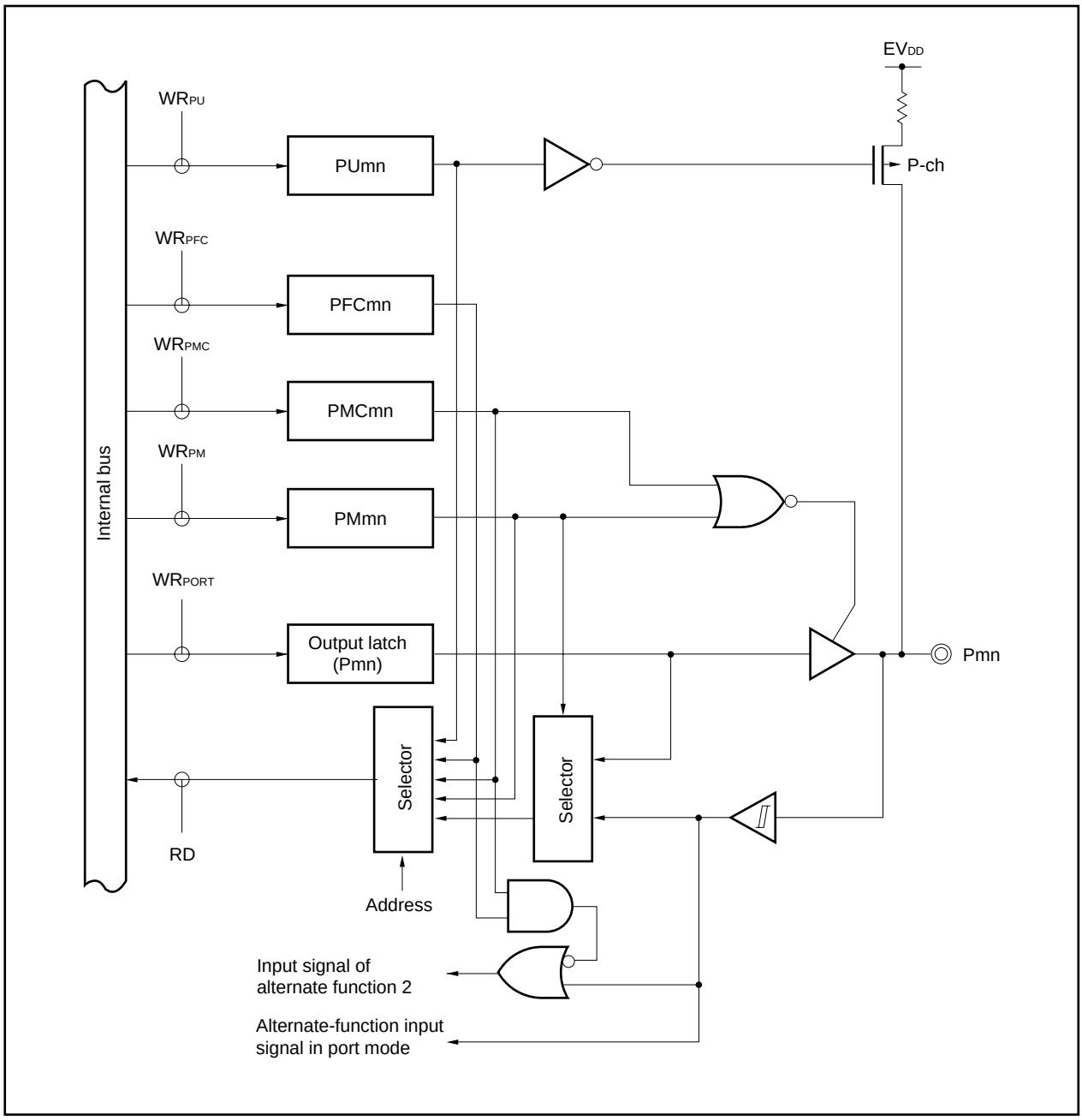
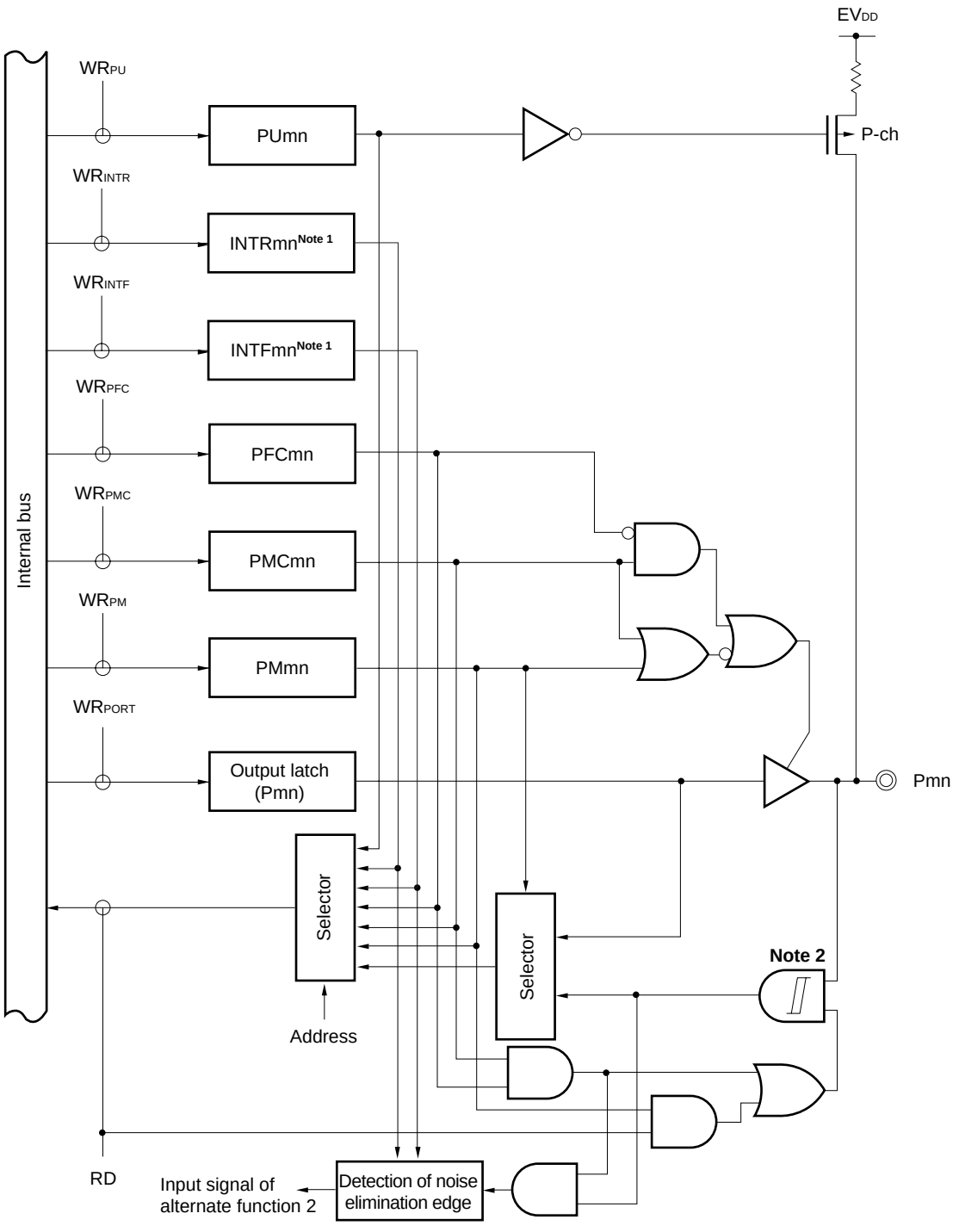


**Figure 4-21. Block Diagram of Type Ex1-SUIL**




**Notes 1.** Refer to **19.4 External Interrupt Request Input Pins (NMI, INTP0 to INTP7)**.

**2.** There are no hysteresis characteristics in the port mode.

**Figure 4-22. Block Diagram of Type Ex1-SUL**

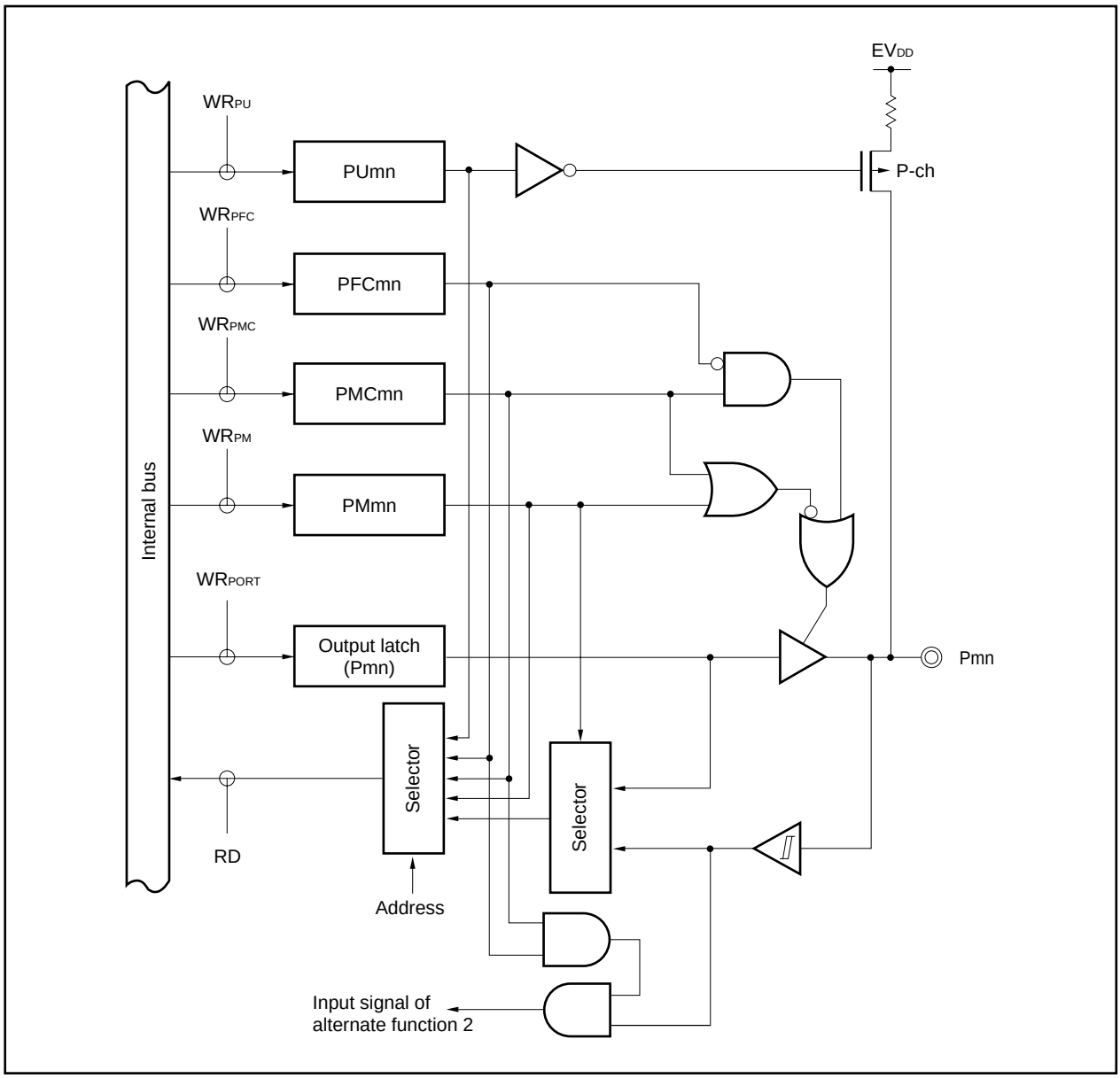
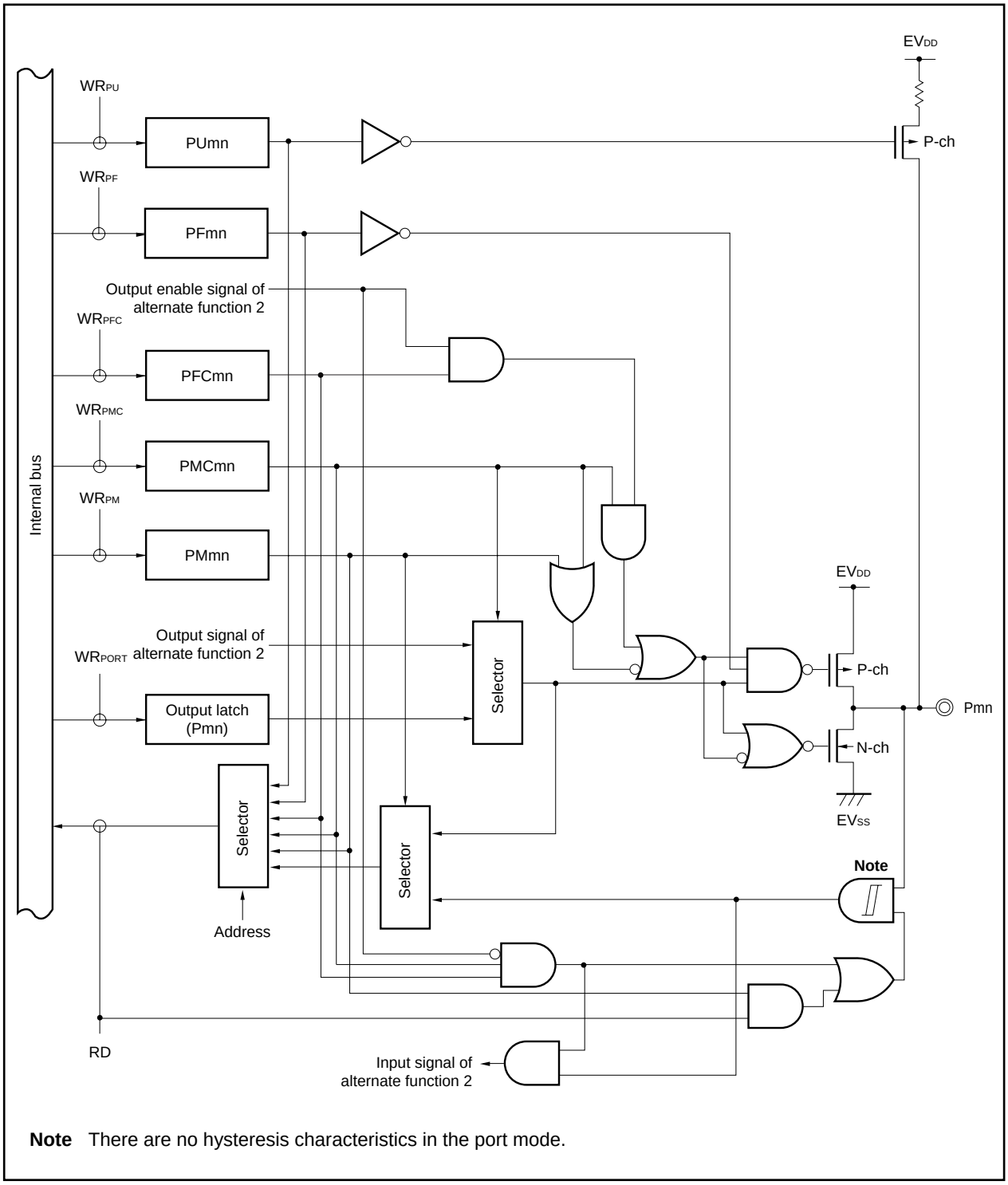
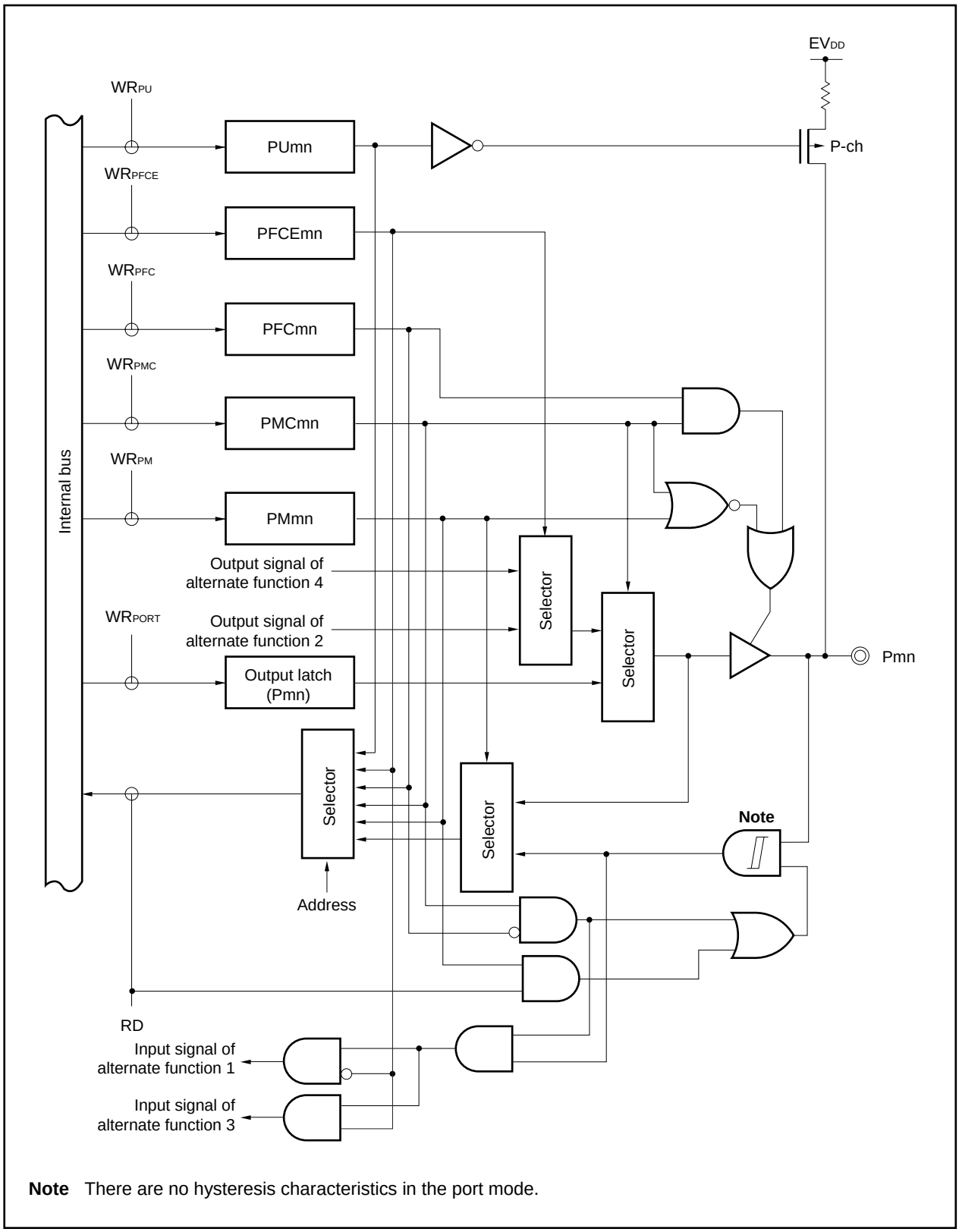


**Figure 4-23. Block Diagram of Type Ex2-SUFL**




**Note** There are no hysteresis characteristics in the port mode.

**Figure 4-24. Block Diagram of Type G1010-SUL**




**Note** There are no hysteresis characteristics in the port mode.

## 4.5 Port Register Setting When Alternate Function Is Used

Table 4-14 shows the port register settings when each port is used for an alternate function.
When using a port pin as an alternate-function pin, refer to description of each pin.

**Table 4-14. Settings When Port Pins Are Used for Alternate Functions (1/5)**

| Pin Name | Alternate Function | | Pnx Bit of Pn Register | PMnx Bit of PMn Register | PMCnx Bit of PMCn Register | PFCEnx Bit of PFCEn Register | PFCnx Re |
|---|---|---|---|---|---|---|---|
| | Function Name | I/O | | | | | |
| P00 | TOH0 | Output | P00 = Setting not required | PM00 = Setting not required | PMC00 = 1 | – | |
| P01 | TOH1 | Output | P01 = Setting not required | PM01 = Setting not required | PMC01 = 1 | – | |
| P02 | NMI | Input | P02 = Setting not required | PM02 = Setting not required | PMC02 = 1 | – | |
| P03 | INTP0 | Input | P03 = Setting not required | PM03 = Setting not required | PMC03 = 1 | – | PFC03 |
| P04 | INTP1 | Input | P04 = Setting not required | PM04 = Setting not required | PMC04 = 1 | – | |
| P05 | INTP2 | Input | P05 = Setting not required | PM05 = Setting not required | PMC05 = 1 | – | |
| P06 | INTP3 | Input | P06 = Setting not required | PM06 = Setting not required | PMC06 = 1 | – | |
| P30 | TXD0 | Output | P30 = Setting not required | PM30 = Setting not required | PMC30 = 1 | – | |
| P31 | RXD0 | Input | P31 = Setting not required | PM31 = Setting not required | PMC31 = 1 | – | **Note 1** |
| | INTP7 | Input | P31 = Setting not required | PM31 = Setting not required | PMC31 = 1 | – | **Note 1**, |
| P32 | ASCK0 | Input | P32 = Setting not required | PM32 = Setting not required | PMC32 = 1 | – | **Note 2**, |
| | ADTRG | Input | P32 = Setting not required | PM32 = Setting not required | PMC32 = 1 | – | **Note 2**, |
| | TO01 | Output | P32 = Setting not required | PM32 = Setting not required | PMC32 = 1 | – | PFC32 |
| P33 | TI000 | Input | P33 = Setting not required | PM33 = Setting not required | PMC33 = 1 | PFCE33 = 0 | PFC33 |
| | TO00 | Output | P33 = Setting not required | PM33 = Setting not required | PMC33 = 1 | PFCE33 = 0 | PFC33 |
| | TIP00 | Input | P33 = Setting not required | PM33 = Setting not required | PMC33 = 1 | PFCE33 = 1 | PFC33 |
| | TOP00 | Output | P33 = Setting not required | PM33 = Setting not required | PMC33 = 1 | PFCE33 = 1 | PFC33 |

**Notes 1.** The INTP7 and RXD0 pins are alternate-function pins. When using the pin as the RXD0 pin, disable edge detection of the alte the INTF3.INTF31 and INTR3.INTR31 bits to 0). When using the pin as the INTP7 pin, stop the UART0 receive operation (clea

**2.** The ASCK0 and ADTRG pins are alternate-function pins. When using the pin as the ASCK0 pin, disable the trigger input of pin (clear the ADS.TRG bit to 0 or set the ADS.ADTMD bit to 1). When using the pin as the ADTRG pin, do not set the UAR input (set the CKSR0.TPS03 to CKSR0.TPS00 bits to other than 1011).

**Table 4-14. Settings When Port Pins Are Used for Alternate Functions (2/5)**

| Pin Name | Alternate Function | | Pnx Bit of Pn Register | PMnx Bit of PMn Register | PMCnx Bit of PMCn Register | PFCEnx Bit of PFCEn Register | PFCnx B Re |
|---|---|---|---|---|---|---|---|
| | Function Name | I/O | | | | | |
| P34 | TI001 | Input | P34 = Setting not required | PM34 = Setting not required | PMC34 = 1 | PFCE34 = 0 | PFC34 = |
| | TO00 | Output | P34 = Setting not required | PM34 = Setting not required | PMC34 = 1 | PFCE34 = 0 | PFC34 = |
| | TIP10 | Input | P34 = Setting not required | PM34 = Setting not required | PMC34 = 1 | PFCE34 = 1 | PFC34 = |
| | TOP10 | Output | P34 = Setting not required | PM34 = Setting not required | PMC34 = 1 | PFCE34 = 1 | PFC34 = |
| P35 | TI010 | Input | P35 = Setting not required | PM35 = Setting not required | PMC35 = 1 | – | PFC35 = |
| | TO01 | Output | P35 = Setting not required | PM35 = Setting not required | PMC35 = 1 | – | PFC35 = |
| P38 | SDA0[Note] | I/O | P38 = Setting not required | PM38 = Setting not required | PMC38 = 1 | – | |
| P39 | SCL0[Note] | I/O | P39 = Setting not required | PM39 = Setting not required | PMC39 = 1 | – | |
| P40 | SI00 | Input | P40 = Setting not required | PM40 = Setting not required | PMC40 = 1 | – | |
| P41 | SO00 | Output | P41 = Setting not required | PM41 = Setting not required | PMC41 = 1 | – | |
| P42 | $\overline{\text{SCK00}}$ | I/O | P42 = Setting not required | PM42 = Setting not required | PMC42 = 1 | – | |

**Note** Only in the $\mu$PD703308Y, 70F3306Y, 70F3308Y

**Table 4-14. Settings When Port Pins Are Used for Alternate Functions (3/5)**

| Pin Name | Alternate Function | | Pnx Bit of Pn Register | PMnx Bit of PMn Register | PMCnx Bit of PMCn Register | PFCnx Bit of PFCn Register | |
|---|---|---|---|---|---|---|---|
| | Function Name | I/O | | | | | |
| P50 | TI011 | Input | P50 = Setting not required | PM50 = Setting not required | PMC50 = 1 | PFC50 = 0 | |
| | RTP00 | Output | P50 = Setting not required | PM50 = Setting not required | PMC50 = 1 | PFC50 = 1 | |
| | KR0 | Input | P50 = Setting not required | PM50 = 1 | PMC50 = 0 | PFC50 = 0 | KRM0 (KR |
| P51 | TI50 | Input | P51 = Setting not required | PM51 = Setting not required | PMC51 = 1 | PFC51 = 0 | |
| | RTP01 | Output | P51 = Setting not required | PM51 = Setting not required | PMC51 = 1 | PFC51 = 1 | |
| | KR1 | Input | P51 = Setting not required | PM51 = 1 | PMC51 = 0 | PFC51 = 0 | KRM1 (KR |
| P52 | TO50 | Output | P52 = Setting not required | PM52 = Setting not required | PMC52 = 1 | PFC52 = 0 | |
| | RTP02 | Output | P52 = Setting not required | PM52 = Setting not required | PMC52 = 1 | PFC52 = 1 | |
| | KR2 | Input | P52 = Setting not required | PM52 = 1 | PMC52 = 0 | PFC52 = 0 | KRM2 (KR |
| P53 | SIA0 | Input | P53 = Setting not required | PM53 = Setting not required | PMC53 = 1 | PFC53 = 0 | |
| | RTP03 | Output | P53 = Setting not required | PM53 = Setting not required | PMC53 = 1 | PFC53 = 1 | |
| | KR3 | Input | P53 = Setting not required | PM53 = 1 | PMC53 = 0 | PFC53 = 0 | KRM3 (K |
| P54 | SOA0 | Output | P54 = Setting not required | PM54 = Setting not required | PMC54 = 1 | PFC54 = 0 | PF54 (PF |
| | RTP04 | Output | P54 = Setting not required | PM54 = Setting not required | PMC54 = 1 | PFC54 = 1 | PF54 (PF |
| | KR4 | Input | P54 = Setting not required | PM54 = 1 | PMC54 = 0 | PFC54 = 0 | PF54 (PF |
| P55 | $\overline{\text{SCKA0}}$ | I/O | P55 = Setting not required | PM55 = Setting not required | PMC55 = 1 | PFC55 = 0 | PF55 (PF |
| | RTP05 | Output | P55 = Setting not required | PM55 = Setting not required | PMC55 = 1 | PFC55 = 1 | PF55 (PF |
| | KR5 | Input | P55 = Setting not required | PM55 = 1 | PMC55 = 0 | PFC55 = 0 | PF55 (PF |
| P70 | ANI0 | Input | P70 = Setting not required | – | – | – | |
| P71 | ANI1 | Input | P71 = Setting not required | – | – | – | |
| P72 | ANI2 | Input | P72 = Setting not required | – | – | – | |
| P73 | ANI3 | Input | P73 = Setting not required | – | – | – | |
| P74 | ANI4 | Input | P74 = Setting not required | – | – | – | |
| P75 | ANI5 | Input | P75 = Setting not required | – | – | – | |
| P76 | ANI6 | Input | P76 = Setting not required | – | – | – | |
| P77 | ANI7 | Input | P77 = Setting not required | – | – | – | |

**Table 4-14. Settings When Port Pins Are Used for Alternate Functions (4/5)**

| Pin Name | Alternate Function | | Pnx Bit of Pn Register | PMnx Bit of PMn Register | PMCnx Bit of PMCn Register | PFCnx Bit of PFCn Register | |
|---|---|---|---|---|---|---|---|
| | Function Name | I/O | | | | | |
| P90 | TXD1 | Output | P90 = Setting not required | PM90 = Setting not required | PMC90 = 1 | PFC90 = 1 | |
| | KR6 | Input | P90 = Setting not required | PM90 = 1 | PMC90 = 0 | PFC90 = 0 | KRM6 ( |
| P91 | RXD1 | Input | P91 = Setting not required | PM91 = Setting not required | PMC91 = 1 | PFC91 = 1 | |
| | KR7 | Input | P91 = Setting not required | PM91 = 1 | PMC91 = 0 | PFC91 = 0 | KRM7 ( |
| P96 | TI51 | Input | P96 = Setting not required | PM96 = 1 | PMC96 = 0 | PFC96 = 0 | |
| | TO51 | Output | P96 = Setting not required | PM96 = Setting not required | PMC96 = 1 | PFC96 = 1 | |
| P97 | SI01 | Input | P97 = Setting not required | PM97 = Setting not required | PMC97 = 1 | PFC97 = 1 | |
| P98 | SO01 | Output | P98 = Setting not required | PM98 = Setting not required | PMC98 = 1 | PFC98 = 1 | PF98 (P |
| P99 | $\overline{\text{SCK01}}$ | I/O | P99 = Setting not required | PM99 = Setting not required | PMC99 = 1 | PFC99 = 1 | PF98 (P |
| P913 | INTP4 | Input | P913 = Setting not required | PM913 = Setting not required | PMC913 = 1 | PFC913 = 1 | |
| P914 | INTP5 | Input | P914 = Setting not required | PM914 = Setting not required | PMC914 = 1 | PFC914 = 1 | |
| P915 | INTP6 | Input | P915 = Setting not required | PM915 = Setting not required | PMC915 = 1 | PFC915 = 1 | |
| PCM0 | $\overline{\text{WAIT}}$ | Input | PCM0 = Setting not required | PMCM0 = Setting not require | PMCCM0 = 1 | – | |
| PCM1 | CLKOUT | Output | PCM1 = Setting not required | PMCM1 = Setting not require | PMCCM1 = 1 | – | |
| PCM2 | $\overline{\text{HLDAK}}$ | Output | PCM2 = Setting not required | PMCM2 = Setting not require | PMCCM2 = 1 | – | |
| PCM3 | $\overline{\text{HLDRQ}}$ | Input | PCM3 = Setting not required | PMCM3 = Setting not require | PMCCM3 = 1 | – | |
| PCS0 | $\overline{\text{CS0}}$ | Output | PCS0 = Setting not required | PMCS0 = Setting not required | PMCCS0 = 1 | – | |
| PCS1 | $\overline{\text{CS1}}$ | Output | PCS1 = Setting not required | PMCS1 = Setting not required | PMCCS1 = 1 | – | |
| PCT0 | $\overline{\text{WR0}}$ | Output | PCT0 = Setting not required | PMCT0 = Setting not required | PMCCT0 = 1 | – | |
| PCT1 | $\overline{\text{WR1}}$ | Output | PCT1 = Setting not required | PMCT1 = Setting not required | PMCCT1 = 1 | – | |
| PCT4 | $\overline{\text{RD}}$ | Output | PCT4 = Setting not required | PMCT4 = Setting not required | PMCCT4 = 1 | – | |
| PCT6 | ASTB | Output | PCT6 = Setting not required | PMCT6 = Setting not required | PMCCT6 = 1 | – | |

**Table 4-14. Settings When Port Pins Are Used for Alternate Functions (5/5)**

| Pin Name | Alternate Function | | Pnx Bit of Pn Register | PMnx Bit of PMn Register | PMCnx Bit of PMCn Register | PFCnx Bit of PFCn Register | |
|---|---|---|---|---|---|---|---|
| | Function Name | I/O | | | | | |
| PDL0 | AD0 | I/O | PDL0 = Setting not required | PMDL0 = Setting not required | PMCDL0 = 1 | – | |
| PDL1 | AD1 | I/O | PDL1 = Setting not required | PMDL1 = Setting not required | PMCDL1 = 1 | – | |
| PDL2 | AD2 | I/O | PDL2 = Setting not required | PMDL2 = Setting not required | PMCDL2 = 1 | – | |
| PDL3 | AD3 | I/O | PDL3 = Setting not required | PMDL3 = Setting not required | PMCDL3 = 1 | – | |
| PDL4 | AD4 | I/O | PDL4 = Setting not required | PMDL4 = Setting not required | PMCDL4 = 1 | – | |
| PDL5 | AD5 | I/O | PDL5 = Setting not required | PMDL5 = Setting not required | PMCDL5 = 1 | – | |
| PDL6 | AD6 | I/O | PDL6 = Setting not required | PMDL6 = Setting not required | PMCDL6 = 1 | – | |
| PDL7 | AD7 | I/O | PDL7 = Setting not required | PMDL7 = Setting not required | PMCDL7 = 1 | – | |
| PDL8 | AD8 | I/O | PDL8 = Setting not required | PMDL8 = Setting not required | PMCDL8 = 1 | – | |
| PDL9 | AD9 | I/O | PDL9 = Setting not required | PMDL9 = Setting not required | PMCDL9 = 1 | – | |
| PDL10 | AD10 | I/O | PDL10 = Setting not required | PMDL10 = Setting not required | PMCDL10 = 1 | – | |
| PDL11 | AD11 | I/O | PDL11 = Setting not required | PMDL11 = Setting not required | PMCDL11 = 1 | – | |
| PDL12 | AD12 | I/O | PDL12 = Setting not required | PMDL12 = Setting not required | PMCDL12 = 1 | – | |
| PDL13 | AD13 | I/O | PDL13 = Setting not required | PMDL13 = Setting not required | PMCDL13 = 1 | – | |
| PDL14 | AD14 | I/O | PDL14 = Setting not required | PMDL14 = Setting not required | PMCDL14 = 1 | – | |
| PDL15 | AD15 | I/O | PDL15 = Setting not required | PMDL15 = Setting not required | PMCDL15 = 1 | – | |

## 4.6 Cautions

### 4.6.1 Cautions on bit manipulation instruction for port n register (Pn)

When a 1-bit manipulation instruction is executed on a port that provides both input and output functions, the value of the output latch of an input port that is not subject to manipulation may be written in addition to the targeted bit.

Therefore, it is recommended to rewrite the output latch when switching a port from input mode to output mode.

<Example> When PDL0 is an output port, PDL1 to PDL7 are input ports (all pin statuses are high level), and the value of the port latch is 00H, if the output of output port PDL0 is changed from low level to high level via a bit manipulation instruction, the value of the port latch is FFH.

Explanation: The targets of writing to and reading from the Pn register of a port whose PMnm bit is 1 are the output latch and pin status, respectively.

A bit manipulation instruction is executed in the following order in the V850ES/KF1+.

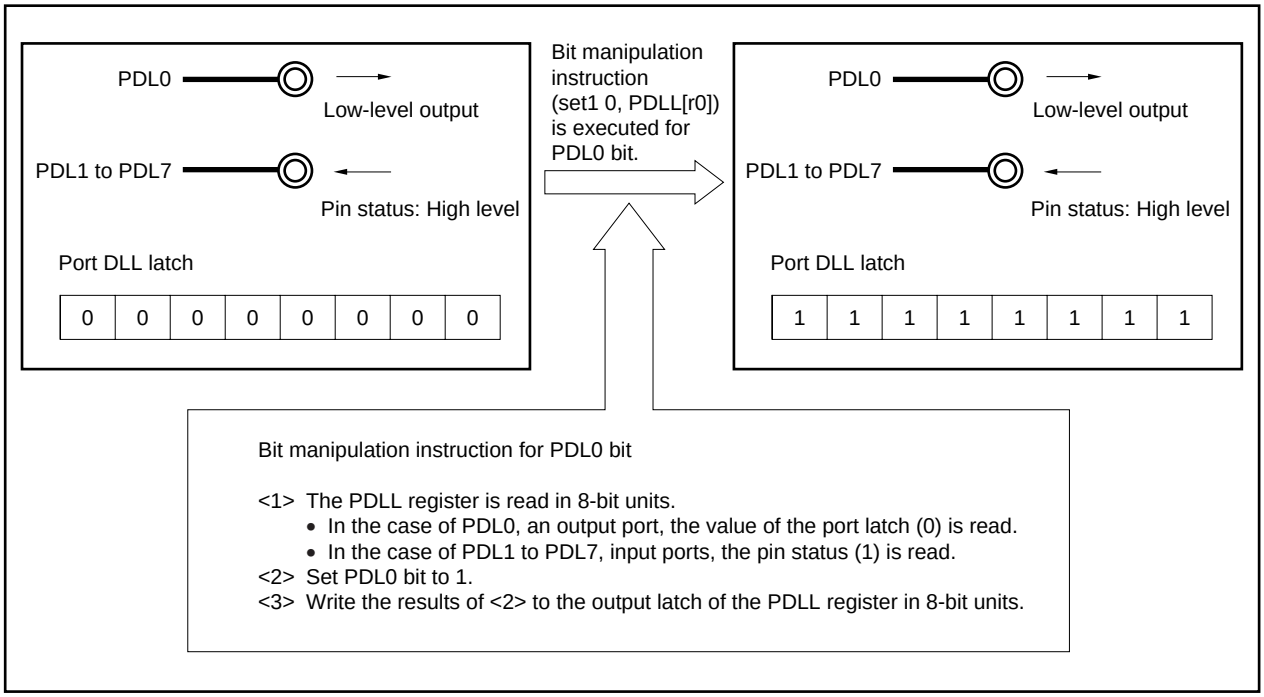
<1> The Pn register is read in 8-bit units.
<2> The targeted one bit is manipulated.
<3> The Pn register is written in 8-bit units.

In step <1>, the value of the output latch (0) of PDL0, which is an output port, is read, while the pin statuses of PDL1 to PDL7, which are input ports, are read. If the pin statuses of PDL1 to PDL7 are high level at this time, the read value is FEH.

The value is changed to FFH by the manipulation in <2>.

FFH is written to the output latch by the manipulation in <3>.

**Figure 4-25. Bit Manipulation Instruction (PDL0)**



### 4.6.2 Hysteresis characteristics

In port mode, the following ports do not have hysteresis characteristics.

P02 to P06
P31 to P35, P38, P39
P40, P42
P97, P99, P913 to P915

# CHAPTER 5 BUS CONTROL FUNCTION

The V850ES/KF1+ is provided with an external bus interface function by which external memories such as ROM and RAM, and I/O can be connected.

## 5.1 Features

- Output is possible from a multiplex bus with a minimum of 3 bus cycles
- Chip select function for up to 2 spaces
- 8-bit/16-bit data bus selectable (for each area selected by chip select function)
- Wait function
  - Programmable wait function of up to 7 states (selectable for each area selected by chip select function)
  - External wait function using $\overline{\text{WAIT}}$ pin
- Idle state function
- Bus hold function

## 5.2 Bus Control Pins

The pins used to connect an external device are listed in the table below.

**Table 5-1. Bus Control Pins**

| Bus Control Pin | Alternate-Function Pin | I/O | Function |
|---|---|---|---|
| AD0 to AD15 | PDL0 to PDL15 | I/O | Address/data bus |
| $\overline{\text{WAIT}}$ | PCM0 | Input | External wait control |
| CLKOUT | PCM1 | Output | Internal system clock output |
| $\overline{\text{CS0}}$, $\overline{\text{CS1}}$ | PCS0, PCS1 | Output | Chip select |
| $\overline{\text{WR0}}$, $\overline{\text{WR1}}$ | PCT0, PCT1 | Output | Write strobe signal |
| $\overline{\text{RD}}$ | PCT4 | Output | Read strobe signal |
| ASTB | PCT6 | Output | Address strobe signal |
| $\overline{\text{HLDRQ}}$ | PCM3 | Input | Bus hold control |
| $\overline{\text{HLDAK}}$ | PCM2 | Output | |

### 5.2.1 Pin status when internal ROM, internal RAM, or on-chip peripheral I/O is accessed

When the internal ROM, internal RAM, or on-chip peripheral I/O is accessed, the status of each pin is as follows.

**Table 5-2. Pin Statuses When Internal ROM, Internal RAM, or On-Chip Peripheral I/O Is Accessed**

| | |
|---|---|
| Address/data bus (AD15 to AD0) | Undefined |
| Control signal | Inactive |

**Caution When a write access is performed to the internal ROM area, address, data, and control signals are activated in the same way as access to the external memory area.**

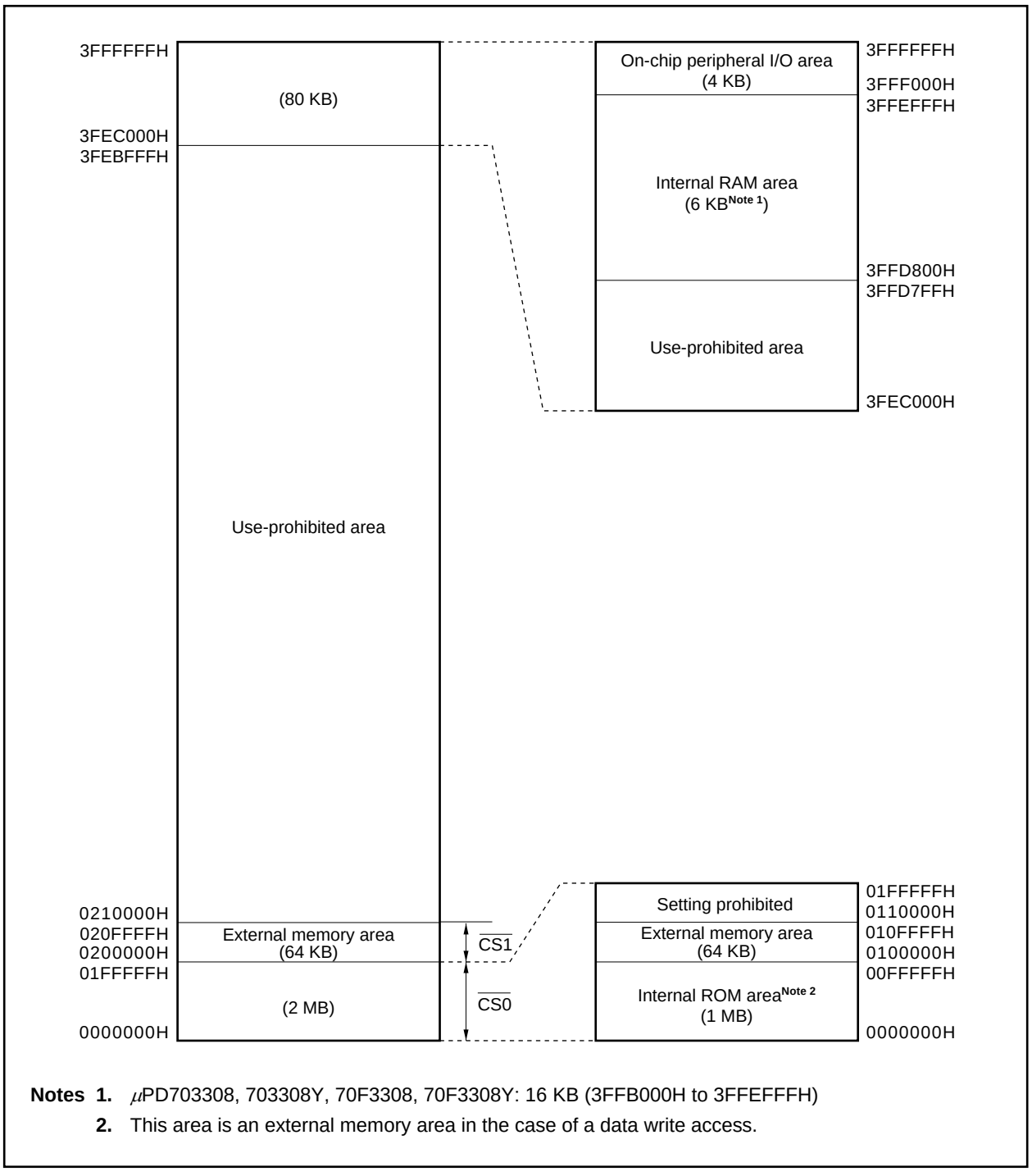
### 5.2.2 Pin status in each operation mode

For the pin status of the V850ES/KF1+ in each operation mode, refer to **2.2 Pin Status**.

## 5.3 Memory Block Function

The 64 MB memory space is divided into chip select areas of (lower) 64 KB and 64 KB. The programmable wait function and bus cycle operation mode for each of these chip select areas can be independently controlled.

**Figure 5-1. Data Memory Map: Physical Address**




**Notes 1.** $\mu$PD703308, 703308Y, 70F3308, 70F3308Y: 16 KB (3FFB000H to 3FFEFFFH)

**2.** This area is an external memory area in the case of a data write access.

### 5.3.1 Chip select control function

Of the 64 MB (linear) address space, two 64 KB spaces (0100000H to 010FFFFH/0200000H to 020FFFFH) include two chip select control functions, $\overline{CS0}$ and $\overline{CS1}$. The areas that can be selected by $\overline{CS0}$ and $\overline{CS1}$ are fixed.

By using these chip select control functions, the memory space can be used effectively. The allocation of the chip select areas is shown in the table below.

| | |
|---|---|
| $\overline{CS0}$ | 0100000H to 010FFFFH (64 KB) |
| $\overline{CS1}$ | 0200000H to 020FFFFH (64 KB) |

## 5.4 Bus Access

### 5.4.1 Number of clocks for access

The following table shows the number of base clocks required for accessing each resource.

| Area (Bus Width) / Bus Cycle Type | Internal ROM (32 Bits) | Internal RAM (32 Bits) | External Memory (16 Bits) | On-Chip Peripheral I/O (16 Bits) |
|---|---|---|---|---|
| Instruction fetch (normal access) | 1 | 1 Note 1 | 3 + n | – |
| Instruction fetch (branch) | 2 | 2 Note 1 | 3 + n | – |
| Operand data access | 3 | 1 | 3 + n | 3 Note 2 |

**Notes 1.** If the access conflicts with a data access, the number of clock is increased by 1.

**2.** This value varies depending on the setting of the VSWC register.

**Remark** Unit: Clocks/access

### 5.4.2 Bus size setting function

The bus size of each external memory area selected by $\overline{CSn}$ can be set to 8 bits or 16 bits by using the BSC register.

The external memory area of the V850ES/KF1+ is selected by $\overline{CS0}$ and $\overline{CS1}$.

**(1) Bus size configuration register (BSC)**

This register can be read or written in 16-bit units.

After reset, BSC is set to 5555H.

**Caution Write to the BSC register after reset, and then do not change the set values. Also, do not access an external memory area until the initial settings of the BSC register are complete.**

After reset: 5555H R/W Address: FFFFF066H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| BSC | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | 0 | 0/1 Note | 0 | 0/1 Note | 0 | BS10 | 0 | BS00 |
| $\overline{CSn}$ signal | | | | | | $\overline{CS1}$ | | $\overline{CS0}$ |

| BSn0 | Data bus width of CSn space (n = 0, 1) |
|---|---|
| 0 | 8 bits |
| 1 | 16 bits |

**Note** Changing the value does not affect the operation.

**Caution Be sure to set bits 14, 12, 10, and 8 to 1, and clear bits 15, 13, 11, 9, 7, 5, 3, and 1 to 0.**

### 5.4.3 Access by bus size

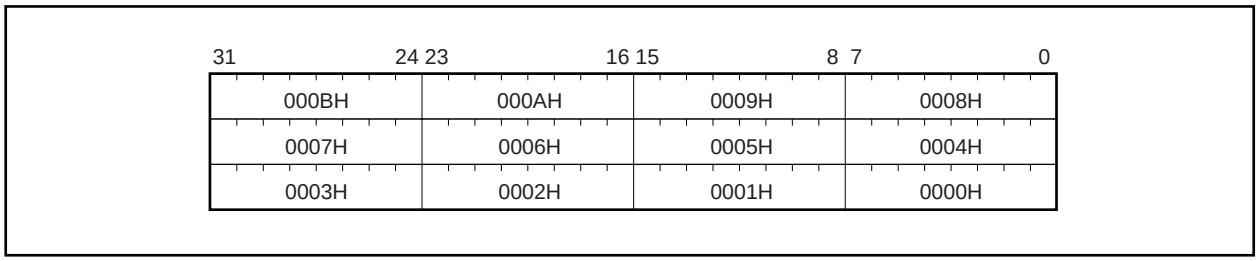
The V850ES/KF1+ accesses the on-chip peripheral I/O and external memory in 8-bit, 16-bit, or 32-bit units. The bus size is as follows.

- The bus size of the on-chip peripheral I/O is fixed to 16 bits.
- The bus size of the external memory is selectable from 8 bits or 16 bits (by using the BSC register).

The operation when each of the above is accessed is described below. All data is accessed starting from the lower side.

The V850ES/KF1+ supports only the little endian format.

**Figure 5-2. Little Endian Address in Word**




**(1) Data space**

The V850ES/KF1+ has an address misalign function.

With this function, data can be placed at all addresses, regardless of the format of the data (word data or halfword data). However, if the word data or halfword data is not aligned at the boundary, a bus cycle is generated at least twice, causing the bus efficiency to drop.

**(a) Halfword-length data access**

A byte-length bus cycle is generated twice if the least significant bit of the address is 1.

**(b) Word-length data access**

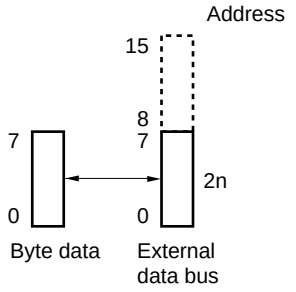
(i) A byte-length bus cycle, halfword-length bus cycle, and byte-length bus cycle are generated in that order if the least significant bit of the address is 1.

(ii) A halfword-length bus cycle is generated twice if the lower 2 bits of the address are 10.

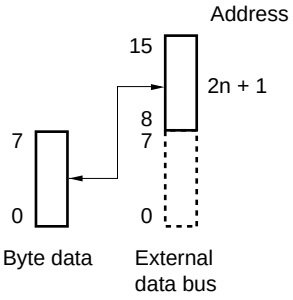
### (2) Byte access (8 bits)

#### (a) 16-bit data bus width

<1> Access to even address (2n)

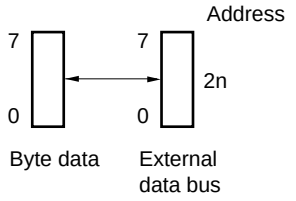



<2> Access to odd address (2n + 1)

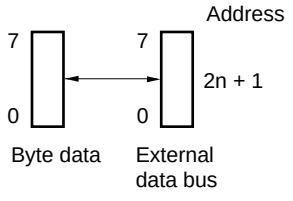



#### (b) 8-bit data bus width

<1> Access to even address (2n)

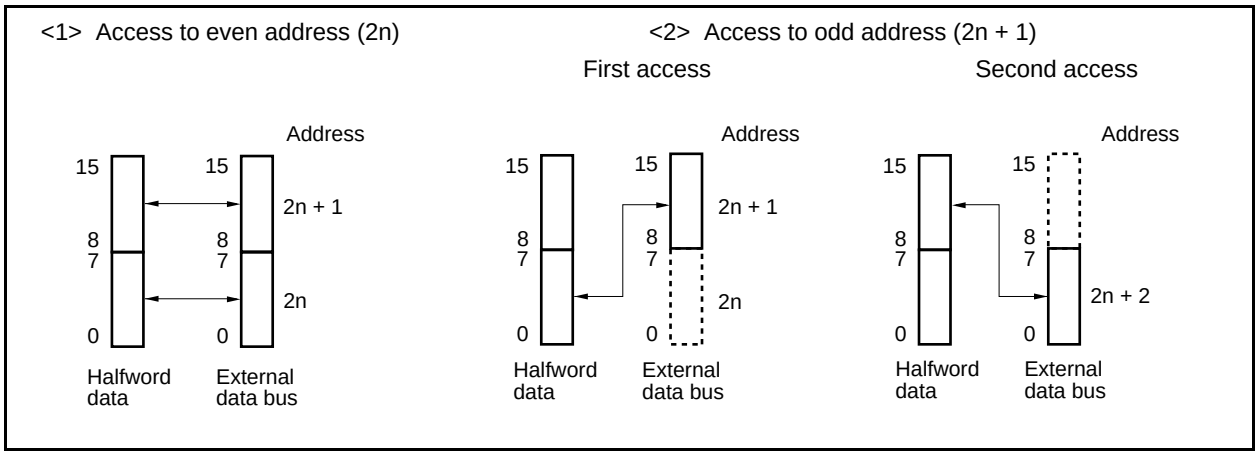



<2> Access to odd address (2n + 1)

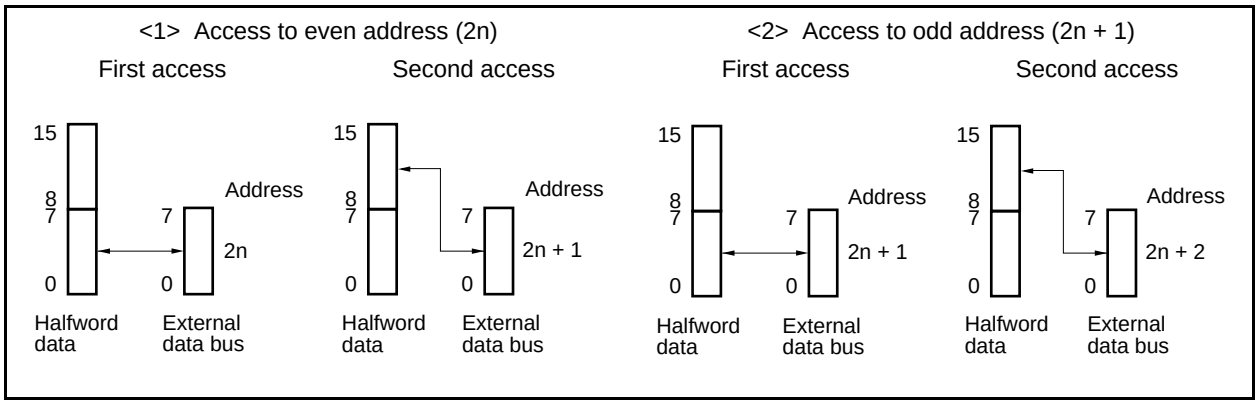


### (3) Halfword access (16 bits)

#### (a) With 16-bit data bus width

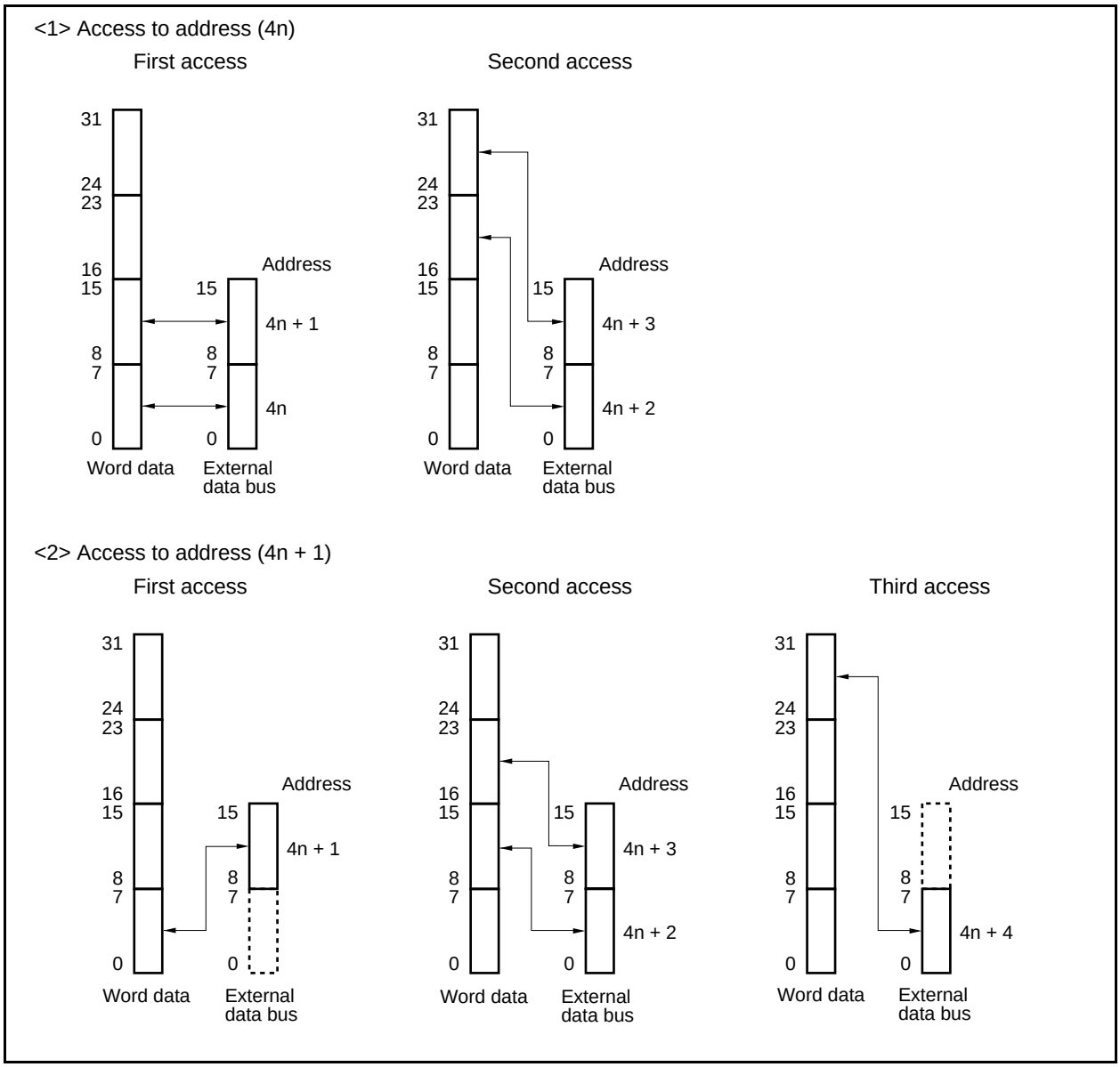



#### (b) 8-bit data bus width



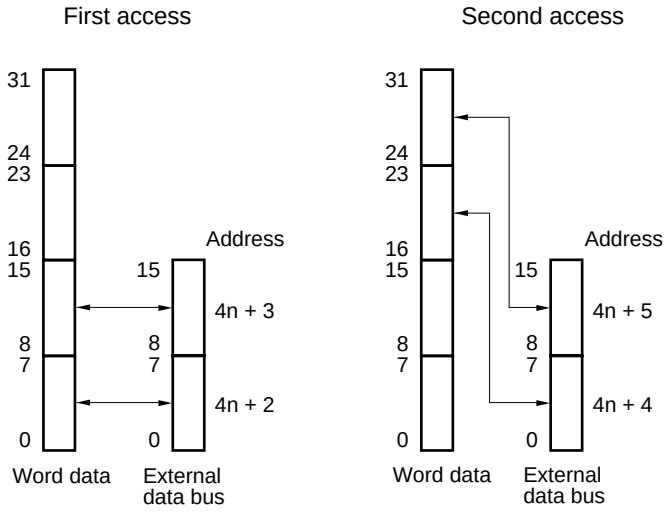
### (4) Word access (32 bits)

#### (a) 16-bit data bus width (1/2)

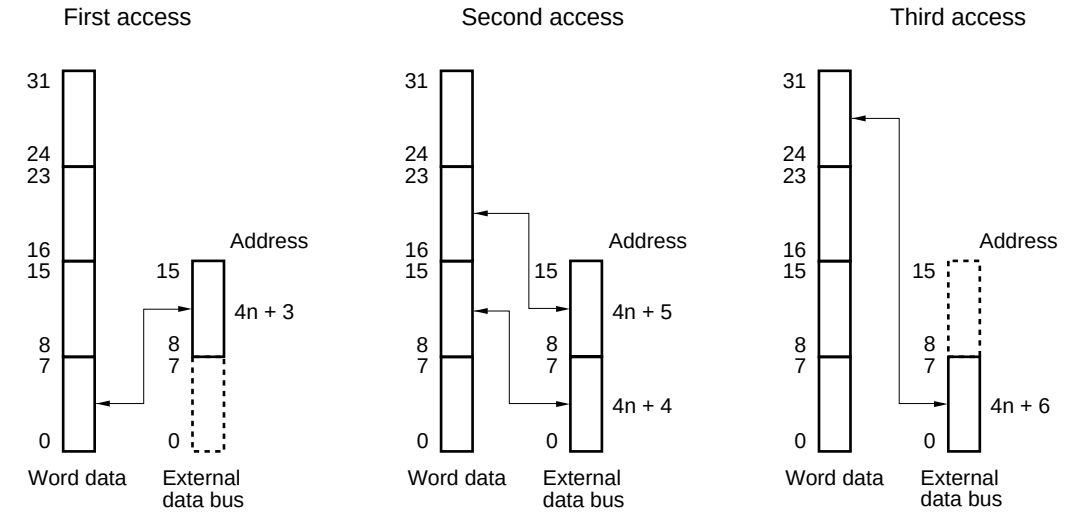


**(a) 16-bit data bus width (2/2)**

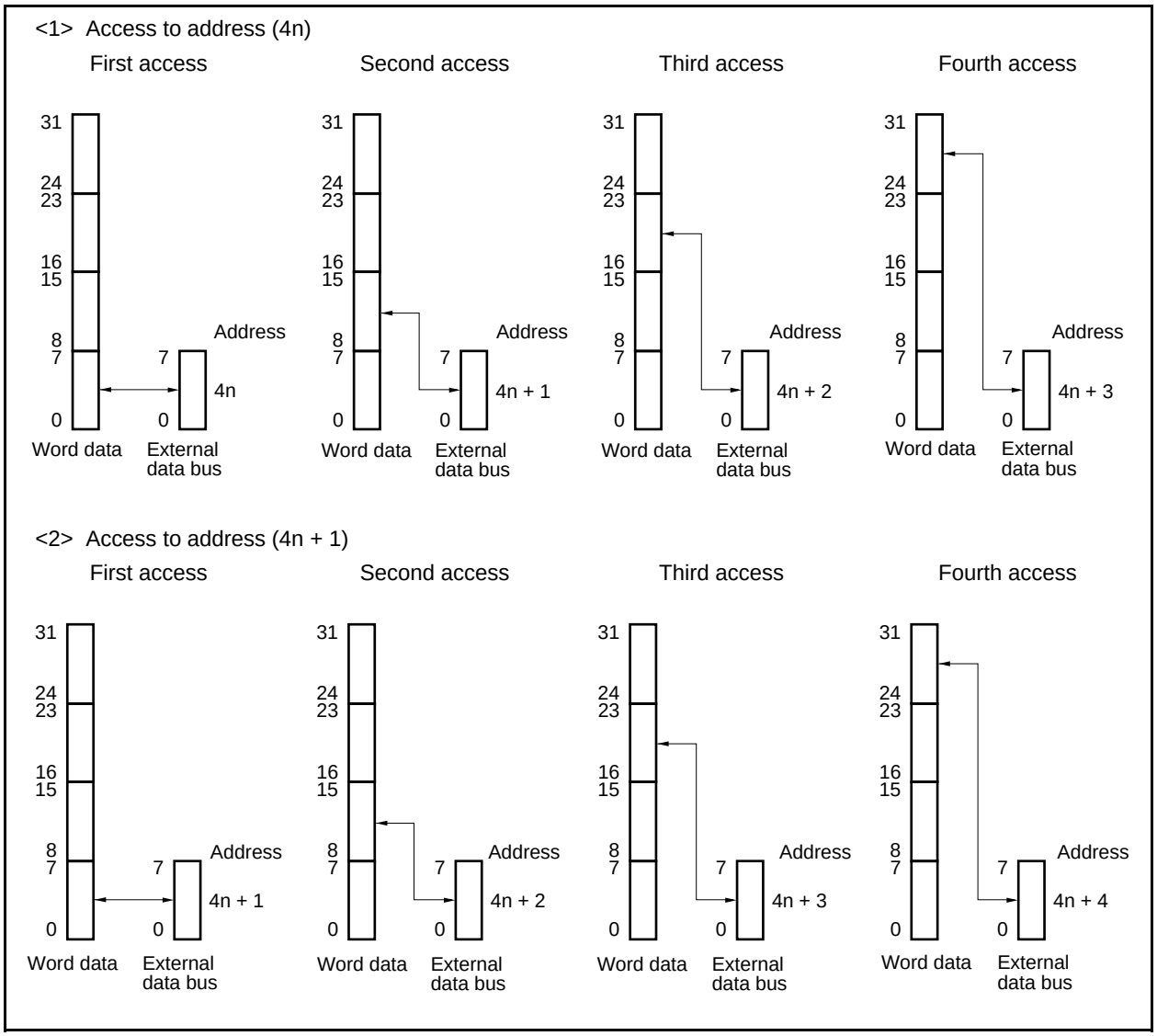
<3> Access to address (4n + 2)



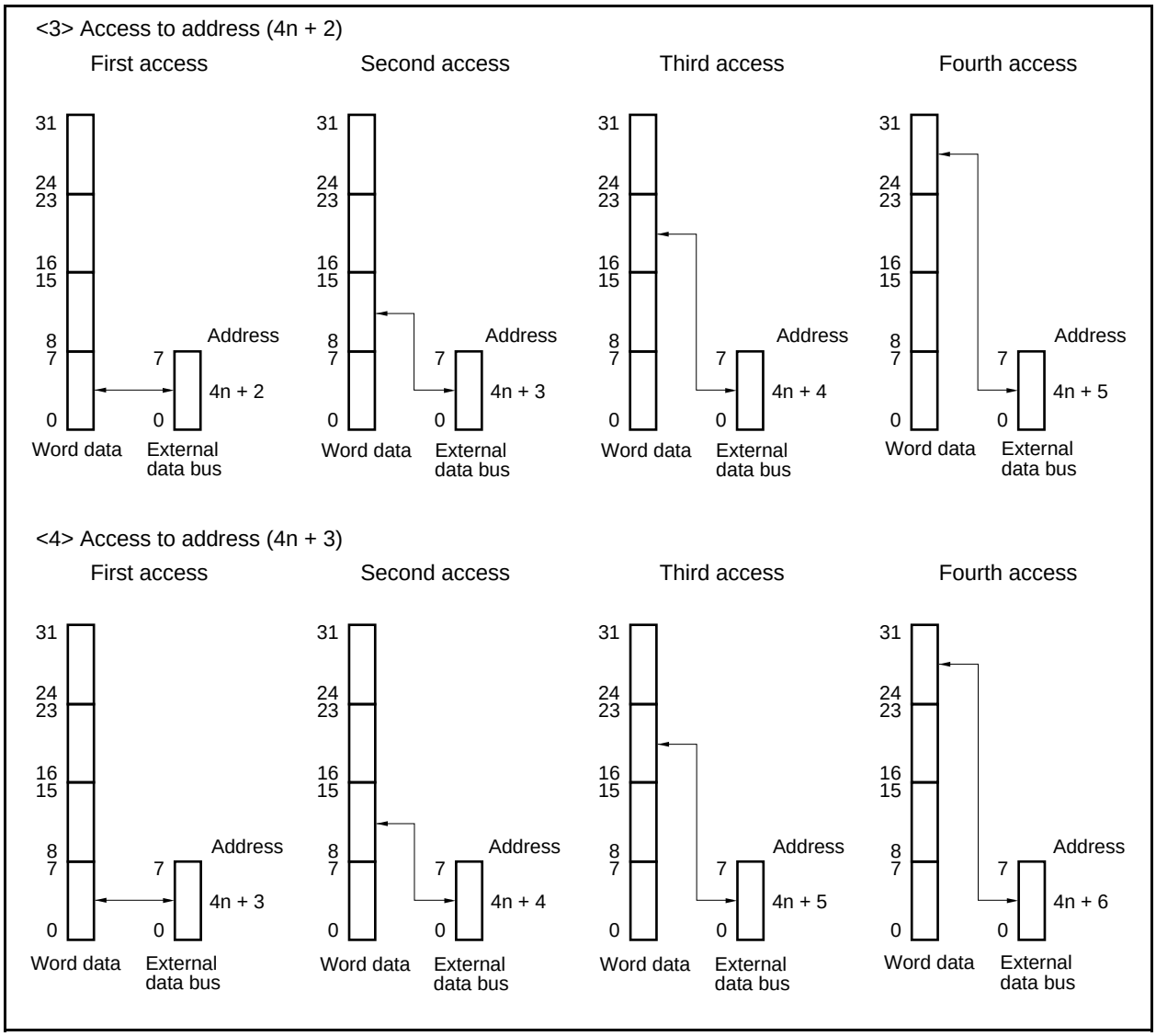

<4> Access to address (4n + 3)



**(b) 8-bit data bus width (1/2)**



**(b) 8-bit data bus width (2/2)**



## 5.5 Wait Function

### 5.5.1 Programmable wait function

**(1) Data wait control register 0 (DWC0)**

To realize interfacing with a low-speed memory or I/O, up to seven data wait states can be inserted in the bus cycle that is executed for each CS space.

The number of wait states can be programmed by using the DWC0 register. Immediately after system reset, 7 data wait states are inserted for all the chip select areas.

The DWC0 register can be read or written in 16-bit units.

After reset, DWC0 is set to 7777H.

**Cautions 1. The internal ROM and internal RAM areas are not subject to programmable wait, and are always accessed without a wait state. The on-chip peripheral I/O area is also not subject to programmable wait, and only wait control from each peripheral function is performed.**

**2. Write to the DWC0 register after reset, and then do not change the set values. Also, do not access an external memory area until the initial settings of the DWC0 register are complete.**

After reset: 7777H R/W Address: FFFFF484H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| DWC0 | 0 | 0/1Note | 0/1Note | 0/1Note | 0 | 0/1Note | 0/1Note | 0/1Note |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | 0 | DW12 | DW11 | DW10 | 0 | DW02 | DW01 | DW00 |

$\overline{CSn}$ signal: $\overline{CS1}$ (DW12, DW11, DW10) $\overline{CS0}$ (DW02, DW01, DW00)

| DWn2 | DWn1 | DWn0 | Number of wait states inserted in CSn space (n = 0, 1) |
|---|---|---|---|
| 0 | 0 | 0 | None |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 2 |
| 0 | 1 | 1 | 3 |
| 1 | 0 | 0 | 4 |
| 1 | 0 | 1 | 5 |
| 1 | 1 | 0 | 6 |
| 1 | 1 | 1 | 7 |

**Note** Changing the value does not affect the operation.

**Caution Be sure to clear bits 15, 11, 7, and 3 to 0.**

### 5.5.2 External wait function

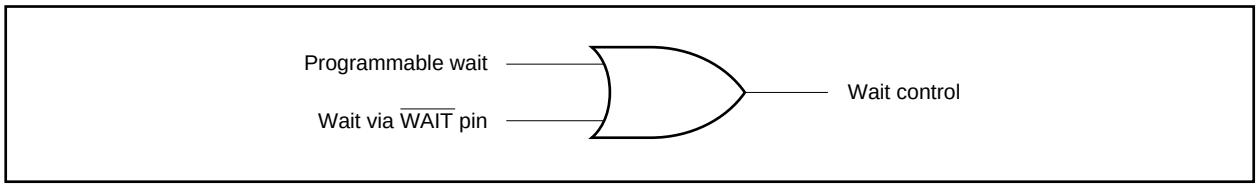
To synchronize an extremely slow memory, I/O, or asynchronous system, any number of wait states can be inserted in the bus cycle by using the external wait pin ($\overline{\text{WAIT}}$).

Access to each area of the internal ROM, internal RAM, and on-chip peripheral I/O is not subject to control by the external wait function, in the same manner as the programmable wait function.

The $\overline{\text{WAIT}}$ signal can be input asynchronously to CLKOUT, and is sampled at the falling edge of the clock in the T2 and TW states of the bus cycle. If the setup/hold time of the sampling timing is not satisfied, a wait state is inserted in the next state, or not inserted at all.

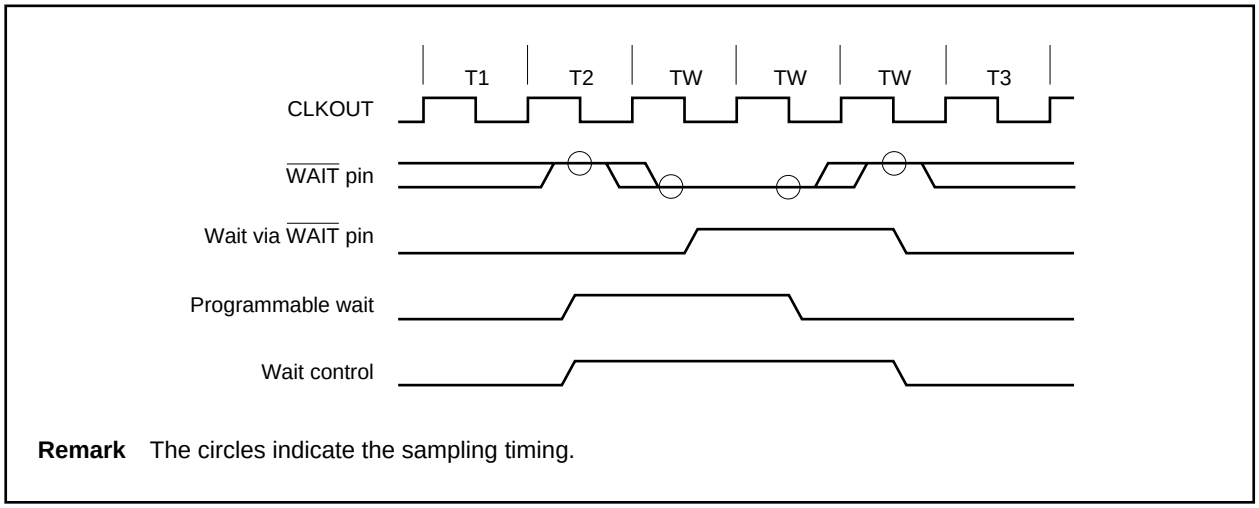
### 5.5.3 Relationship between programmable wait and external wait

Wait cycles are inserted as the result of an OR operation between the wait cycles specified by the set value of the programmable wait and the wait cycles controlled by the $\overline{\text{WAIT}}$ pin.




For example, if the timing of the programmable wait and the $\overline{\text{WAIT}}$ pin signal is as illustrated below, three wait states will be inserted in the bus cycle.

**Figure 5-3. Example of Inserting Wait States**




**Remark** The circles indicate the sampling timing.

### 5.5.4 Programmable address wait function

Address-setup or address-hold waits to be inserted in each bus cycle can be set by using the AWC register. Address wait insertion is set for each chip select area ($\overline{CS0}$ and $\overline{CS1}$).

If an address setup wait is inserted, it seems that the high-clock period of T1 state is extended by 1 clock. If an address hold wait is inserted, it seems that the low-clock period of T1 state is extended by 1 clock.

**(1) Address wait control register (AWC)**

This register can be read or written in 16-bit units.

After reset, AWC is set to FFFFH.

**Cautions 1. The internal ROM, internal RAM, and on-chip peripheral I/O areas are not subject to address setup wait or address hold wait insertion.**

**2. Write the AWC register after reset, and then do not change the set values. Also, do not access an external memory area until the initial settings of the AWC register are complete.**

After reset: FFFFH R/W Address: FFFFF488H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| AWC | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0/1[Note] | 0/1[Note] | 0/1[Note] | 0/1[Note] | AHW1 | ASW1 | AHW0 | ASW0 |

$\overline{CSn}$ signal: AHW1 → $\overline{CS1}$, AHW0 → $\overline{CS0}$

| AHWn | Specifies insertion of address hold wait (n = 0, 1) |
|---|---|
| 0 | Not inserted |
| 1 | Inserted |

| ASWn | Specifies insertion of address setup wait (n = 0, 1) |
|---|---|
| 0 | Not inserted |
| 1 | Inserted |

**Note** Changing the value does not affect the operation.

**Caution Be sure to set bits 15 to 8 to 1.**

## 5.6 Idle State Insertion Function

To facilitate interfacing with low-speed memories, one idle state (TI) can be inserted after the T3 state in the bus cycle that is executed for each space selected by $\overline{CSn}$. By inserting idle states, the data output float delay time of the memory can be secured during read access (an idle state cannot be inserted during write access).

Whether the idle state is to be inserted can be programmed by using the BCC register.

An idle state is inserted for all the areas immediately after system reset.

**(1) Bus cycle control register (BCC)**

This register can be read or written in 16-bit units.

After reset, BCC is set to AAAAH.

**Cautions 1. The internal ROM, internal RAM, and on-chip peripheral I/O areas are not subject to idle state insertion.**

**2. Write to the BCC register after reset, and then do not change the set values. Also, do not access an external memory area until the initial settings of the BCC register are complete.**

After reset: AAAAH R/W Address: FFFFF48AH

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| BCC | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| 0/1[Note] | 0 | 0/1[Note] | 0 | BC11 | 0 | BC01 | 0 |

$\overline{CSn}$ signal: $\overline{CS1}$ (BC11), $\overline{CS0}$ (BC01)

| BCn1 | Specifies insertion of idle state (n = 0, 1) |
|---|---|
| 0 | Not inserted |
| 1 | Inserted |

**Note** Changing the value does not affect the operation.

**Caution Be sure to set bits 15, 13, 11, and 9 to 1, and clear bits 14, 12, 10, 8, 6, 4, 2, and 0 to 0.**

## 5.7 Bus Hold Function

### 5.7.1 Functional outline

The $\overline{\text{HLDRQ}}$ and $\overline{\text{HLDAK}}$ functions are valid if the PCM2 and PCM3 pins are set to their alternate functions.

When the $\overline{\text{HLDRQ}}$ pin is asserted (low level), indicating that another bus master has requested bus mastership, the external address/data bus goes into a high-impedance state and is released (bus hold status). If the request for the bus mastership is cleared and the $\overline{\text{HLDRQ}}$ pin is deasserted (high level), driving these pins is started again.

During the bus hold period, execution of the program in the internal ROM and internal RAM is continued until a peripheral I/O register or the external memory is accessed.

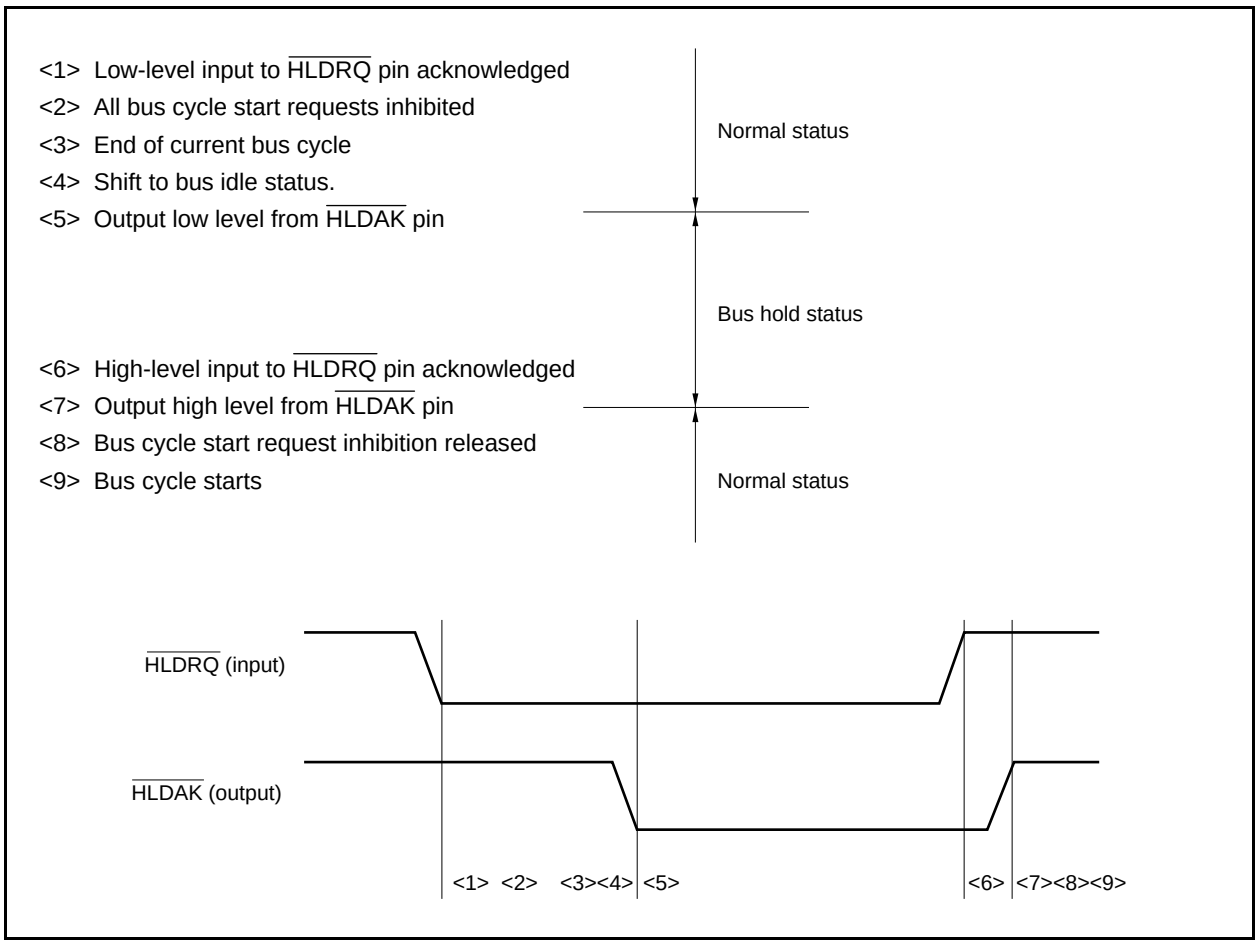
The bus hold status is indicated by assertion (low level) of the $\overline{\text{HLDAK}}$ pin. The bus hold function enables the configuration of multi-processor type systems in which two or more bus masters exist.

Note that the bus hold request is not acknowledged during a multiple-access cycle initiated by the bus sizing function or a bit manipulation instruction.

| Status | Data Bus Width | Access Type | Timing in Which Bus Hold Request Not Acknowledged |
|---|---|---|---|
| CPU bus lock | 16 bits | Word access to even address | Between first and second access |
| | | Word access to odd address | Between first and second access |
| | | | Between second and third access |
| | | Halfword access to odd address | Between first and second access |
| | 8 bits | Word access | Between first and second access |
| | | | Between second and third access |
| | | | Between third and fourth access |
| | | Halfword access | Between first and second access |
| Read-modify-write access of bit manipulation instruction | – | – | Between read access and write access |

### 5.7.2 Bus hold procedure

The bus hold status transition procedure is shown below.




### 5.7.3 Operation in power save mode

Because the internal system clock is stopped in the STOP and IDLE modes, the bus hold status is not entered even if the HLDRQ pin is asserted.

In the HALT mode, the HLDAK pin is asserted as soon as the HLDRQ pin has been asserted, and the bus hold status is entered. When the HLDRQ pin is later deasserted, the HLDAK pin is also deasserted, and the bus hold status is cleared.

## 5.8 Bus Priority

Bus hold, instruction fetch (branch), instruction fetch (successive), and operand data access are executed in the external bus cycle.

Bus hold has the highest priority, followed by operand data access, instruction fetch (branch), and instruction fetch (successive).

An instruction fetch may be inserted between the read access and write access in a read-modify-write access.

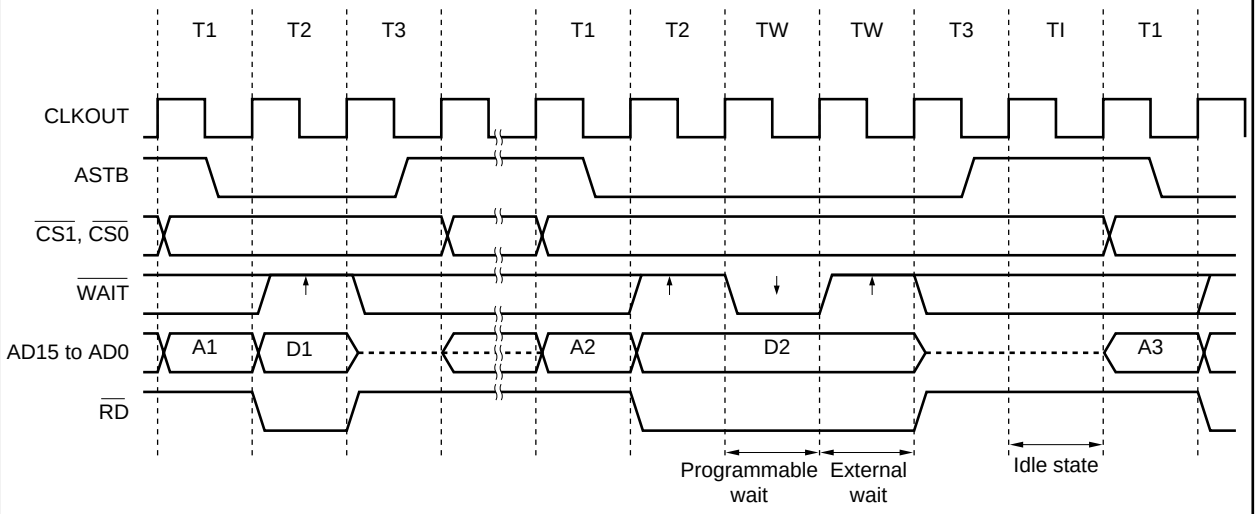
If an instruction is executed for two or more accesses, an instruction fetch and bus hold are not inserted between accesses due to bus size limitations.

**Table 5-3. Bus Priority**

| Priority | External Bus Cycle | Bus Master |
|---|---|---|
| High | Bus hold | External device |
| ↕ | Operand data access | CPU |
| | Instruction fetch (branch) | CPU |
| Low | Instruction fetch (successive) | CPU |

## 5.9 Bus Timing

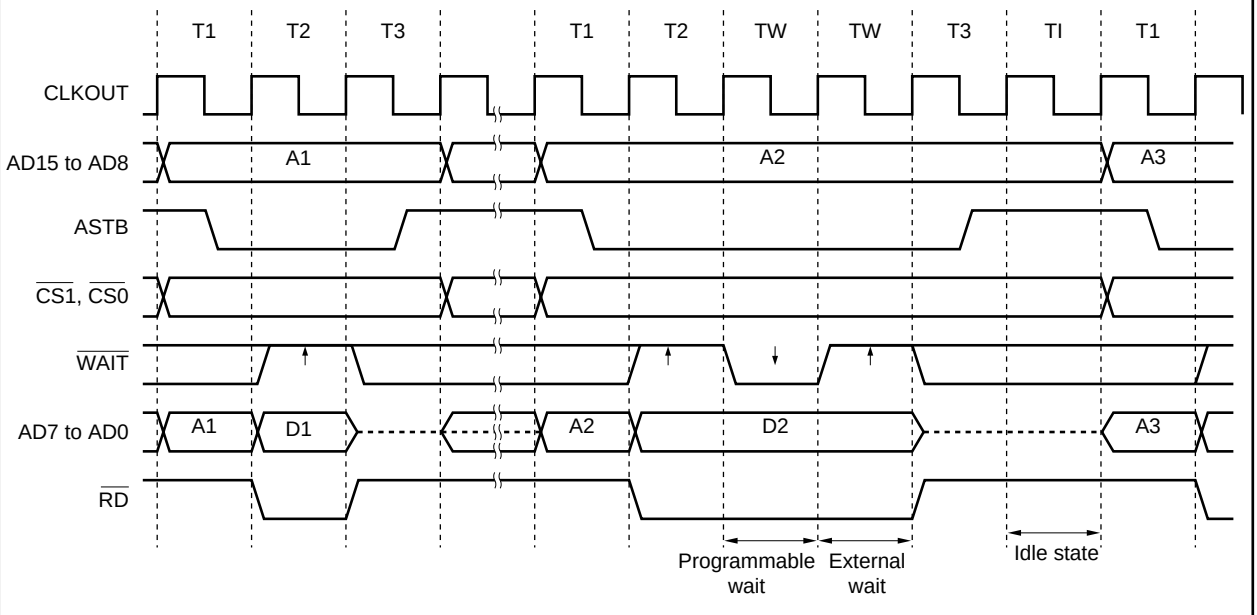
**Figure 5-4. Multiplex Bus Read Timing (Bus Size: 16 Bits, 16-Bit Access)**




| 8-bit access | Odd address | Even address |
|---|---|---|
| AD15 to AD8 | Active | Hi-Z |
| AD7 to AD0 | Hi-Z | Active |

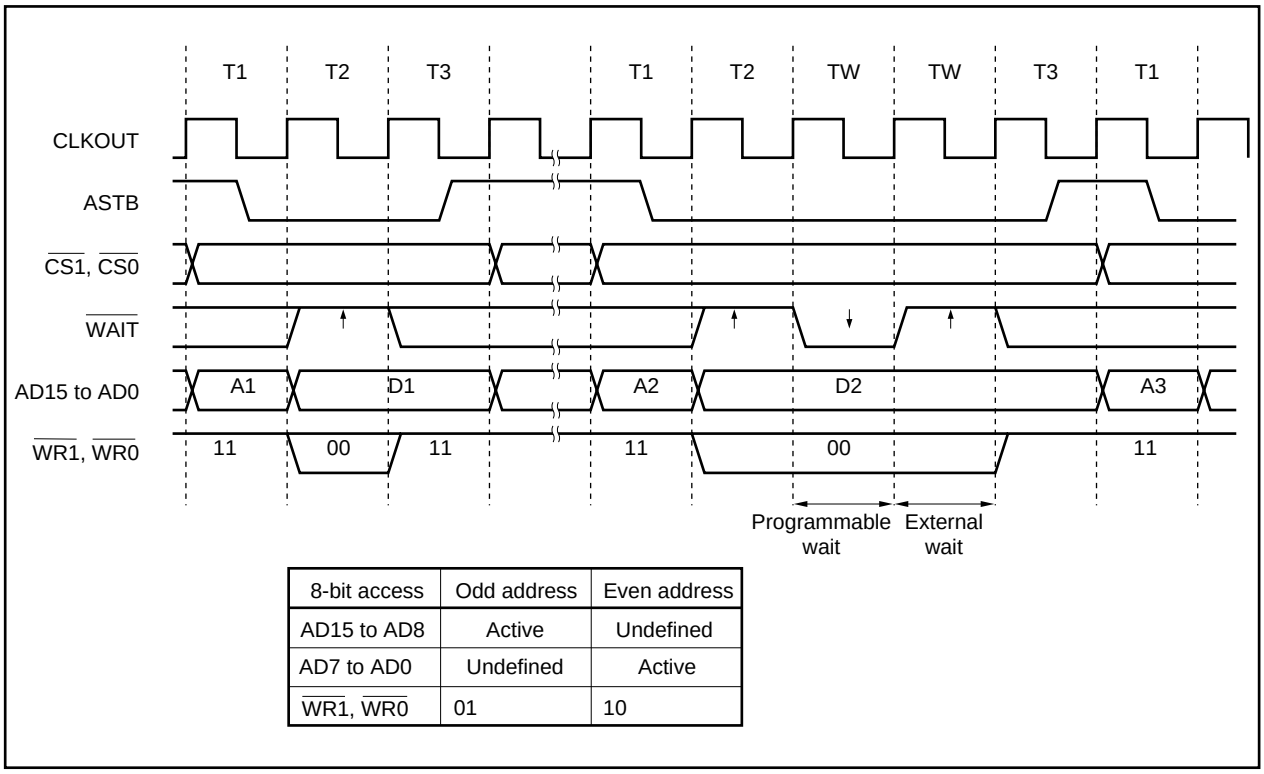
**Remark** The broken lines indicate high impedance.

**Figure 5-5. Multiplex Bus Read Timing (Bus Size: 8 Bits)**




**Remark** The broken lines indicate high impedance.

**Figure 5-6. Multiplex Bus Write Timing (Bus Size: 16 Bits, 16-Bit Access)**




| 8-bit access | Odd address | Even address |
|---|---|---|
| AD15 to AD8 | Active | Undefined |
| AD7 to AD0 | Undefined | Active |
| WR1, WR0 | 01 | 10 |

**Figure 5-7. Multiplex Bus Write Timing (Bus Size: 8 Bits)**

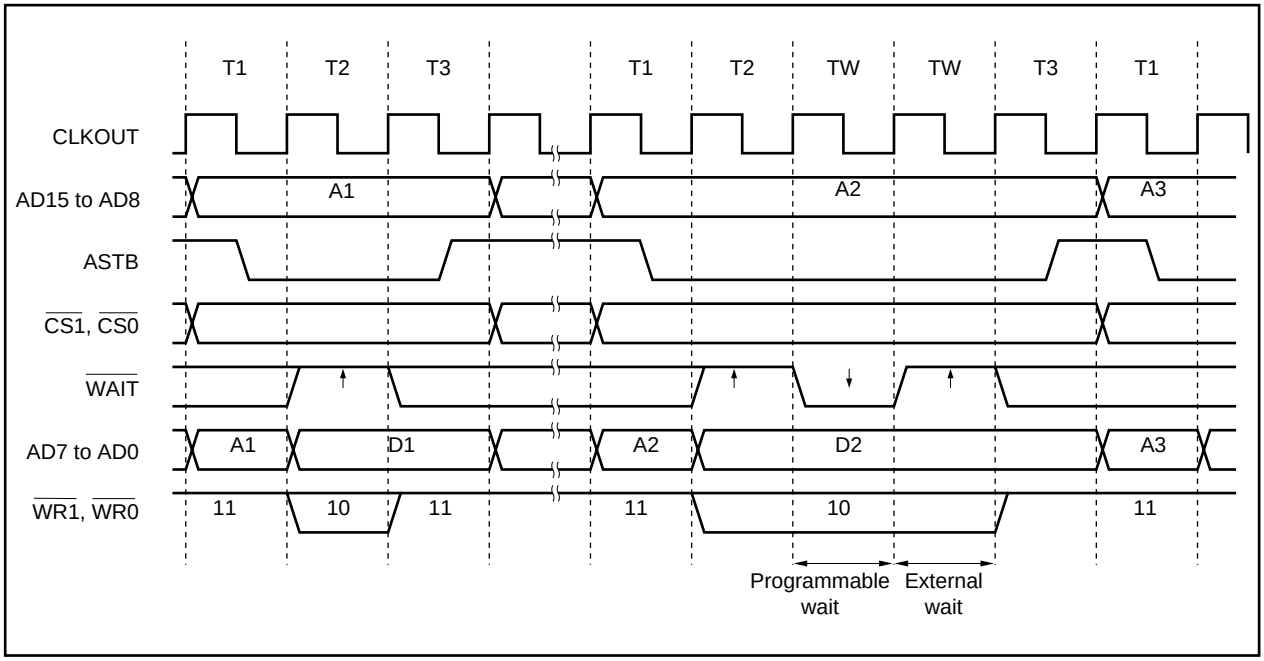
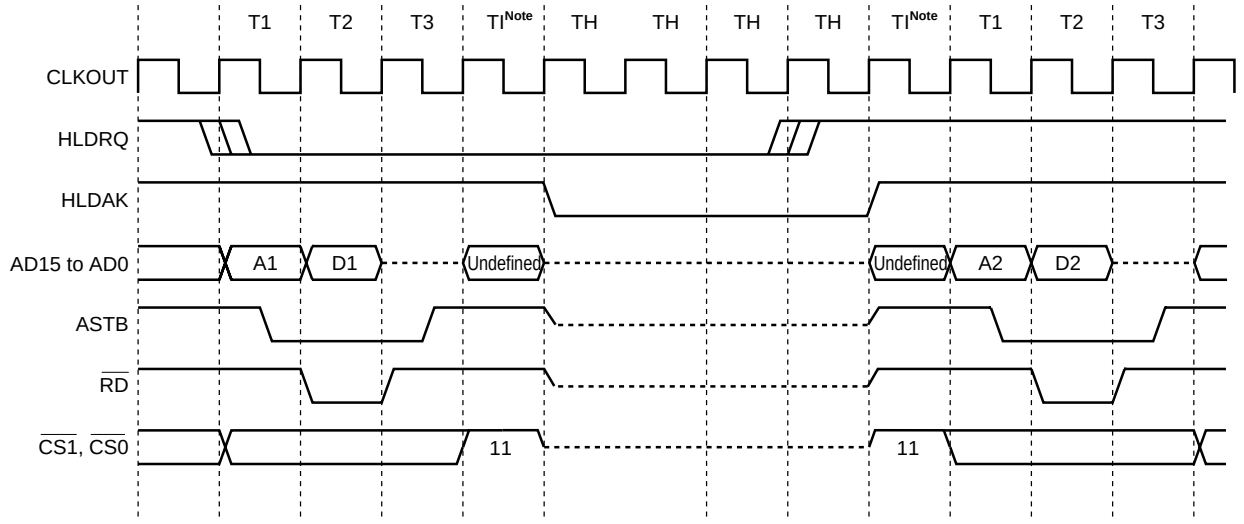


**Figure 5-8. Multiplex Bus Hold Timing (Bus Size: 16 Bits, 16-Bit Access)**

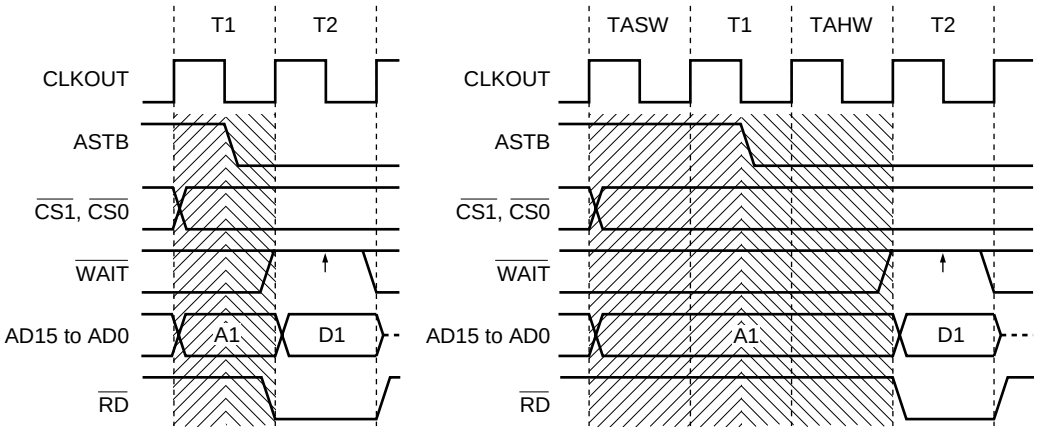



**Note** This idle state (TI) does not depend on the BCC register settings.

**Remarks 1.** Refer to **Table 2-2 Pin Operation Status in Operation Modes** for the pin statuses in the bus hold mode.

**2.** The broken lines indicate high impedance.

**Figure 5-9. Address Wait Timing (Bus Size: 16 Bits, 16-Bit Access)**




**Remarks 1.** TASW (address setup wait): Image of high-level width of T1 state expanded.

**2.** TAHW (address hold wait): Image of low-level width of T1 state expanded.

**3.** The broken lines indicate high impedance.

## 5.10 Cautions

With the external bus function, signals may not be output at the correct timing under the following conditions.

<Operating conditions>

- CLKOUT asynchronous (2.7 V ≤ $V_{DD}$ = $EV_{DD}$ = $AV_{REF0}$ ≤ 5.5 V)
  When 1/$f_{CPU}$ < 84 ns

<Countermeasure>

When used under the above conditions, be sure to insert an address setup/hold wait using the AWC register (n = 0, 1).

- 70 ns < 1/$f_{CPU}$ < 84 ns
  Set an address setup wait (ASWn bit = 1).
- 62.5 ns < 1/$f_{CPU}$ < 70 ns
  Set an address setup wait (ASWn bit = 1) and address hold wait (AHWn bit = 1).

# CHAPTER 6 CLOCK GENERATION FUNCTION

## 6.1 Overview

The following clock generation functions are available.

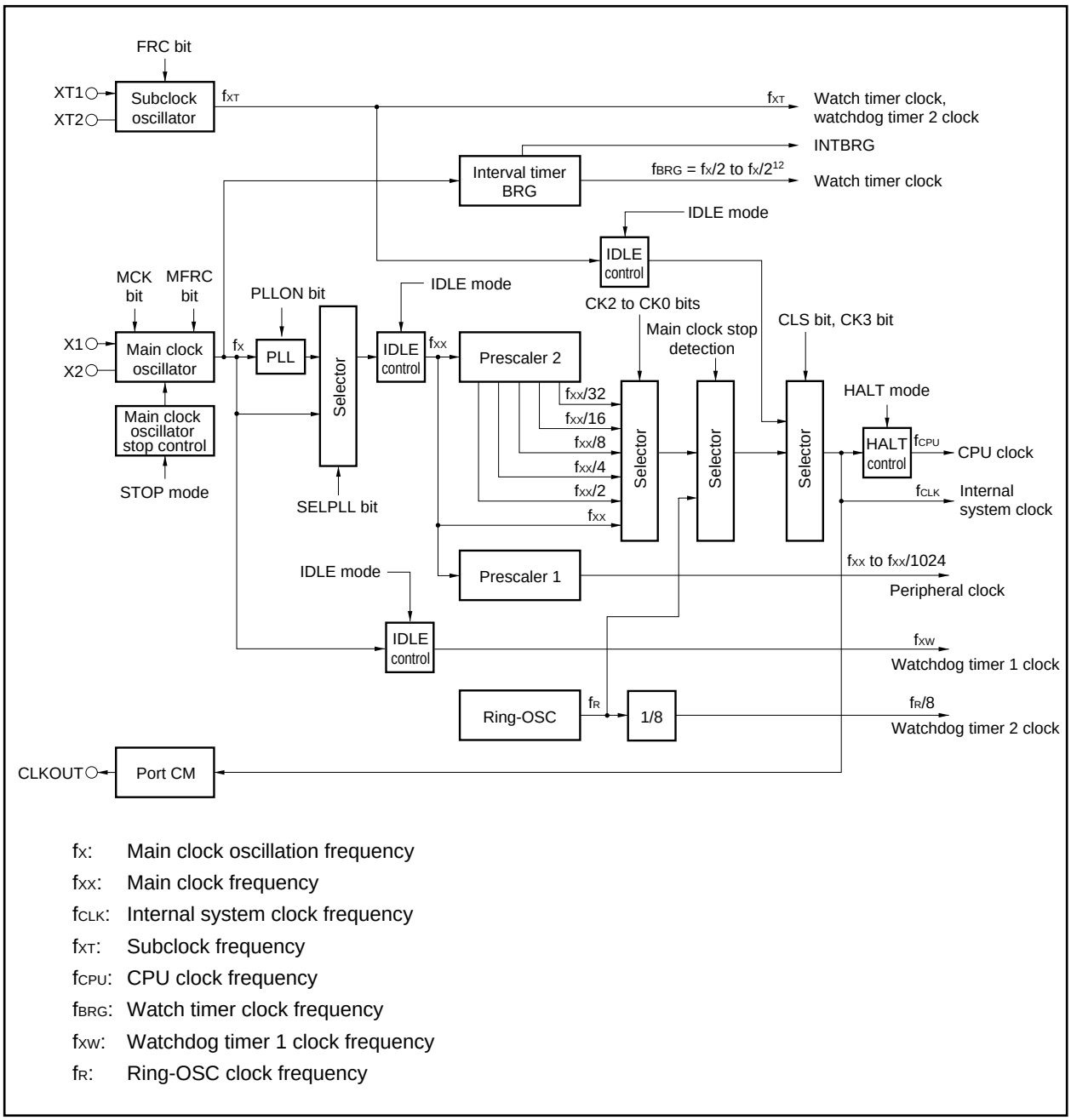
- Main clock oscillator
  - $f_X$ = 2 MHz ($f_{XX}$ = 8 MHz, REGC = $V_{DD}$ = 2.7 to 5.5 V, in PLL mode)
  - $f_X$ = 2 to 5 MHz ($f_{XX}$ = 8 to 20 MHz, REGC = $V_{DD}$ = 4.5 to 5.5 V, in PLL mode)
  - $f_X$ = 2 MHz ($f_{XX}$ = 8 MHz, REGC = capacitor, $V_{DD}$ = 4.0 to 5.5 V, in PLL mode)
  - $f_X$ = 2 to 8^Note MHz ($f_{XX}$ = 2 to 8^Note MHz, REGC = $V_{DD}$ = 2.7 to 5.5 V, in clock-through mode)
- Subclock oscillator
  - 32.768 kHz
- On-chip ring oscillator (Ring-OSC)
  - $f_R$ = 120 to 480 kHz (240 kHz (TYP.))
- Multiplication (×4) function by PLL (Phase Locked Loop)
  - Clock-through mode/PLL mode selectable
  - Usable voltage: $V_{DD}$ = 2.7 to 5.5 V
- Internal system clock generation
  - 7 steps ($f_{XX}$, $f_{XX}$/2, $f_{XX}$/4, $f_{XX}$/8, $f_{XX}$/16, $f_{XX}$/32, $f_{XT}$)
  - Operates at $f_R$ after the reset signal for the clock monitor is generated upon detection of main clock stop.
- Peripheral clock generation
- Clock output function

**Note** This value may change after evaluation.

**Remark** $f_X$: Main clock oscillation frequency
$f_{XX}$: Main clock frequency
$f_R$: Ring-OSC clock frequency

## 6.2 Configuration

**Figure 6-1. Clock Generator**




fX: Main clock oscillation frequency
fXX: Main clock frequency
fCLK: Internal system clock frequency
fXT: Subclock frequency
fCPU: CPU clock frequency
fBRG: Watch timer clock frequency
fXW: Watchdog timer 1 clock frequency
fR: Ring-OSC clock frequency

**(1) Main clock oscillator**

The main clock oscillator oscillates the following frequencies ($f_X$).

- $f_X$ = 2 MHz (REGC = $V_{DD}$ = 2.7 to 5.5 V, in PLL mode)
- $f_X$ = 2 to 5 MHz (REGC = $V_{DD}$ = 4.5 to 5.5 V, in PLL mode)
- $f_X$ = 2 to 8[Note] MHz (REGC = $V_{DD}$ = 2.7 to 5.5 V, in clock-through mode)

**Note** This value may change after evaluation.

**(2) Subclock oscillator**

The subclock oscillator oscillates a frequency of 32.768 kHz ($f_{XT}$).

**(3) Main clock oscillator stop control**

This circuit generates a control signal that stops oscillation of the main clock oscillator.

Oscillation of the main clock oscillator is stopped in the STOP mode or when the PCC.MCK bit = 1 (valid only when the PCC.CLS bit = 1).

**(4) Prescaler 1**

This prescaler generates the clock ($f_{XX}$ to $f_{XX}$/1024) to be supplied to the following on-chip peripheral functions: TMP0, TM00, TM01, TM50, TM51, TMH0, TMH1, CSI00, CSI01, CSIA0, UART0, UART1, $I^2C0$, and ADC

**(5) Prescaler 2**

This circuit divides the main clock ($f_{XX}$).

The clock generated by prescaler 2 ($f_{XX}$ to $f_{XX}$/32) is supplied to the selector that generates the CPU clock ($f_{CPU}$) and internal system clock ($f_{CLK}$).

$f_{CLK}$ is the clock supplied to the INTC, ROM correction, ROM, and RAM blocks, and can be output from the CLKOUT pin.

**(6) Interval timer BRG**

This circuit divides the clock ($f_X$) generated by the main clock oscillator to a specific frequency (32.768 kHz) and supplies that clock to the watch timer block. It can also be used as an interval timer.

For details, refer to **CHAPTER 11 INTERVAL TIMER, WATCH TIMER**.

**(7) PLL**

This circuit multiplies the clock ($f_X$) generated by the main clock oscillator.

It operates in two modes: clock-through mode in which $f_X$ is output as is, and PLL mode in which a multiplied clock is output. These modes can be selected by using the PLLCTL.SELPLL bit.

Operation of the PLL can be started or stopped by the PLLCTL.PLLON bit.

**(8) Ring-OSC (on-chip ring oscillator)**

The Ring-OSC oscillator oscillates a frequency ($f_R$) of 120 to 480 kHz (240 kHz (TYP.)).

## 6.3 Registers

### (1) Processor clock control register (PCC)

The PCC register is a special register. Data can be written to this register only in combination of specific sequences (refer to **3.4.7 Special registers**).

This register can be read or written in 8-bit or 1-bit units.

After reset, PCC is set to 03H.

(1/2)

After reset: 03H R/W Address: FFFFF828H

| | 7 | <6> | 5 | <4> | <3> | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PCC | FRC | MCK | MFRC | CLS[Note] | CK3 | CK2 | CK1 | CK0 |

| FRC | Use of subclock on-chip feedback resistor |
|---|---|
| 0 | Used |
| 1 | Not used |

| MCK | Control of main clock oscillator |
|---|---|
| 0 | Oscillation enabled |
| 1 | Oscillation stopped |

- Even if the MCK bit is set to 1 while the system is operating with the main clock as the CPU clock, the operation of the main clock does not stop. It stops after the CPU clock has been changed to the subclock.
- When the main clock is stopped and the device is operating on the subclock, clear the MCK bit to 0 and wait until the oscillation stabilization time has been secured by the program before switching back to the main clock.

| MFRC | Use of main clock on-chip feedback resistor |
|---|---|
| 0 | Used |
| 1 | Not used |

| CLS[Note] | Status of CPU clock ($f_{CPU}$) |
|---|---|
| 0 | Main clock operation |
| 1 | Subclock operation |

**Note** The CLS bit is a read-only bit.

(2/2)

| CK3 | CK2 | CK1 | CK0 | Clock selection ($f_{CLK}/f_{CPU}$) |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | $f_{XX}$ |
| 0 | 0 | 0 | 1 | $f_{XX}/2$ |
| 0 | 0 | 1 | 0 | $f_{XX}/4$ |
| 0 | 0 | 1 | 1 | $f_{XX}/8$ (default value) |
| 0 | 1 | 0 | 0 | $f_{XX}/16$ |
| 0 | 1 | 0 | 1 | $f_{XX}/32$ |
| 0 | 1 | 1 | × | Setting prohibited |
| 1 | × | × | × | $f_{XT}$ |

**Cautions 1. Do not change the CPU clock (by using the CK3 to CK0 bits) while CLKOUT is being output.**

**2. Use a bit manipulation instruction to manipulate the CK3 bit. When using an 8-bit manipulation instruction, do not change the set values of the CK2 to CK0 bits.**

**3. When the CPU operates on the subclock and no clock is input to the X1 pin, do not access a register in which a wait occurs using an access method that causes a wait (refer to 3.4.8 (2) Access to special on-chip peripheral I/O register for details of the access methods). If a wait occurs, it can only be released by a reset.**

**Remark** ×: don't care

**(a) Example of setting main clock operation → subclock operation**

<1> CK3 bit ← 1: Use of a bit manipulation instruction is recommended. Do not change the CK2 to CK0 bits.

<2> Subclock operation: Read the CLS bit to check if subclock operation has started. It takes the following time after the CK3 bit is set until subclock operation is started.

Max.: $1/f_{XT}$ (1/subclock frequency)

<3> MCK bit ← 1: Set the MCK bit to 1 only when stopping the main clock.

**Cautions 1. When stopping the main clock, stop the PLL.**

**2. If the following conditions are not satisfied, change the CK2 to CK0 bits so that the conditions are satisfied, then change to the subclock operation mode.**

**Main clock ($f_{XX}$) > Subclock ($f_{XT}$: 32.768 kHz) × 4**

[Description example]

```
<1>  _SET_SUB_RUN :
     st.b       r0, PRCMD[r0]
     set1       3, PCC[r0]          -- CK3 bit ← 1
<2>  _CHECK_CLS :
     tst1       4, PCC[r0]          -- Wait until subclock operation starts.
     bz         _CHECK_CLS
<3>  _STOP_MAIN_CLOCK :
     st.b       r0, PRCMD[r0]
     set1       6, PCC[r0]          -- MCK bit ← 1, main clock is stopped
```

**Remark** The above description is an example. Note with caution that the CLS bit is read in a closed loop in <2>.

**(b) Example of setting subclock operation → main clock operation**

<1> MCK bit ← 0: Main clock starts oscillating

<2> Insert waits by the program and wait until the oscillation stabilization time of the main clock elapses.

<3> CK3 bit ← 0: Use of a bit manipulation instruction is recommended. Do not change the CK2 to CK0 bits.

<4> Main clock operation: It takes the following time after the CK3 bit is set until main clock operation is started.

Max.: $1/f_{XT}$ (1/subclock frequency)

Therefore, insert one NOP instruction immediately after setting the CK3 bit to 0 or read the CLS bit to check if main clock operation has started.

[Description example]

```
<1> _START_MAIN_OSC :
    st.b        r0, PRCMD[r0]       -- Release of protection of special registers
    clr1        6, PCC[r0]          -- Main clock starts oscillating
<2> movea       0x55, r0, r11       -- Wait for oscillation stabilization time
    _WAIT_OST :
    nop
    nop
    nop
    addi        -1, r11, r11
    mp          r0, r11
    bne         _PROGRAM_WAIT
<3> st.b        r0, PRCMD[r0]
    clr1        3, PCC[r0]          -- CK3 ← 0
<4> _CHECK_CLS :
    tst1        4, PCC[r0]          -- Wait until main clock operation starts
    bnz         _CHECK_CLS
```

**Remark** The above description is an example. Note with caution that the CLS bit is read in a closed loop in <4>.

**(2) Ring-OSC mode register (RCM)**

The RCM register is an 8-bit register that sets the operation mode of the Ring-OSC oscillator.

This register can be read or written in 8-bit or 1-bit units.

After reset, RCM is cleared to 00H.

**Caution The settings of the RCM register differ for a mask ROM version and flash memory version. Refer to CHAPTER 28 MASK OPTION/OPTION BYTE for details.**

- **Mask ROM version (μPD703308, 703308Y)**
  **Valid when "(Ring-OSC) Can be stopped by software" is selected by the mask option.**
- **Flash memory version (μPD70F3306, 70F3306Y, 70F3308, 70F3308Y)**
  **Valid when RINGSTP is cleared to 0 by the option byte setting.**

After reset: 00H R/W Address: FFFFF80CH

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | <0> |
|---|---|---|---|---|---|---|---|---|
| RCM | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RSTOP |

| RSTOP | Enables/disables Ring-OSC oscillation |
|---|---|
| 0 | Ring-OSC oscillation enabled. |
| 1 | Ring-OSC oscillation disabled (stopped). |

**(3) CPU operation clock status register (CCLS)**

The CCLS register indicates the CPU operation clock status.

This register is read-only, in 8-bit or 1-bit units.

After reset, CCLS is cleared to 00H.

After reset: 00H R Address: FFFFF82EH

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CCLS | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CCLSF |

| CCLSF | CPU operation clock status |
|---|---|
| 0 | Operates on main clock ($f_X$) or subclock ($f_{XT}$). |
| 1 | Operates on Ring-OSC ($f_R$). |

## 6.4 Operation

### 6.4.1 Operation of each clock

The following table shows the operation status of each clock.

**Table 6-1. Operation Status of Each Clock**

| Register Setting and Operation Status / Target Clock | PCC Register | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | CLS bit = 0, MCK bit = 0 | | | | | CLS bit = 1, MCK bit = 0 | | CLS bit = 1, MCK bit = 1 | |
| | During reset | During oscillation stabilization time count | HALT mode | IDLE mode | STOP mode | Subclock mode | Sub-IDLE mode | Subclock mode | Sub-IDLE mode |
| Main clock oscillator ($f_X$) | × | ○ | ○ | ○ | × | ○ | ○ | × | × |
| Subclock oscillator ($f_{XT}$) | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| CPU clock ($f_{CPU}$) | × | × | × | × | × | ○ | × | ○ | × |
| Internal system clock ($f_{CLK}$) | × | × | ○ | × | × | ○ | × | ○ | × |
| Peripheral clock ($f_{XX}$ to $f_{XX}/1024$) | × | × | ○ | × | × | ○ | × | × | × |
| WT clock (main) | × | ○ | ○ | ○ | × | ○ | ○ | × | × |
| WT clock (sub) | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| WDT1 clock ($f_{XW}$) | × | ○ | ○ | ○ | × | ○ | ○ | × | × |
| WDT2 clock (Ring-OSC) | × | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |
| WDT2 clock (sub) | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ | ○ |

**Remark** ○: Operable
×: Stopped

### 6.4.2 Clock output function

The clock output function is used to output the internal system clock ($f_{CLK}$) from the CLKOUT pin.

The internal system clock ($f_{CLK}$) is selected by using the PCC.CK3 to PCC.CK0 bits.

The CLKOUT pin functions alternately as the PCM1 pin and functions as a clock output pin if so specified by the control register of port CM.

The status of the CLKOUT pin is the same as the internal system clock in Table 6-1 and the pin can output the clock when it is in the operable status. It outputs a low level in the stopped status. However, the port mode (PCM1: input mode) is selected until the CLKOUT pin output is set after reset. Consequently, the CLKOUT pin goes into a high-impedance state.

### 6.4.3 External clock input function

An external clock can be directly input to the oscillator. Input the clock to the X1 pin and its inverse signal to the X2 pin. Set the PCC.MFRC bit to 1 (on-chip feedback resistor not used). Note, however, that oscillation stabilization time is inserted even in the external clock mode. Connect $V_{DD}$ directly to the REGC pin.

## 6.5 PLL Function

### 6.5.1 Overview

The PLL function is used to output the operating clock of the CPU and peripheral macro at a frequency 4 times higher than the oscillation frequency, and select the clock-through mode.

When PLL function is used: Input clock = 2 to 5 MHz ($f_{XX}$: 8 to 20 MHz)
Clock-through mode: Input clock = 2 to 10 MHz ($f_{XX}$: 2 to 10 MHz)

### 6.5.2 Register

**(1) PLL control register (PLLCTL)**

The PLLCTL register is an 8-bit register that controls the security function of PLL and RTO.
This register can be read or written in 8-bit or 1-bit units.
After reset, PLLCTL is set to 01H.

After reset: 01H R/W Address: FFFFF806H

| | 7 | 6 | 5 | 4 | 3 | <2> | <1> | <0> |
|---|---|---|---|---|---|---|---|---|
| PLLCTL | 0 | 0 | 0 | 0 | 0 | RTOST0[Note] | SELPLL | PLLON |

| PLLON | PLL operation stop register |
|---|---|
| 0 | PLL stopped |
| 1 | PLL operating |

| SELPLL | PLL clock selection register |
|---|---|
| 0 | Clock-through operation |
| 1 | PLL operation |

**Note** For the RTOST0 bit, refer to **CHAPTER 13 REAL-TIME OUTPUT FUNCTION (RTO)**.

**Caution Be sure to clear bits 4 to 7 to 0. Changing bit 3 does not affect the operation.**

### 6.5.3 Usage

**(1) When PLL is used**

- After reset has been released, the PLL operates (PLLCTL.PLLON bit = 1), but because the default mode is the clock-through mode (PLLCTL.SELPLL bit = 0), select the PLL mode (SELPLL bit = 1).
- To set the STOP mode in which the main clock is stopped, or to set the IDLE mode, first select the clock-through mode and then stop the PLL. To return from the IDLE or STOP mode, first enable PLL operation (PLLON bit = 1), and then select the PLL mode (SELPLL bit = 1).
- To enable the PLL operation, first set the PLLON bit to 1, wait for 200 $\mu$s, and then set the SELPLL bit to 1. To stop the PLL, first select the clock-through mode (SELPLL bit = 0), wait for 8 clocks or more, and then stop the PLL (PLLON bit = 0).

**(2) When PLL is not used**

- The clock-through mode (SELPLL bit = 0) is selected after reset has been released, but the PLL is operating (PLLON bit = 1) and must therefore be stopped (PLLON bit = 0).

**Remark** The PLL is operable in the IDLE mode. To realize low power consumption, stop the PLL. Be sure to stop the PLL when shifting to the STOP mode.

# CHAPTER 7 16-BIT TIMER/EVENT COUNTER P (TMP)

Timer P (TMP) is a 16-bit timer/event counter.
The V850ES/KF1+ incorporates TMP0.

## 7.1 Overview

An outline of TMP0 is shown below.

- Clock selection: 8 ways
- Capture trigger input pins: 2
- External event count input pins: 1
- External trigger input pins: 1
- Timer/counters: 1
- Capture/compare registers: 2
- Capture/compare match interrupt request signals: 2
- Timer output pins: 2

## 7.2 Functions

TMP0 has the following functions.

- Interval timer
- External event counter
- External trigger pulse output
- One-shot pulse output
- PWM output
- Free-running timer
- Pulse width measurement

## 7.3 Configuration

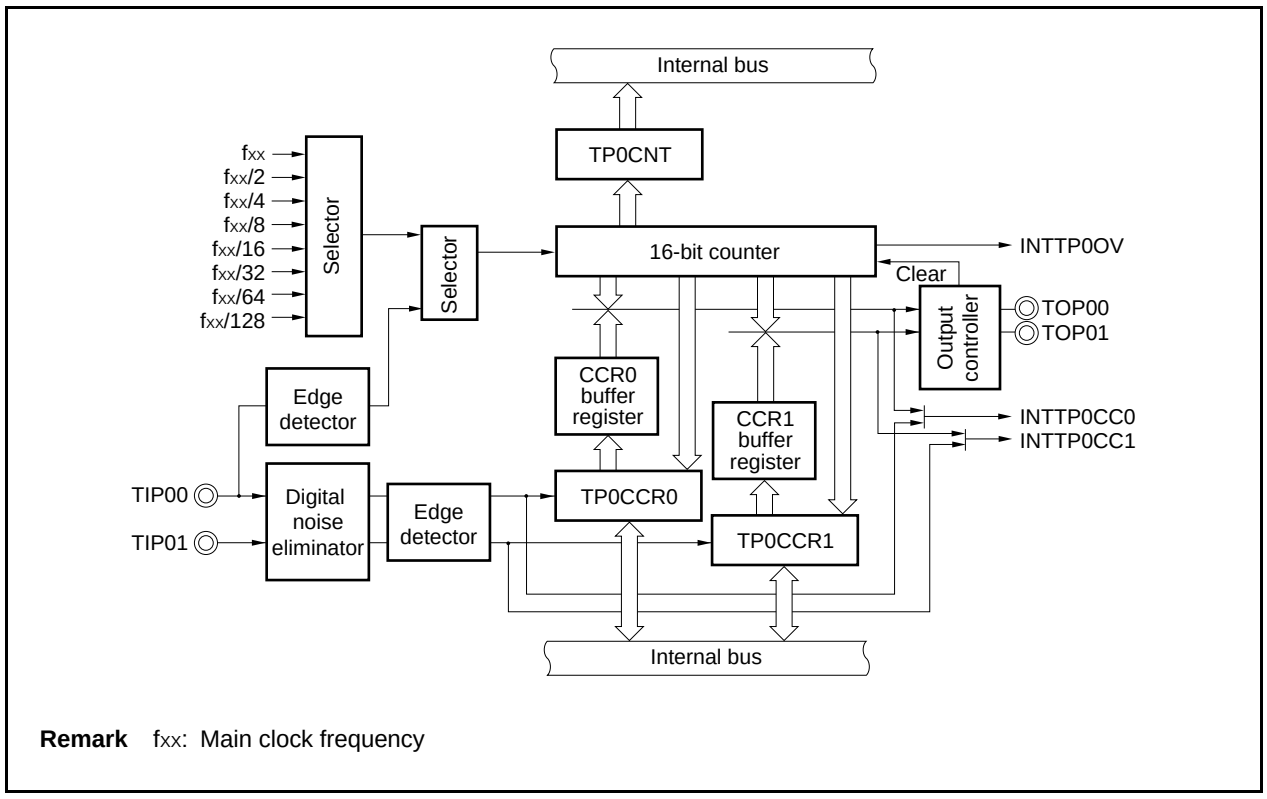
TMP0 includes the following hardware.

**Table 7-1. Configuration of TMP0**

| Item | Configuration |
|---|---|
| Timer register | 16-bit counter |
| Registers | TMP0 capture/compare registers 0, 1 (TP0CCR0, TP0CCR1)<br>TMP0 counter read buffer register (TP0CNT)<br>CCR0, CCR1 buffer registers |
| Timer inputs | 2 (TIP00[Note], TIP01 pins) |
| Timer outputs | 2 (TOP00, TOP01 pins) |
| Control registers | TMP0 control registers 0, 1 (TP0CTL0, TP0CTL1)<br>TMP0 I/O control registers 0 to 2 (TP0IOC0 to TP0IOC2)<br>TMP0 option registers 0, 1 (TP0OPT0, TP0OPT1) |

**Note** The TIP00 pin functions alternately as a capture trigger input signal, external event count input signal, and external trigger input signal.

**Figure 7-1. Block Diagram of TMP0**




**Remark** $f_{XX}$: Main clock frequency

**(1) 16-bit counter**

This 16-bit counter can count internal clocks or external events.

The count value of this counter can be read by using the TP0CNT register.

When the TP0CTL0.TP0CE bit = 0, the value of the 16-bit counter is FFFFH. If the TP0CNT register is read at this time, 0000H is read.

Reset input clears the TP0CE bit to 0. Therefore, the 16-bit counter is set to FFFFH.

**(2) CCR0 buffer register**

This is a 16-bit compare register that compares the count value of the 16-bit counter.

When the TP0CCR0 register is used as a compare register, the value written to the TP0CCR0 register is transferred to the CCR0 buffer register. When the count value of the 16-bit counter matches the value of the CCR0 buffer register, a compare match interrupt request signal (INTTP0CC0) is generated.

The CCR0 buffer register cannot be read or written directly.

The CCR0 buffer register is cleared to 0000H after reset, as the TP0CCR0 register is cleared to 0000H.

**(3) CCR1 buffer register**

This is a 16-bit compare register that compares the count value of the 16-bit counter.

When the TP0CCR1 register is used as a compare register, the value written to the TP0CCR1 register is transferred to the CCR1 buffer register. When the count value of the 16-bit counter matches the value of the CCR1 buffer register, a compare match interrupt request signal (INTTP0CC1) is generated.

The CCR1 buffer register cannot be read or written directly.

The CCR1 buffer register is cleared to 0000H after reset, as the TP0CCR1 register is cleared to 0000H.

**(4) Edge detector**

This circuit detects the valid edges input to the TIP00 and TIP01 pins. No edge, rising edge, falling edge, or both the rising and falling edges can be selected as the valid edge by using the TP0IOC1 and TP0IOC2 registers.

**(5) Output controller**

This circuit controls the output of the TOP00 and TOP01 pins. The output controller is controlled by the TP0IOC0 register.

**(6) Selector**

This selector selects the count clock for the 16-bit counter. Eight types of internal clocks or an external event can be selected as the count clock.

**(7) Digital noise eliminator**

This circuit is valid only when the TIP00 and TIP01 pins are used as a capture trigger input pin.

This circuit is controlled by the P0NFC and P1NFC registers.

## 7.4 Registers

### (1) TMP0 control register 0 (TP0CTL0)

The TP0CTL0 register is an 8-bit register that controls the operation of TMP0.

This register can be read or written in 8-bit or 1-bit units.

Reset input clears this register to 00H.

The same value can always be written to the TP0CTL0 register by software.

After reset: 00H R/W Address: FFFFF5A0H

| | <7> | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| TP0CTL0 | TP0CE | 0 | 0 | 0 | 0 | TP0CKS2 | TP0CKS1 | TP0CKS0 |

| TP0CE | TMP0 operation control |
|---|---|
| 0 | TMP0 operation disabled (TMP0 reset asynchronously[Note]). |
| 1 | TMP0 operation enabled. TMP0 operation started. |

| TP0CKS2 | TP0CKS1 | TP0CKS0 | Internal count 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 | 0 | 0 | $f_{XX}/16$ |
| 1 | 0 | 1 | $f_{XX}/32$ |
| 1 | 1 | 0 | $f_{XX}/64$ |
| 1 | 1 | 1 | $f_{XX}/128$ |

**Note** TP0OPT0.TP0OVF bit, 16-bit counter, timer output (TOP00, TOP01 pins)

**Cautions 1. Set the TP0CKS2 to TP0CKS0 bits when the TP0CE bit = 0. When the value of the TP0CE bit is changed from 0 to 1, the TP0CKS2 to TP0CKS0 bits can be set simultaneously.**

**2. Be sure to clear bits 3 to 6 to 0.**

**Remark** $f_{XX}$: Main clock frequency

**(2) TMP0 control register 1 (TP0CTL1)**

The TP0CTL1 register is an 8-bit register that controls the operation of TMP0.

This register can be read or written in 8-bit or 1-bit units.

Reset input clears this register to 00H.

After reset: 00H R/W Address: FFFFF5A1H

| | 7 | <6> | <5> | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| TP0CTL1 | 0 | TP0EST | TP0EEE | 0 | 0 | TP0MD2 | TP0MD1 | TP0MD0 |

| TP0EST | Software trigger control |
|---|---|
| 0 | – |
| 1 | Generate a valid signal for external trigger input.<br>• In one-shot pulse output mode: A one-shot pulse is output with writing 1 to the TP0EST bit as the trigger.<br>• In external trigger pulse output mode: A PWM waveform is output with writing 1 to the TP0EST bit as the trigger. |

| TP0EEE | Count clock selection |
|---|---|
| 0 | Disable operation with external event count input.<br>(Perform counting with the count clock selected by the TP0CTL0.TP0CK0 to TP0CTL0.TP0CK2 bits.) |
| 1 | Enable operation with external event count input.<br>(Perform counting at the valid edge of the external event count input signal.) |
| The TP0EEE bit selects whether counting is performed with the internal count clock or the valid edge of the external event count input. | |

| TP0MD2 | TP0MD1 | TP0MD0 | Timer mode selection |
|---|---|---|---|
| 0 | 0 | 0 | Interval timer mode |
| 0 | 0 | 1 | External event count mode |
| 0 | 1 | 0 | External trigger pulse output mode |
| 0 | 1 | 1 | One-shot pulse output mode |
| 1 | 0 | 0 | PWM output mode |
| 1 | 0 | 1 | Free-running timer mode |
| 1 | 1 | 0 | Pulse width measurement mode |
| 1 | 1 | 1 | Setting prohibited |

**Cautions 1. The TP0EST bit is valid only in the external trigger pulse output mode or one-shot pulse output mode. In any other mode, writing 1 to this bit is ignored.**

**2. External event count input is selected in the external event count mode regardless of the value of the TP0EEE bit.**

**3. Set the TP0EEE and TP0MD2 to TP0MD0 bits when the TP0CTL0.TP0CE bit = 0. (The same value can be written when the TP0CE bit = 1.) The operation is not guaranteed when rewriting is performed with the TP0CE bit = 1. If rewriting was mistakenly performed, clear the TP0CE bit to 0 and then set the bits again.**

**4. Be sure to clear bits 3, 4, and 7 to 0.**

**(3) TMP0 I/O control register 0 (TP0IOC0)**

The TP0IOC0 register is an 8-bit register that controls the timer output (TOP00, TOP01 pins).

This register can be read or written in 8-bit or 1-bit units.

Reset input clears this register to 00H.

After reset: 00H R/W Address: FFFFF5A2H

| | 7 | 6 | 5 | 4 | 3 | <2> | 1 | <0> |
|---|---|---|---|---|---|---|---|---|
| TP0IOC0 | 0 | 0 | 0 | 0 | TP0OL1 | TP0OE1 | TP0OL0 | TP0OE0 |

| TP0OL1 | TOP01 pin output level setting |
|---|---|
| 0 | TOP01 pin output inversion disabled |
| 1 | TOP01 pin output inversion enabled |

| TP0OE1 | TOP01 pin output setting |
|---|---|
| 0 | Timer output disabled<br>• When TP0OL1 bit = 0: Low level is output from the TOP01 pin<br>• When TP0OL1 bit = 1: High level is output from the TOP01 pin |
| 1 | Timer output enabled (a square wave is output from the TOP01 pin). |

| TP0OL0 | TOP00 pin output level setting |
|---|---|
| 0 | TOP00 pin output inversion disabled |
| 1 | TOP00 pin output inversion enabled |

| TP0OE0 | TOP00 pin output setting |
|---|---|
| 0 | Timer output disabled<br>• When TP0OL0 bit = 0: Low level is output from the TOP00 pin<br>• When TP0OL0 bit = 1: High level is output from the TOP00 pin |
| 1 | Timer output enabled (a square wave is output from the TOP00 pin). |

**Cautions 1. Rewrite the TP0OL1, TP0OE1, TP0OL0, and TP0OE0 bits when the TP0CTL0.TP0CE bit = 0. (The same value can be written when the TP0CE bit = 1.) If rewriting was mistakenly performed, clear the TP0CE bit to 0 and then set the bits again.**

**2. Even if the TP0OLa bit is manipulated when the TP0CE and TP0OEa bits are 0, the TOP0a pin output level varies (a = 0, 1).**

**(4) TMP0 I/O control register 1 (TP0IOC1)**

The TP0IOC1 register is an 8-bit register that controls the valid edge of the capture trigger input signals (TIP00, TIP01 pins).

This register can be read or written in 8-bit units.

Reset input clears this register to 00H.

After reset: 00H R/W Address: FFFFF5A3H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| TP0IOC1 | 0 | 0 | 0 | 0 | TP0IS3 | TP0IS2 | TP0IS1 | TP0IS0 |

| TP0IS3 | TP0IS2 | Capture trigger input signal (TIP01 pin) valid edge setting |
|---|---|---|
| 0 | 0 | No edge detection (capture operation invalid) |
| 0 | 1 | Detection of rising edge |
| 1 | 0 | Detection of falling edge |
| 1 | 1 | Detection of both edges |

| TP0IS1 | TP0IS0 | Capture trigger input signal (TIP00 pin) valid edge setting |
|---|---|---|
| 0 | 0 | No edge detection (capture operation invalid) |
| 0 | 1 | Detection of rising edge |
| 1 | 0 | Detection of falling edge |
| 1 | 1 | Detection of both edges |

**Cautions 1. Rewrite the TP0IS3 to TP0IS0 bits when the TP0CTL0.TP0CE bit = 0. (The same value can be written when the TP0CE bit = 1.) If rewriting was mistakenly performed, clear the TP0CE bit to 0 and then set the bits again.**

**2. The TP0IS3 to TP0IS0 bits are valid only in the free-running timer mode and the pulse width measurement mode. In all other modes, a capture operation is not possible.**

**(5) TMP0 I/O control register 2 (TP0IOC2)**

The TP0IOC2 register is an 8-bit register that controls the valid edge of the external event count input signal (TIP00 pin) and external trigger input signal (TIP00 pin).

This register can be read or written in 8-bit or 1-bit units.

Reset input clears this register to 00H.

After reset: 00H R/W Address: FFFFF5A4H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| TP0IOC2 | 0 | 0 | 0 | 0 | TP0EES1 | TP0EES0 | TP0ETS1 | TP0ETS0 |

| TP0EES1 | TP0EES0 | External event count input signal (TIP00 pin) valid edge setting |
|---|---|---|
| 0 | 0 | No edge detection (external event count invalid) |
| 0 | 1 | Detection of rising edge |
| 1 | 0 | Detection of falling edge |
| 1 | 1 | Detection of both edges |

| TP0ETS1 | TP0ETS0 | External trigger input signal (TIP00 pin) valid edge setting |
|---|---|---|
| 0 | 0 | No edge detection (external trigger invalid) |
| 0 | 1 | Detection of rising edge |
| 1 | 0 | Detection of falling edge |
| 1 | 1 | Detection of both edges |

**Cautions 1. Rewrite the TP0EES1, TP0EES0, TP0ETS1, and TP0ETS0 bits when the TP0CTL0.TP0CE bit = 0. (The same value can be written when the TP0CE bit = 1.) If rewriting was mistakenly performed, clear the TP0CE bit to 0 and then set the bits again.**

**2. The TP0EES1 and TP0EES0 bits are valid only when the TP0CTL1.TP0EEE bit = 1 or when the external event count mode (TP0CTL1.TP0MD2 to TP0CTL1.TP0MD0 bits = 001) has been set.**

**(6) TMP0 option register 0 (TP0OPT0)**

The TP0OPT0 register is an 8-bit register used to set the capture/compare operation and detect an overflow.

This register can be read or written in 8-bit or 1-bit units.

Reset input clears this register to 00H.

After reset: 00H R/W Address: FFFFF5A5H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | <0> |
|---|---|---|---|---|---|---|---|---|
| TP0OPT0 | 0 | 0 | TP0CCS1 | TP0CCS0 | 0 | 0 | 0 | TP0OVF |

| TP0CCS1 | TP0CCR1 register capture/compare selection |
|---|---|
| 0 | Compare register selected |
| 1 | Capture register selected |
| The TP0CCS1 bit setting is valid only in the free-running timer mode. | |

| TP0CCS0 | TP0CCR0 register capture/compare selection |
|---|---|
| 0 | Compare register selected |
| 1 | Capture register selected |
| The TP0CCS0 bit setting is valid only in the free-running timer mode. | |

| TP0OVF | TMP0 overflow detection flag |
|---|---|
| Set (1) | Overflow occurred |
| Reset (0) | TP0OVF bit 0 written or TP0CTL0.TP0CE bit = 0 |
| • The TP0OVF bit is reset when the 16-bit counter count value overflows from FFFFH to 0000H in the free-running timer mode or the pulse width measurement mode.<br>• An interrupt request signal (INTTP0OV) is generated at the same time that the TP0OVF bit is set to 1. The INTTP0OV signal is not generated in modes other than the free-running timer mode and the pulse width measurement mode.<br>• The TP0OVF bit is not cleared even when the TP0OVF bit or the TP0OPT0 register are read when the TP0OVF bit = 1.<br>• The TP0OVF bit can be both read and written, but the TP0OVF bit cannot be set to 1 by software. Writing 1 has no influence on the operation of TMP0. | |

**Cautions 1. Rewrite the TP0CCS1 and TP0CCS0 bits when the TP0CE bit = 0. (The same value can be written when the TP0CE bit = 1.) If rewriting was mistakenly performed, clear the TP0CE bit to 0 and then set the bits again.**

**2. Be sure to clear bits 1 to 3, 6, and 7 to 0.**

**(7) TMP0 capture/compare register 0 (TP0CCR0)**

The TP0CCR0 register can be used as a capture register or a compare register depending on the mode.

This register can be used as a capture register or a compare register only in the free-running timer mode, depending on the setting of the TP0OPT0.TP0CCS0 bit. In the pulse width measurement mode, the TP0CCR0 register can be used only as a capture register. In any other mode, this register can be used only as a compare register.

The TP0CCR0 register can be read or written during operation.

This register can be read or written in 16-bit units.

Reset input clears this register to 0000H.

**Caution Accessing the TP0CCR0 register is disabled during subclock operation with the main clock stopped. For details, refer to 3.4.8 (2).**

After reset: 0000H R/W Address: FFFFF5A6H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TP0CCR0 | | | | | | | | | | | | | | | | |

**(a) Function as compare register**

The TP0CCR0 register can be rewritten even when the TP0CTL0.TP0CE bit = 1.

The set value of the TP0CCR0 register is transferred to the CCR0 buffer register. When the value of the 16-bit counter matches the value of the CCR0 buffer register, a compare match interrupt request signal (INTTP0CC0) is generated. If TOP00 pin output is enabled at this time, the output of the TOP00 pin is inverted.

When the TP0CCR0 register is used as a cycle register in the interval timer mode, external event count mode, external trigger pulse output mode, one-shot pulse output mode, or PWM output mode, the value of the 16-bit counter is cleared (0000H) if its count value matches the value of the CCR0 buffer register.

**(b) Function as capture register**

When the TP0CCR0 register is used as a capture register in the free-running timer mode, the count value of the 16-bit counter is stored in the TP0CCR0 register if the valid edge of the capture trigger input pin (TIP00 pin) is detected. In the pulse width measurement mode, the count value of the 16-bit counter is stored in the TP0CCR0 register and the 16-bit counter is cleared (0000H) if the valid edge of the capture trigger input pin (TIP00 pin) is detected.

Even if the capture operation and reading the TP0CCR0 register conflict, the correct value of the TP0CCR0 register can be read.

The following table shows the functions of the capture/compare register in each mode, and how to write data to the compare register.

**Table 7-2. Function of Capture/Compare Register in Each Mode and How to Write Compare Register**

| Operation Mode | Capture/Compare Register | How to Write Compare Register |
|---|---|---|
| Interval timer | Compare register | Anytime write |
| External event counter | Compare register | Anytime write |
| External trigger pulse output | Compare register | Batch write |
| One-shot pulse output | Compare register | Anytime write |
| PWM output | Compare register | Batch write |
| Free-running timer | Capture/compare register | Anytime write |
| Pulse width measurement | Capture register | – |

**(8) TMP0 capture/compare register 1 (TP0CCR1)**

The TP0CCR1 register can be used as a capture register or a compare register depending on the mode.

This register can be used as a capture register or a compare register only in the free-running timer mode, depending on the setting of the TP0OPT0.TP0CCS1 bit. In the pulse width measurement mode, the TP0CCR1 register can be used only as a capture register. In any other mode, this register can be used only as a compare register.

The TP0CCR1 register can be read or written during operation.

This register can be read or written in 16-bit units.

Reset input clears this register to 0000H.

**Caution Accessing the TP0CCR1 register is disabled during subclock operation with the main clock stopped. For details, refer to 3.4.8 (2).**

After reset: 0000H R/W Address: FFFFF5A8H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TP0CCR1 | | | | | | | | | | | | | | | | |

**(a) Function as compare register**

The TP0CCR1 register can be rewritten even when the TP0CTL0.TP0CE bit = 1.

The set value of the TP0CCR1 register is transferred to the CCR1 buffer register. When the value of the 16-bit counter matches the value of the CCR1 buffer register, a compare match interrupt request signal (INTTP0CC1) is generated. If TOP01 pin output is enabled at this time, the output of the TOP01 pin is inverted.

**(b) Function as capture register**

When the TP0CCR1 register is used as a capture register in the free-running timer mode, the count value of the 16-bit counter is stored in the TP0CCR1 register if the valid edge of the capture trigger input pin (TIP01 pin) is detected. In the pulse width measurement mode, the count value of the 16-bit counter is stored in the TP0CCR1 register and the 16-bit counter is cleared (0000H) if the valid edge of the capture trigger input pin (TIP01 pin) is detected.

Even if the capture operation and reading the TP0CCR1 register conflict, the correct value of the TP0CCR1 register can be read.

The following table shows the functions of the capture/compare register in each mode, and how to write data to the compare register.

**Table 7-3. Function of Capture/Compare Register in Each Mode and How to Write Compare Register**

| Operation Mode | Capture/Compare Register | How to Write Compare Register |
|---|---|---|
| Interval timer | Compare register | Anytime write |
| External event counter | Compare register | Anytime write |
| External trigger pulse output | Compare register | Batch write |
| One-shot pulse output | Compare register | Anytime write |
| PWM output | Compare register | Batch write |
| Free-running timer | Capture/compare register | Anytime write |
| Pulse width measurement | Capture register | – |

**(9) TMP0 counter read buffer register (TP0CNT)**

The TP0CNT register is a read buffer register that can read the count value of the 16-bit counter.

If this register is read when the TP0CTL0.TP0CE bit = 1, the count value of the 16-bit timer can be read.

This register is read-only, in 16-bit units.

The value of the TP0CNT register is cleared to 0000H when the TP0CE bit = 0. If the TP0CNT register is read at this time, the value of the 16-bit counter (FFFFH) is not read, but 0000H is read.

The value of the TP0CNT register is cleared to 0000H after reset, as the TP0CE bit is cleared to 0.

**Caution Accessing the TP0CNT register is disabled during subclock operation with the main clock stopped. For details, refer to 3.4.8 (2).**

After reset: 0000H R Address: FFFFF5AAH

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TP0CNT | | | | | | | | | | | | | | | | |

## 7.5 Operation

TMP0 can perform the following operations.

| Operation | TP0CTL1.TP0EST Bit (Software Trigger Bit) | TIP00 Pin (External Trigger Input) | Capture/Compare Register Setting | Compare Register Write |
|---|---|---|---|---|
| Interval timer mode | Invalid | Invalid | Compare only | Anytime write |
| External event count mode[Note 1] | Invalid | Invalid | Compare only | Anytime write |
| External trigger pulse output mode[Note 2] | Valid | Valid | Compare only | Batch write |
| One-shot pulse output mode[Note 2] | Valid | Valid | Compare only | Anytime write |
| PWM output mode | Invalid | Invalid | Compare only | Batch write |
| Free-running timer mode | Invalid | Invalid | Switching enabled | Anytime write |
| Pulse width measurement mode[Note 2] | Invalid | Invalid | Capture only | Not applicable |

**Notes 1.** To use the external event count mode, specify that the valid edge of the TIP00 pin capture trigger input is not detected (by clearing the TP0IOC1.TP0IS1 and TP0IOC1.TP0IS0 bits to “00”).

**2.** When using the external trigger pulse output mode, one-shot pulse output mode, and pulse width measurement mode, select the internal clock as the count clock (by clearing the TP0CTL1.TP0EEE bit to 0).

### 7.5.1 Interval timer mode (TP0MD2 to TP0MD0 bits = 000)

In the interval timer mode, an interrupt request signal (INTTP0CC0) is generated at the specified interval if the TP0CTL0.TP0CE bit is set to 1. A square wave whose half cycle is equal to the interval can be output from the TOP00 pin.

Usually, the TP0CCR1 register is not used in the interval timer mode.

**Figure 7-2. Configuration of Interval Timer**

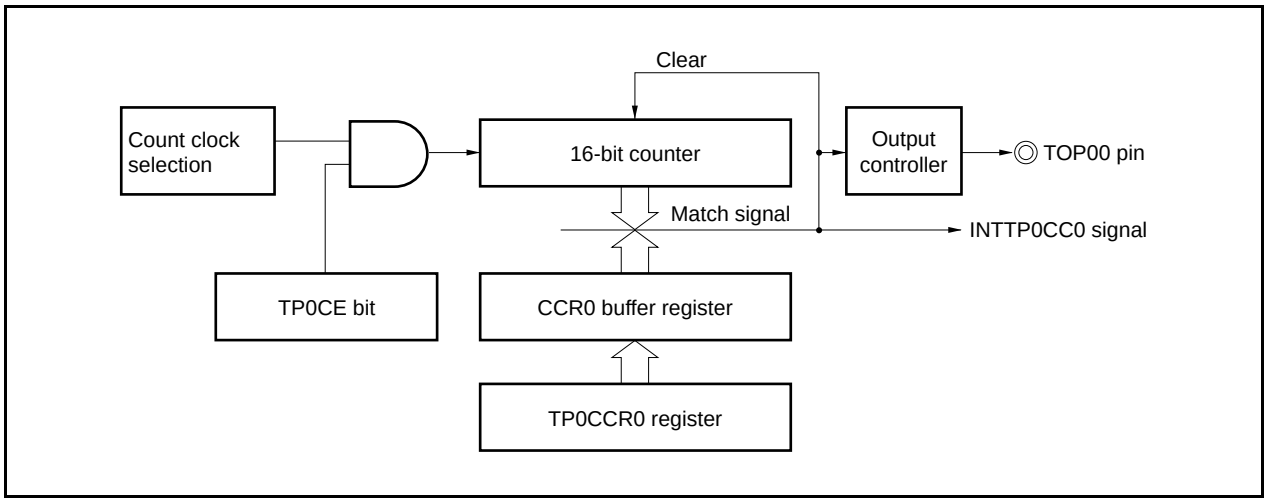
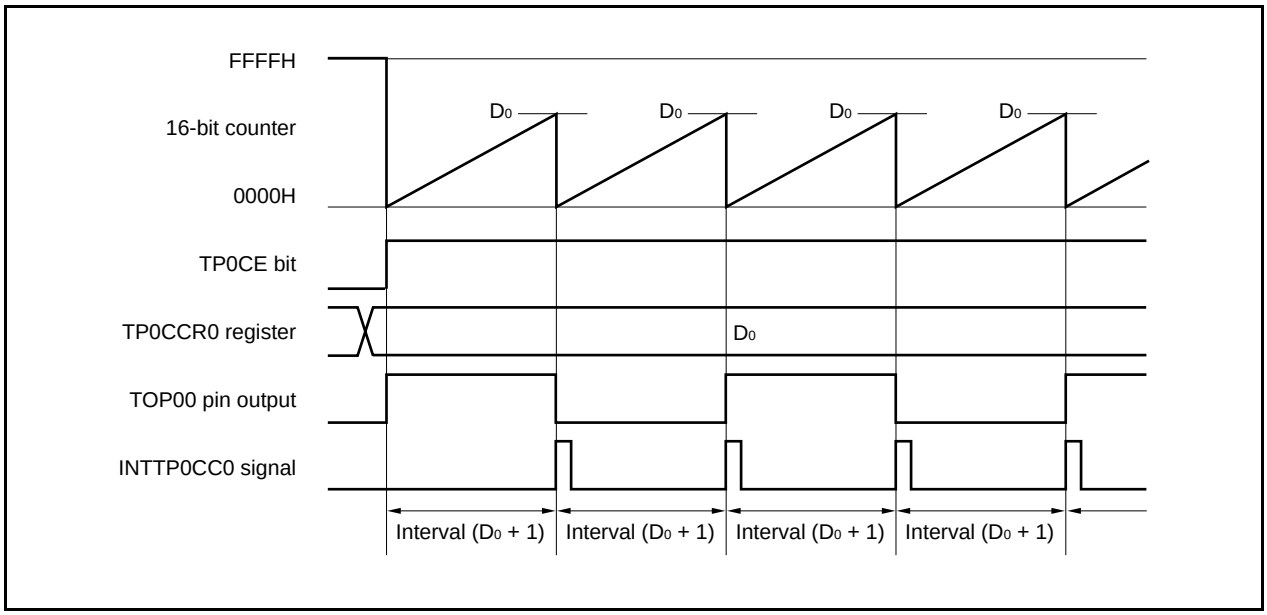



**Figure 7-3. Basic Timing of Operation in Interval Timer Mode**



When the TP0CE bit is set to 1, the value of the 16-bit counter is cleared from FFFFH to 0000H in synchronization with the count clock, and the counter starts counting. At this time, the output of the TOP00 pin is inverted. Additionally, the set value of the TP0CCR0 register is transferred to the CCR0 buffer register.

When the count value of the 16-bit counter matches the value of the CCR0 buffer register, the 16-bit counter is cleared to 0000H, the output of the TOP00 pin is inverted, and a compare match interrupt request signal (INTTP0CC0) is generated.

The interval can be calculated by the following expression.

Interval = (Set value of TP0CCR0 register + 1) × Count clock cycle

**Figure 7-4. Register Setting for Interval Timer Mode Operation (1/2)**

**(a) TMP0 control register 0 (TP0CTL0)**

| | TP0CE | | | | | TP0CKS2 | TP0CKS1 | TP0CKS0 |
|---|---|---|---|---|---|---|---|---|
| TP0CTL0 | 0/1 | 0 | 0 | 0 | 0 | 0/1 | 0/1 | 0/1 |

Select count clock

0: Stop counting
1: Enable counting

**(b) TMP0 control register 1 (TP0CTL1)**

| | | TP0EST | TP0EEE | | | | TP0MD2 | TP0MD1 | TP0MD0 |
|---|---|---|---|---|---|---|---|---|---|
| TP0CTL1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

0, 0, 0:
Interval timer mode

**(c) TMP0 I/O control register 0 (TP0IOC0)**

| | | | | | TP0OL1 | TP0OE1 | TP0OL0 | TP0OE0 |
|---|---|---|---|---|---|---|---|---|
| TP0IOC0 | 0 | 0 | 0 | 0 | 0/1 | 0/1 | 0/1 | 0/1 |

0: Disable TOP00 pin output
1: Enable TOP00 pin output

Setting of output level with operation of TOP00 pin disabled
0: Low level
1: High level

0: Disable TOP01 pin output
1: Enable TOP01 pin output

Setting of output level with operation of TOP01 pin disabled
0: Low level
1: High level

**Figure 7-4. Register Setting for Interval Timer Mode Operation (2/2)**

**(d) TMP0 counter read buffer register (TP0CNT)**

By reading the TP0CNT register, the count value of the 16-bit counter can be read.

**(e) TMP0 capture/compare register 0 (TP0CCR0)**

If the TP0CCR0 register is set to $D_0$, the interval is as follows.

Interval = $(D_0 + 1) \times$ Count clock cycle

**(f) TMP0 capture/compare register 1 (TP0CCR1)**

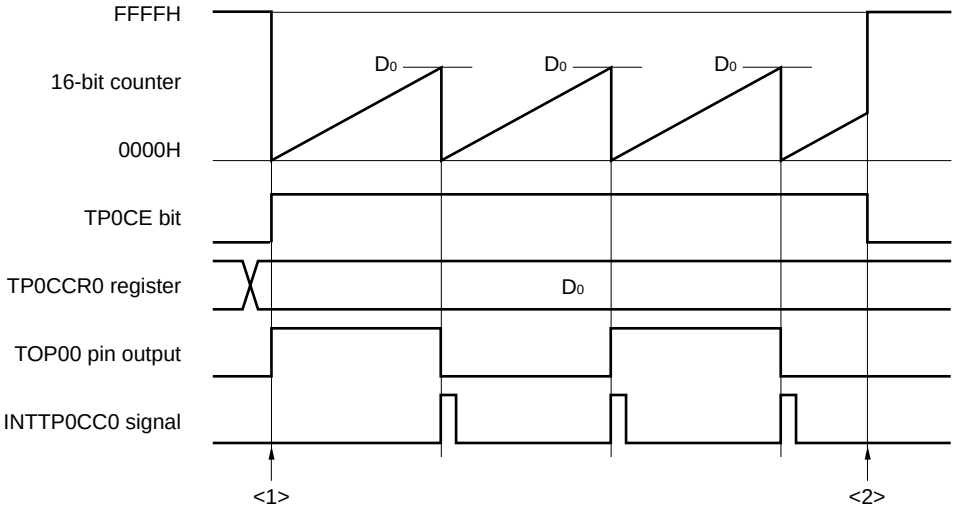
Usually, the TP0CCR1 register is not used in the interval timer mode. However, the set value of the TP0CCR1 register is transferred to the CCR1 buffer register. A compare match interrupt request signal (INTTP0CC1) is generated when the count value of the 16-bit counter matches the value of the CCR1 buffer register.

Therefore, mask the interrupt request by using the corresponding interrupt mask flag (TP0CCMK1).

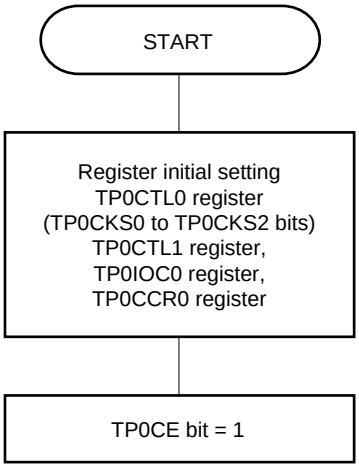
**Remark** TMP0 I/O control register 1 (TP0IOC1), TMP0 I/O control register 2 (TP0IOC2), and TMP0 option register 0 (TP0OPT0) are not used in the interval timer mode.

**(1) Interval timer mode operation flow**

**Figure 7-5. Software Processing Flow in Interval Timer Mode**



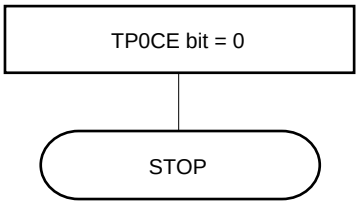

<1> Count operation start flow




Initial setting of these registers is performed before setting the TP0CE bit to 1.

The TP0CKS0 to TP0CKS2 bits can be set at the same time when counting has been started (TP0CE bit = 1).

<2> Count operation stop flow



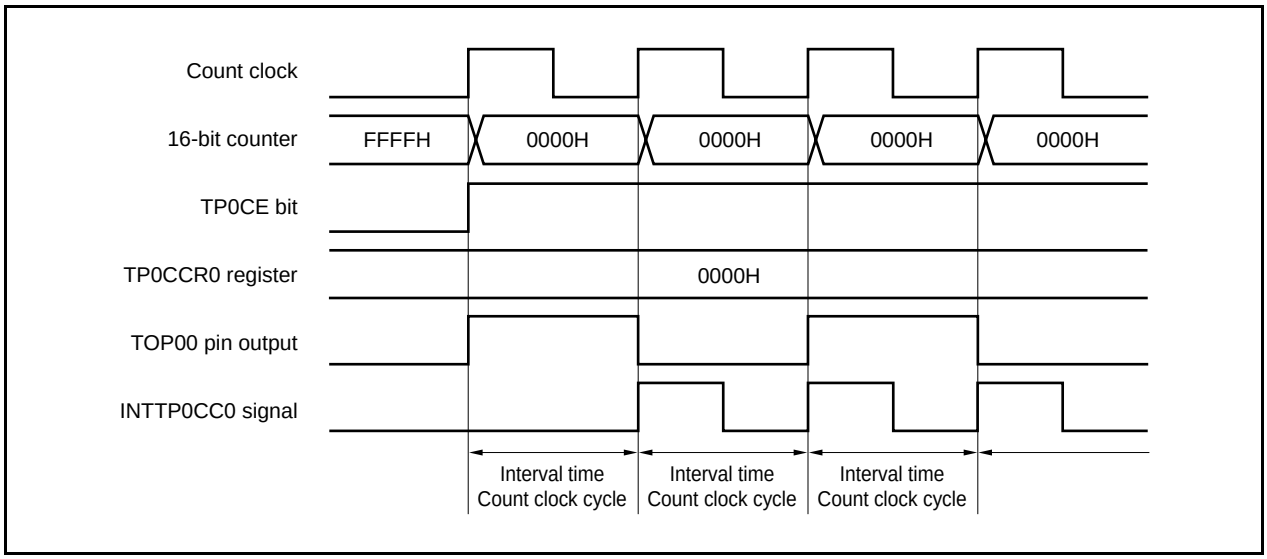

The counter is initialized and counting is stopped by clearing the TP0CE bit to 0.

### (2) Interval timer mode operation timing

#### (a) Operation if TP0CCR0 register is cleared to 0000H

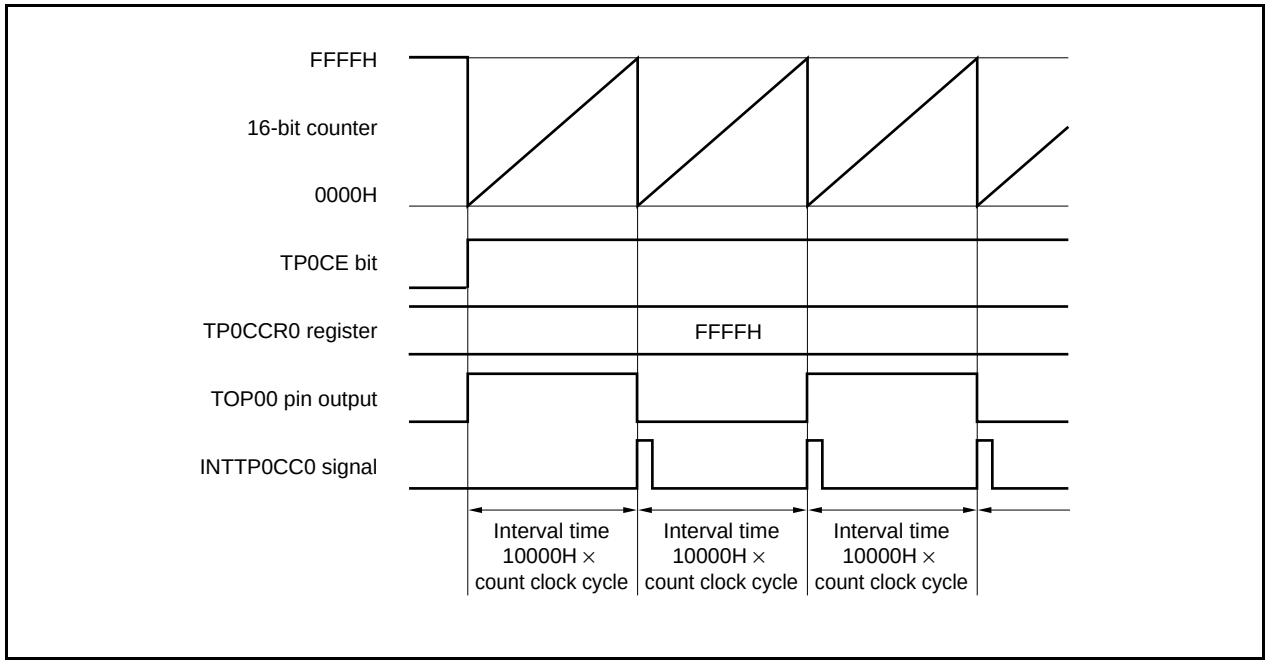
If the TP0CCR0 register is cleared to 0000H, the INTTP0CC0 signal is generated at each count clock, and the output of the TOP00 pin is inverted.

The value of the 16-bit counter is always 0000H.



**(b) Operation if TP0CCR0 register is set to FFFFH**

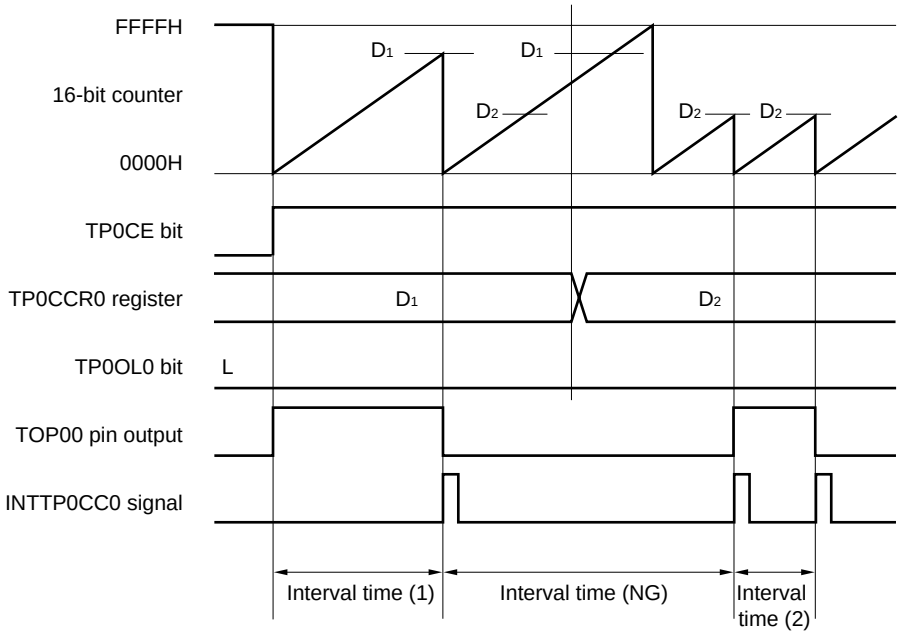
If the TP0CCR0 register is set to FFFFH, the 16-bit counter counts up to FFFFH. The counter is cleared to 0000H in synchronization with the next count-up timing. The INTTP0CC0 signal is generated and the output of the TOP00 pin is inverted. At this time, an overflow interrupt request signal (INTTP0OV) is not generated, nor is the overflow flag (TP0OPT0.TP0OVF bit) set to 1.



**(c) Notes on rewriting TP0CCR0 register**

To change the value of the TP0CCR0 register to a smaller value, stop counting once and then change the set value.

If the value of the TP0CCR0 register is rewritten to a smaller value during counting, the 16-bit counter may overflow.




**Remark** Interval time (1): $(D_1 + 1) \times$ Count clock cycle
Interval time (NG): $(10000H + D_2 + 1) \times$ Count clock cycle
Interval time (2): $(D_2 + 1) \times$ Count clock cycle

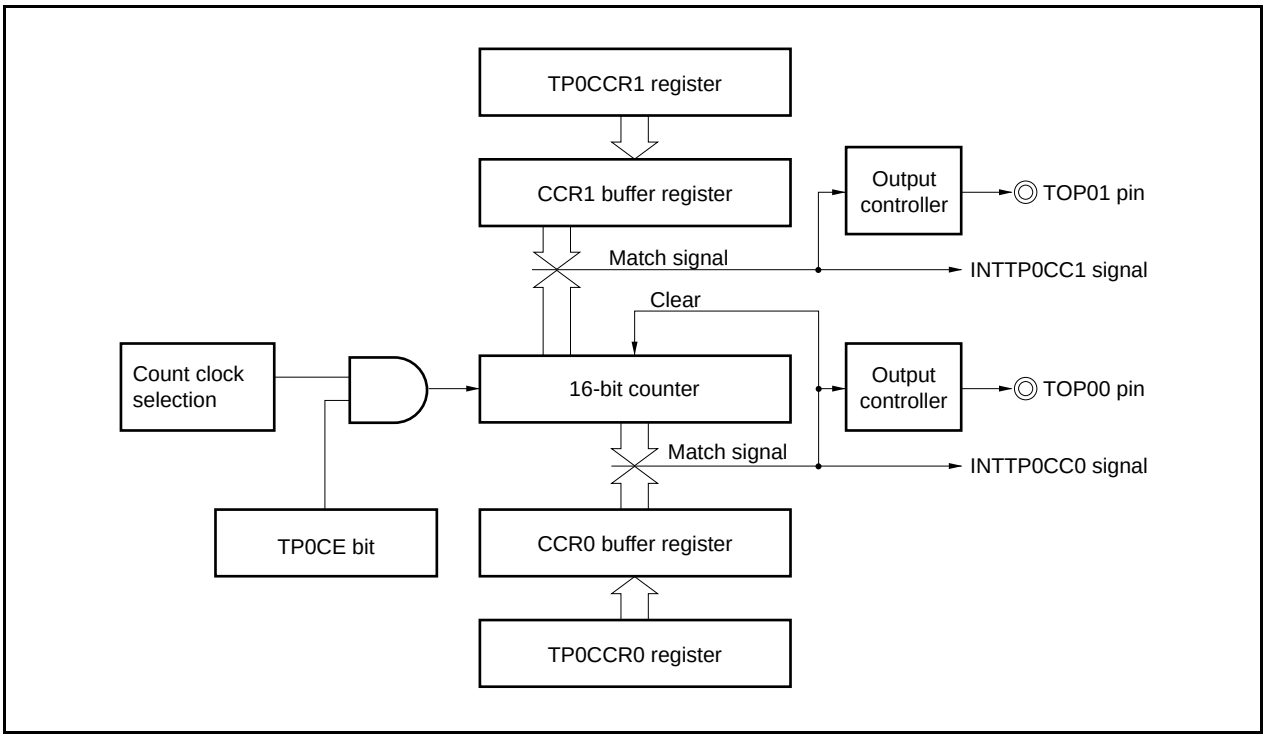
If the value of the TP0CCR0 register is changed from $D_1$ to $D_2$ while the count value is greater than $D_2$ but less than $D_1$, the count value is transferred to the CCR0 buffer register as soon as the TP0CCR0 register has been rewritten. Consequently, the value of the 16-bit counter that is compared is $D_2$.

Because the count value has already exceeded $D_2$, however, the 16-bit counter counts up to FFFFH, overflows, and then counts up again from 0000H. When the count value matches $D_2$, the INTTP0CC0 signal is generated and the output of the TOP00 pin is inverted.

Therefore, the INTTP0CC0 signal may not be generated at the interval time "$(D_1 + 1) \times$ Count clock cycle" or "$(D_2 + 1) \times$ Count clock cycle" originally expected, but may be generated at an interval of "$(10000H + D_2 + 1) \times$ Count clock cycle".

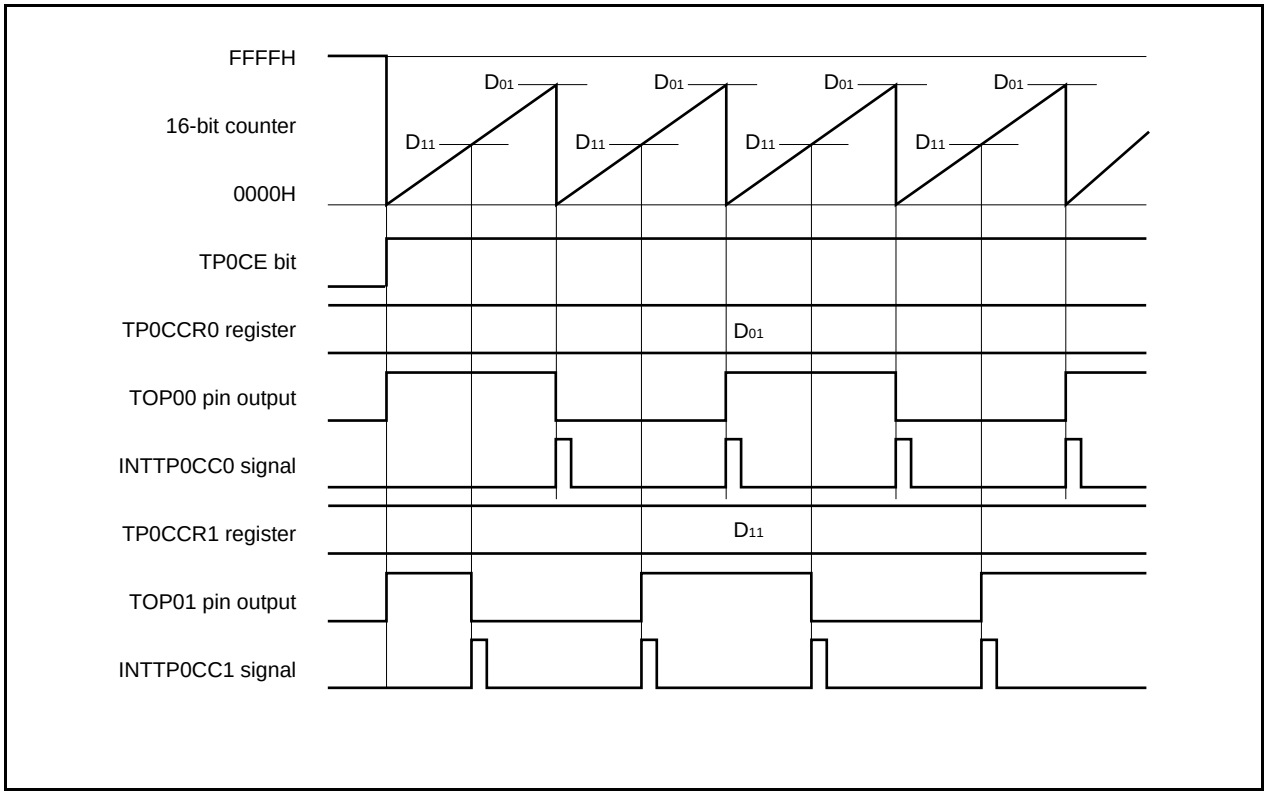
### (d) Operation of TP0CCR1 register

**Figure 7-6. Configuration of TP0CCR1 Register**



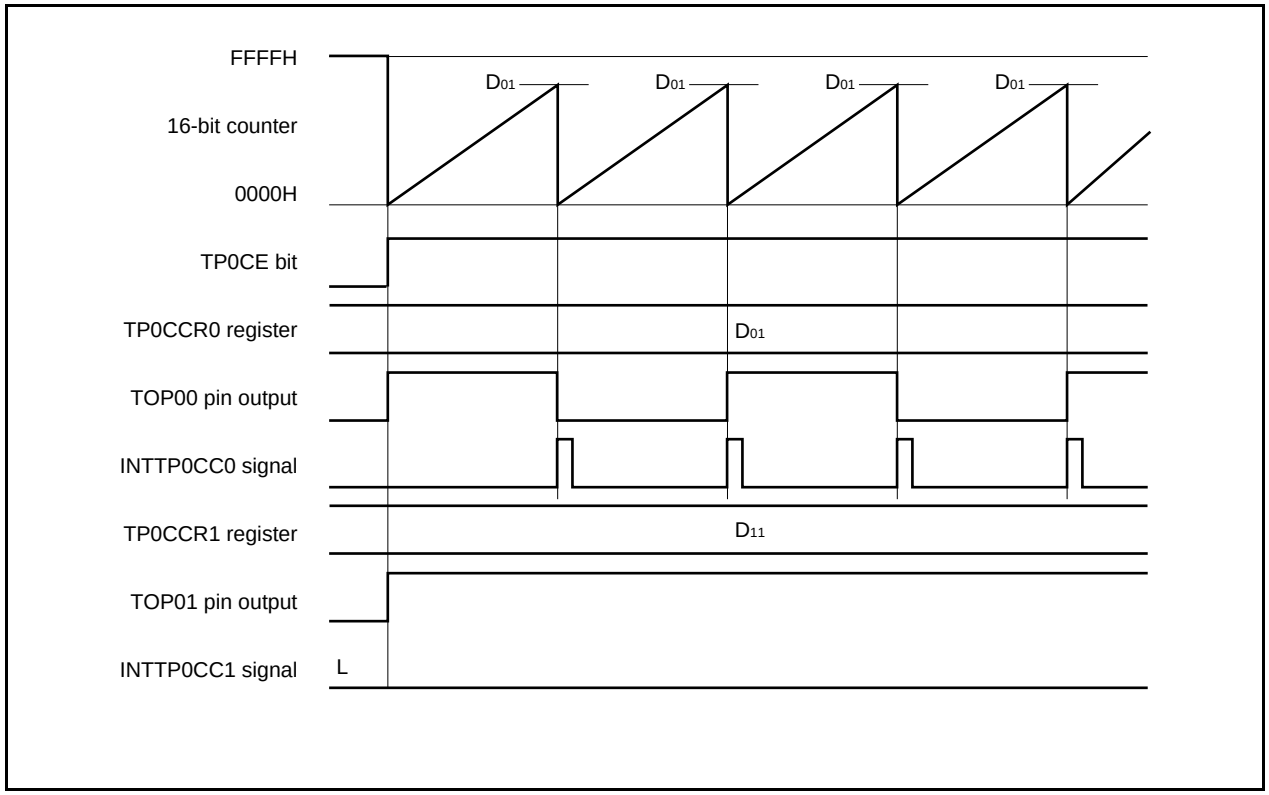
If the set value of the TP0CCR1 register is less than the set value of the TP0CCR0 register, the INTTP0CC1 signal is generated once per cycle. At the same time, the output of the TOP01 pin is inverted. The TOP01 pin outputs a square wave with the same cycle as that output by the TOP00 pin.

**Figure 7-7. Timing Chart When $D_{01} \geq D_{11}$**



If the set value of the TP0CCR1 register is greater than the set value of the TP0CCR0 register, the count value of the 16-bit counter does not match the value of the TP0CCR1 register. Consequently, the INTTP0CC1 signal is not generated, nor is the output of the TOP01 pin changed.

**Figure 7-8. Timing Chart When $D_{01} < D_{11}$**



### 7.5.2 External event count mode (TP0MD2 to TP0MD0 bits = 001)

In the external event count mode, the valid edge of the external event count input is counted when the TP0CTL0.TP0CE bit is set to 1, and an interrupt request signal (INTTP0CC0) is generated each time the specified number of edges have been counted. The TOP00 pin cannot be used.

Usually, the TP0CCR1 register is not used in the external event count mode.

**Figure 7-9. Configuration in External Event Count Mode**

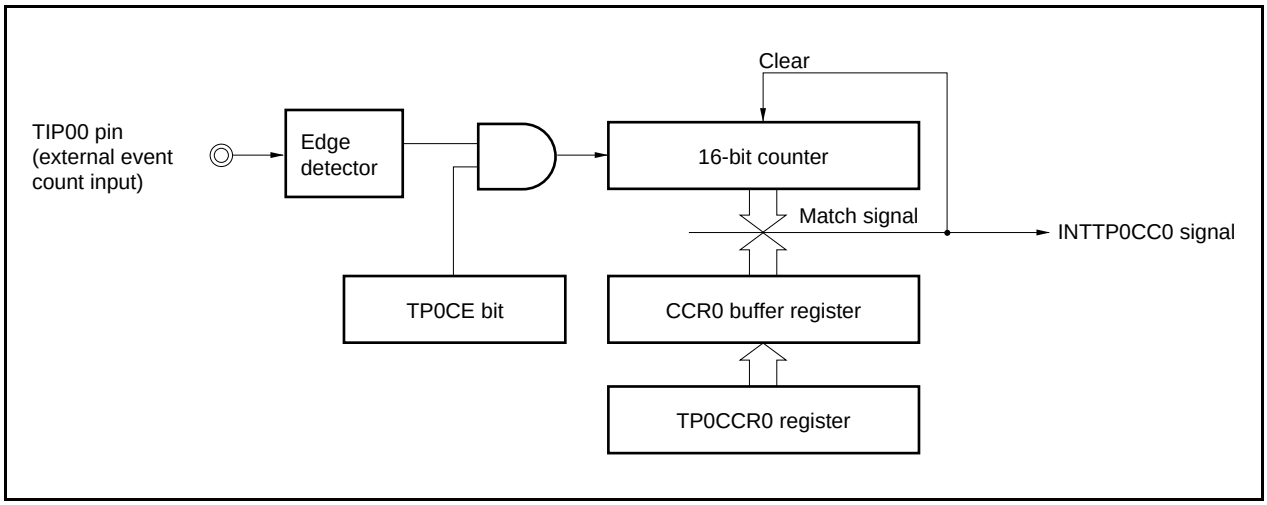
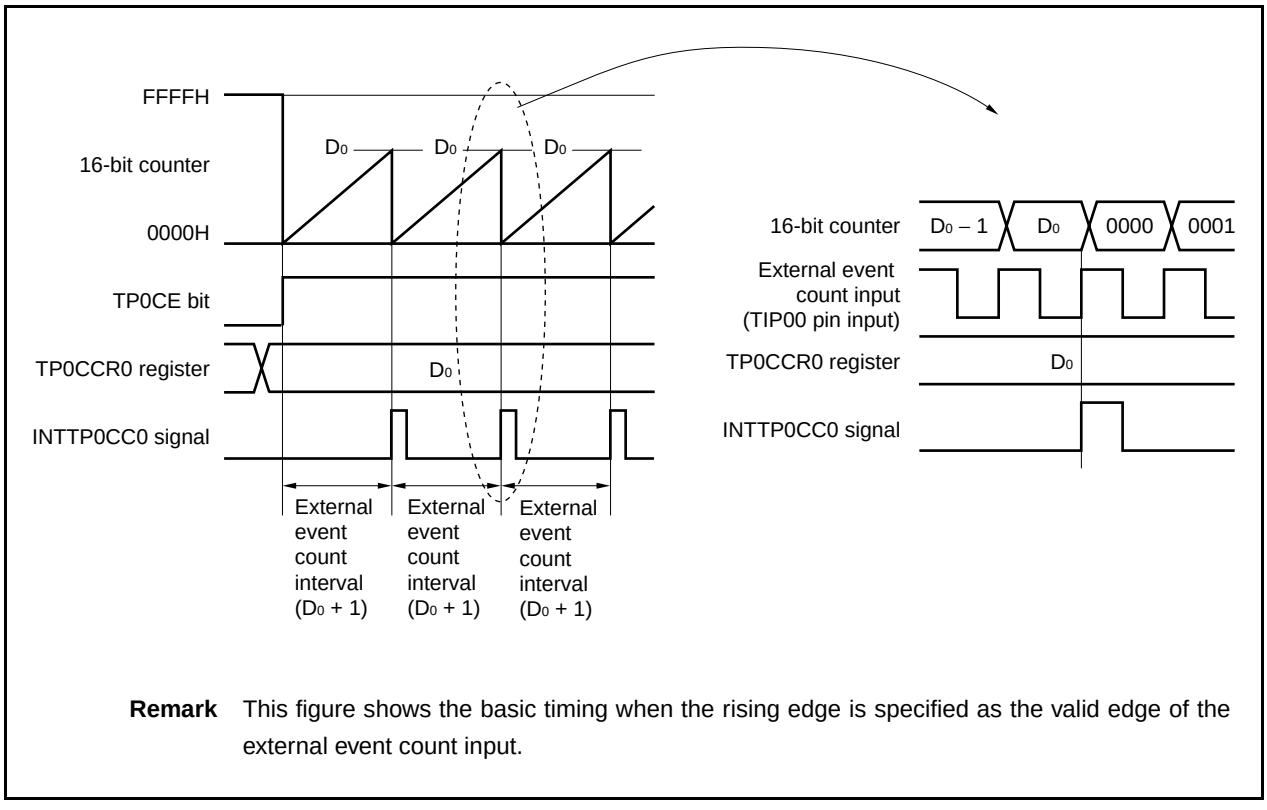



**Figure 7-10. Basic Timing in External Event Count Mode**




**Remark** This figure shows the basic timing when the rising edge is specified as the valid edge of the external event count input.

When the TP0CE bit is set to 1, the value of the 16-bit counter is cleared from FFFFH to 0000H. The counter counts each time the valid edge of external event count input is detected. Additionally, the set value of the TP0CCR0 register is transferred to the CCR0 buffer register.

When the count value of the 16-bit counter matches the value of the CCR0 buffer register, the 16-bit counter is cleared to 0000H, and a compare match interrupt request signal (INTTP0CC0) is generated.

The INTTP0CC0 signal is generated each time the valid edge of the external event count input has been detected (set value of TP0CCR0 register + 1) times.

**Figure 7-11. Register Setting for Operation in External Event Count Mode (1/2)**

**(a) TMP0 control register 0 (TP0CTL0)**

| | TP0CE | | | | | TP0CKS2 | TP0CKS1 | TP0CKS0 |
|---|---|---|---|---|---|---|---|---|
| TP0CTL0 | 0/1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

0: Stop counting
1: Enable counting

**(b) TMP0 control register 1 (TP0CTL1)**

| | | TP0EST | TP0EEE | | | TP0MD2 | TP0MD1 | TP0MD0 |
|---|---|---|---|---|---|---|---|---|
| TP0CTL1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

0, 0, 1:
External event count mode

**(c) TMP0 I/O control register 0 (TP0IOC0)**

| | | | | | TP0OL1 | TP0OE1 | TP0OL0 | TP0OE0 |
|---|---|---|---|---|---|---|---|---|
| TP0IOC0 | 0 | 0 | 0 | 0 | 0/1 | 0/1 | 0 | 0 |

0: Disable TOP00 pin output
0: Disable TOP01 pin output
1: Enable TOP01 pin output
Setting of output level with operation of TOP01 pin disabled
0: Low level
1: High level

**(d) TMP0 I/O control register 2 (TP0IOC2)**

| | | | | | TP0EES1 | TP0EES0 | TP0ETS1 | TP0ETS0 |
|---|---|---|---|---|---|---|---|---|
| TP0IOC2 | 0 | 0 | 0 | 0 | 0/1 | 0/1 | 0 | 0 |

Select valid edge of external event count input

**Figure 7-11. Register Setting for Operation in External Event Count Mode (2/2)**

**(e) TMP0 counter read buffer register (TP0CNT)**

The count value of the 16-bit counter can be read by reading the TP0CNT register.

**(f) TMP0 capture/compare register 0 (TP0CCR0)**

If $D_0$ is set to the TP0CCR0 register, the counter is cleared and a compare match interrupt request signal (INTTP0CC0) is generated when the number of external event counts reaches ($D_0$ + 1).

**(g) TMP0 capture/compare register 1 (TP0CCR1)**

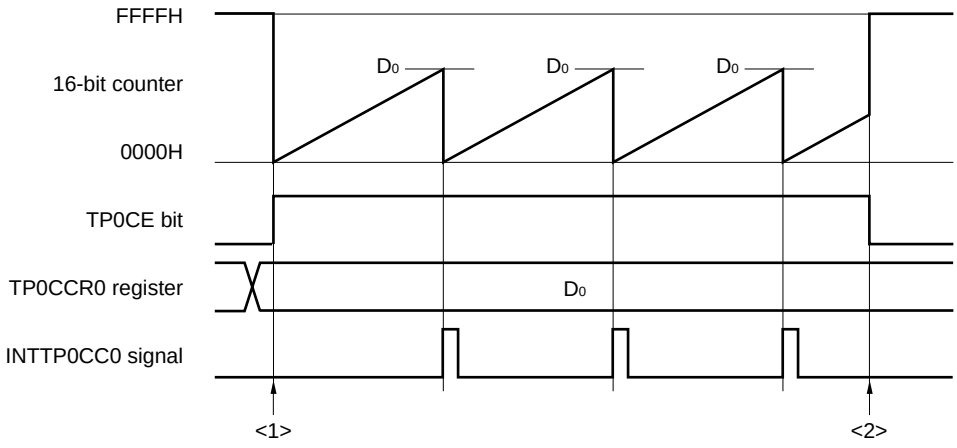
Usually, the TP0CCR1 register is not used in the external event count mode. However, the set value of the TP0CCR1 register is transferred to the CCR1 buffer register. When the count value of the 16-bit counter matches the value of the CCR1 buffer register, a compare match interrupt request signal (INTTP0CC1) is generated.

Therefore, mask the interrupt signal by using the interrupt mask flag (TP0CCMK1).

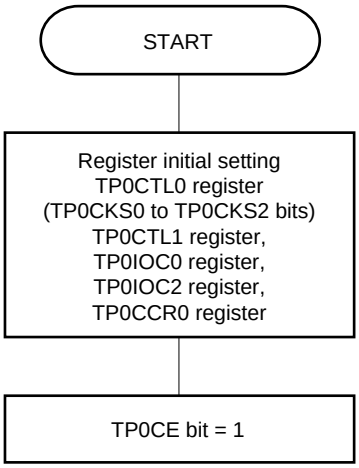
**Remark** TMP0 I/O control register 1 (TP0IOC1) and TMP0 option register 0 (TP0OPT0) are not used in the external event count mode.

**(1) External event count mode operation flow**

**Figure 7-12. Flow of Software Processing in External Event Count Mode**



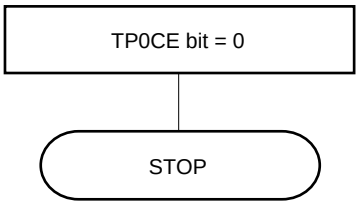

<1> Count operation start flow




Initial setting of these registers is performed before setting the TP0CE bit to 1.

The TP0CKS0 to TP0CKS2 bits can be set at the same time when counting has been started (TP0CE bit = 1).

<2> Count operation stop flow



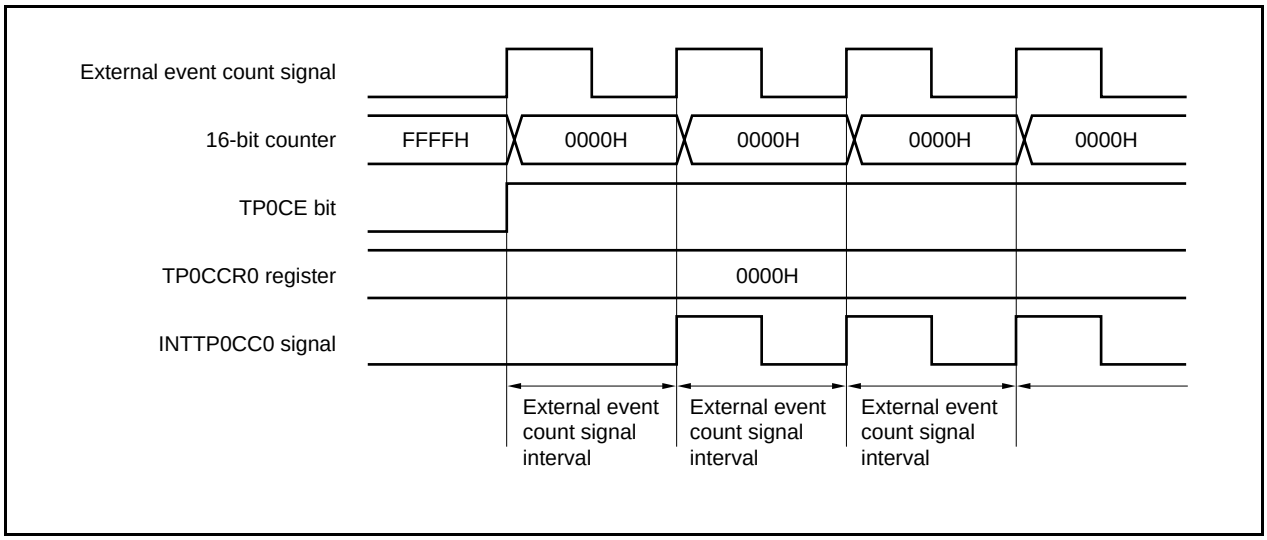

The counter is initialized and counting is stopped by clearing the TP0CE bit to 0.

### (2) Operation timing in external event count mode

#### (a) Operation if TP0CCR0 register is cleared to 0000H

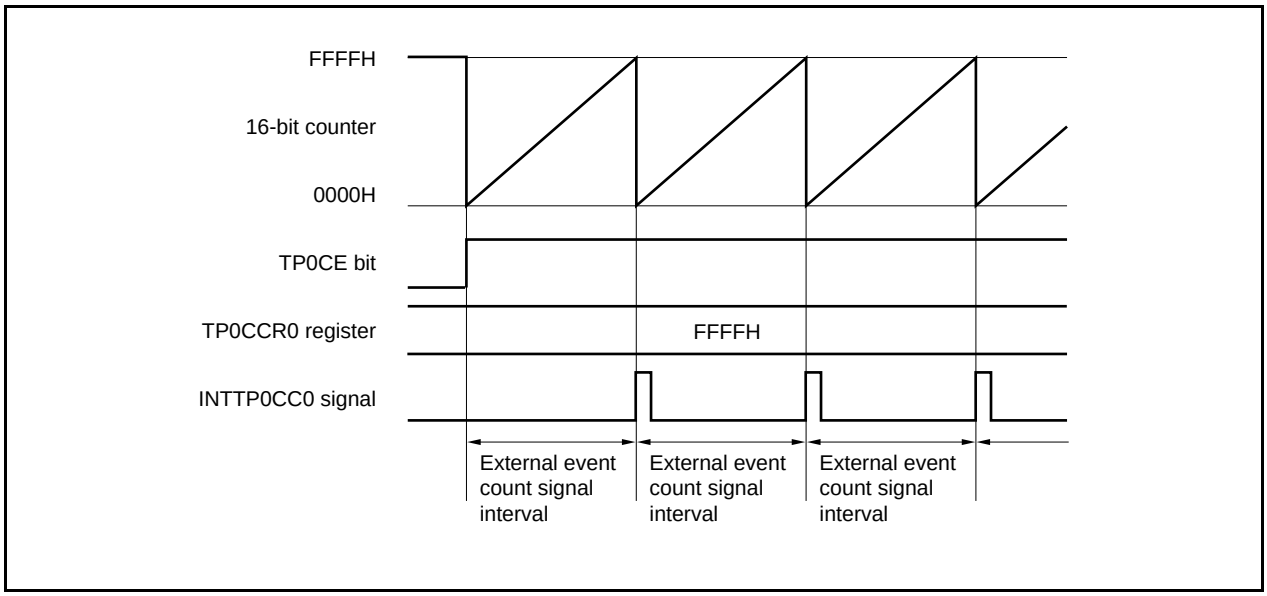
If the TP0CCR0 register is cleared to 0000H, the INTTP0CC0 signal is generated each time the valid signal of the external event count signal has been detected.

The 16-bit counter is always 0000H.




#### (b) Operation if TP0CCR0 register is set to FFFFH

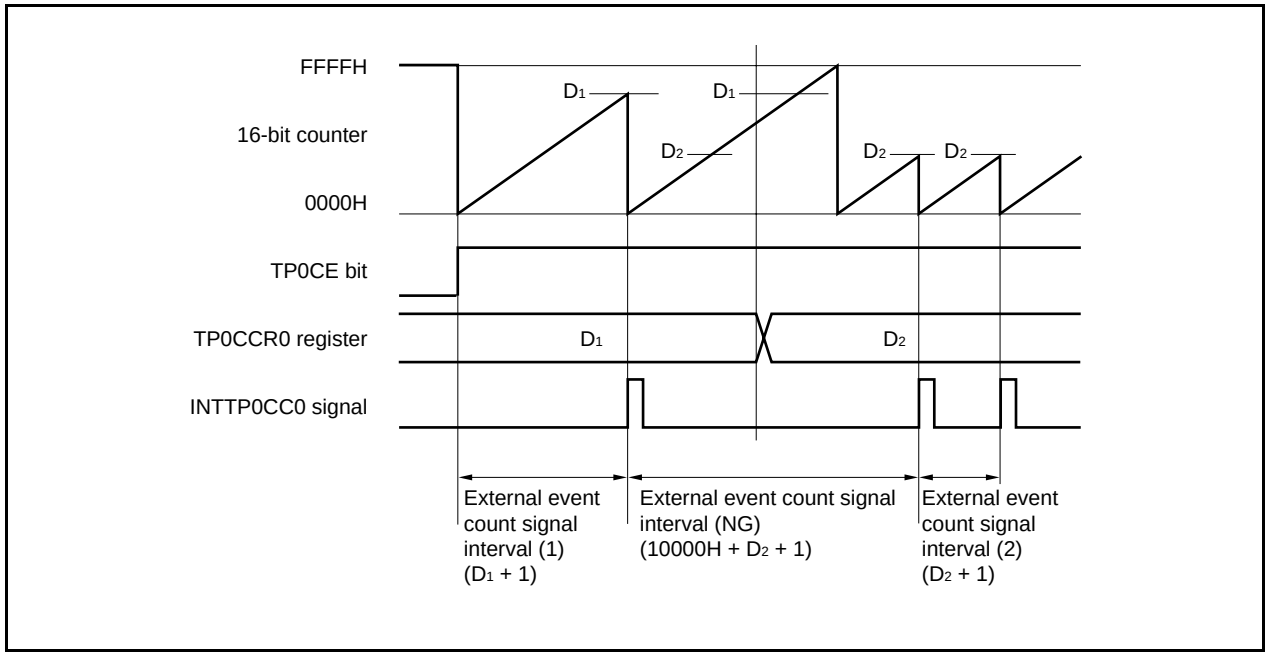
If the TP0CCR0 register is set to FFFFH, the 16-bit counter counts to FFFFH each time the valid edge of the external event count signal has been detected. The 16-bit counter is cleared to 0000H in synchronization with the next count-up timing, and the INTTP0CC0 signal is generated. At this time, the TP0OPT0.TP0OVF bit is not set.



**(c) Notes on rewriting the TP0CCR0 register**

To change the value of the TP0CCR0 register to a smaller value, stop counting once and then change the set value.

If the value of the TP0CCR0 register is rewritten to a smaller value during counting, the 16-bit counter may overflow.



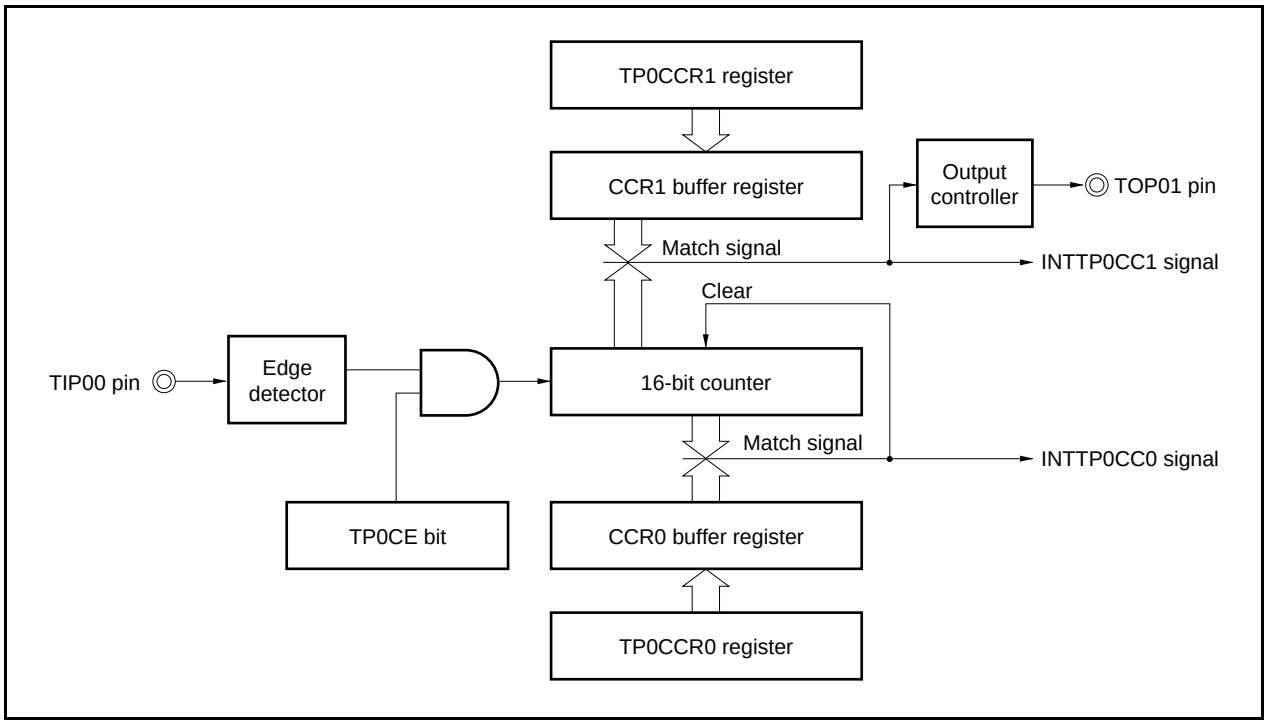

If the value of the TP0CCR0 register is changed from $D_1$ to $D_2$ while the count value is greater than $D_2$ but less than $D_1$, the count value is transferred to the CCR0 buffer register as soon as the TP0CCR0 register has been rewritten. Consequently, the value that is compared with the 16-bit counter is $D_2$.

Because the count value has already exceeded $D_2$, however, the 16-bit counter counts up to FFFFH, overflows, and then counts up again from 0000H. When the count value matches $D_2$, the INTTP0CC0 signal is generated.

Therefore, the INTTP0CC0 signal may not be generated at the valid edge count of "($D_1$ + 1) times" or "($D_2$ + 1) times" originally expected, but may be generated at the valid edge count of "(10000H + $D_2$ + 1) times".

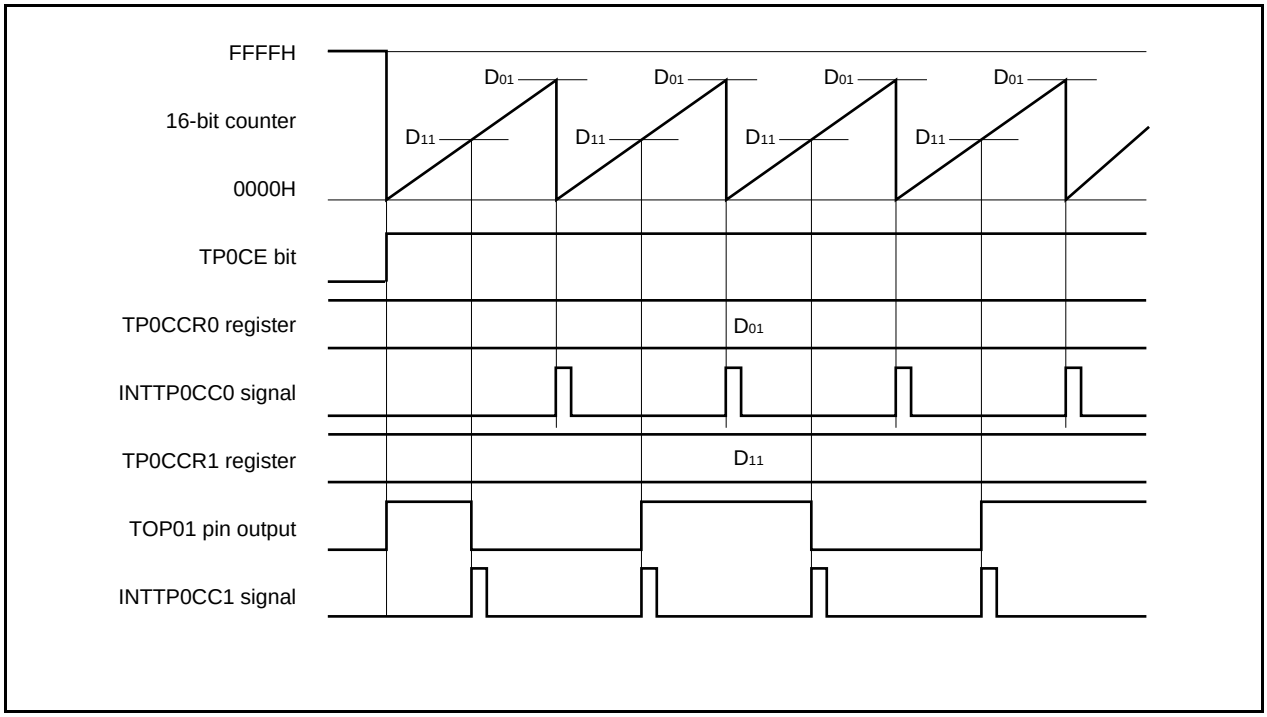
### (d) Operation of TP0CCR1 register

**Figure 7-13. Configuration of TP0CCR1 Register**



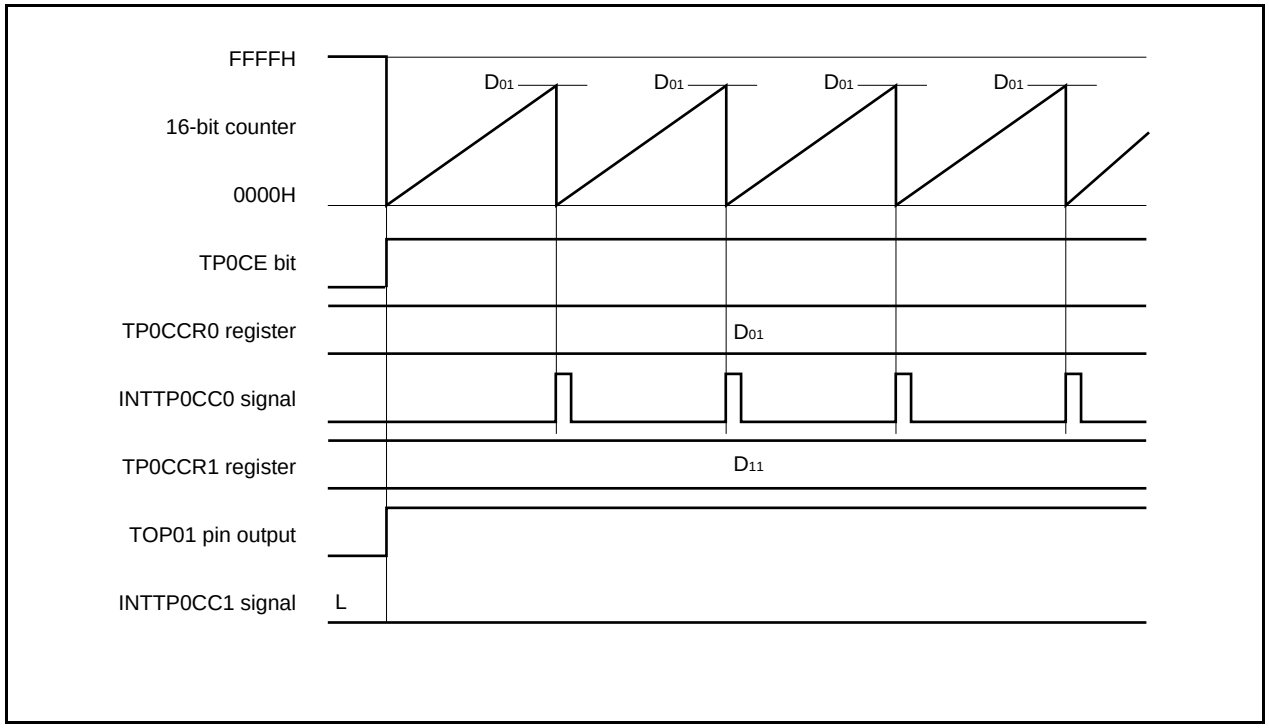

If the set value of the TP0CCR1 register is smaller than the set value of the TP0CCR0 register, the INTTP0CC1 signal is generated once per cycle. At the same time, the output signal of the TOP01 pin is inverted.

**Figure 7-14. Timing Chart When $D_{01} \geq D_{11}$**



If the set value of the TP0CCR1 register is greater than the set value of the TP0CCR0 register, the INTTP0CC1 signal is not generated because the count value of the 16-bit counter and the value of the TP0CCR1 register do not match. Nor is the output signal of the TOP01 pin changed.

**Figure 7-15. Timing Chart When $D_{01} < D_{11}$**



### 7.5.3 External trigger pulse output mode (TP0MD2 to TP0MD0 bits = 010)

In the external trigger pulse output mode, 16-bit timer/event counter P waits for a trigger when the TP0CTL0.TP0CE bit is set to 1. When the valid edge of an external trigger input signal is detected, 16-bit timer/event counter P starts counting, and outputs a PWM waveform from the TOP01 pin.

Pulses can also be output by generating a software trigger instead of using the external trigger. When using a software trigger, a square wave that has one cycle of the PWM waveform as half its cycle can also be output from the TOP00 pin.

**Figure 7-16. Configuration in External Trigger Pulse Output Mode**

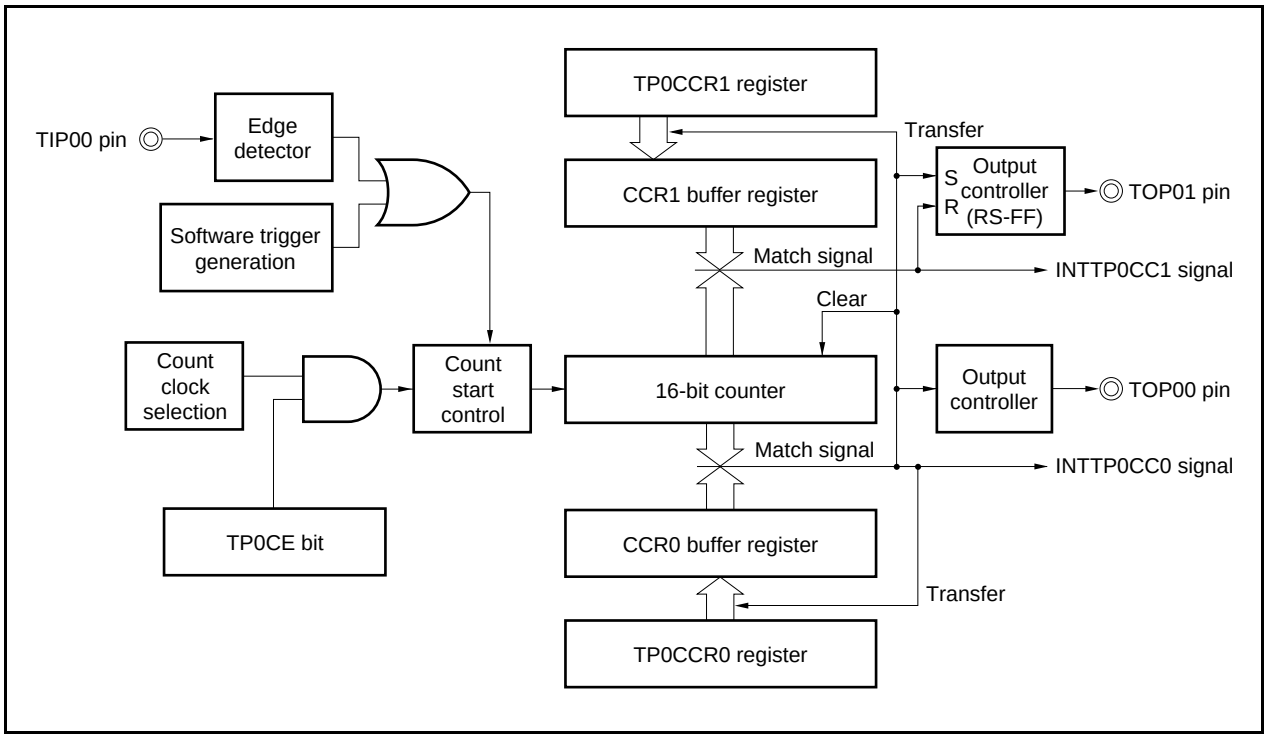
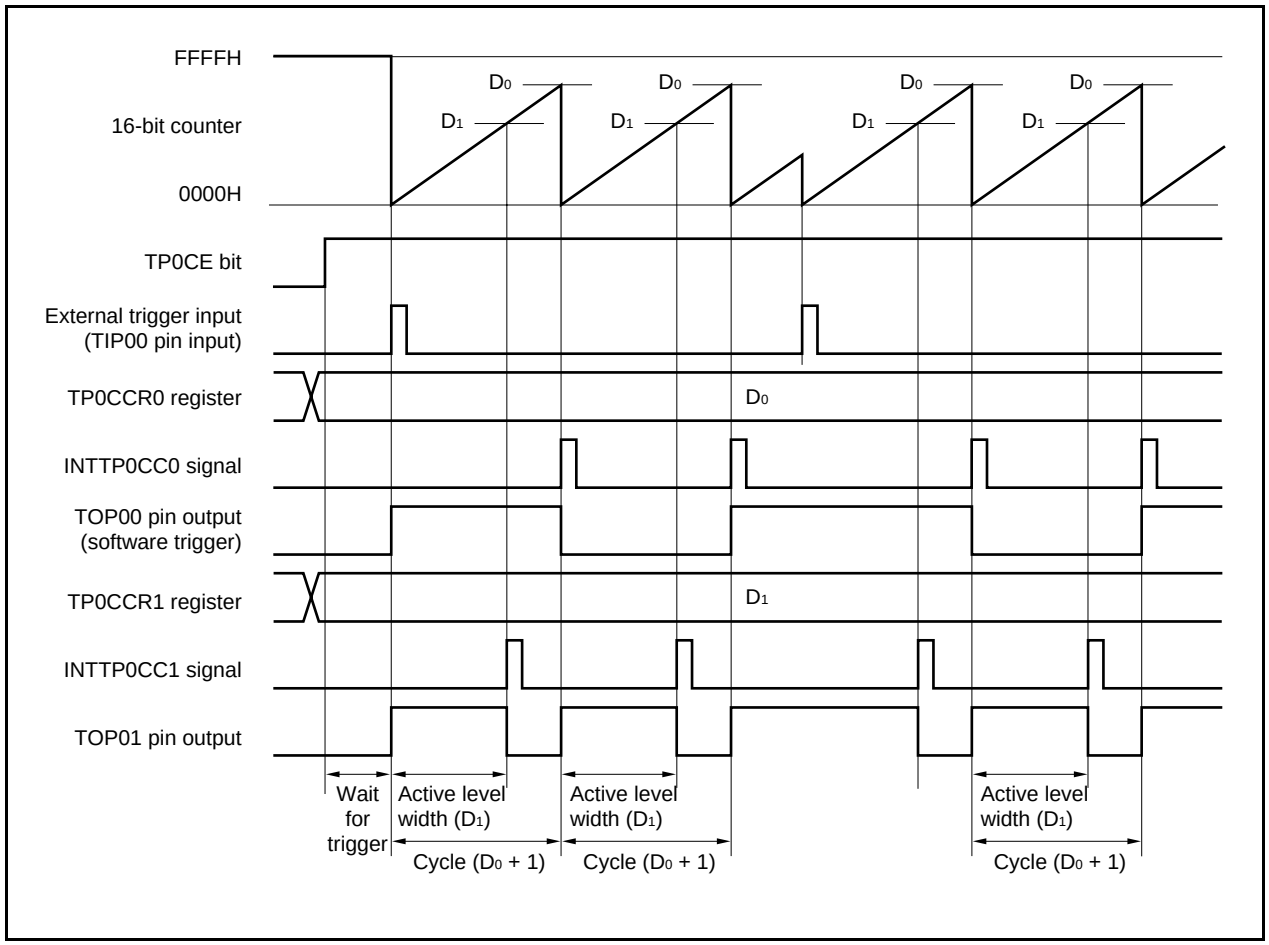


**Figure 7-17. Basic Timing in External Trigger Pulse Output Mode**




16-bit timer/event counter P waits for a trigger when the TP0CE bit is set to 1. When the trigger is generated, the 16-bit counter is cleared from FFFFH to 0000H, starts counting at the same time, and outputs a PWM waveform from the TOP01 pin.

If the trigger is generated again while the counter is operating, the counter is cleared to 0000H and restarted.

The active level width, cycle, and duty factor of the PWM waveform can be calculated as follows.

Active level width = (Set value of TP0CCR1 register) × Count clock cycle
Cycle = (Set value of TP0CCR0 register + 1) × Count clock cycle
Duty factor = (Set value of TP0CCR1 register)/(Set value of TP0CCR0 register + 1)

The compare match interrupt request signal INTTP0CC0 is generated when the 16-bit counter counts next time after its count value matches the value of the CCR0 buffer register, and the 16-bit counter is cleared to 0000H. The compare match interrupt request signal INTTP0CC1 is generated when the count value of the 16-bit counter matches the value of the CCR1 buffer register.

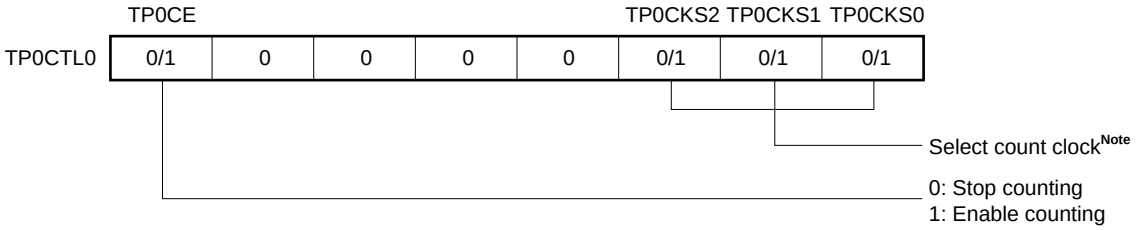
The value set to the TP0CCRa register is transferred to the CCRa buffer register when the count value of the 16-bit counter matches the value of the CCRa buffer register and the 16-bit counter is cleared to 0000H.

The valid edge of an external trigger input signal, or setting the software trigger (TP0CTL1.TP0EST bit) to 1 is used as the trigger.

**Remark** a = 0, 1

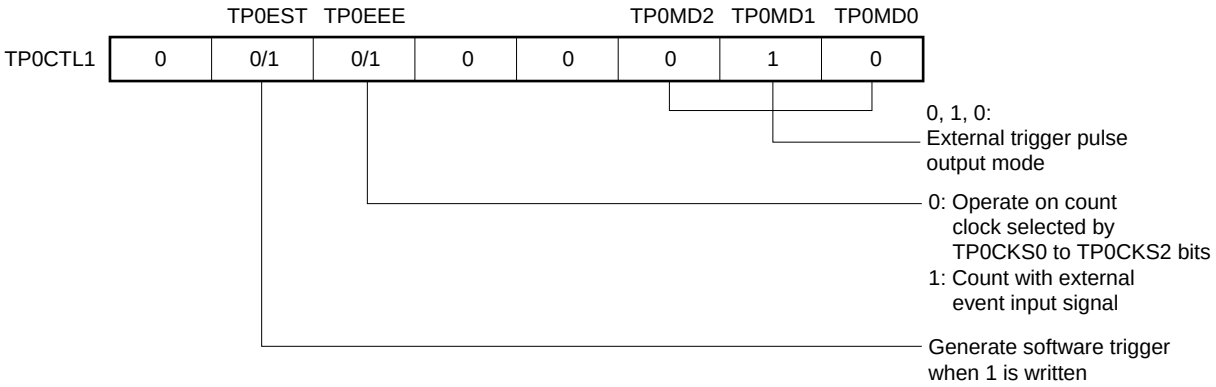
**Figure 7-18. Setting of Registers in External Trigger Pulse Output Mode (1/2)**

**(a) TMP0 control register 0 (TP0CTL0)**

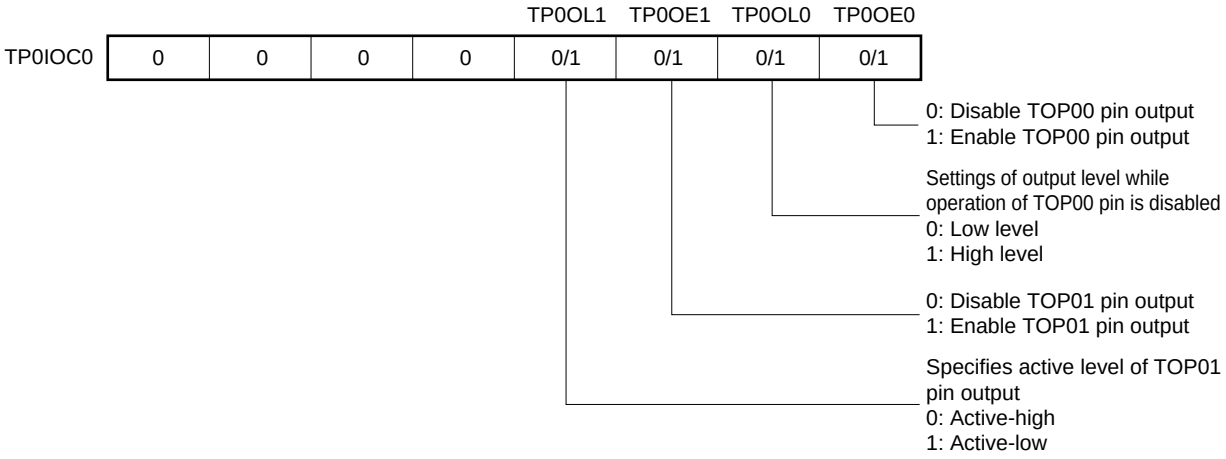



**Note** The setting is invalid when the TP0CTL1.TP0EEE bit = 1.

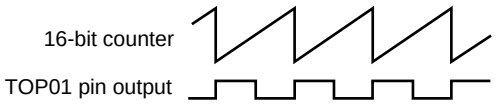
**(b) TMP0 control register 1 (TP0CTL1)**




**(c) TMP0 I/O control register 0 (TP0IOC0)**




- When TP0OL1 bit = 0




- When TP0OL1 bit = 1

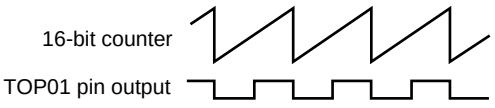
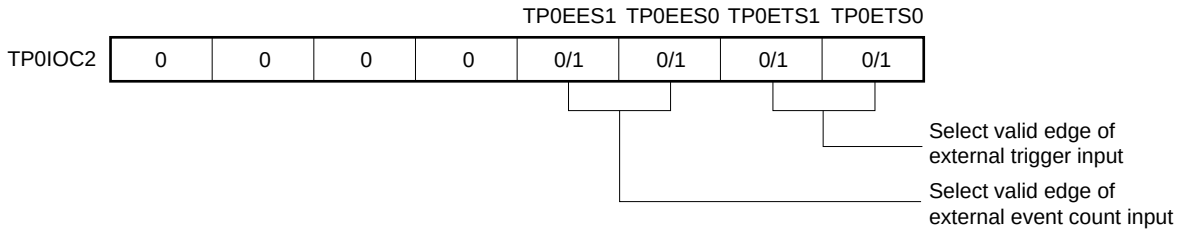


**Figure 7-18. Setting of Registers in External Trigger Pulse Output Mode (2/2)**

**(d) TMP0 I/O control register 2 (TP0IOC2)**




**(e) TMP0 counter read buffer register (TP0CNT)**

The value of the 16-bit counter can be read by reading the TP0CNT register.

**(f) TMP0 capture/compare registers 0 and 1 (TP0CCR0 and TP0CCR1)**

If $D_0$ is set to the TP0CCR0 register and $D_1$ to the TP0CCR1 register, the cycle and active level of the PWM waveform are as follows.

Cycle = $(D_0 + 1) \times$ Count clock cycle
Active level width = $D_1 \times$ Count clock cycle

**Remark** TMP0 I/O control register 1 (TP0IOC1) and TMP0 option register 0 (TP0OPT0) are not used in the external trigger pulse output mode.

**(1) Operation flow in external trigger pulse output mode**

**Figure 7-19. Software Processing Flow in External Trigger Pulse Output Mode (1/2)**

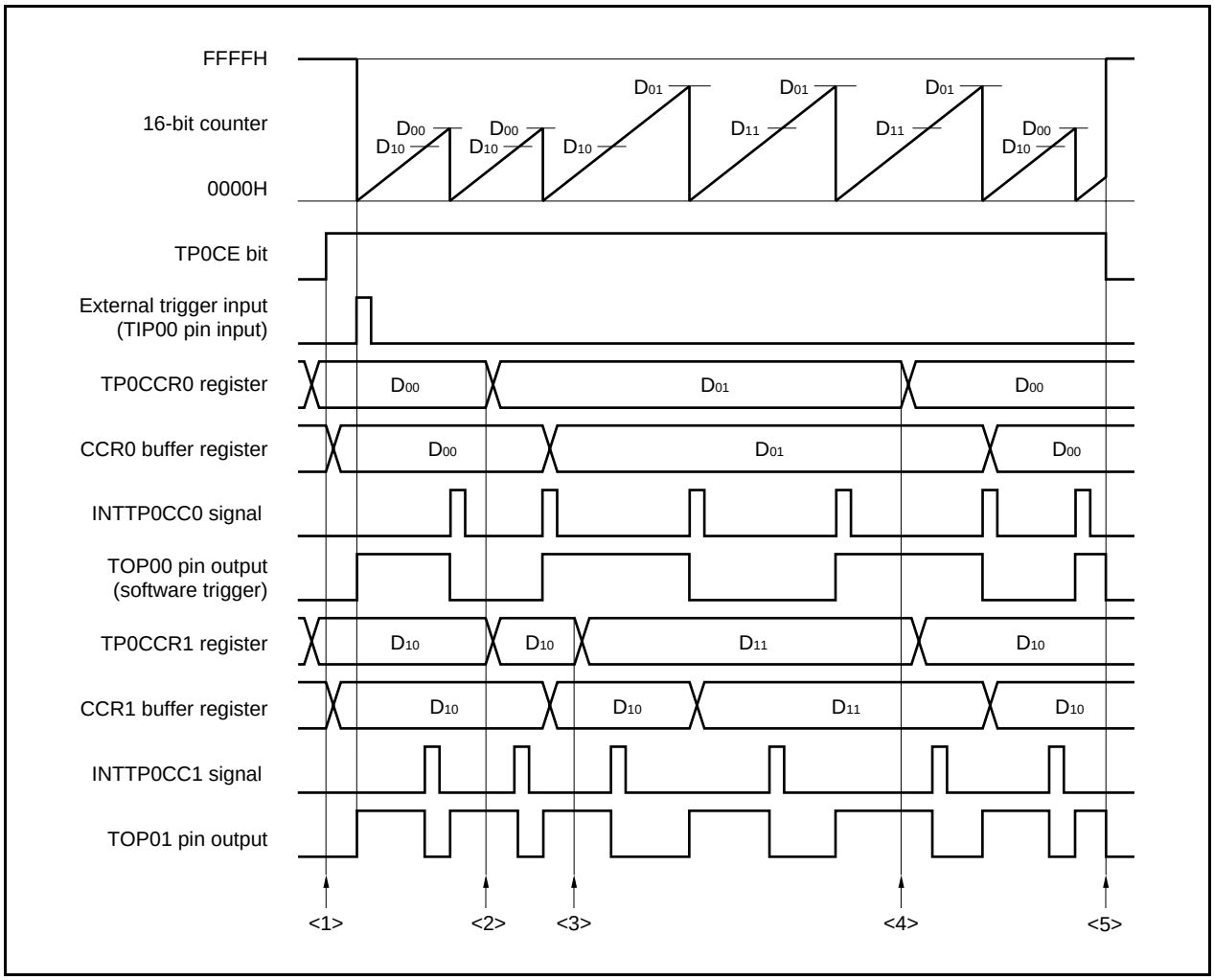
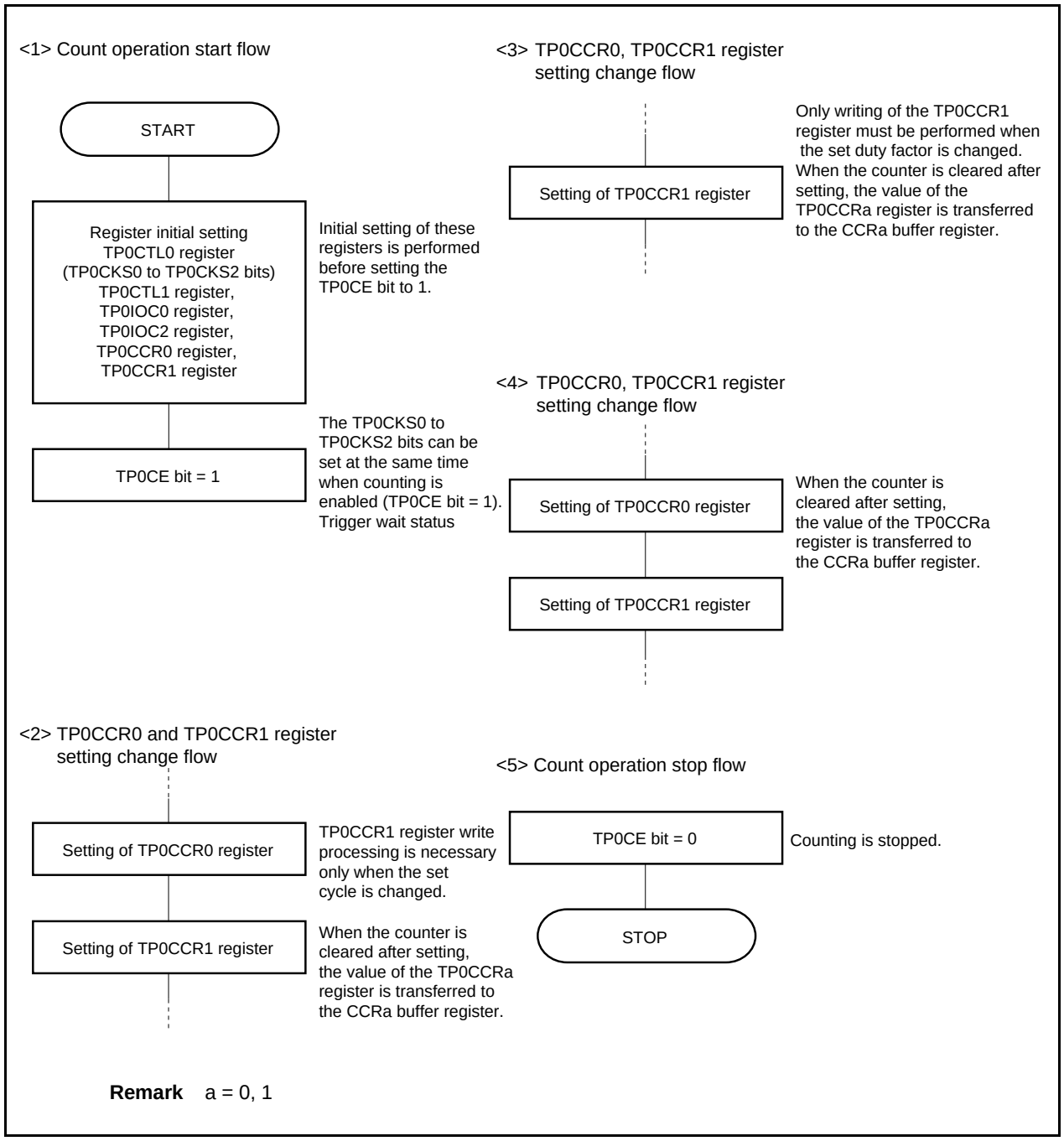


**Figure 7-19. Software Processing Flow in External Trigger Pulse Output Mode (2/2)**



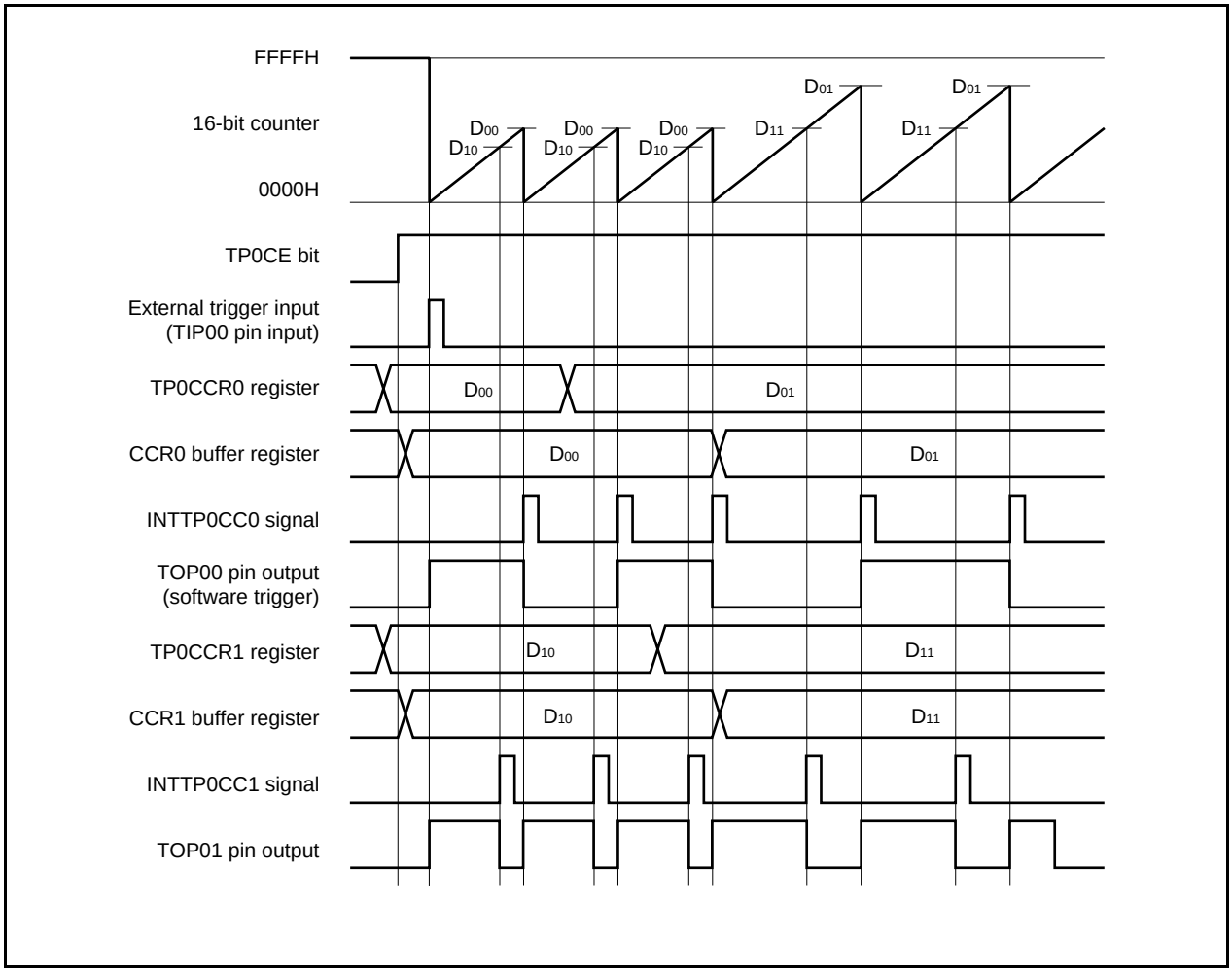

**Remark** a = 0, 1

### (2) External trigger pulse output mode operation timing

#### (a) Note on changing pulse width during operation

To change the PWM waveform while the counter is operating, write the TP0CCR1 register last.

Rewrite the TP0CCRa register after writing the TP0CCR1 register after the INTTP0CC0 signal is detected.



In order to transfer data from the TP0CCRa register to the CCRa buffer register, the TP0CCR1 register must be written.

To change both the cycle and active level width of the PWM waveform at this time, first set the cycle to the TP0CCR0 register and then set the active level width to the TP0CCR1 register.

To change only the cycle of the PWM waveform, first set the cycle to the TP0CCR0 register, and then write the same value to the TP0CCR1 register.

To change only the active level width (duty factor) of the PWM waveform, only the TP0CCR1 register has to be set.

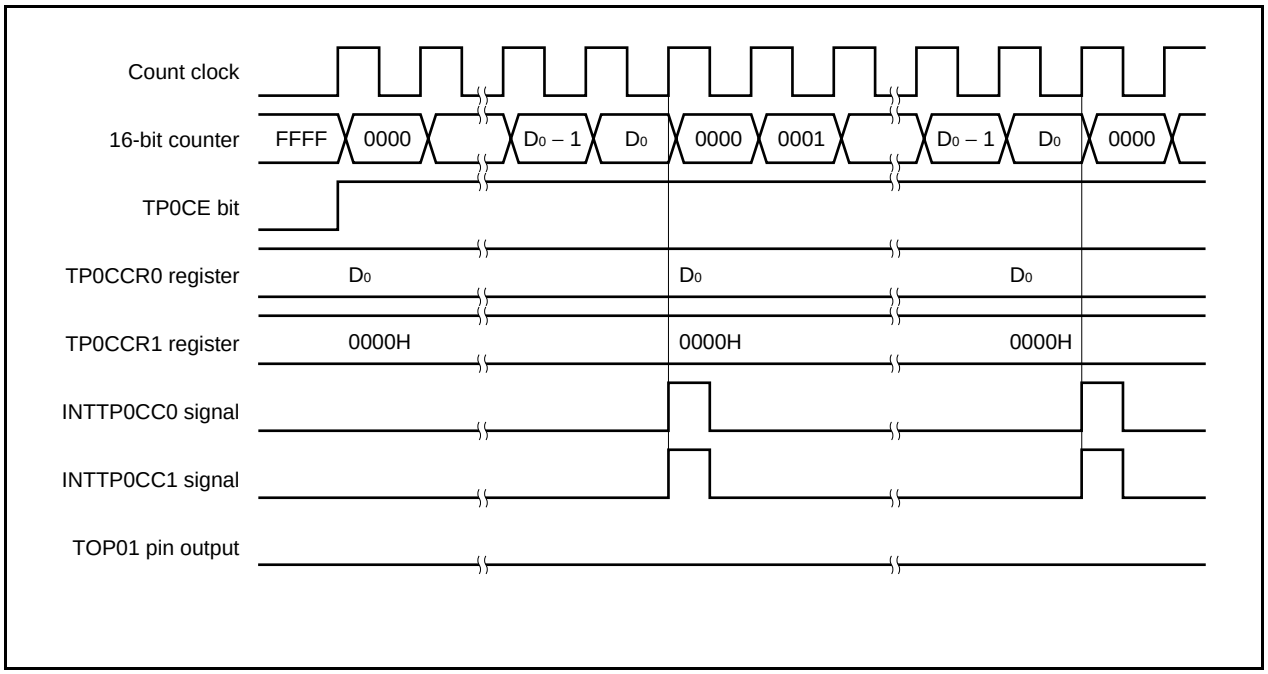
After data is written to the TP0CCR1 register, the value written to the TP0CCRa register is transferred to the CCRa buffer register in synchronization with clearing of the 16-bit counter, and is used as the value compared with the 16-bit counter.

To write the TP0CCR0 or TP0CCR1 register again after writing the TP0CCR1 register once, do so after the INTTP0CC0 signal is generated. Otherwise, the value of the CCRa buffer register may become undefined because the timing of transferring data from the TP0CCRa register to the CCRa buffer register conflicts with writing the TP0CCRa register.

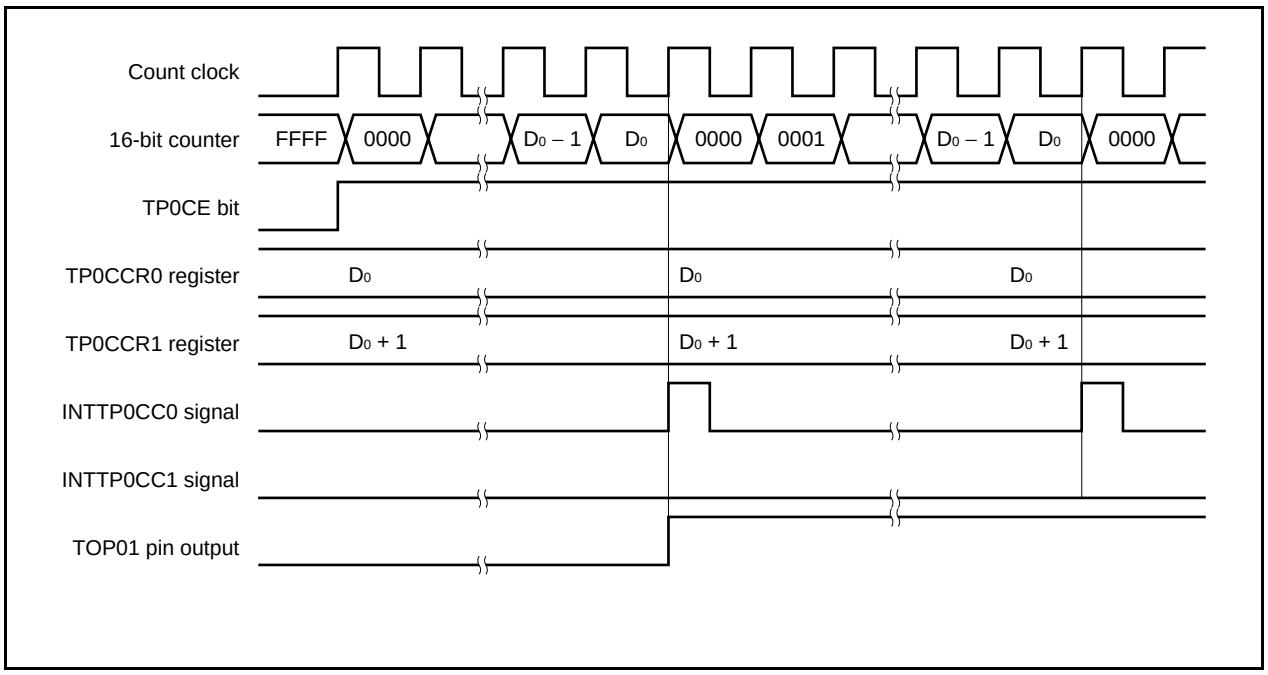
**Remark** a = 0, 1

**(b) 0%/100% output of PWM waveform**

To output a 0% waveform, clear the TP0CCR1 register to 0000H. If the set value of the TP0CCR0 register is FFFFH, the INTTP0CC1 signal is generated periodically.

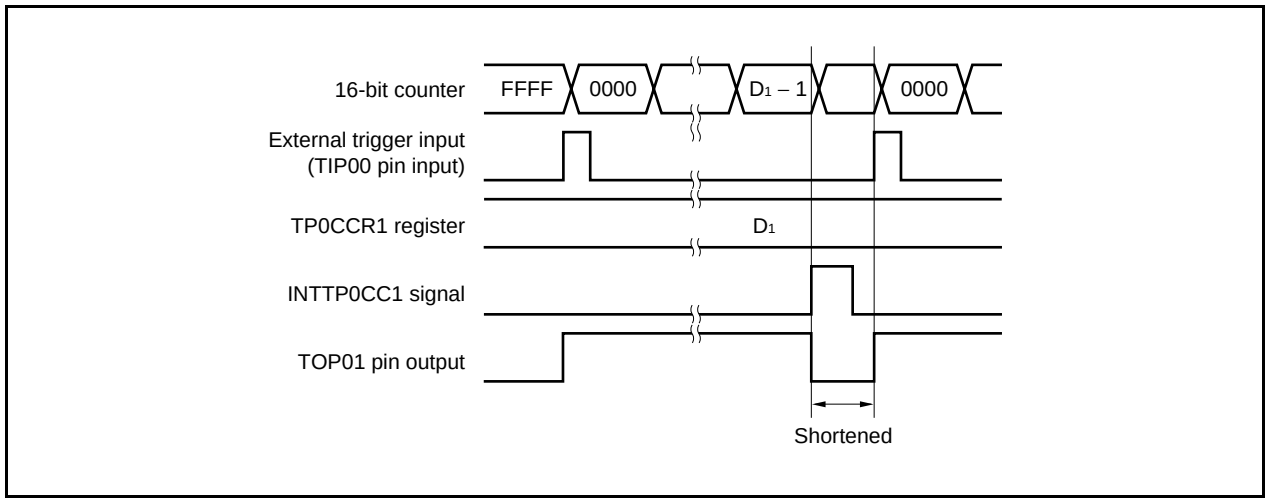



To output a 100% waveform, set a value of (set value of TP0CCR0 register + 1) to the TP0CCR1 register. If the set value of the TP0CCR0 register is FFFFH, 100% output cannot be produced.

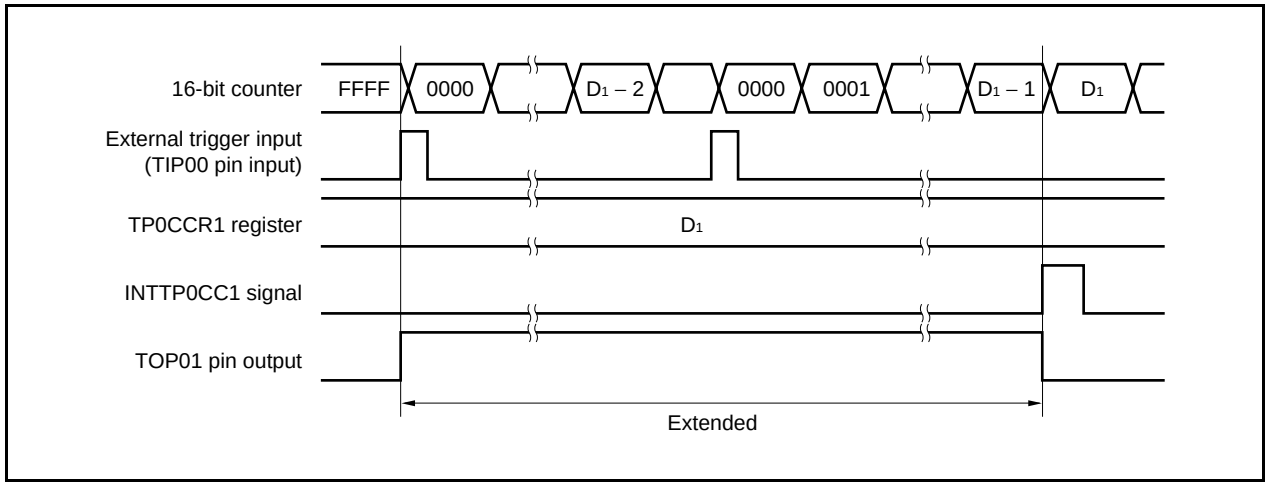


**(c) Conflict between trigger detection and match with TP0CCR1 register**

If the trigger is detected immediately after the INTTP0CC1 signal is generated, the 16-bit counter is immediately cleared to 0000H, the output signal of the TOP01 pin is asserted, and the counter continues counting. Consequently, the inactive period of the PWM waveform is shortened.

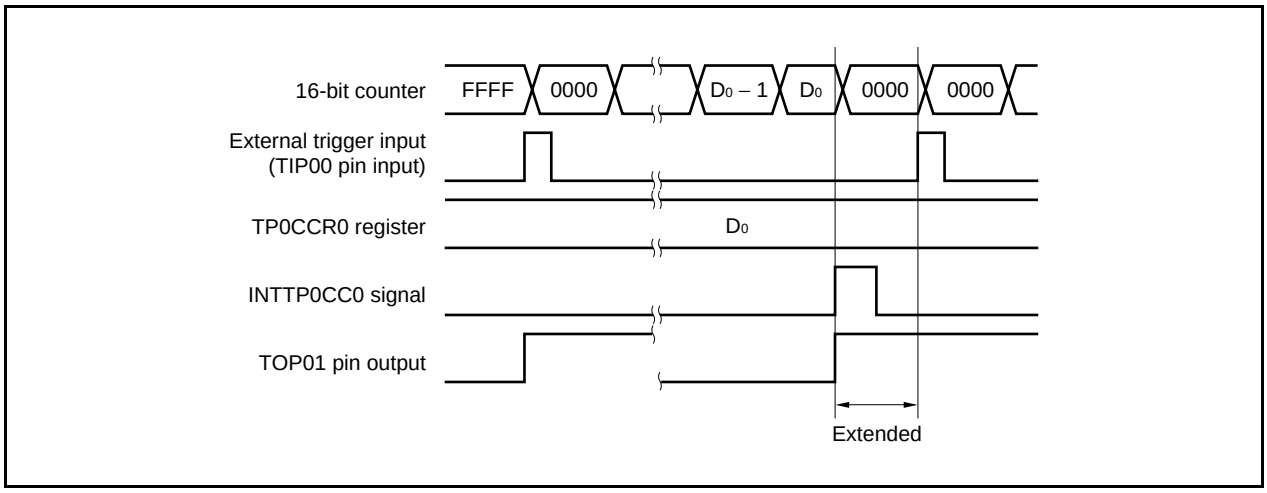



If the trigger is detected immediately before the INTTP0CC1 signal is generated, the INTTP0CC1 signal is not generated, and the 16-bit counter is cleared to 0000H and continues counting. The output signal of the TOP01 pin remains active. Consequently, the active period of the PWM waveform is extended.

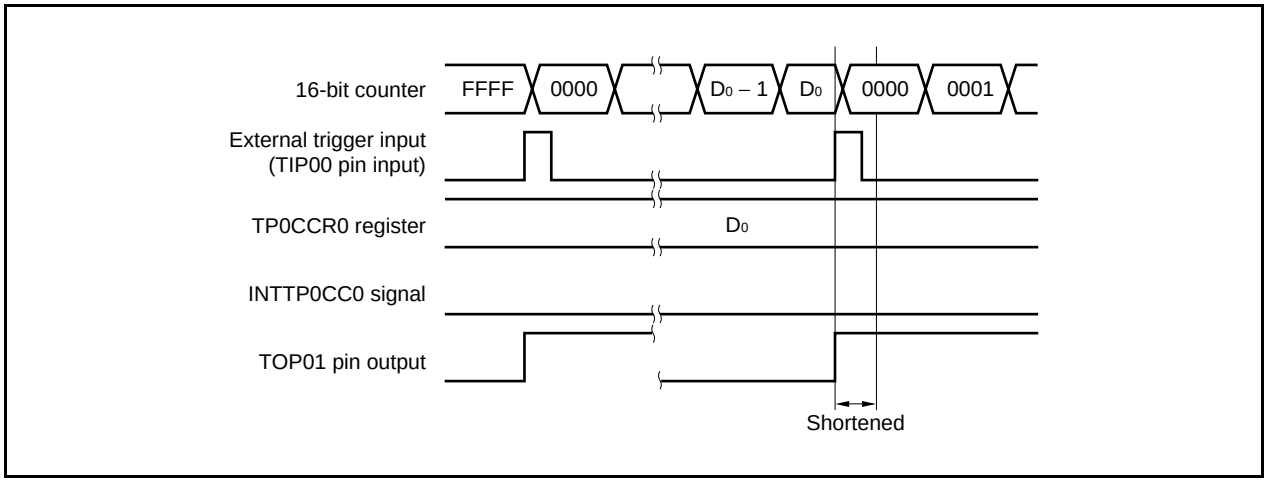


**(d) Conflict between trigger detection and match with TP0CCR0 register**

If the trigger is detected immediately after the INTTP0CC0 signal is generated, the 16-bit counter is cleared to 0000H and continues counting up. Therefore, the active period of the TOP01 pin is extended by time from generation of the INTTP0CC0 signal to trigger detection.

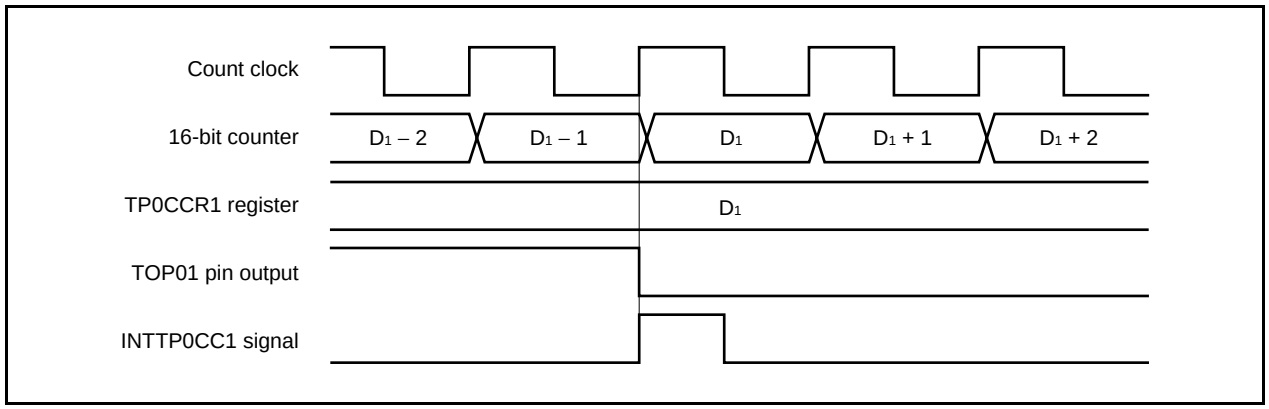



If the trigger is detected immediately before the INTTP0CC0 signal is generated, the INTTP0CC0 signal is not generated. The 16-bit counter is cleared to 0000H, the TOP01 pin is asserted, and the counter continues counting. Consequently, the inactive period of the PWM waveform is shortened.



**(e) Generation timing of compare match interrupt request signal (INTTP0CC1)**

The timing of generation of the INTTP0CC1 signal in the external trigger pulse output mode differs from the timing of other INTTP0CC1 signals; the INTTP0CC1 signal is generated when the count value of the 16-bit counter matches the value of the TP0CCR1 register.




Usually, the INTTP0CC1 signal is generated in synchronization with the next count up, after the count value of the 16-bit counter matches the value of the TP0CCR1 register.

In the external trigger pulse output mode, however, it is generated one clock earlier. This is because the timing is changed to match the timing of changing the output signal of the TOP01 pin.

### 7.5.4 One-shot pulse output mode (TP0MD2 to TP0MD0 bits = 011)

In the one-shot pulse output mode, 16-bit timer/event counter P waits for a trigger when the TP0CTL0.TP0CE bit is set to 1. When the valid edge of an external trigger input is detected, 16-bit timer/event counter P starts counting, and outputs a one-shot pulse from the TOP01 pin.

Instead of the external trigger, a software trigger can also be generated to output the pulse. When the software trigger is used, the TOP00 pin outputs the active level while the 16-bit counter is counting, and the inactive level when the counter is stopped (waiting for a trigger).

**Figure 7-20. Configuration in One-Shot Pulse Output Mode**

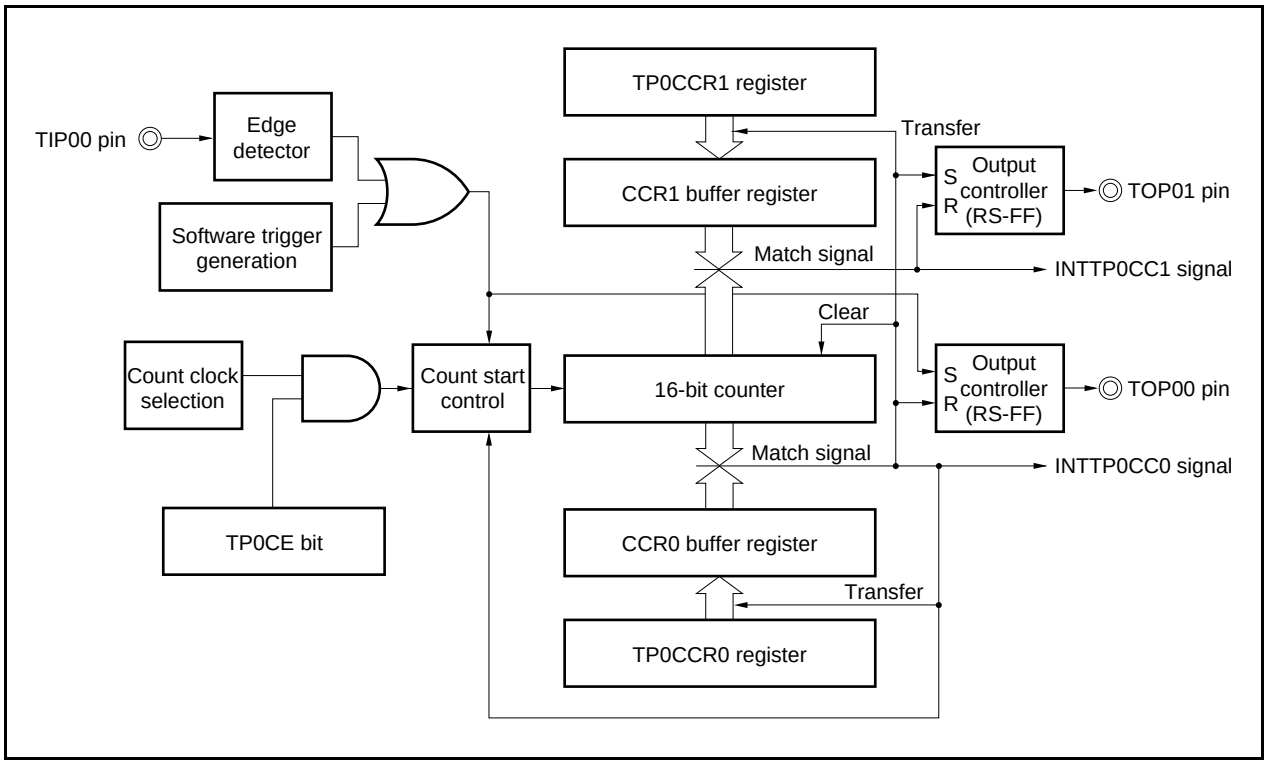
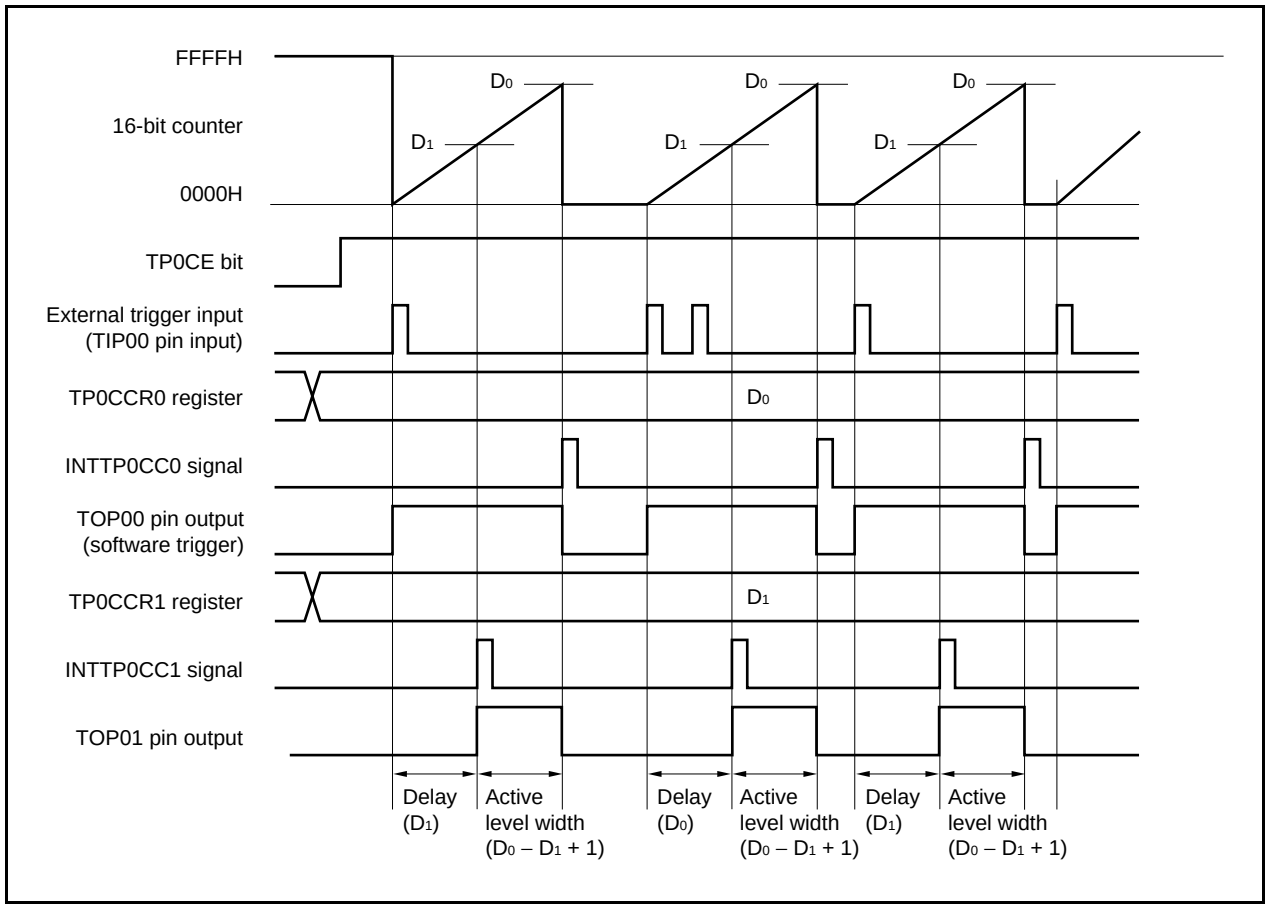


**Figure 7-21. Basic Timing in One-Shot Pulse Output Mode**




When the TP0CE bit is set to 1, 16-bit timer/event counter P waits for a trigger. When the trigger is generated, the 16-bit counter is cleared from FFFFH to 0000H, starts counting, and outputs a one-shot pulse from the TOP01 pin. After the one-shot pulse is output, the 16-bit counter is set to FFFFH, stops counting, and waits for a trigger. If a trigger is generated again while the one-shot pulse is being output, it is ignored.

The output delay period and active level width of the one-shot pulse can be calculated as follows.

Output delay period = (Set value of TP0CCR1 register) × Count clock cycle

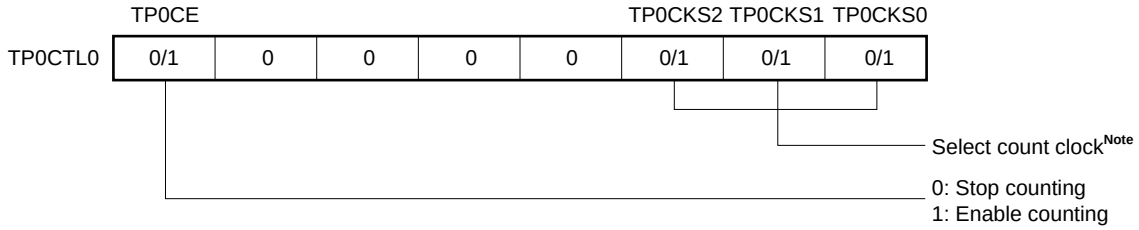
Active level width = (Set value of TP0CCR0 register − Set value of TP0CCR1 register + 1) × Count clock cycle

The compare match interrupt request signal INTTP0CC0 is generated when the 16-bit counter counts after its count value matches the value of the CCR0 buffer register. The compare match interrupt request signal INTTP0CC1 is generated when the count value of the 16-bit counter matches the value of the CCR1 buffer register.

The valid edge of an external trigger input or setting the software trigger (TP0CTL1.TP0EST bit) to 1 is used as the trigger.

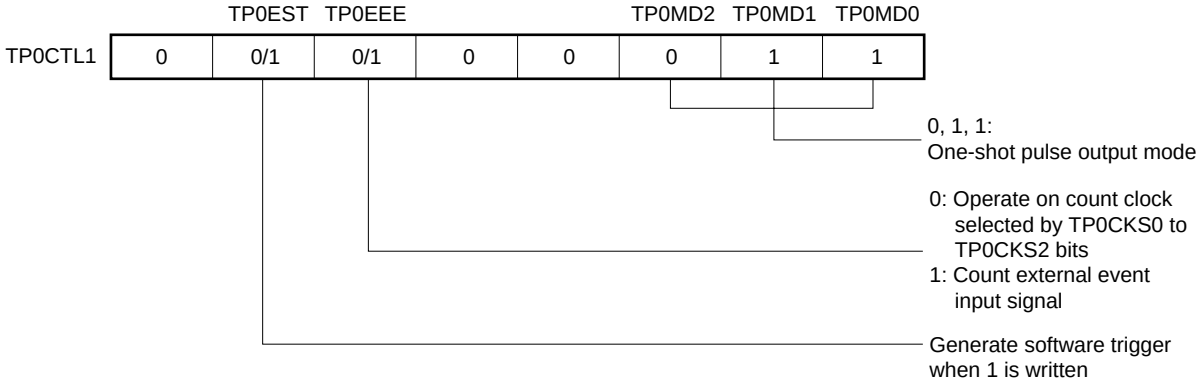
**Figure 7-22. Setting of Registers in One-Shot Pulse Output Mode (1/2)**

**(a) TMP0 control register 0 (TP0CTL0)**

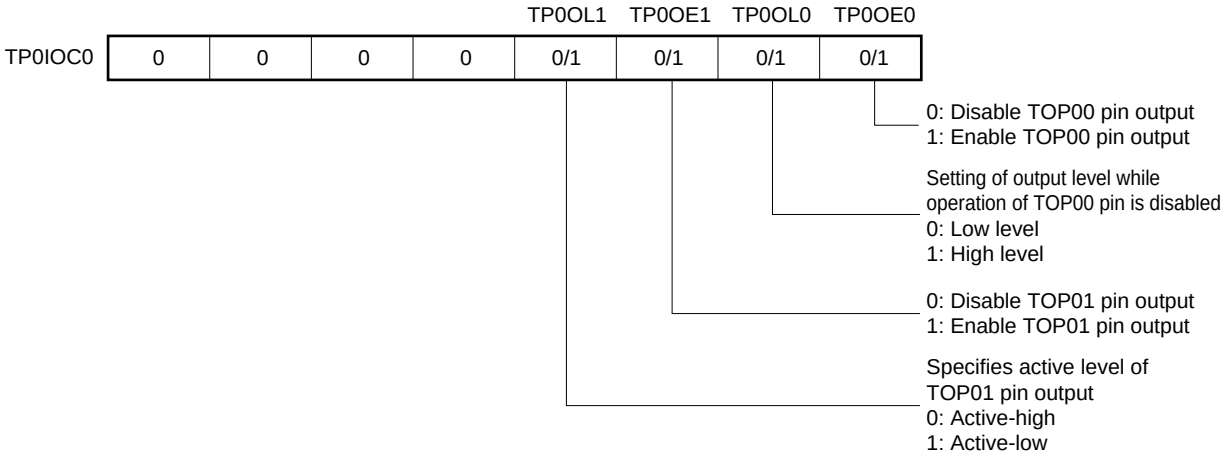



**Note** The setting is invalid when the TP0CTL1.TP0EEE bit = 1.

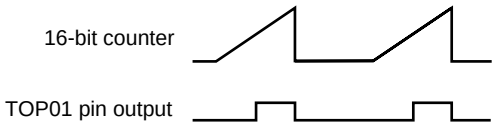
**(b) TMP0 control register 1 (TP0CTL1)**




**(c) TMP0 I/O control register 0 (TP0IOC0)**




• When TP0OL1 bit = 0




• When TP0OL1 bit = 1

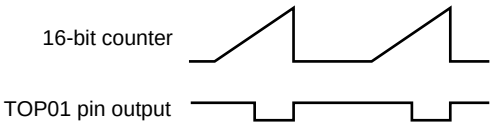
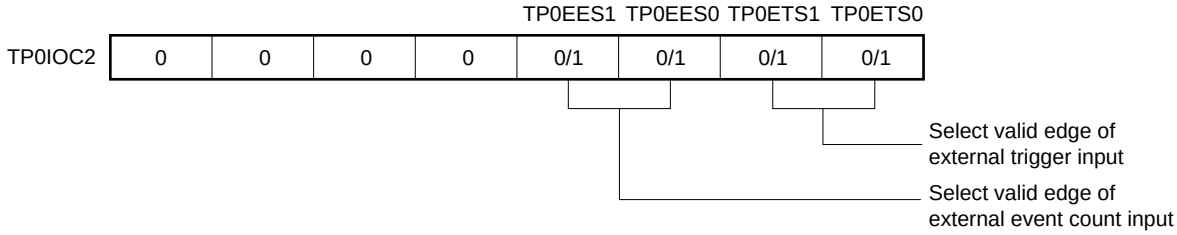


**Figure 7-22. Setting of Registers in One-Shot Pulse Output Mode (2/2)**

**(d) TMP0 I/O control register 2 (TP0IOC2)**




**(e) TMP0 counter read buffer register (TP0CNT)**

The value of the 16-bit counter can be read by reading the TP0CNT register.

**(f) TMP0 capture/compare registers 0 and 1 (TP0CCR0 and TP0CCR1)**

If $D_0$ is set to the TP0CCR0 register and $D_1$ to the TP0CCR1 register, the active level width and output delay period of the one-shot pulse are as follows.

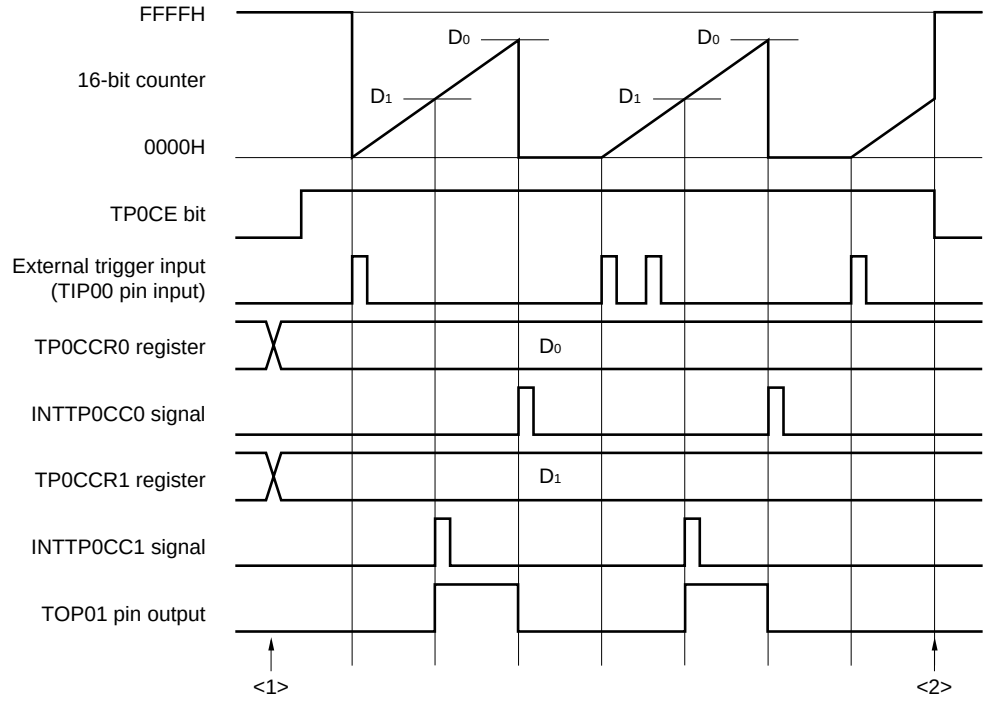
Active level width = $(D_1 - D_0 + 1) \times$ Count clock cycle

Output delay period = $D_1 \times$ Count clock cycle

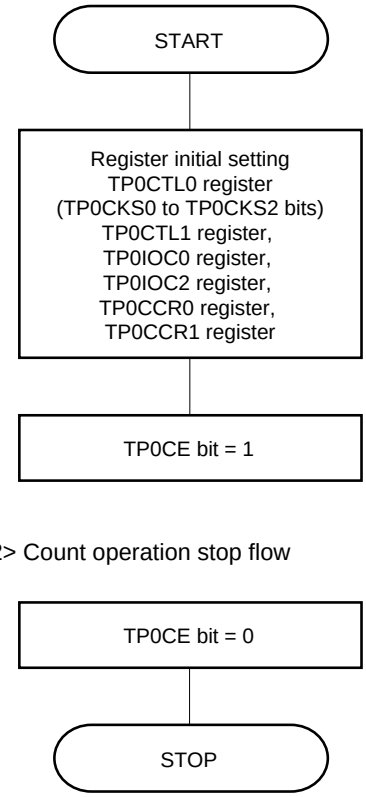
**Remark** TMP0 I/O control register 1 (TP0IOC1) and TMP0 option register 0 (TP0OPT0) are not used in the one-shot pulse output mode.

### (1) Operation flow in one-shot pulse output mode

**Figure 7-23. Software Processing Flow in One-Shot Pulse Output Mode**




<1> Count operation start flow




Initial setting of these registers is performed before setting the TP0CE bit to 1.

The TP0CKS0 to TP0CKS2 bits can be set at the same time when counting has been started (TP0CE bit = 1).
Trigger wait status

<2> Count operation stop flow



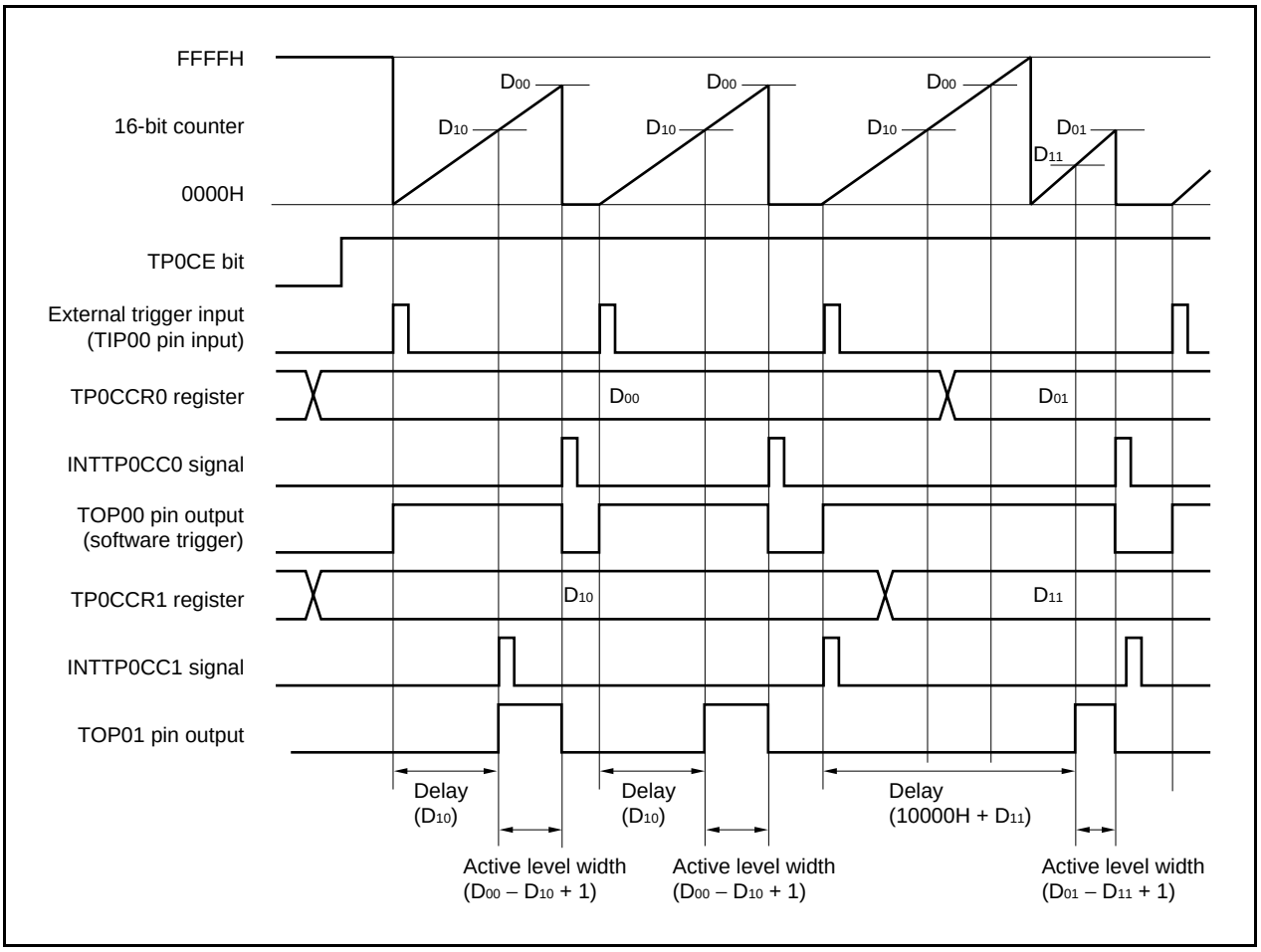

Count operation is stopped

### (2) Operation timing in one-shot pulse output mode

#### (a) Note on rewriting TP0CCRa register

To change the set value of the TP0CCRa register to a smaller value, stop counting once, and then change the set value.

If the value of the TP0CCRa register is rewritten to a smaller value during counting, the 16-bit counter may overflow.



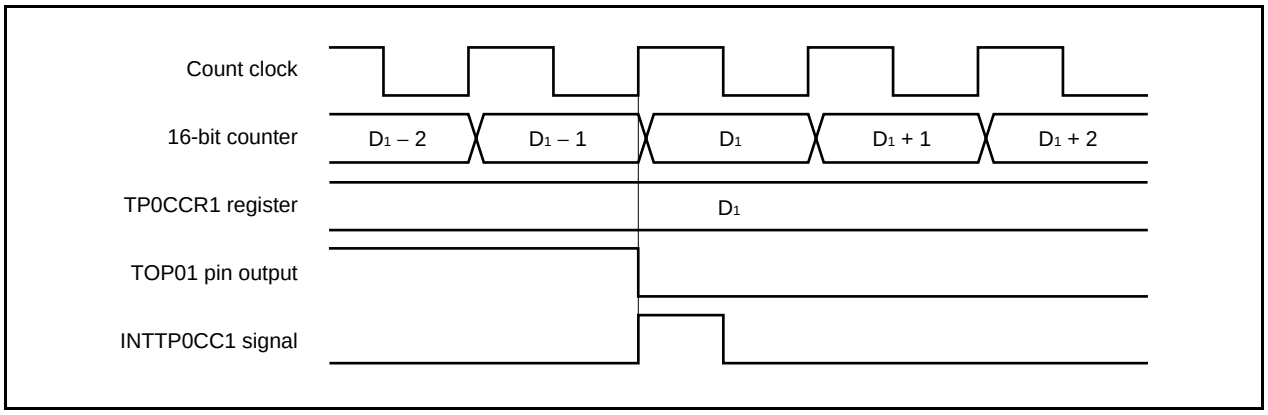

When the TP0CCR0 register is rewritten from $D_{00}$ to $D_{01}$ and the TP0CCR1 register from $D_{10}$ to $D_{11}$ where $D_{00} > D_{01}$ and $D_{10} > D_{11}$, if the TP0CCR1 register is rewritten when the count value of the 16-bit counter is greater than $D_{11}$ and less than $D_{10}$ and if the TP0CCR0 register is rewritten when the count value is greater than $D_{01}$ and less than $D_{00}$, each set value is reflected as soon as the register has been rewritten and compared with the count value. The counter counts up to FFFFH and then counts up again from 0000H. When the count value matches $D_{11}$, the counter generates the INTTP0CC1 signal and asserts the TOP01 pin. When the count value matches $D_{01}$, the counter generates the INTTP0CC0 signal, deasserts the TOP01 pin, and stops counting.

Therefore, the counter may output a pulse with a delay period or active period different from that of the one-shot pulse that is originally expected.

**Remark** a = 0, 1

**(b) Generation timing of compare match interrupt request signal (INTTP0CC1)**

The generation timing of the INTTP0CC1 signal in the one-shot pulse output mode is different from other INTTP0CC1 signals; the INTTP0CC1 signal is generated when the count value of the 16-bit counter matches the value of the TP0CCR1 register.




Usually, the INTTP0CC1 signal is generated when the 16-bit counter counts up next time after its count value matches the value of the TP0CCR1 register.

In the one-shot pulse output mode, however, it is generated one clock earlier. This is because the timing is changed to match the change timing of the TOP01 pin.

### 7.5.5 PWM output mode (TP0MD2 to TP0MD0 bits = 100)

In the PWM output mode, a PWM waveform is output from the TOP01 pin when the TP0CTL0.TP0CE bit is set to 1. In addition, a pulse with one cycle of the PWM waveform as half its cycle is output from the TOP00 pin.

**Figure 7-24. Configuration in PWM Output Mode**

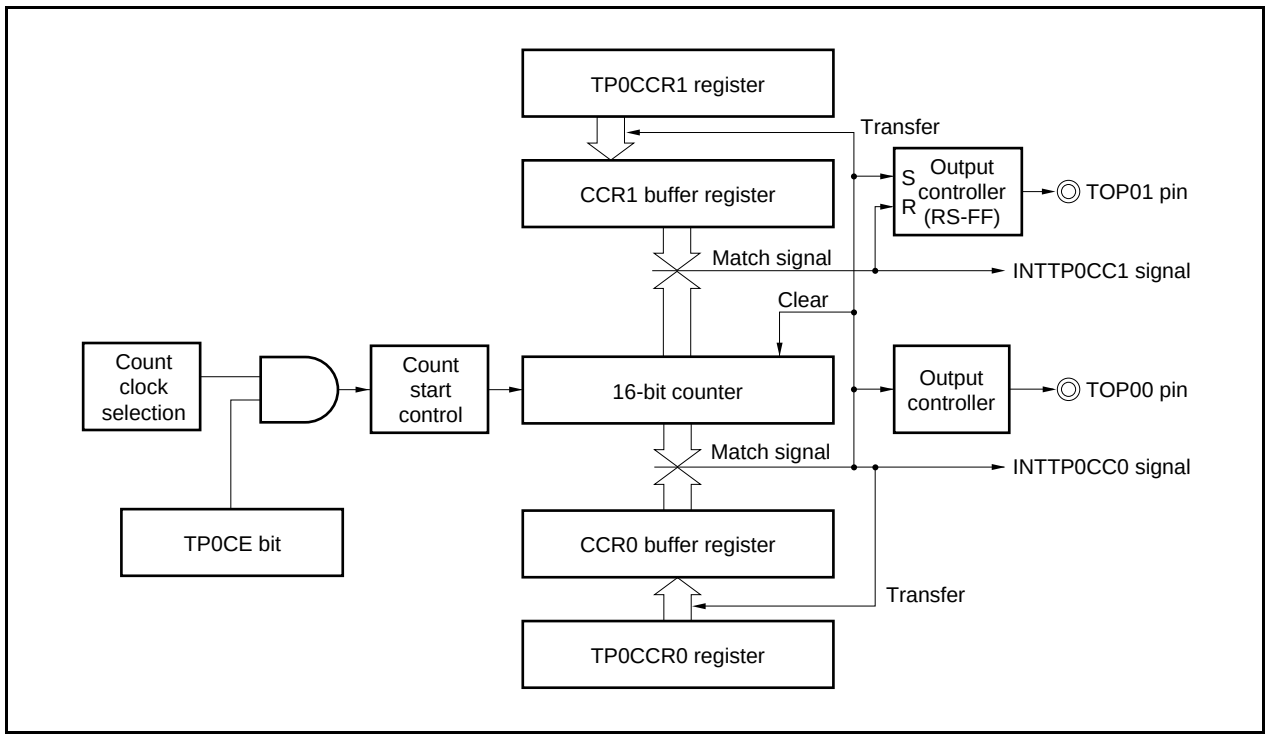
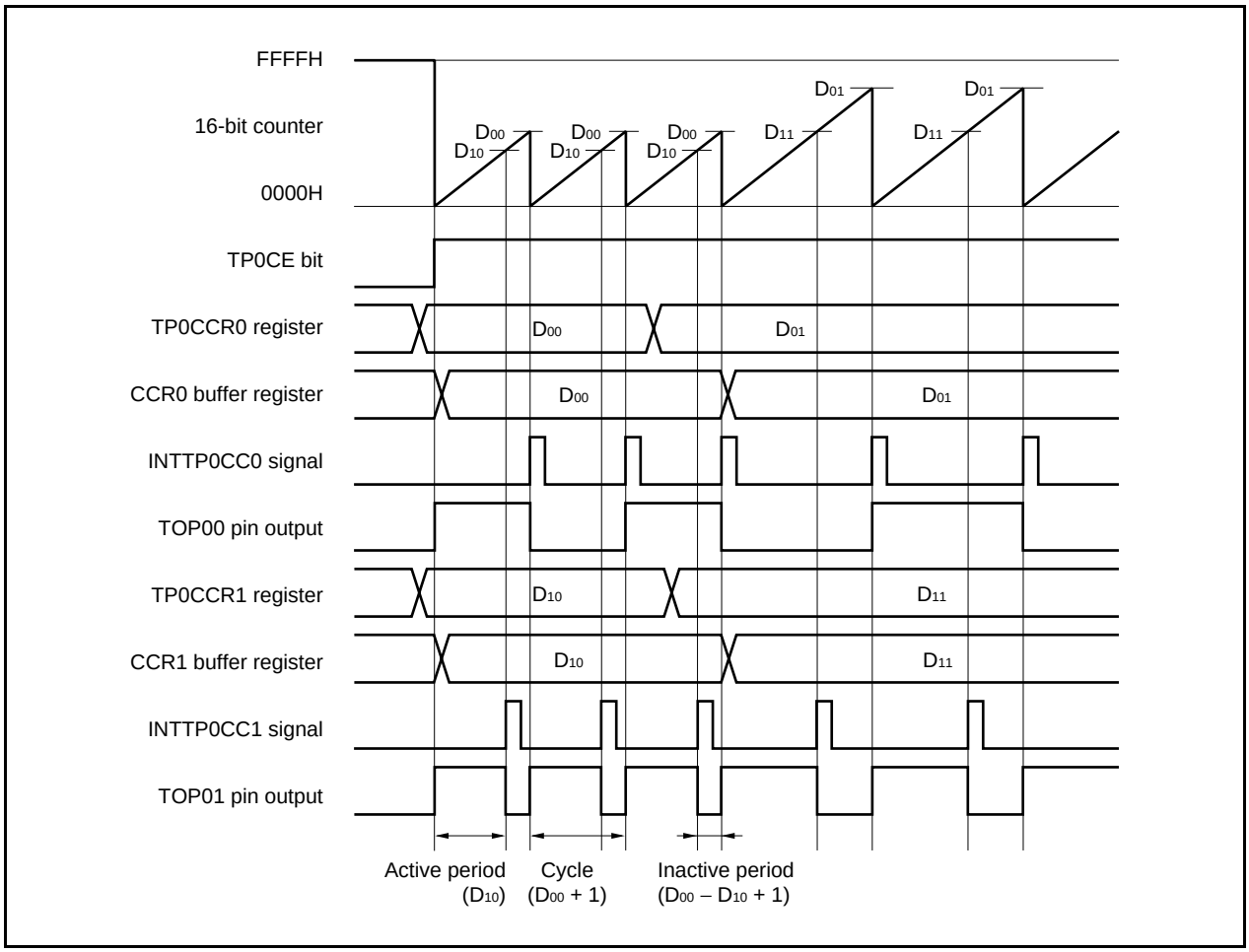


**Figure 7-25. Basic Timing in PWM Output Mode**




When the TP0CE bit is set to 1, the 16-bit counter is cleared from FFFFH to 0000H, starts counting, and outputs a PWM waveform from the TOP01 pin.

The active level width, cycle, and duty factor of the PWM waveform can be calculated as follows.

Active level width = (Set value of TP0CCR1 register ) × Count clock cycle
Cycle = (Set value of TP0CCR0 register + 1) × Count clock cycle
Duty factor = (Set value of TP0CCR1 register)/(Set value of TP0CCR0 register + 1)

The PWM waveform can be changed by rewriting the TP0CCRa register while the counter is operating. The newly written value is reflected when the count value of the 16-bit counter matches the value of the CCR0 buffer register and the 16-bit counter is cleared to 0000H.

The compare match interrupt request signal INTTP0CC0 is generated when the 16-bit counter counts next time after its count value matches the value of the CCR0 buffer register, and the 16-bit counter is cleared to 0000H. The compare match interrupt request signal INTTP0CC1 is generated when the count value of the 16-bit counter matches the value of the CCR1 buffer register.

The value set to the TP0CCRa register is transferred to the CCRa buffer register when the count value of the 16-bit counter matches the value of the CCRa buffer register and the 16-bit counter is cleared to 0000H.

**Remark** a = 0, 1

**Figure 7-26. Register Setting in PWM Output Mode (1/2)**

**(a) TMP0 control register 0 (TP0CTL0)**

| | TP0CE | | | | | TP0CKS2 | TP0CKS1 | TP0CKS0 |
|---|---|---|---|---|---|---|---|---|
| TP0CTL0 | 0/1 | 0 | 0 | 0 | 0 | 0/1 | 0/1 | 0/1 |

Select count clock[Note]

0: Stop counting
1: Enable counting

**Note** The setting is invalid when the TP0CTL1.TP0EEE bit = 1.

**(b) TMP0 control register 1 (TP0CTL1)**

| | | TP0EST | TP0EEE | | | | TP0MD2 | TP0MD1 | TP0MD0 |
|---|---|---|---|---|---|---|---|---|---|
| TP0CTL1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

1, 0, 0:
PWM output mode

**(c) TMP0 I/O control register 0 (TP0IOC0)**

| | | | | | TP0OL1 | TP0OE1 | TP0OL0 | TP0OE0 |
|---|---|---|---|---|---|---|---|---|
| TP0IOC0 | 0 | 0 | 0 | 0 | 0/1 | 0/1 | 0/1 | 0/1 |

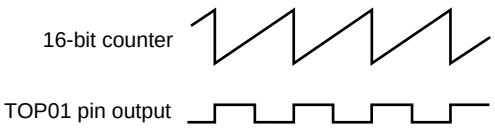
0: Disable TOP00 pin output
1: Enable TOP00 pin output

Setting of output level while operation of TOP00 pin is disabled
0: Low level
1: High level

0: Disable TOP01 pin output
1: Enable TOP01 pin output

Specifies active level of TOP01 pin output
0: Active-high
1: Active-low

- When TP0OL1 bit = 0




- When TP0OL1 bit = 1

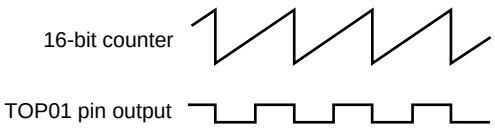
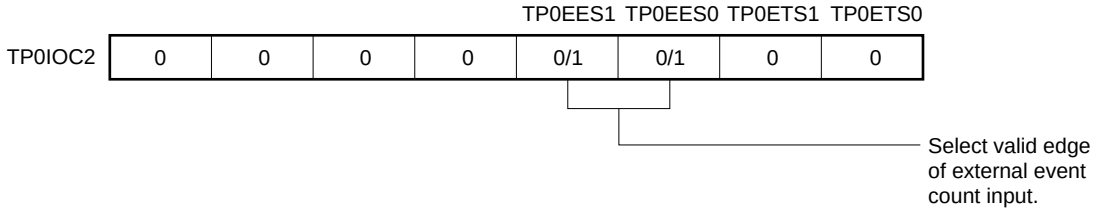


**Figure 7-26. Register Setting in PWM Output Mode (2/2)**

**(d) TMP0 I/O control register 2 (TP0IOC2)**




**(e) TMP0 counter read buffer register (TP0CNT)**

The value of the 16-bit counter can be read by reading the TP0CNT register.

**(f) TMP0 capture/compare registers 0 and 1 (TP0CCR0 and TP0CCR1)**

If $D_0$ is set to the TP0CCR0 register and $D_1$ to the TP0CCR1 register, the cycle and active level of the PWM waveform are as follows.

Cycle = $(D_0 + 1) \times$ Count clock cycle
Active level width = $D_1 \times$ Count clock cycle

**Remark** TMP0 I/O control register 1 (TP0IOC1) and TMP0 option register 0 (TP0OPT0) are not used in the PWM output mode.

**(1) Operation flow in PWM output mode**

**Figure 7-27. Software Processing Flow in PWM Output Mode (1/2)**

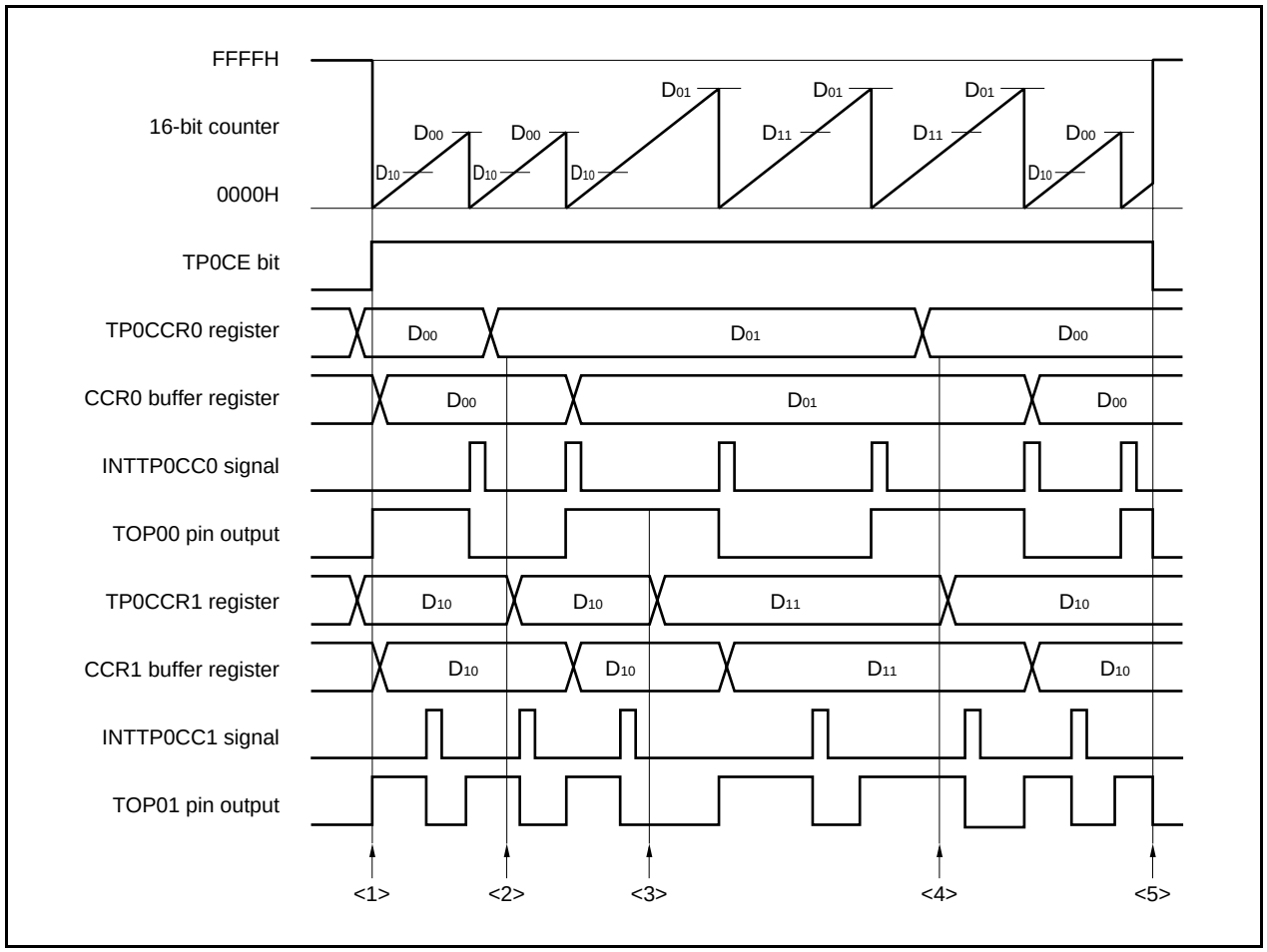
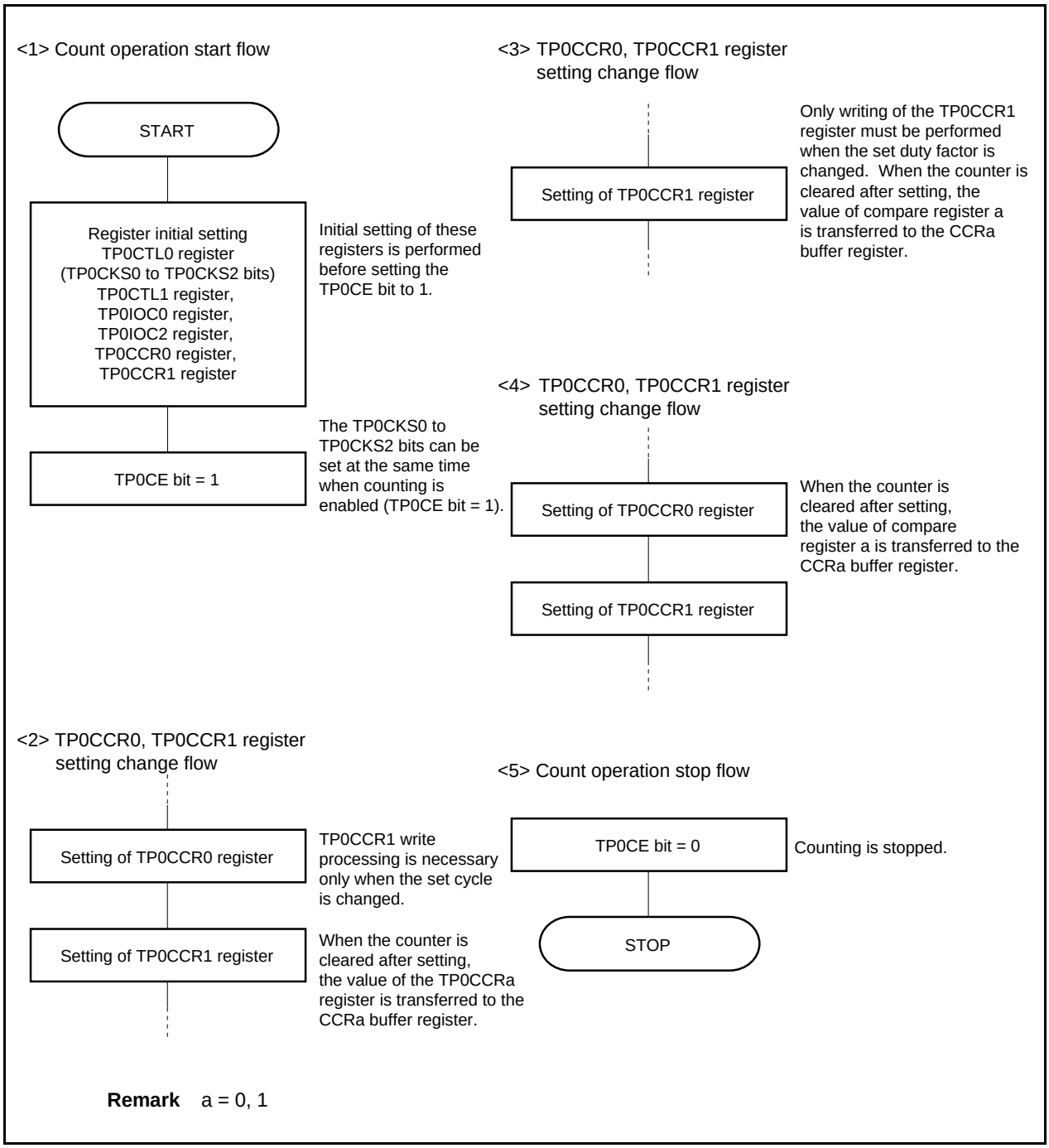


**Figure 7-27. Software Processing Flow in PWM Output Mode (2/2)**



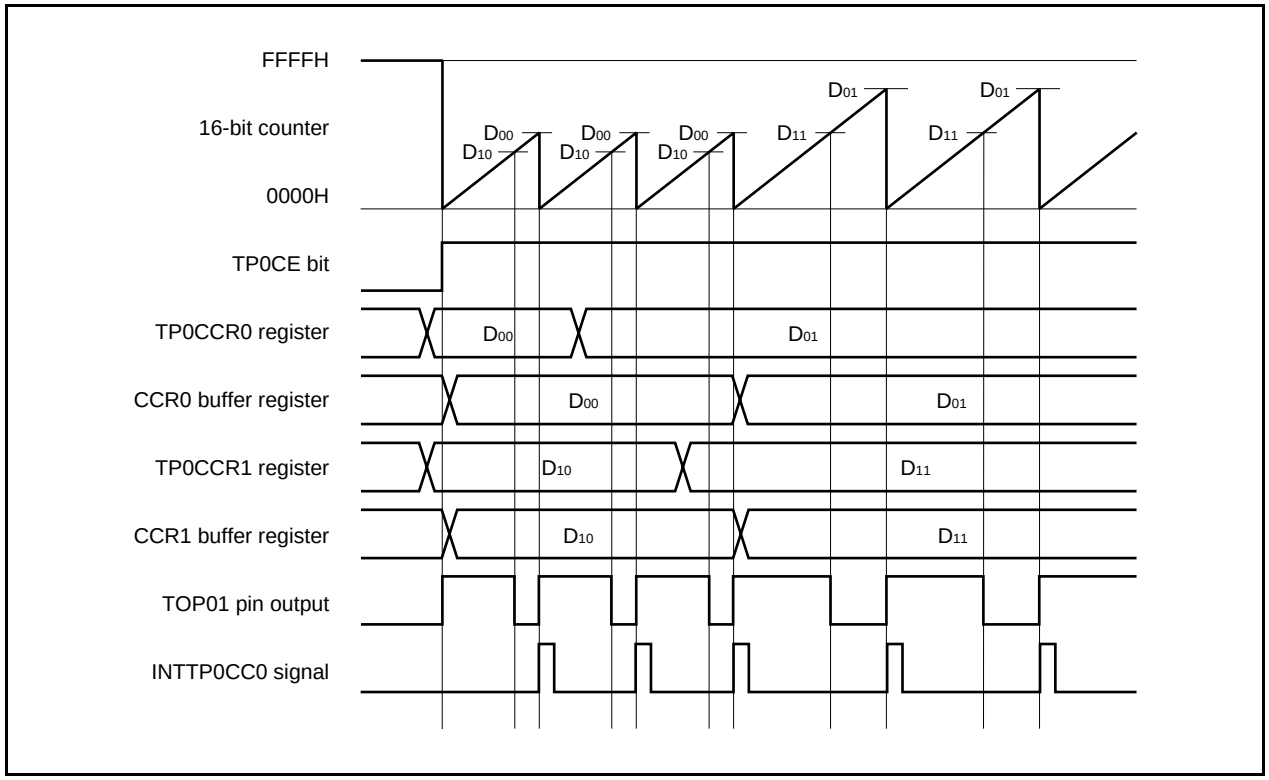

**Remark** a = 0, 1

### (2) PWM output mode operation timing

#### (a) Changing pulse width during operation

To change the PWM waveform while the counter is operating, write the TP0CCR1 register last.

Rewrite the TP0CCRa register after writing the TP0CCR1 register after the INTTP0CC1 signal is detected.




To transfer data from the TP0CCRa register to the CCRa buffer register, the TP0CCR1 register must be written.

To change both the cycle and active level of the PWM waveform at this time, first set the cycle to the TP0CCR0 register and then set the active level to the TP0CCR1 register.

To change only the cycle of the PWM waveform, first set the cycle to the TP0CCR0 register, and then write the same value to the TP0CCR1 register.

To change only the active level width (duty factor) of the PWM waveform, only the TP0CCR1 register has to be set.

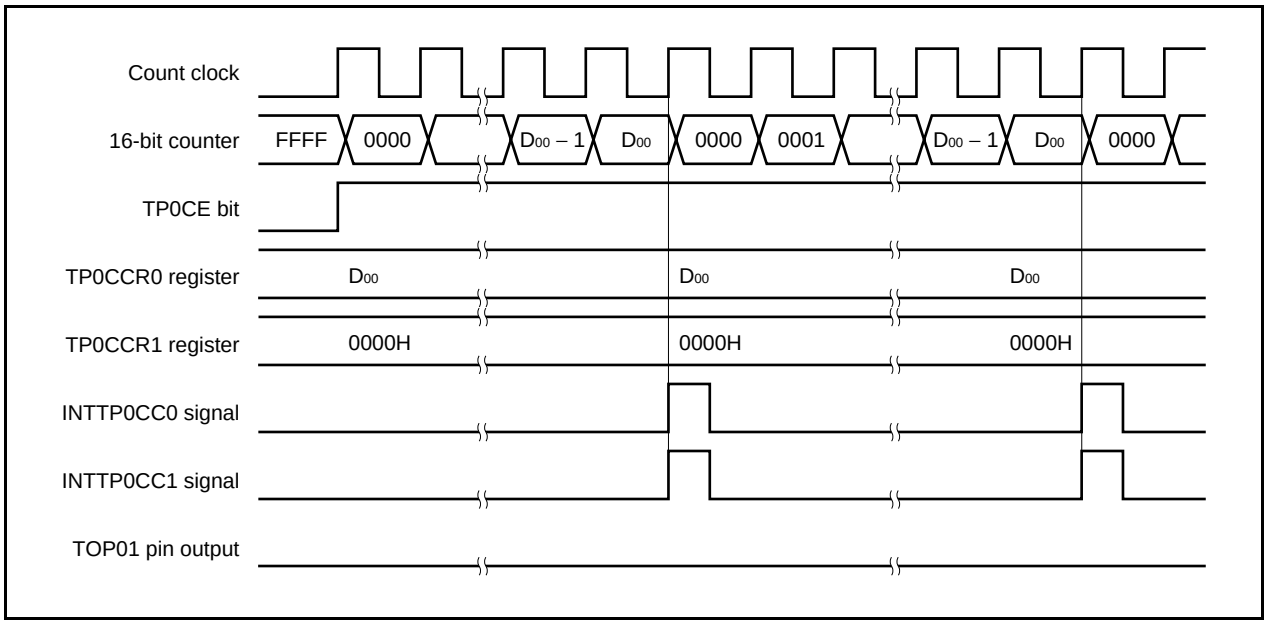
After data is written to the TP0CCR1 register, the value written to the TP0CCRa register is transferred to the CCRa buffer register in synchronization with clearing of the 16-bit counter, and is used as the value compared with the 16-bit counter.

To write the TP0CCR0 or TP0CCR1 register again after writing the TP0CCR1 register once, do so after the INTTP0CC0 signal is generated. Otherwise, the value of the CCRa buffer register may become undefined because the timing of transferring data from the TP0CCRa register to the CCRa buffer register conflicts with writing the TP0CCRa register.

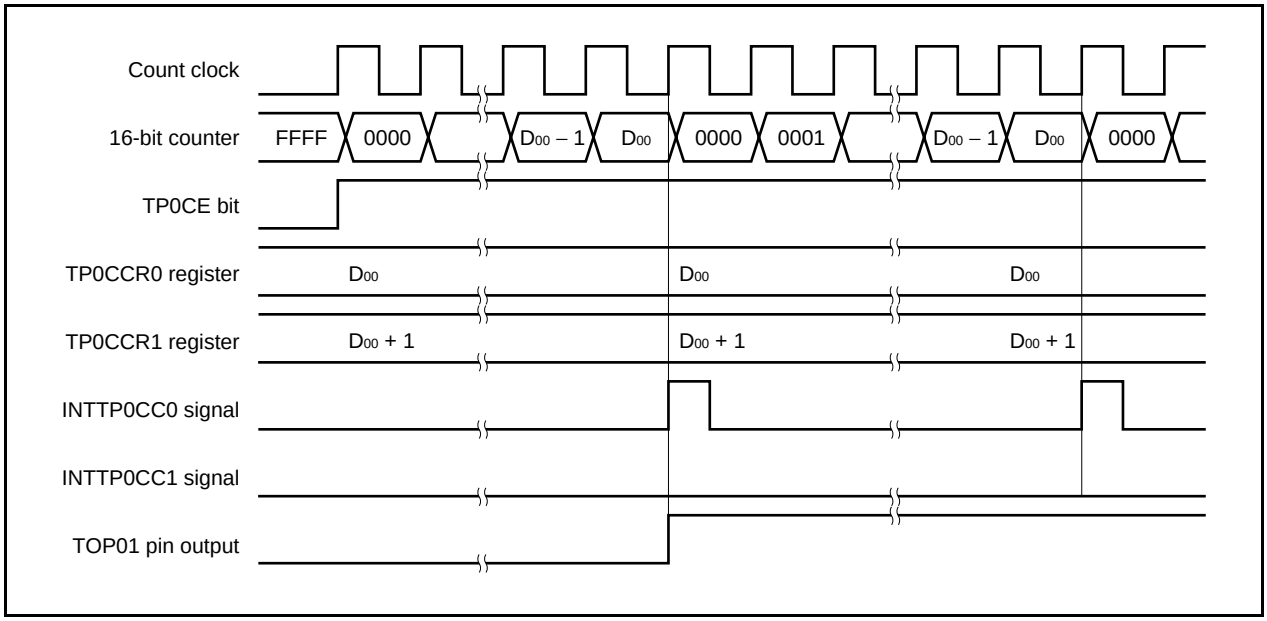
**Remark** a = 0, 1

**(b) 0%/100% output of PWM waveform**

To output a 0% waveform, set the TP0CCR1 register to 0000H. If the set value of the TP0CCR0 register is FFFFH, the INTTP0CC1 signal is generated periodically.

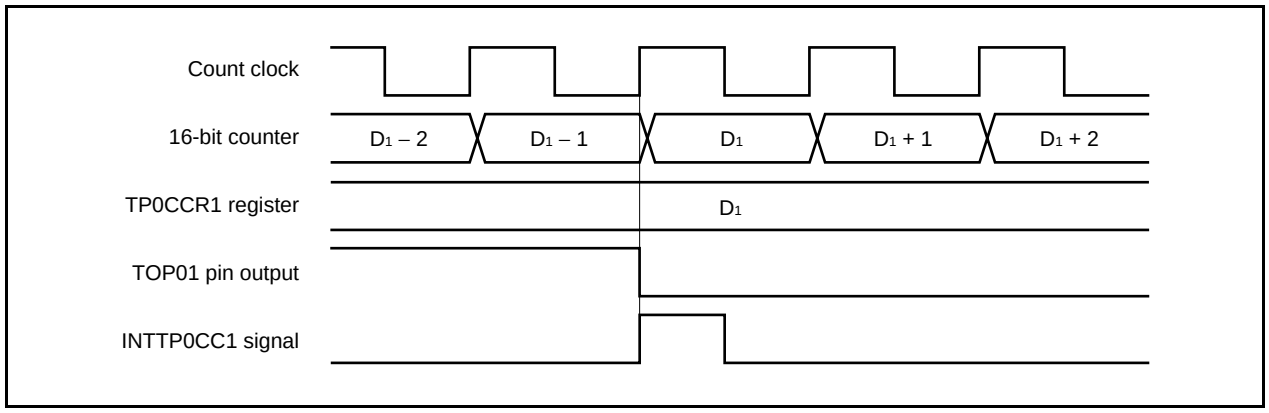



To output a 100% waveform, set a value of (set value of TP0CCR0 register + 1) to the TP0CCR1 register. If the set value of the TP0CCR0 register is FFFFH, 100% output cannot be produced.



**(c) Generation timing of compare match interrupt request signal (INTTP0CC1)**

The timing of generation of the INTTP0CC1 signal in the PWM output mode differs from the timing of other INTTP0CC1 signals; the INTTP0CC1 signal is generated when the count value of the 16-bit counter matches the value of the TP0CCR1 register.



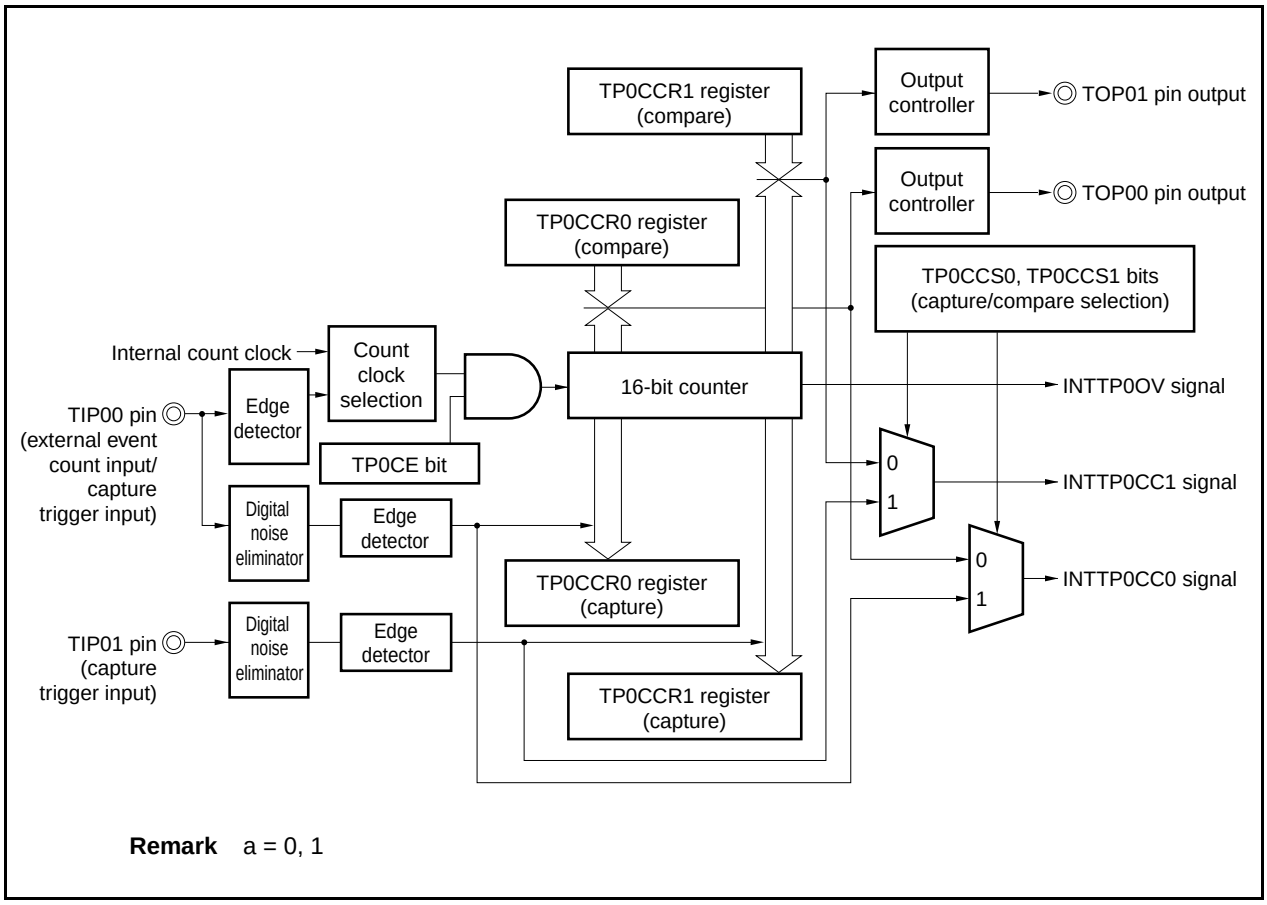

Usually, the INTTP0CC1 signal is generated in synchronization with the next counting up after the count value of the 16-bit counter matches the value of the TP0CCR1 register.

In the PWM output mode, however, it is generated one clock earlier. This is because the timing is changed to match the change timing of the output signal of the TOP01 pin.

### 7.5.6 Free-running timer mode (TP0MD2 to TP0MD0 bits = 101)

In the free-running timer mode, 16-bit timer/event counter P starts counting when the TP0CTL0.TP0CE bit is set to 1. At this time, the TP0CCRa register can be used as a compare register or a capture register, depending on the setting of the TP0OPT0.TP0CCS0 and TP0OPT0.TP0CCS1 bits.

**Figure 7-28. Configuration in Free-Running Timer Mode**



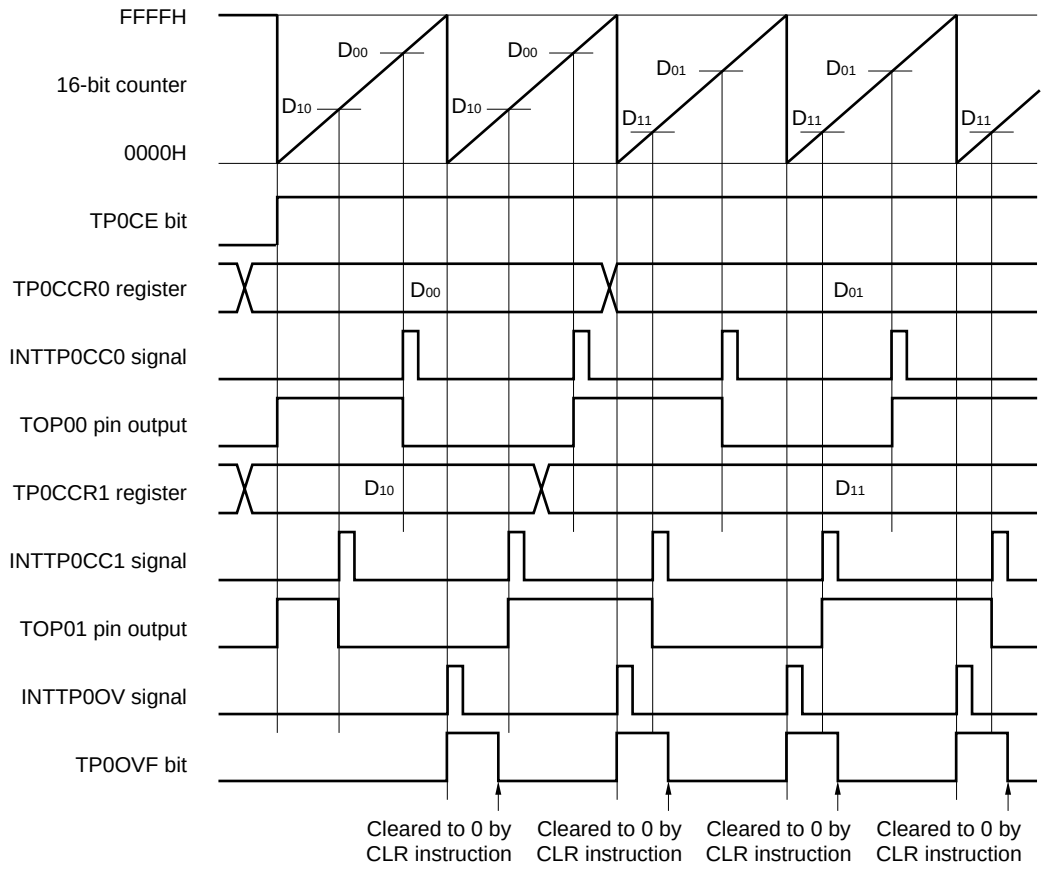

**Remark** a = 0, 1

When the TP0CE bit is set to 1, 16-bit timer/event counter P starts counting, and the output signals of the TOP00 and TOP01 pins are inverted. When the count value of the 16-bit counter later matches the set value of the TP0CCRa register, a compare match interrupt request signal (INTTP0CCa) is generated, and the output signal of the TOP0a pin is inverted.

The 16-bit counter continues counting in synchronization with the count clock. When it counts up to FFFFH, it generates an overflow interrupt request signal (INTTP0OV) at the next clock, is cleared to 0000H, and continues counting. At this time, the overflow flag (TP0OPT0.TP0OVF bit) is also set to 1. Clear the overflow flag to 0 by executing the CLR instruction by software.

The TP0CCRa register can be rewritten while the counter is operating. If it is rewritten, the new value is reflected at that time, and compared with the count value.

**Figure 7-29. Basic Timing in Free-Running Timer Mode (Compare Function)**




**Remark** a = 0, 1

When the TP0CE bit is set to 1, the 16-bit counter starts counting. When the valid edge input to the TIP0a pin is detected, the count value of the 16-bit counter is stored in the TP0CCRa register, and a capture interrupt request signal (INTTP0CCa) is generated.

The 16-bit counter continues counting in synchronization with the count clock. When it counts up to FFFFH, it generates an overflow interrupt request signal (INTTP0OV) at the next clock, is cleared to 0000H, and continues counting. At this time, the overflow flag (TP0OPT0.TP0OVF bit) is also set to 1. Clear the overflow flag to 0 by executing the CLR instruction by software.

**Figure 7-30. Basic Timing in Free-Running Timer Mode (Capture Function)**

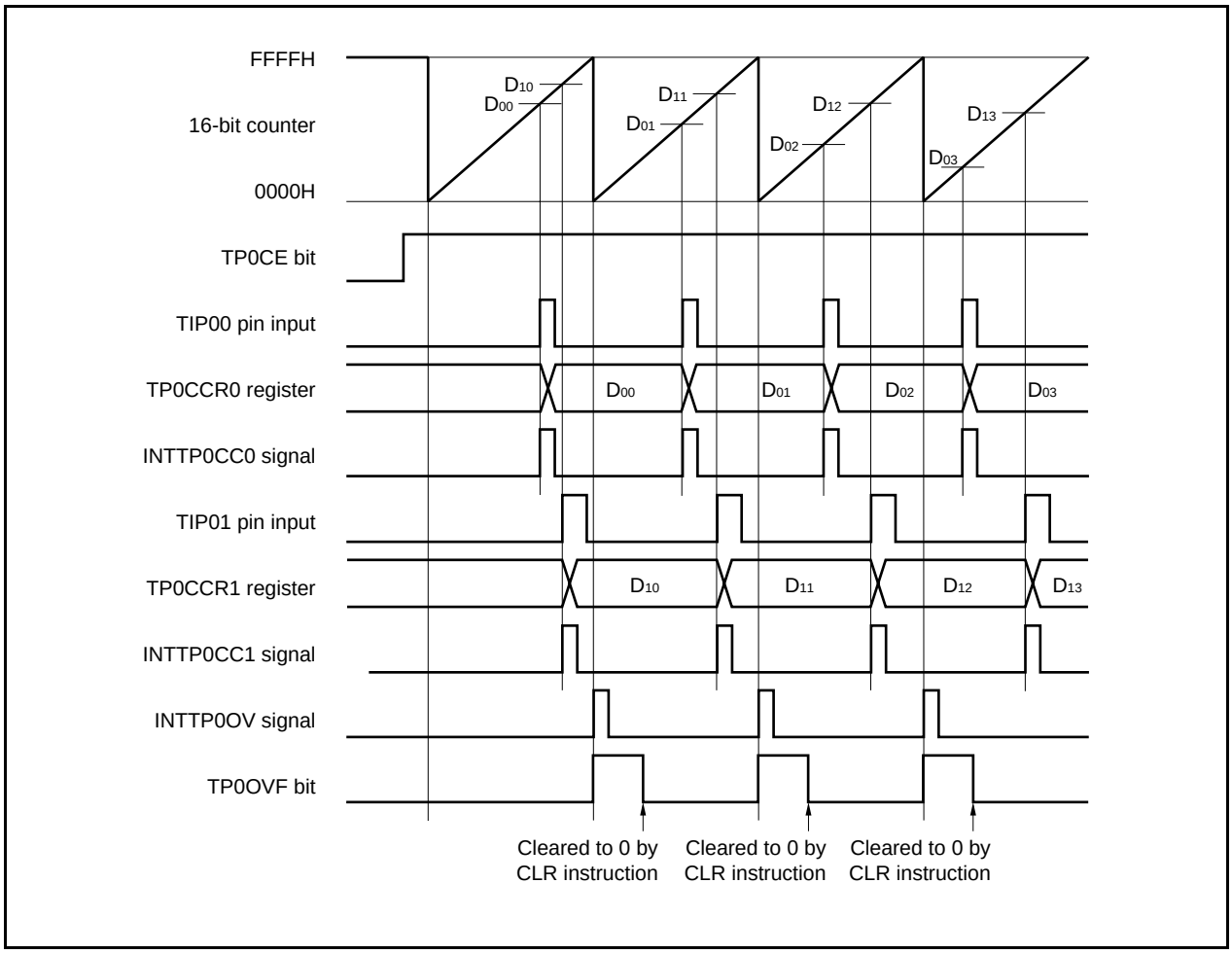
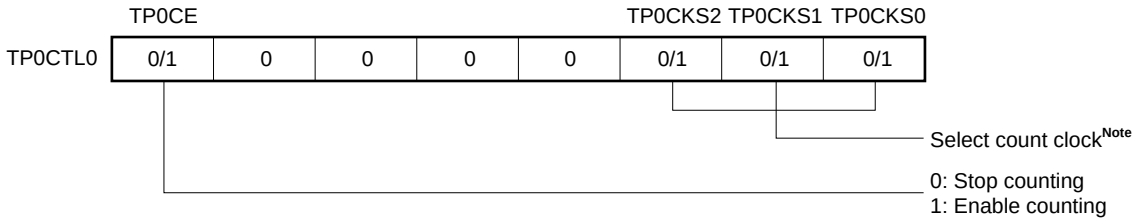


**Figure 7-31. Register Setting in Free-Running Timer Mode (1/2)**

**(a) TMP0 control register 0 (TP0CTL0)**




**Note** The setting is invalid when the TP0CTL1.TP0EEE bit = 1

**(b) TMP0 control register 1 (TP0CTL1)**


**(c) TMP0 I/O control register 0 (TP0IOC0)**

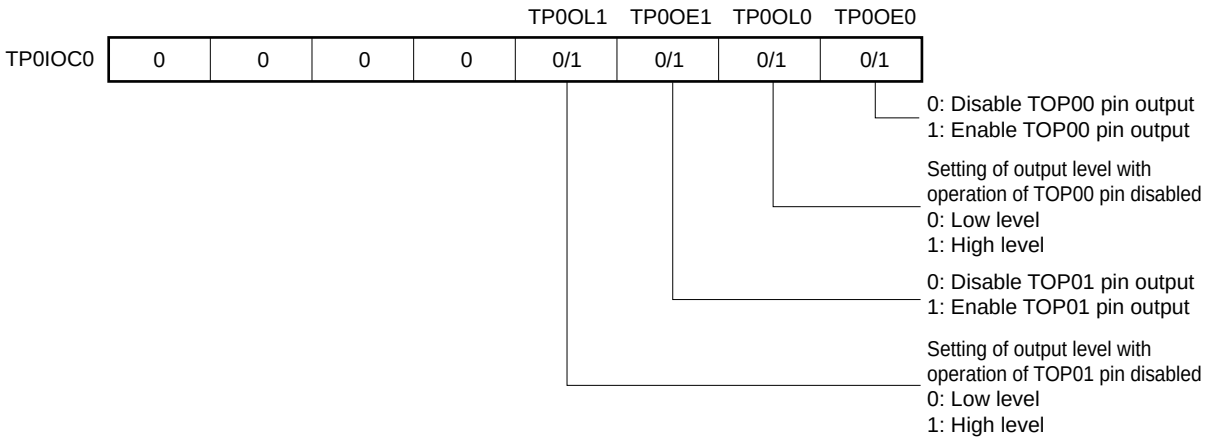
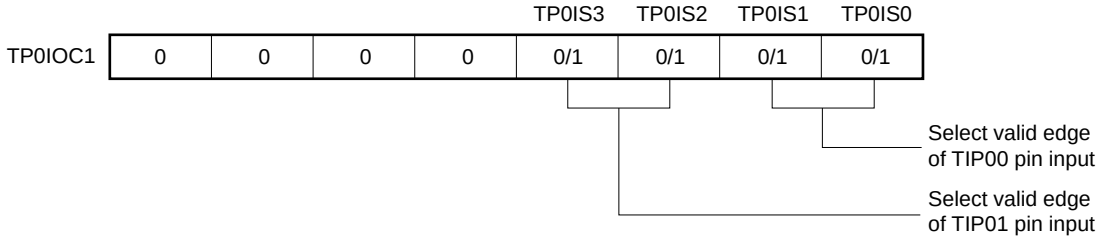
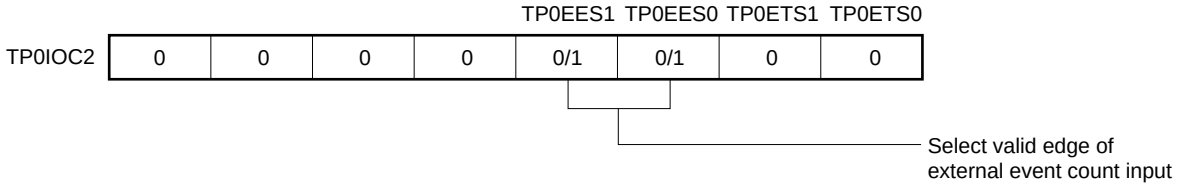


**Figure 7-31. Register Setting in Free-Running Timer Mode (2/2)**

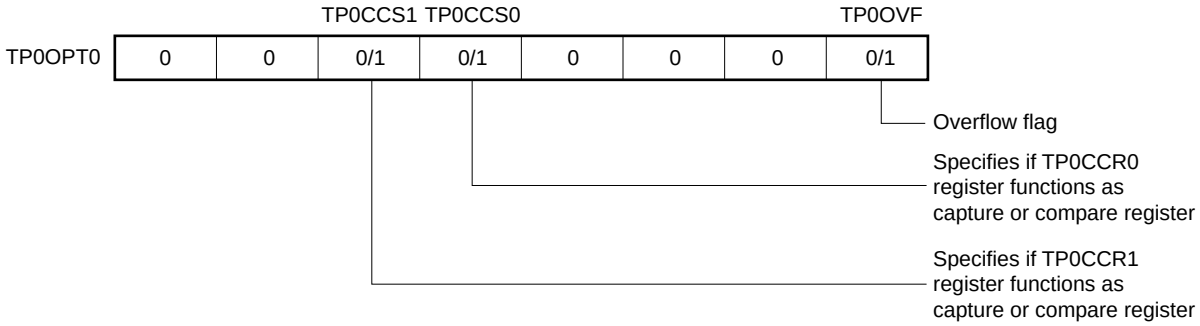
**(d) TMP0 I/O control register 1 (TP0IOC1)**




**(e) TMP0 I/O control register 2 (TP0IOC2)**




**(f) TMP0 option register 0 (TP0OPT0)**




**(g) TMP0 counter read buffer register (TP0CNT)**

The value of the 16-bit counter can be read by reading the TP0CNT register.

**(h) TMP0 capture/compare registers 0 and 1 (TP0CCR0 and TP0CCR1)**

These registers function as capture registers or compare registers depending on the setting of the TP0OPT0.TP0CCSa bit.

When the registers function as capture registers, they store the count value of the 16-bit counter when the valid edge input to the TIP0a pin is detected.

When the registers function as compare registers and when $D_a$ is set to the TP0CCRa register, the INTTP0CCa signal is generated when the counter reaches ($D_a$ + 1), and the output signal of the TOP0a pin is inverted.

**Remark** a = 0, 1

### (1) Operation flow in free-running timer mode

#### (a) When using capture/compare register as compare register

**Figure 7-32. Software Processing Flow in Free-Running Timer Mode (Compare Function) (1/2)**

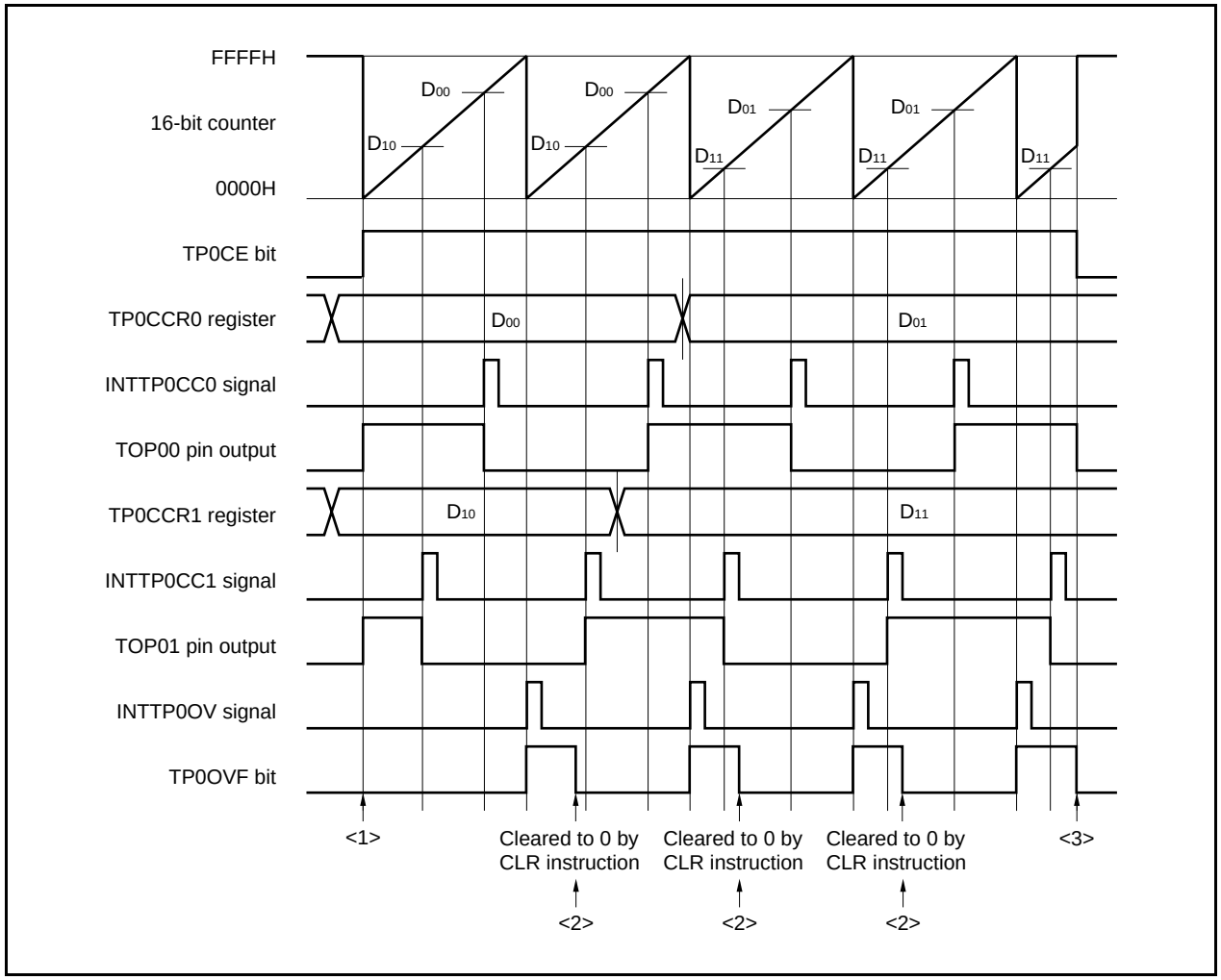
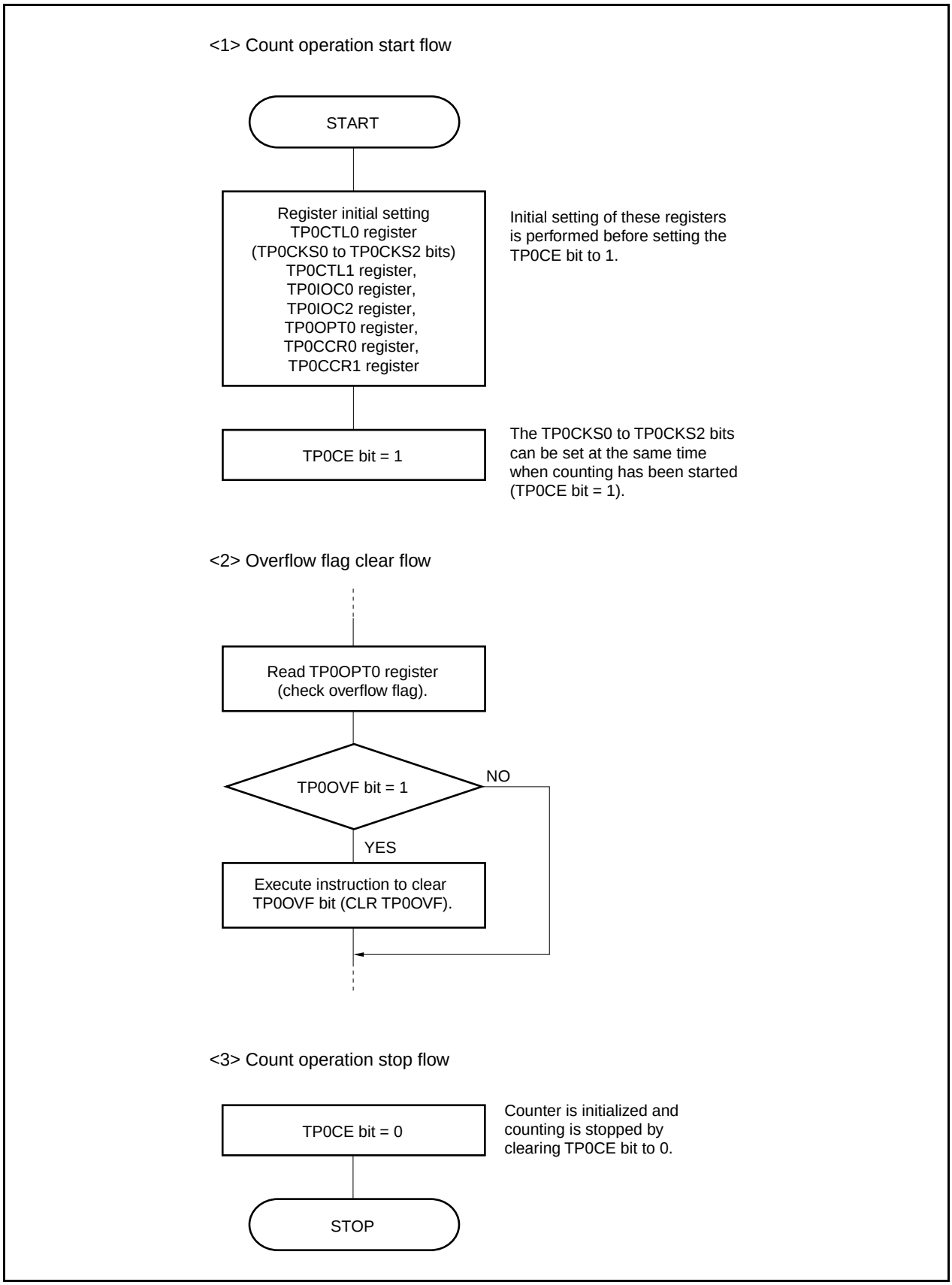


**Figure 7-32. Software Processing Flow in Free-Running Timer Mode (Compare Function) (2/2)**



**(b) When using capture/compare register as capture register**

**Figure 7-33. Software Processing Flow in Free-Running Timer Mode (Capture Function) (1/2)**

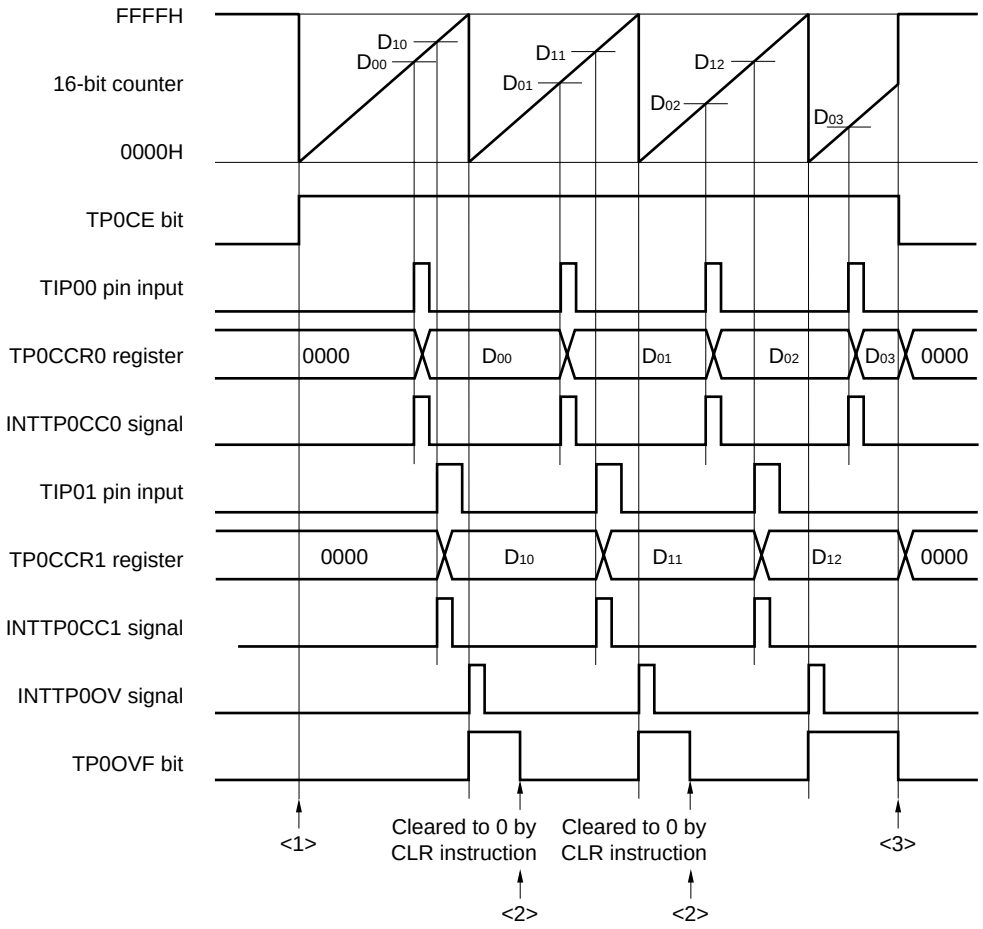
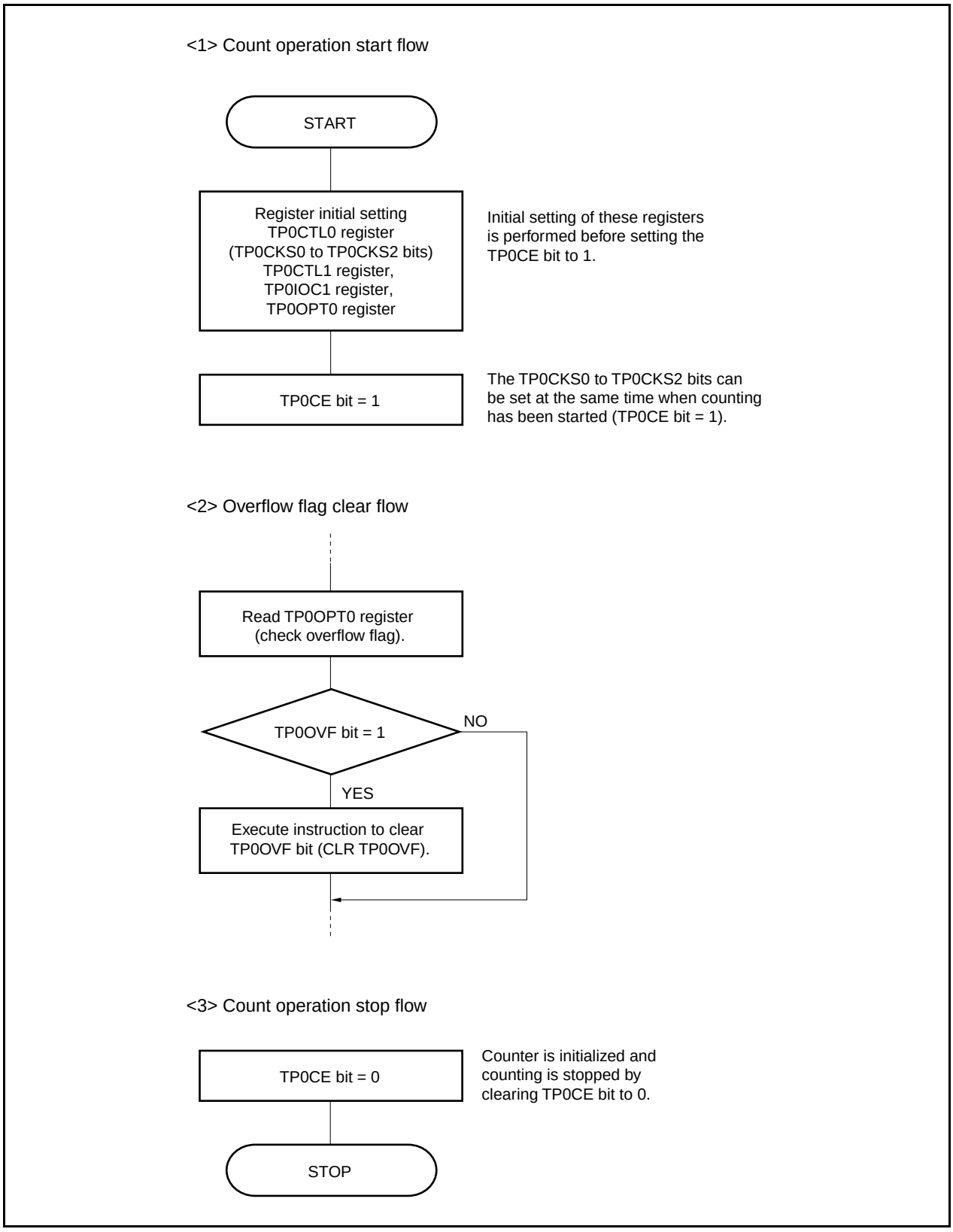


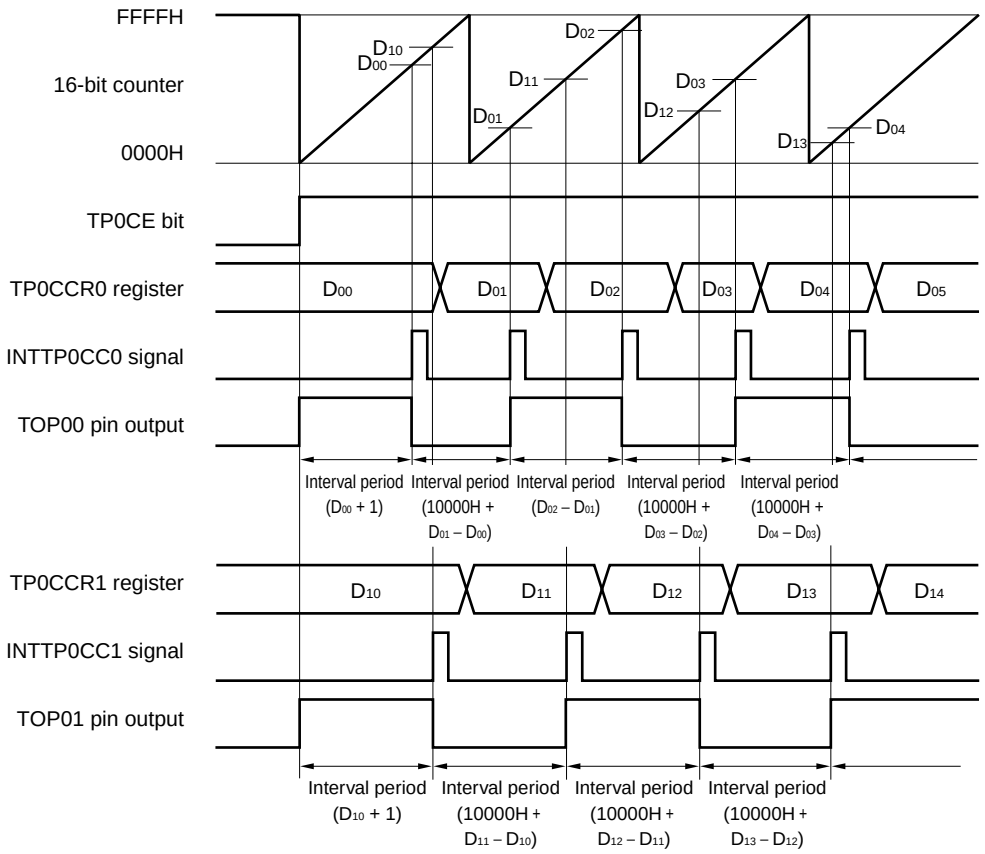
**Figure 7-33. Software Processing Flow in Free-Running Timer Mode (Capture Function) (2/2)**



### (2) Operation timing in free-running timer mode

#### (a) Interval operation with compare register

When 16-bit timer/event counter P is used as an interval timer with the TP0CCRa register used as a compare register, software processing is necessary for setting a comparison value to generate the next interrupt request signal each time the INTTP0CCa signal has been detected.




When performing an interval operation in the free-running timer mode, two intervals can be set with one channel.

To perform the interval operation, the value of the corresponding TP0CCRa register must be re-set in the interrupt servicing that is executed when the INTTP0CCa signal is detected.

The set value for re-setting the TP0CCRa register can be calculated by the following expression, where "$D_a$" is the interval period.

Compare register default value: $D_a - 1$

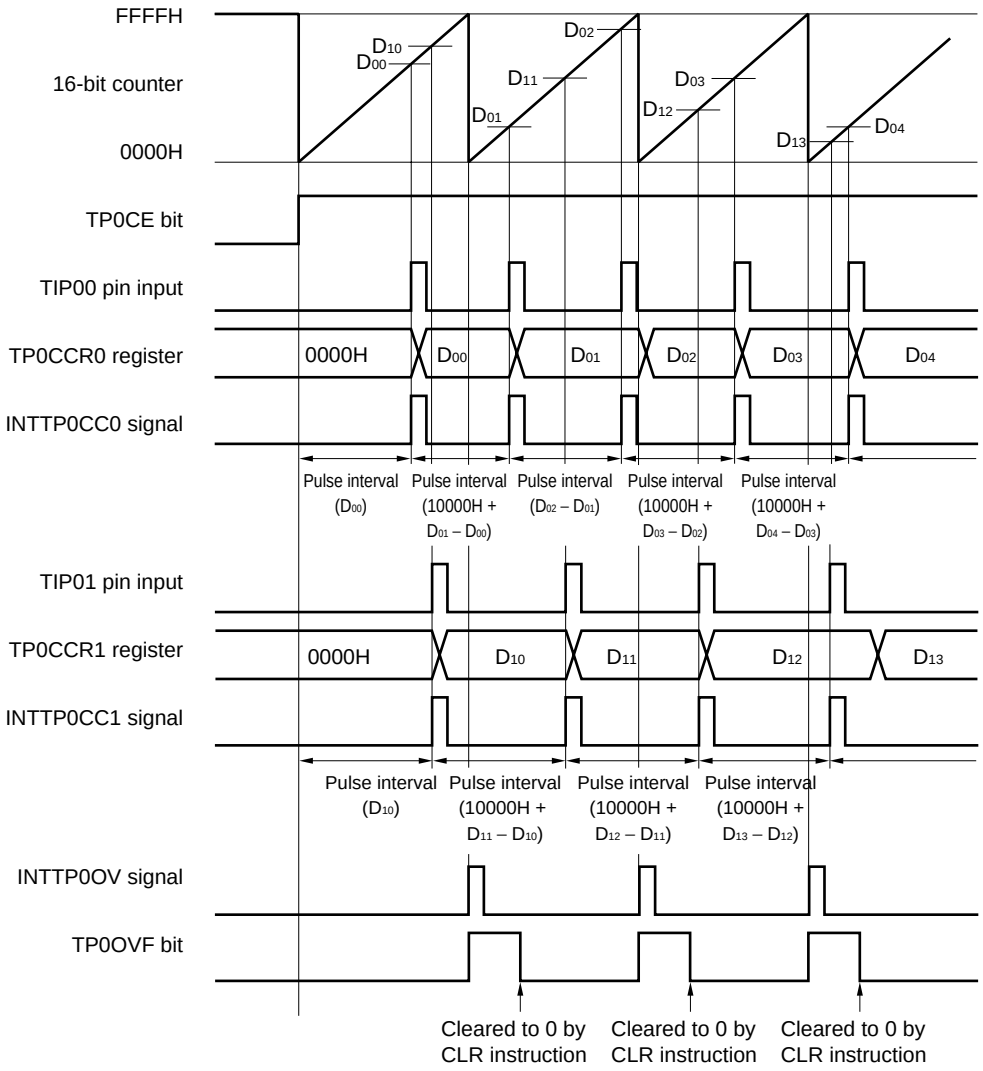
Value set to compare register second and subsequent time: Previous set value + $D_a$

(If the calculation result is greater than FFFFH, subtract 10000H from the result and set this value to the register.)

**Remark** a = 0, 1

**(b) Pulse width measurement with capture register**

When pulse width measurement is performed with the TP0CCRa register used as a capture register, software processing is necessary for reading the capture register each time the INTTP0CCa signal has been detected and for calculating an interval.




When executing pulse width measurement in the free-running timer mode, two pulse widths can be measured with one channel.

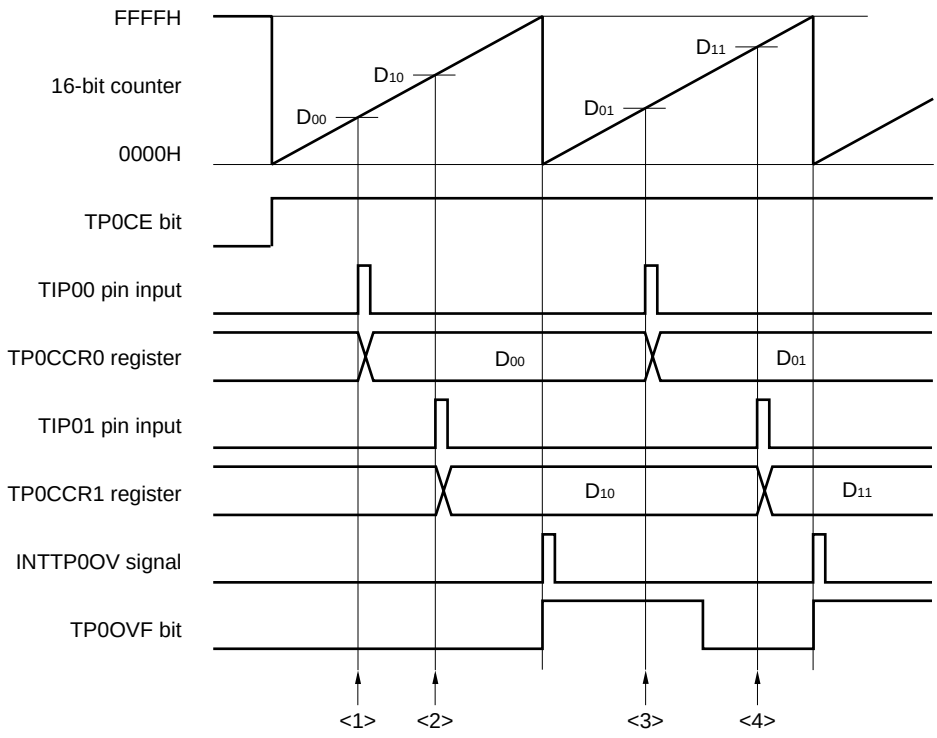
To measure a pulse width, the pulse width can be calculated by reading the value of the TP0CCRa register in synchronization with the INTTP0CCa signal, and calculating the difference between the read value and the previously read value.

**Remark** a = 0, 1

**(c) Processing of overflow when two capture registers are used**

Care must be exercised in processing the overflow flag when two capture registers are used. First, an example of incorrect processing is shown below.

**Example of incorrect processing when two capture registers are used**

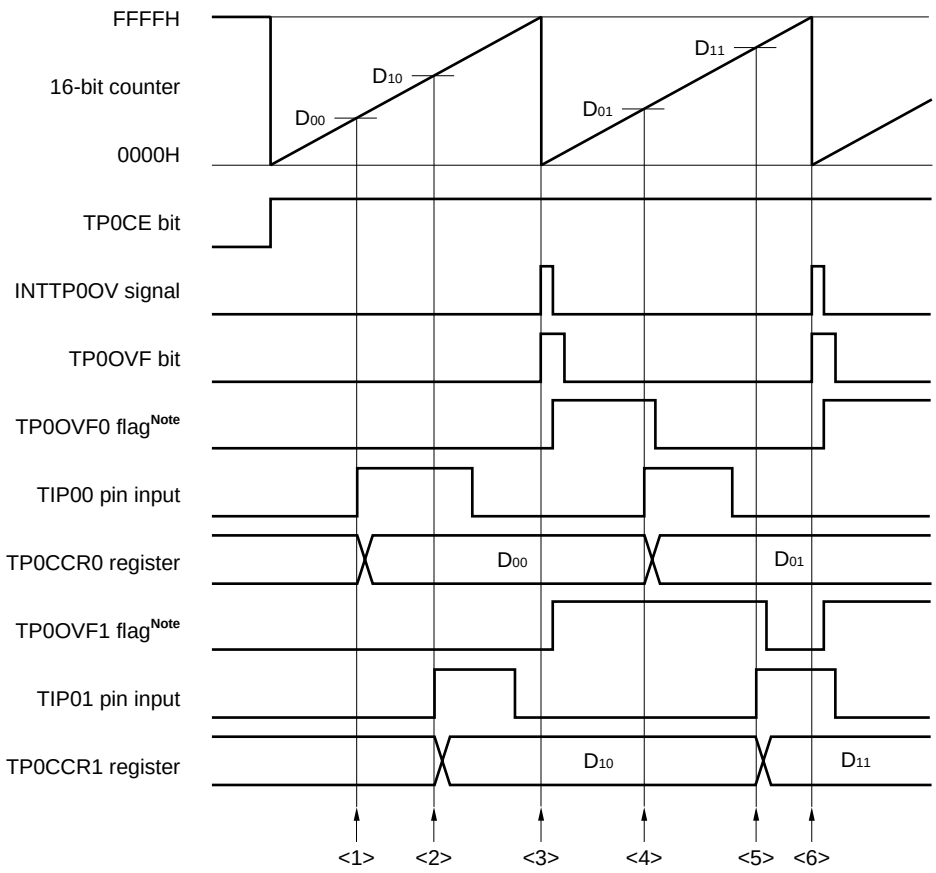



The following problem may occur when two pulse widths are measured in the free-running timer mode.

<1> Read the TP0CCR0 register (setting of the default value of the TIP00 pin input).
<2> Read the TP0CCR1 register (setting of the default value of the TIP01 pin input).
<3> Read the TP0CCR0 register.
Read the overflow flag. If the overflow flag is 1, clear it to 0.
Because the overflow flag is 1, the pulse width can be calculated by (10000H + $D_{01} - D_{00}$).
<4> Read the TP0CCR1 register.
Read the overflow flag. Because the flag is cleared in <3>, 0 is read.
Because the overflow flag is 0, the pulse width can be calculated by ($D_{11} - D_{10}$) (incorrect).

When two capture registers are used, and if the overflow flag is cleared to 0 by one capture register, the other capture register may not obtain the correct pulse width.
Use software when using two capture registers. An example of how to use software is shown below.

**Example when two capture registers are used (using overflow interrupt)**




**Note** The TP0OVF0 and TP0OVF1 flags are set on the internal RAM by software.

<1> Read the TP0CCR0 register (setting of the default value of the TIP00 pin input).

<2> Read the TP0CCR1 register (setting of the default value of the TIP01 pin input).

<3> An overflow occurs. Set the TP0OVF0 and TP0OVF1 flags to 1 in the overflow interrupt servicing, and clear the overflow flag to 0.

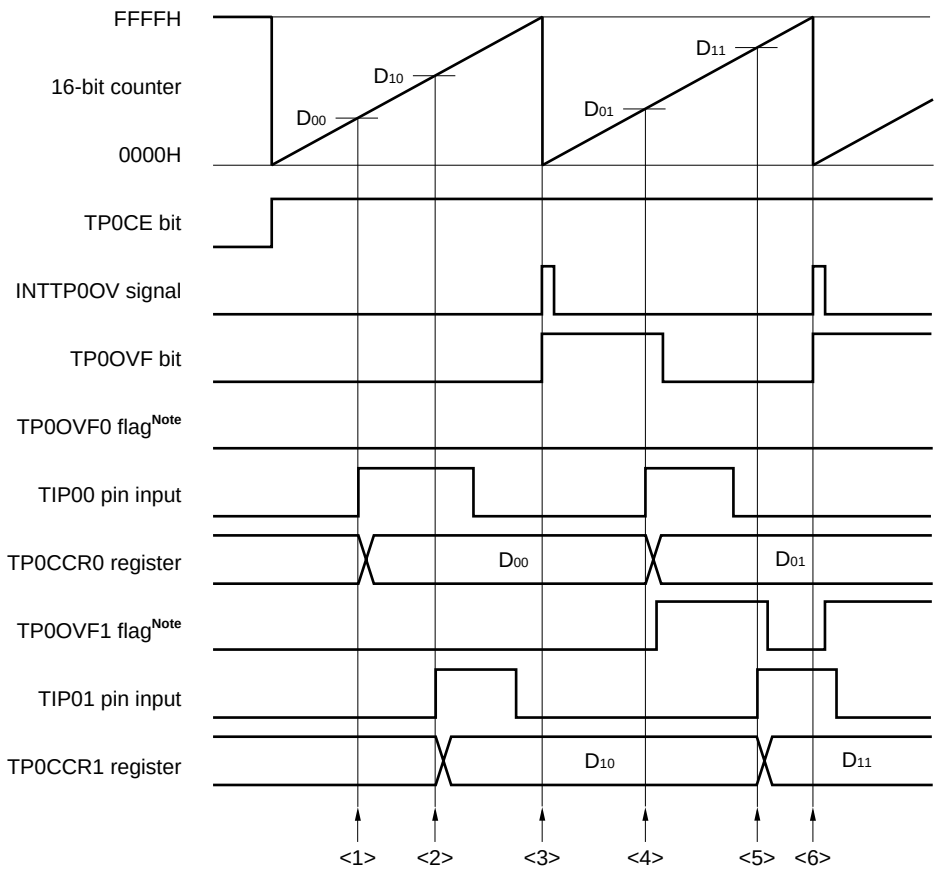
<4> Read the TP0CCR0 register.
Read the TP0OVF0 flag. If the TP0OVF0 flag is 1, clear it to 0.
Because the TP0OVF0 flag is 1, the pulse width can be calculated by (10000H + $D_{01} - D_{00}$).

<5> Read the TP0CCR1 register.
Read the TP0OVF1 flag. If the TP0OVF1 flag is 1, clear it to 0 (the TP0OVF0 flag is cleared in <4>, and the TP0OVF1 flag remains 1).
Because the TP0OVF1 flag is 1, the pulse width can be calculated by (10000H + $D_{11} - D_{10}$) (correct).

<6> Same as <3>

(2/2)

**Example when two capture registers are used (without using overflow interrupt)**




**Note** The TP0OVF0 and TP0OVF1 flags are set on the internal RAM by software.

<1> Read the TP0CCR0 register (setting of the default value of the TIP00 pin input).

<2> Read the TP0CCR1 register (setting of the default value of the TIP01 pin input).

<3> An overflow occurs. Nothing is done by software.

<4> Read the TP0CCR0 register.

Read the overflow flag. If the overflow flag is 1, set only the TP0OVF1 flag to 1, and clear the overflow flag to 0.

Because the overflow flag is 1, the pulse width can be calculated by (10000H + $D_{01} - D_{00}$).

<5> Read the TP0CCR1 register.

Read the overflow flag. Because the overflow flag is cleared in <4>, 0 is read.

Read the TP0OVF1 flag. If the TP0OVF1 flag is 1, clear it to 0.

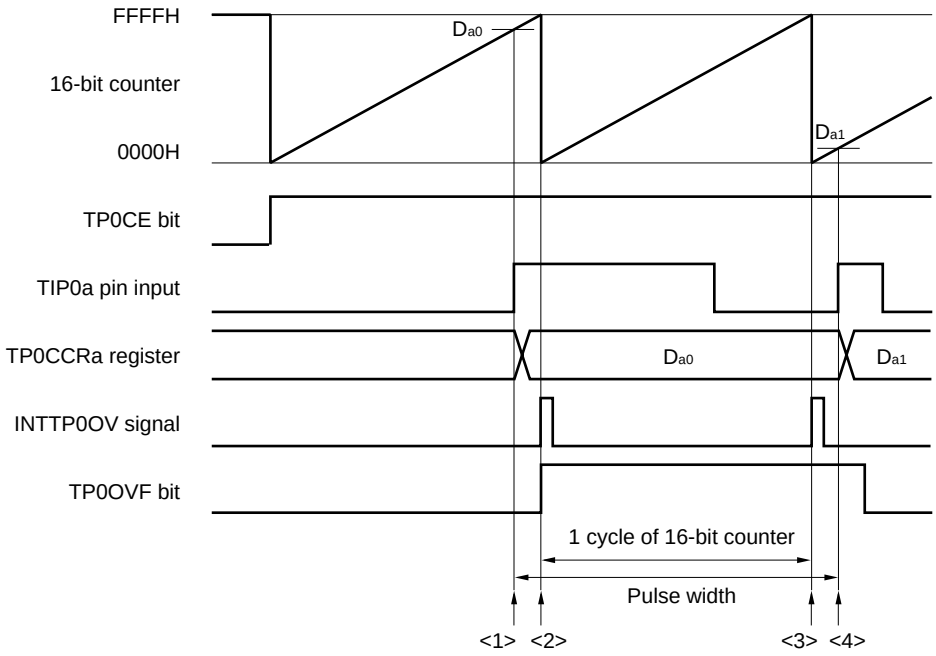
Because the TP0OVF1 flag is 1, the pulse width can be calculated by (10000H + $D_{11} - D_{10}$) (correct).

<6> Same as <3>

**(d) Processing of overflow if capture trigger interval is long**

If the pulse width is greater than one cycle of the 16-bit counter, care must be exercised because an overflow may occur more than once from the first capture trigger to the next. First, an example of incorrect processing is shown below.

**Example of incorrect processing when capture trigger interval is long**



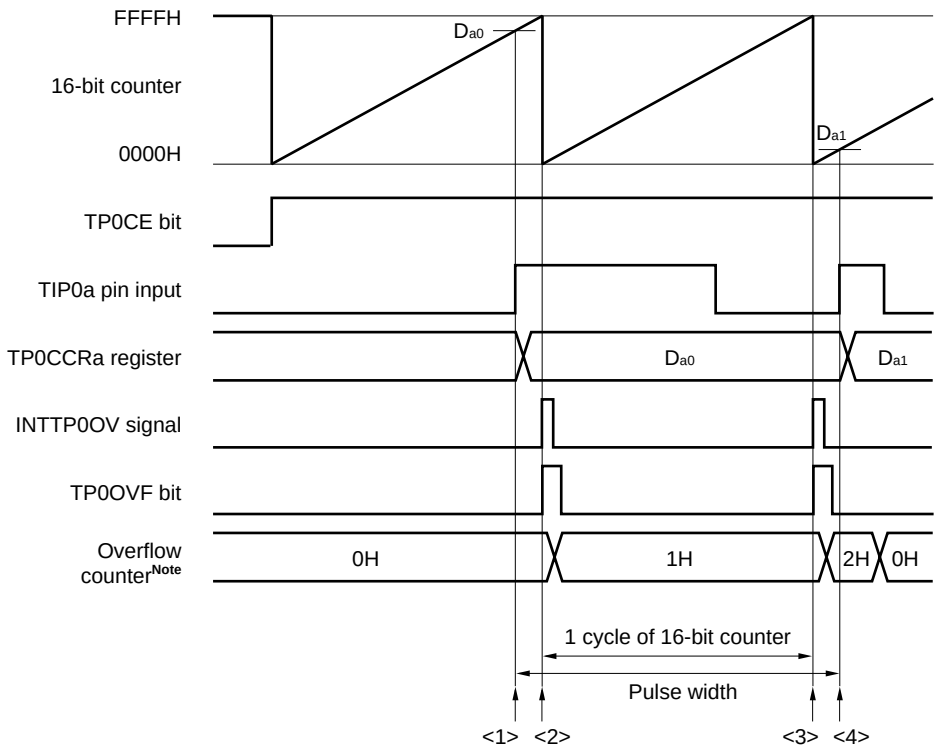

The following problem may occur when long pulse width is measured in the free-running timer mode.

<1> Read the TP0CCRa register (setting of the default value of the TIP0a pin input).
<2> An overflow occurs. Nothing is done by software.
<3> An overflow occurs a second time. Nothing is done by software.
<4> Read the TP0CCRa register.
Read the overflow flag. If the overflow flag is 1, clear it to 0.
Because the overflow flag is 1, the pulse width can be calculated by (10000H + $D_{a1} - D_{a0}$) (incorrect).
Actually, the pulse width must be (20000H + $D_{a1} - D_{a0}$) because an overflow occurs twice.

If an overflow occurs twice or more when the capture trigger interval is long, the correct pulse width may not be obtained.

If the capture trigger interval is long, slow the count clock to lengthen one cycle of the 16-bit counter, or use software. An example of how to use software is shown next.

**Example when capture trigger interval is long**




**Note** The overflow counter is set arbitrarily by software on the internal RAM.

<1> Read the TP0CCRa register (setting of the default value of the TIP0a pin input).

<2> An overflow occurs. Increment the overflow counter and clear the overflow flag to 0 in the overflow interrupt servicing.

<3> An overflow occurs a second time. Increment (+1) the overflow counter and clear the overflow flag to 0 in the overflow interrupt servicing.

<4> Read the TP0CCRa register.

Read the overflow counter.

→ When the overflow counter is “N”, the pulse width can be calculated by ($N \times 10000H + D_{a1} - D_{a0}$).

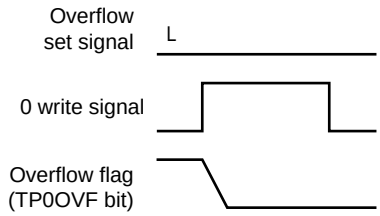
In this example, the pulse width is ($20000H + D_{a1} - D_{a0}$) because an overflow occurs twice.

Clear the overflow counter (0H).

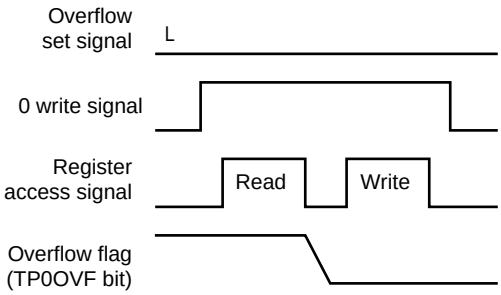
**(e) Clearing overflow flag**

The overflow flag can be cleared to 0 by clearing the TP0OVF bit to 0 with the CLR instruction and by writing 8-bit data (bit 0 is 0) to the TP0OPT0 register. To accurately detect an overflow, read the TP0OVF bit when it is 1, and then clear the overflow flag by using a bit manipulation instruction.

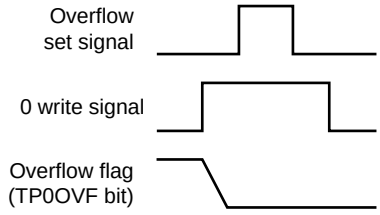
(i) Operation to write 0 (without conflict with setting)



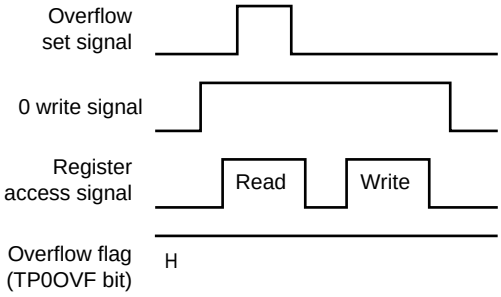

(iii) Operation to clear to 0 (without conflict with setting)




(ii) Operation to write 0 (conflict with setting)




(iv) Operation to clear to 0 (conflict with setting)




To clear the overflow flag to 0, read the overflow flag to check if it is set to 1, and clear it with the CLR instruction. If 0 is written to the overflow flag without checking if the flag is 1, the set information of overflow may be erased by writing 0 ((ii) in the above chart). Therefore, software may judge that no overflow has occurred even when an overflow actually has occurred.

If execution of the CLR instruction conflicts with occurrence of an overflow when the overflow flag is cleared to 0 with the CLR instruction, the overflow flag remains set even after execution of the clear instruction.

### 7.5.7 Pulse width measurement mode (TP0MD2 to TP0MD0 bits = 110)

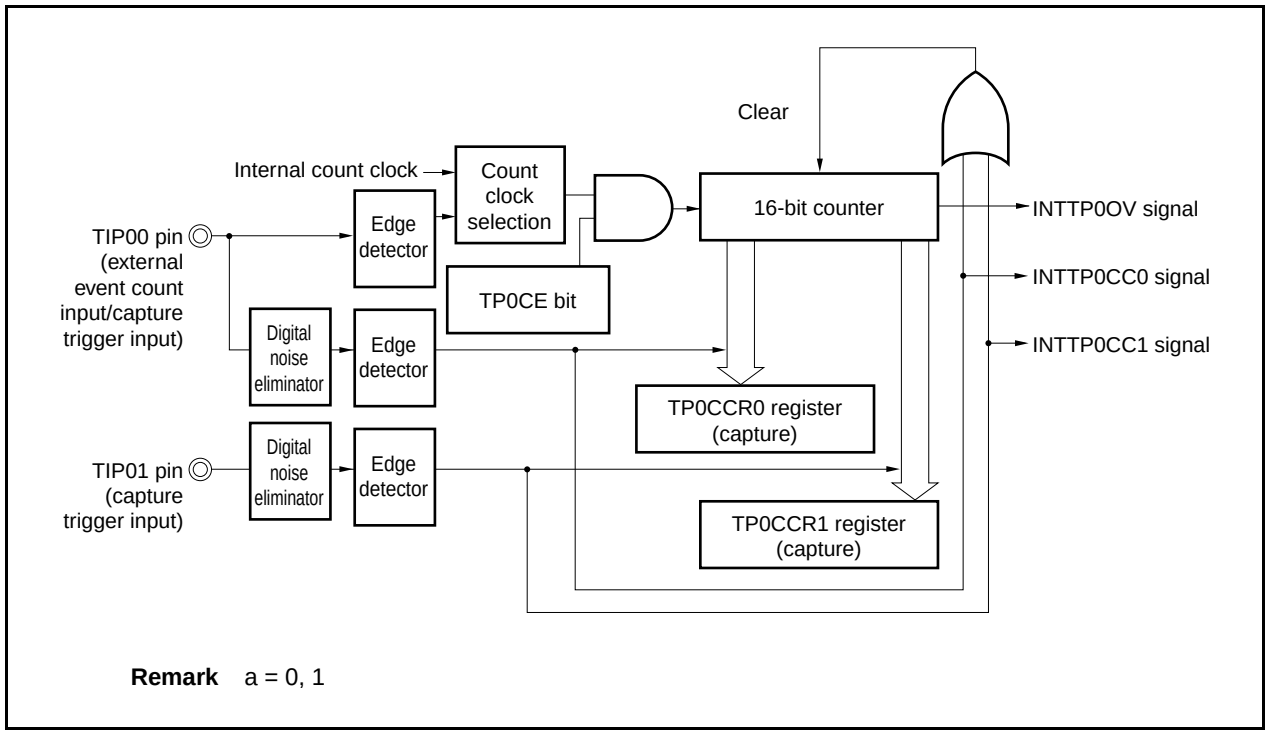
In the pulse width measurement mode, 16-bit timer/event counter P starts counting when the TP0CTL0.TP0CE bit is set to 1. Each time the valid edge input to the TIP0a pin has been detected, the count value of the 16-bit counter is stored in the TP0CCRa register, and the 16-bit counter is cleared to 0000H.

The interval of the valid edge can be measured by reading the TP0CCRa register after a capture interrupt request signal (INTTP0CCa) occurs.

Select either the TIP00 or TIP01 pin as the capture trigger input pin. Specify “No edge detected” by using the TP0IOC1 register for the unused pins.

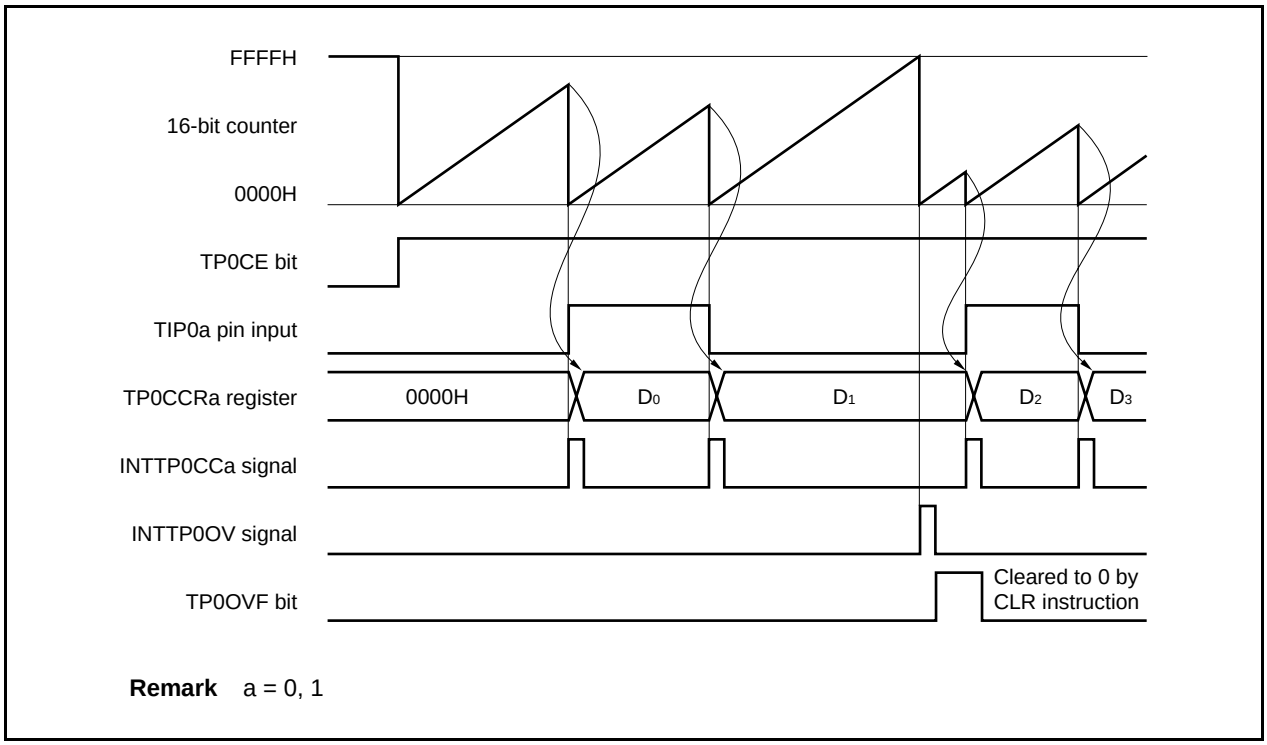
When an external clock is used as the count clock, measure the pulse width of the TIP01 pin because the external clock is fixed to the TIP00 pin. At this time, clear the TP0IOC1.TP0IS1 and TP0IOC1.TP0IS0 bits to 00 (capture trigger input (TIP00 pin): No edge detected).

**Figure 7-34. Configuration in Pulse Width Measurement Mode**




**Remark** a = 0, 1

**Figure 7-35. Basic Timing in Pulse Width Measurement Mode**




**Remark** a = 0, 1

When the TP0CE bit is set to 1, the 16-bit counter starts counting. When the valid edge input to the TIP0a pin is later detected, the count value of the 16-bit counter is stored in the TP0CCRa register, the 16-bit counter is cleared to 0000H, and a capture interrupt request signal (INTTP0CCa) is generated.

The pulse width is calculated as follows.

First pulse width = $(D_0 + 1) \times$ Count clock cycle

Second and subsequent pulse width = $(D_N - D_{N-1}) \times$ Count clock cycle

If the valid edge is not input to the TIP0a pin even when the 16-bit counter counted up to FFFFH, an overflow interrupt request signal (INTTP0OV) is generated at the next count clock, and the counter is cleared to 0000H and continues counting. At this time, the overflow flag (TP0OPT0.TP0OVF bit) is also set to 1. Clear the overflow flag to 0 by executing the CLR instruction via software.

If the overflow flag is set to 1, the pulse width can be calculated as follows.

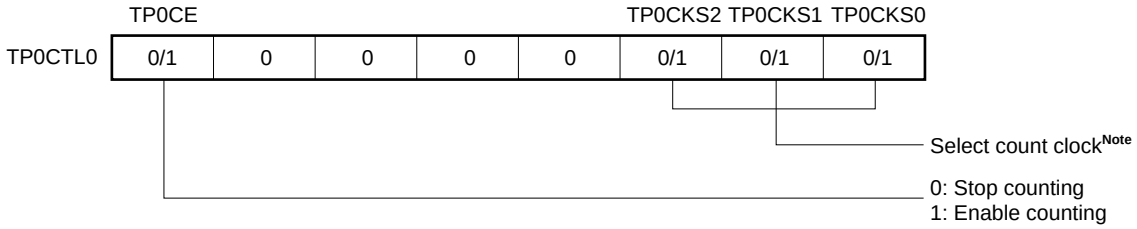
First pulse width = $(D_0 + 10001H) \times$ Count clock cycle

Second and subsequent pulse width = $(10000H + D_N - D_{N-1}) \times$ Count clock cycle

**Remark** a = 0, 1

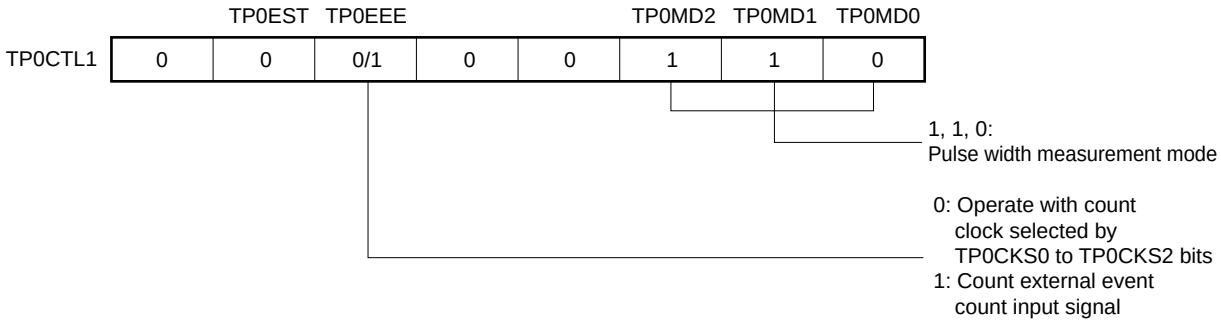
**Figure 7-36. Register Setting in Pulse Width Measurement Mode (1/2)**

**(a) TMP0 control register 0 (TP0CTL0)**

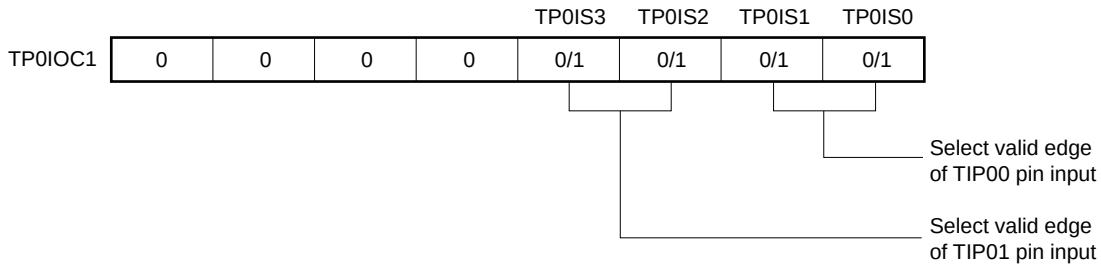



**Note** Setting is invalid when the TP0EEE bit = 1.

**(b) TMP0 control register 1 (TP0CTL1)**




**(c) TMP0 I/O control register 1 (TP0IOC1)**




**(d) TMP0 I/O control register 2 (TP0IOC2)**

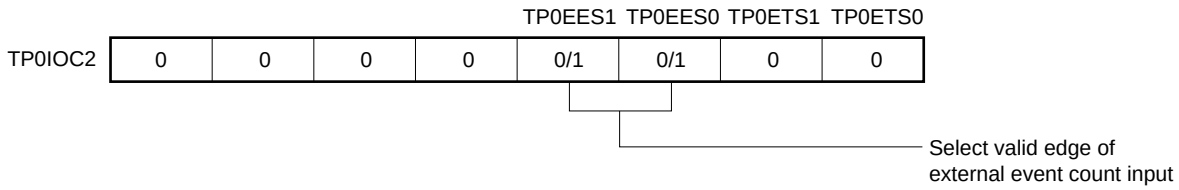


**Figure 7-36. Register Setting in Pulse Width Measurement Mode (2/2)**

**(e) TMP0 option register 0 (TP0OPT0)**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | TP0CCS1 | TP0CCS0 | | | | TP0OVF |
| TP0OPT0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0/1 |

Overflow flag

**(f) TMP0 counter read buffer register (TP0CNT)**

The value of the 16-bit counter can be read by reading the TP0CNT register.

**(g) TMP0 capture/compare registers 0 and 1 (TP0CCR0 and TP0CCR1)**

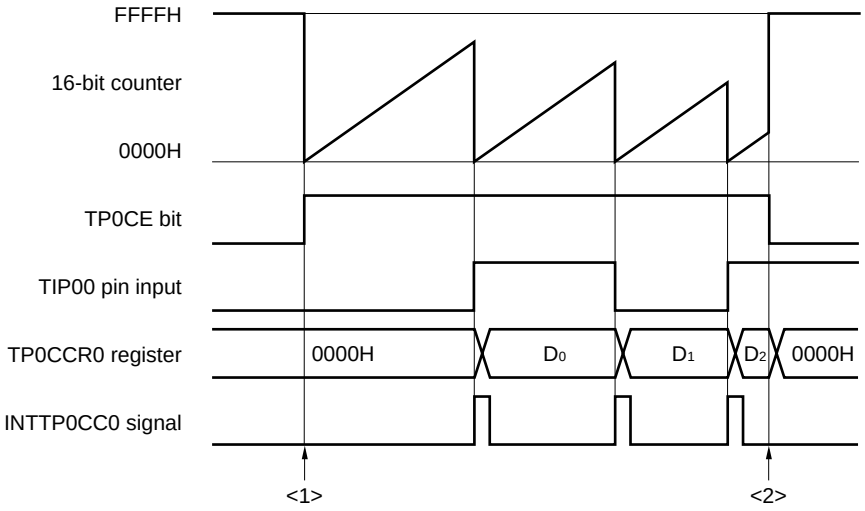
These registers store the count value of the 16-bit counter when the valid edge input to the TIP0a pin is detected.

**Remarks 1.** TMP0 I/O control register 0 (TP0IOC0) is not used in the pulse width measurement mode.

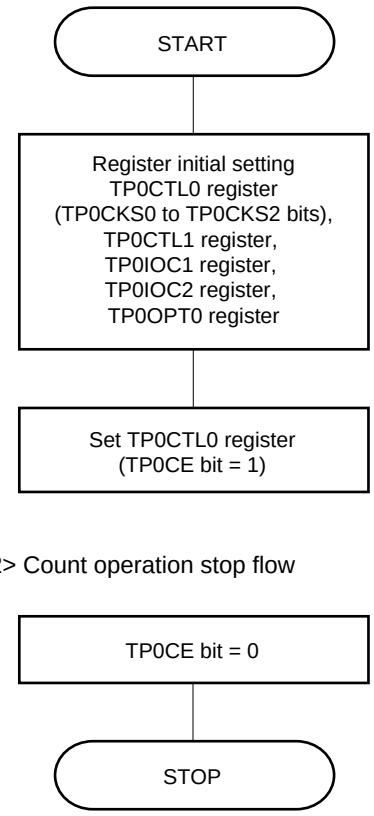
**2.** a = 0, 1

**(1) Operation flow in pulse width measurement mode**

**Figure 7-37. Software Processing Flow in Pulse Width Measurement Mode**




<1> Count operation start flow




Initial setting of these registers is performed before setting the TP0CE bit to 1.

The TP0CKS0 to TP0CKS2 bits can be set at the same time when counting has been started (TP0CE bit = 1).

<2> Count operation stop flow



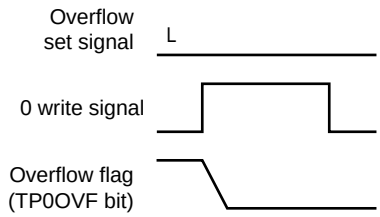

The counter is initialized and counting is stopped by clearing the TP0CE bit to 0.

### (2) Operation timing in pulse width measurement mode

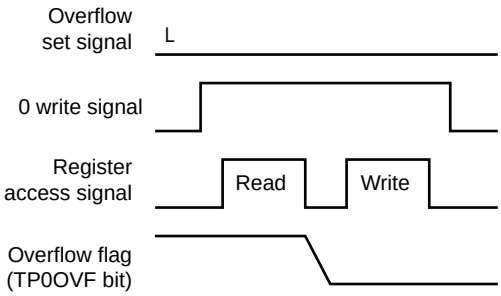
#### (a) Clearing overflow flag

The overflow flag can be cleared to 0 by clearing the TP0OVF bit to 0 with the CLR instruction and by writing 8-bit data (bit 0 is 0) to the TP0OPT0 register. To accurately detect an overflow, read the TP0OVF bit when it is 1, and then clear the overflow flag by using a bit manipulation instruction.

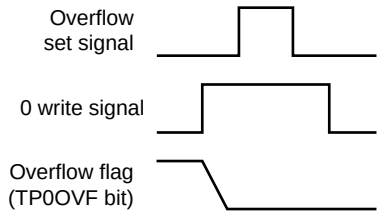
(i) Operation to write 0 (without conflict with setting)



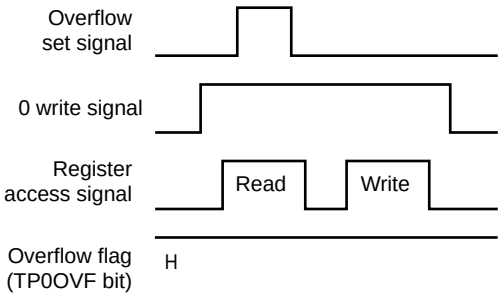

(iii) Operation to clear to 0 (without conflict with setting)




(ii) Operation to write 0 (conflict with setting)




(iv) Operation to clear to 0 (conflict with setting)




To clear the overflow flag to 0, read the overflow flag to check if it is set to 1, and clear it with the CLR instruction. If 0 is written to the overflow flag without checking if the flag is 1, the set information of overflow may be erased by writing 0 ((ii) in the above chart). Therefore, software may judge that no overflow has occurred even when an overflow actually has occurred.

If execution of the CLR instruction conflicts with occurrence of an overflow when the overflow flag is cleared to 0 with the CLR instruction, the overflow flag remains set even after execution of the clear instruction.

### 7.5.8 Timer output operations

The following table shows the operations and output levels of the TOP00 and TOP01 pins.

**Table 7-4. Timer Output Control in Each Mode**

| Operation Mode | TOP01 Pin | TOP00 Pin |
|---|---|---|
| Interval timer mode | Square wave output | |
| External event count mode | Square wave output | – |
| External trigger pulse output mode | External trigger pulse output | Square wave output |
| One-shot pulse output mode | One-shot pulse output | |
| PWM output mode | PWM output | |
| Free-running timer mode | Square wave output (only when compare function is used) | |
| Pulse width measurement mode | – | |

**Table 7-5. Truth Table of TOP00 and TOP01 Pins Under Control of Timer Output Control Bits**

| TP0IOC0.TP0OLa Bit | TP0IOC0.TP0OEa Bit | TP0CTL0.TP0CE Bit | Level of TOP0a Pin |
|---|---|---|---|
| 0 | 0 | × | Low-level output |
| | 1 | 0 | Low-level output |
| | | 1 | Low level immediately before counting, high level after counting is started |
| 1 | 0 | × | High-level output |
| | 1 | 0 | High-level output |
| | | 1 | High level immediately before counting, low level after counting is started |

**Remark** a = 0, 1

## 7.6 Eliminating Noise on Capture Trigger Input Pin (TIP0a)

The TIP0a pin has a digital noise eliminator.

However, this circuit is valid only when the pin is used as a capture trigger input pin; it is invalid when the pin is used as an external event count input pin or external trigger input pin.

Digital noise can be eliminated by specifying the alternate function of the TIP0a pin using the PMC3, PFC3, and PFCE3 registers.

The number of times of sampling can be selected from three or two by using the PaNFC.PaNFSTS bit. The sampling clock can be selected from $f_{XX}$, $f_{XX}/2$, $f_{XX}/4$, $f_{XX}/16$, $f_{XX}/32$, or $f_{XX}/64$, by using the PaNFC.PaNFC2 to PaNFC.PaNFC0 bits.

**(1) TIP0a noise elimination control register (PaNFC)**

This register is used to select the sampling clock and the number of times of sampling for eliminating digital noise.

This register can be read or written in 8-bit or 1-bit units.

Reset input clears this register to 00H.

After reset: 00H R/W Address: P0NFC FFFFFB00H, P1NFC FFFFFB04H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PaNFC (a = 0, 1) | 0 | PaNFSTS | 0 | 0 | 0 | PaNFC2 | PaNFC1 | PaNFC0 |

| PaNFSTS | Setting of number of times of sampling for eliminating digital noise |
|---|---|
| 0 | Number of times of sampling = 3 |
| 1 | Number of times of sampling = 2 |

| PaNFC2 | PaNFC1 | PaNFC0 | Sampling clock selection |
|---|---|---|---|
| 0 | 0 | 0 | $f_{XX}$ |
| 0 | 0 | 1 | $f_{XX}/2$ |
| 0 | 1 | 0 | $f_{XX}/4$ |
| 0 | 1 | 1 | $f_{XX}/16$ |
| 1 | 0 | 0 | $f_{XX}/32$ |
| 1 | 0 | 1 | $f_{XX}/64$ |
| Other than above | | | Setting prohibited |

**Cautions 1. Enable starting the 16-bit counter of TMP0 (TP0CTL.TP0CE bit = 1) after the lapse of the sampling clock period × number of times of sampling.**

**2. Be sure to clear bits 7, 5 to 3 to 0.**

**<Setting procedure>**

<1> Select the number of times of sampling and the sampling clock by using the PaNFC register.
<2> Select the alternate function (of the TIP0a pin) by using the PMC3, PFC3, and PFCE3 registers.
<3> Set the operating mode of TMP0 (such as the capture mode or the valid edge of the capture trigger).
<4> Enable the TMP0 count operation.

**<Noise elimination width>**

The digital noise elimination width ($t_{WTIP0a}$) is as follows, where T is the sampling clock period and M is the number of times of sampling.

- $t_{WTIP0a} < (M - 1)T$: Accurately eliminated as noise
- $(M - 1)T \leq t_{WTIP0a} < MT$: Eliminated as noise or detected as valid edge
- $t_{WTIP0a} \geq MT$: Accurately detected as valid edge

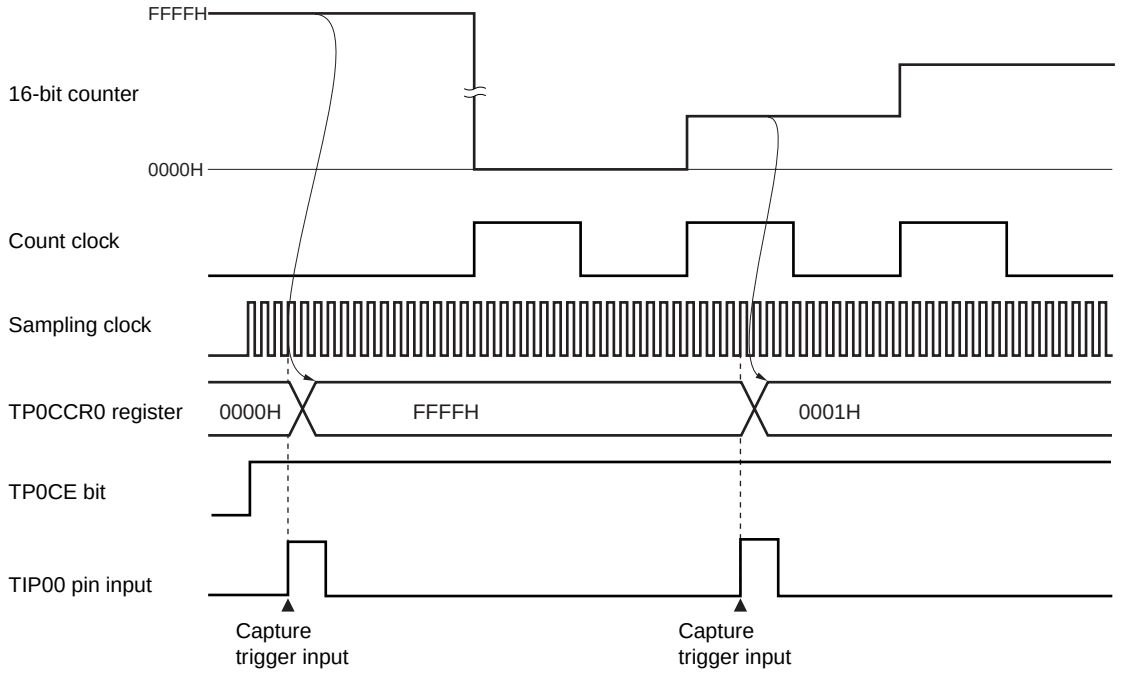
Therefore, a pulse width of MT or longer must be input so that the valid edge of the capture trigger input can be accurately detected.

## 7.7 Cautions

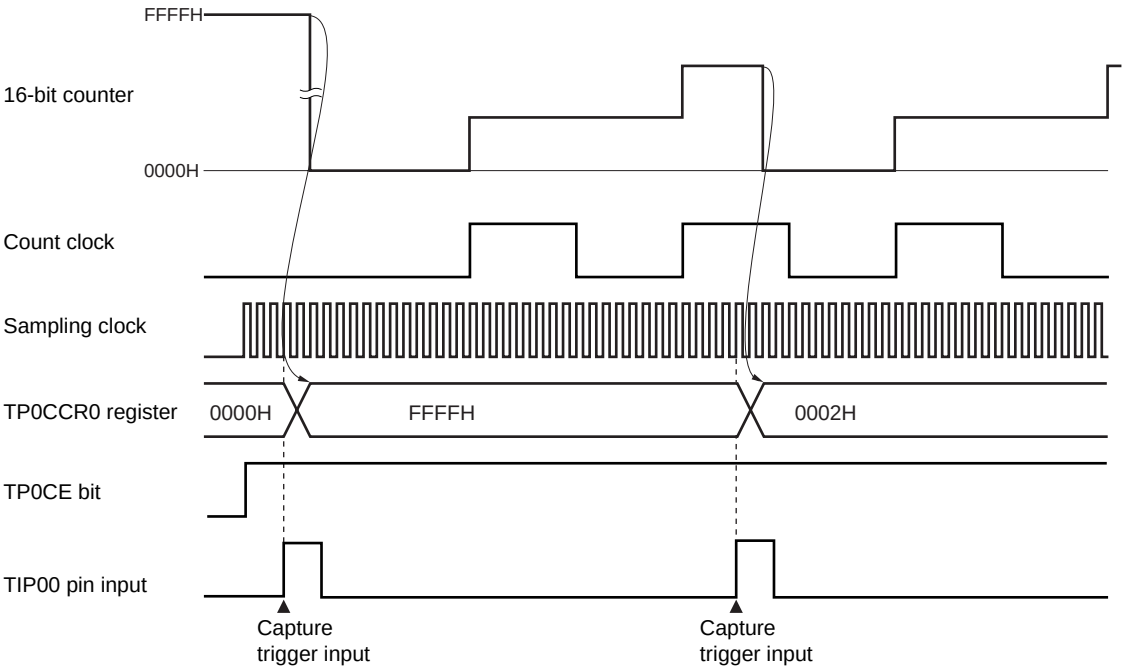
### (1) Capture operation

When the capture operation is used and a slow clock is selected as the count clock, FFFFH, not 0000H, may be captured in the TP0CCRn register if the capture trigger is input immediately after the TP0CE bit is set to 1.

**(a) Free-running timer mode**




**(b) Pulse width measurement mode**



# CHAPTER 8 16-BIT TIMER/EVENT COUNTER 0

In the V850ES/KF1+, two channels of 16-bit timer/event counter 0 are provided.

## 8.1 Functions

16-bit timer/event counter 0n has the following functions (n = 0, 1).

**(1) Interval timer**

Generates an interrupt at predetermined time intervals.

**(2) PPG output**

Can output a rectangular wave with any frequency and any output pulse width.

**(3) Pulse width measurement**

Can measure the pulse width of a signal input from an external source.

**(4) External event counters**

Can measure the number of pulses of a signal input from an external source.

**(5) Square-wave output**

Can output a square wave of any frequency.

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

Can output a one-shot pulse with any output pulse width.

## 8.2 Configuration

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

**Table 8-1. Configuration of 16-Bit Timer/Event Counter 0n**

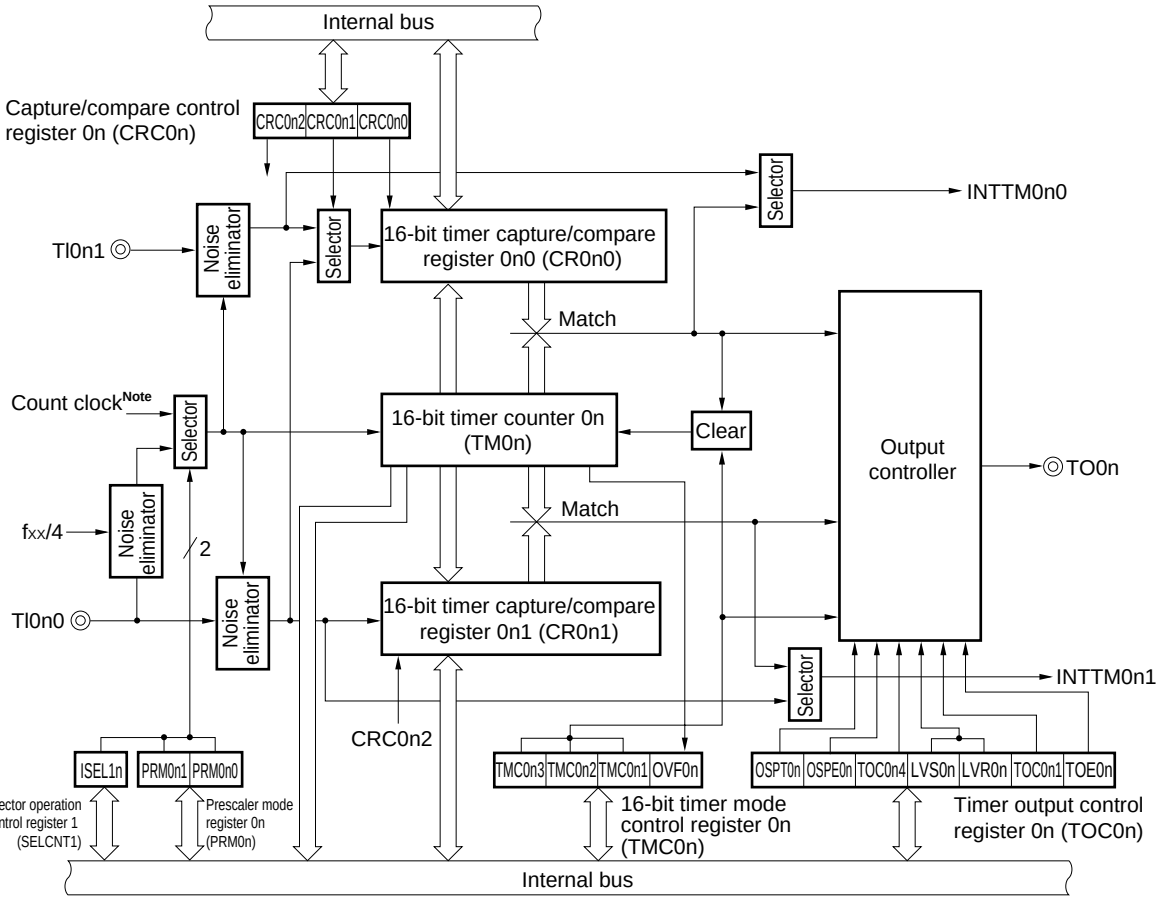
| Item | Configuration |
|---|---|
| Timer/counters | 16-bit timer counter 0n × 1 (TM0n) |
| Registers | 16-bit timer capture/compare register: 16 bits × 2 (CR0n0, CR0n1) |
| Timer inputs | 2 (TI0n0, TI0n1 pins) |
| Timer outputs | 1 (TO0n pin), output controller |
| Control registers[Note] | 16-bit timer mode control register n (TMC0n)<br>Capture/compare control register n (CRC0n)<br>16-bit timer output control register n (TOC0n)<br>Prescaler mode register 0n (PRM0n)<br>Selector operation control register 1 (SELCNT1) |

**Note** To use the TI0n0, TI0n1, and TO0n pin functions, refer to **Table 4-14 Settings When Port Pins Are Used for Alternate Functions**.

**Remark** n = 0, 1

The block diagram is shown below.

**Figure 8-1. Block Diagram of 16-Bit Timer/Event Counter 0n**




**Note** Set with the PRM0n register and SELCNT1 register.

**Remarks 1.** n = 0, 1

**2.** fxx: Main clock frequency

**(1) 16-bit timer counter 0n (TM0n)**

The TM0n register is a 16-bit read-only register that counts count pulses. The counter is incremented in synchronization with the rising edge of the input clock.

After reset: 0000H R Address: TM00 FFFFF600H, TM01 FFFFF610H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| TM0n | | | | | | | | | | | | | | | | |

(n = 0, 1)

The count value is reset to 0000H in the following cases.

<1> Reset
<2> If the TMC0n.TMC0n3 and TMC0n.TMC0n2 bits are cleared (0).
<3> If the valid edge of the TI0n0 pin is input in the mode in which clear & start occurs when inputting the valid edge of the TI0n0 pin
<4> If the TM0n register and the CR0n0 register match each other in the mode in which clear & start occurs on a match between the TM0n register and the CR0n0 register
<5> If the TOC0n.OSPT0n bit is set (1) or if the valid edge of the TI0n0 pin is input in the one-shot pulse output mode

**Remark** n = 0, 1

**(2) 16-bit timer capture/compare register 0n0 (CR0n0)**

The CR0n0 register is a 16-bit register that combines capture register and compare register functions.
The CRC0n.CRC0n0 bit is used to set whether to use the CR0n0 register as a capture register or as a compare register.
The CR0n0 register can be read or written in 16-bit units.
After reset, this register is cleared to 0000H.

After reset: 0000H R/W Address: CR000 FFFFF602H, CR010 FFFFF612H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CR0n0 | | | | | | | | | | | | | | | | |

(n = 0, 1)

**(a) When using the CR0n0 register as a compare register**

The value set to the CR0n0 register and the count value set to the TM0n register are always compared and when these values match, an interrupt request signal (INTTM0n0) is generated. The values are retained until rewritten.

**(b) When using the CR0n0 register as a capture register**

The TM0n register count value is captured to the CR0n0 register by inputting a capture trigger.

The valid edge of the TI0n0 pin or TI0n1 pin can be selected as the capture trigger. The valid edge of the TI0n0 pin is set with the PRM0n.ESn01 and PRM0n.ESn00 bits. The valid edge of the TI0n1 pin is set with the PRM0n.ESn11 and PRM0n.ESn10 bits.

Table 8-2 shows the settings when the valid edge of the TI0n0 pin is specified as the capture trigger, and Table 8-3 shows the settings when the valid edge of the TI0n1 pin is specified as the capture trigger.

**Table 8-2. Capture Trigger of CR0n0 Register and Valid Edge of TI0n0 Pin**

| Capture Trigger of CR0n0 | Valid Edge of TI0n0 Pin | | |
|---|---|---|---|
| | | ESn01 | ESn00 |
| Falling edge | Rising edge | 0 | 1 |
| Rising edge | Falling edge | 0 | 0 |
| No capture operation | Both rising and falling edges | 1 | 1 |

**Remarks 1.** n = 0, 1

**2.** Setting the ESn01 and ESn00 bits to 10 is prohibited.

**Table 8-3. Capture Trigger of CR0n0 Register and Valid Edge of TI0n1 Pin**

| Capture Trigger of CR0n0 | Valid Edge of TI0n1 Pin | | |
|---|---|---|---|
| | | ESn11 | ESn10 |
| Falling edge | Falling edge | 0 | 0 |
| Rising edge | Rising edge | 0 | 1 |
| Both rising and falling edges | Both rising and falling edges | 1 | 1 |

**Remarks 1.** n = 0, 1

**2.** Setting the ESn11 and ESn10 bits to 10 is prohibited.

**Cautions 1. Set a value other than 0000H to the CR0n0 register in the mode in which clear & start occurs upon a match of the values of the TM0n register and CR0n0 register. However, if 0000H is set to the CR0n0 register in the free-running timer mode or the TI0n0 pin valid edge clear & start mode, an interrupt request signal (INTTM0n0) is generated when the value changes from 0000H to 0001H after an overflow (FFFFH).**

**2. When the P33 and P35 pins are used as the valid edges of TI000 and TI010, and the timer output function is used, set the P34 and P32 pins as the timer output pins (TO00, TO01).**

**3. If, when the CR0n0 register is used as a capture register, the register read interval and capture trigger input conflict, the read data becomes undefined (but the capture data itself is normal). Moreover, when the count stop input and capture trigger input conflict, the capture data becomes undefined.**

**4. The CR0n0 register cannot be rewritten during timer count operation.**

**(3) 16-bit timer capture/compare register 0n1 (CR0n1)**

The CR0n1 register is a 16-bit register that combines capture register and compare register functions. The CRC0n.CRC0n2 bit is used to set whether to use the CR0n1 register as a capture register or as a compare register.

The CR0n1 register can be read or written in 16-bit units.

After reset, this register is cleared to 0000H.

After reset: 0000H R/W Address: CR001 FFFFF604H, CR011 FFFFF614H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CR0n1 | | | | | | | | | | | | | | | | |

(n = 0, 1)

**(a) When using the CR0n1 register as a compare register**

The value set to the CR0n1 register and the count value of the TM0n register are always compared and when these values match, an interrupt request signal (INTTM0n1) is generated.

**(b) When using the CR0n1 register as a capture register**

The TM0n register count value is captured to the CR0n1 register by inputting a capture trigger.

The valid edge of the TI0n0 pin can be selected as the capture trigger. The valid edge of the TI0n0 pin is set with the PRM0n.ESn01 and PRM0n.ESn00 bits.

Table 8-4 shows the settings when the valid edge of the TI0n0 pin is specified as the capture trigger.

**Table 8-4. Capture Trigger of CR0n1 Register and Valid Edge of TI0n0 Pin**

| Capture Trigger of CR0n1 | Valid Edge of TI0n0 Pin | | |
|---|---|---|---|
| | | ESn01 | ESn00 |
| Falling edge | Falling edge | 0 | 0 |
| Rising edge | Rising edge | 0 | 1 |
| Both rising and falling edges | Both rising and falling edges | 1 | 1 |

**Remarks 1.** n = 0, 1

**2.** Setting the ESn01 and ESn00 bits to 10 is prohibited.

**Cautions 1. If 0000H is set to the CR0n1 register, an interrupt request signal (INTTM0n1) is generated after overflow of the TM0n register, after clear & start on a match between the TM0n register and CR0n0 register, after clear by the valid edge of the TI0n0 pin, or after clear by a one-shot pulse output trigger.**

**2. When the P33 and P35 pins are used as the valid edges of TI000 and TI010, and the timer output function is used, set the P34 and P32 pins as the timer output pins (TO00, TO01).**

**3. If, when the CR0n1 register is used as a capture register, the register read interval and capture trigger input conflict, the read data becomes undefined (but the capture data itself is normal). Moreover, when the count stop input and capture trigger input conflict, the capture data becomes undefined.**

**4. The CR0n1 register can be rewritten during TM0n register operation only in the PPG output mode. Refer to 8.4.2 PPG output operation.**

## 8.3 Registers

The registers that control 16-bit timer/event counter 0n are as follows.

- 16-bit timer mode control register 0n (TMC0n)
- Capture/compare control register 0n (CRC0n)
- 16-bit timer output control register 0n (TOC0n)
- Prescaler mode register 0n (PRM0n)
- Selector operation control register 1 (SELCNT1)

**Remark** To use the TI0n0, TI0n1, and TO0n pin functions, refer to **Table 4-14 Settings When Port Pins Are Used for Alternate Functions**.

**(1) 16-bit timer mode control register 0n (TMC0n)**

The TMC0n register is used to set the operation mode of 16-bit timer/event counter 0n, the clear mode of the TM0n register, the output timing, and to detect overflow.

The TMC0n register can be read or written in 8-bit or 1-bit units.

After reset, this register is cleared to 00H.

**Cautions 1. 16-bit timer/event counter 0n starts operating when a value other than 00 (operation stop mode) is set to the TMC0n.TMC0n3 and TMC0n.TMC0n2 bits. To stop the operation, set 00 to the TMC0n3 and TMC0n2 bits.**

**2. When the main clock is stopped and the CPU operates on the subclock, do not access the TMC0n register using an access method that causes a wait.**
**For details, refer to 3.4.8 (2).**

**Remark** n = 0, 1

After reset: 00H R/W Address: TMC00 FFFFF606H, TMC01 FFFFF616H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | <0> |
|---|---|---|---|---|---|---|---|---|
| TMC0n | 0 | 0 | 0 | 0 | TMC0n3 | TMC0n2 | TMC0n1[Note] | OVF0n |

(n = 0, 1)

| TMC0n3 | TMC0n2 | TMC0n1[Note] | Selection of operation mode and clear mode | Selection of TO0n output inverse timing | Generation of interrupt |
|---|---|---|---|---|---|
| 0 | 0 | 0 | Operation stop (TM0n cleared to 0) | Unchanged | Not generated |
| 0 | 0 | 1 | | | |
| 0 | 1 | 0 | Free-running timer mode | Match of TM0n and CR0n0 or match of TM0n and CR0n1 | Generated upon match of TM0n and CR0n0 and match of TM0n and CR0n1 |
| 0 | 1 | 1 | | Match of TM0n and CR0n0, match of TM0n and CR0n1, or valid edge of TI0n0 | |
| 1 | 0 | 0 | Clear & start with valid edge of TI0n0 | Match of TM0n and CR0n0 or match of TM0n and CR0n1 | |
| 1 | 0 | 1 | | Match of TM0n and CR0n0, match of TM0n and CR0n1, or valid edge of TI0n0 | |
| 1 | 1 | 0 | Clear & start upon match of TM0n and CR0n0 | Match of TM0n and CR0n0 or match of TM0n and CR0n1 | |
| 1 | 1 | 1 | | Match of TM0n and CR0n0, match of TM0n and CR0n1, or valid edge of TI0n0 | |

| OVF0n | Detection of overflow of 16-bit timer register 0n |
|---|---|
| 0 | No overflow |
| 1 | Overflow |

**Note** Be sure to clear the TMC0n1 bit to 0 when the TO0n pin and TI0n0 pin are used alternately.

**Cautions 1. Write to bits other than the OVF0n flag after stopping the timer operation.**

**2. The valid edge of the TI0n0 pin is set by the PRM0n register.**

**3. When the mode in which the timer is cleared and started upon match of TM0n and CR0n0 is selected, the setting value of CR0n0 is FFFFH, and when the value of TM0n changes from FFFFH to 0000H, the OVF0n flag is set to 1.**

**Remark** TO0n: Output pin of 16-bit timer/event counter 0n
TI0n0: Input pin of 16-bit timer/event counter 0n
TM0n: 16-bit timer counter 0n
CR0n0: 16-bit timer capture/compare register 0n0
CR0n1: 16-bit timer capture/compare register 0n1

**(2) Capture/compare control register 0n (CRC0n)**

The CRC0n register controls the operation of the CR0n0 and CR0n1 registers.

The CRC0n register can be read or written in 8-bit or 1-bit units.

After reset, CRC0n is cleared to 00H.

After reset: 00H R/W Address: CRC00 FFFFF608H, CRC01 FFFFF618H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CRC0n | 0 | 0 | 0 | 0 | 0 | CRC0n2 | CRC0n1 | CRC0n0 |

(n = 0, 1)

| CRC0n2 | Selection of operation mode of CR0n1 register |
|---|---|
| 0 | Operation as compare register |
| 1 | Operation as capture register |

| CRC0n1 | Selection of capture trigger of CR0n0 register |
|---|---|
| 0 | Capture at valid edge of TI0n1 pin |
| 1 | Capture at inverse phase of valid edge of TI0n0 pin |

| CRC0n0 | Selection of operation mode of CR0n0 register |
|---|---|
| 0 | Operation as compare register |
| 1 | Operation as capture register |

**Cautions 1. Before setting the CRC0n register, be sure to stop the timer operation.**

**2. When the mode in which the timer is cleared and started upon match of the TM0n register and CR0n0 register is selected by the TMC0n register, do not specify the CR0n0 register as the capture register.**

**3. When both the rising and falling edges are specified for the TI0n0 pin valid edge, capture operation is not performed.**

**4. To ensure reliable capture operation, a pulse longer than two cycles of the count clock selected by the PRM0n and SELCNT1 registers is required.**

### (3) 16-bit timer output control register 0n (TOC0n)

The TOC0n register controls the operation of the 16-bit timer/event counter 0n output controller by setting or resetting the timer output F/F, enabling or disabling inverse output, enabling or disabling the timer of 16-bit timer/event counter 0n, enabling or disabling the one-shot pulse output operation, and selecting an output trigger for a one-shot pulse by software.

The TOC0n register can be read or written in 8-bit or 1-bit units.

After reset, TOC0n is cleared to 00H.

(1/2)

After reset: 00H R/W Address: TOC00 FFFFF609H, TOC01 FFFFF619H

| | 7 | <6> | <5> | 4 | <3> | <2> | 1 | <0> |
|---|---|---|---|---|---|---|---|---|
| TOC0n | 0 | OSPT0n | OSPE0n | TOC0n4 | LVS0n | LVR0n | TOC0n1 | TOE0n |

(n = 0, 1)

| OSPT0n | Output trigger for one-shot pulse by software |
|---|---|
| 0 | – |
| 1 | One-shot pulse output |

| OSPE0n | Control of one-shot pulse output operation |
|---|---|
| 0 | Successive pulse output |
| 1 | One-shot pulse output[Note] |

| TOC0n4 | Control of timer output F/F upon match of CR0n1 register and TM0n register |
|---|---|
| 0 | Inversion operation disabled |
| 1 | Inversion operation enabled |

| LVS0n | LVR0n | Setting of status of timer output F/F |
|---|---|---|
| 0 | 0 | Unchanged |
| 0 | 1 | Reset timer output F/F (0) |
| 1 | 0 | Set timer output F/F (1) |
| 1 | 1 | Setting prohibited |

| TOC0n1 | Control of timer output F/F upon match of CR0n0 register and TM0n register |
|---|---|
| 0 | Inversion operation disabled |
| 1 | Inversion operation enabled |

| TOE0n | Control of timer output |
|---|---|
| 0 | Output disabled (output is fixed to low level) |
| 1 | Output enabled |

**Note** The one-shot pulse output operates normally in the free-running timer mode and the mode in which clear & start occurs on the valid edge of the TI0n0 pin. In the mode in which clear & start occurs on match between the TM0n register and the CR0n0 register, one-shot pulse output is not performed because no overflow occurs.

(2/2)

**Cautions 1. Be sure to stop the timer operation before setting other than the TOC0n4 bit.**

**2. The LVS0n and LVR0n bits are 0 when read.**

**3. The OSPT0n bit is 0 when read because it is automatically cleared after data has been set.**

**4. Do not set the OSPT0n bit (1) other than for one-shot pulse output.**

**5. When performing successive writes to the OSPT0n bit, place an interval between writes of two or more cycles of the count clock selected by the PRM0n register.**

**6. Do not set the LVS0n bit (1) before setting the TOE0n bit.**
**Do not set the LVS0n bit and TOE0n bit (1) at the same time.**

**7. Do not set <1> and <2> below at the same time. Set as follows.**
**<1> Set the TOC0n1, TOC0n4, TOE0n, and OSPE0n bits: Setting of timer output operation**
**<2> Set the LVS0n and LVR0n bits: Setting of timer output F/F**

**(4) Prescaler mode register 0n (PRM0n)**

The PRM0n register sets the count clock of the TM0n register and the valid edge of the TI0n0 and TI0n1 pin inputs. The PRMn01 and PRMn00 bits are set in combination with the SELCNT1.ISEL1n bit. Refer to **8.3 (6) Count clock setting for 16-bit timer/event counter 0n** for details.

The PRM0n register can be read or written in 8-bit or 1-bit units.

After reset, PRM0n is cleared to 00H.

**Cautions 1. When setting the count clock to the TI0n0 pin valid edge, do not set the mode in which clear & start occurs on TI0n0 pin valid edge and do not set the TI0n0 pin as a capture trigger.**

**2. Before setting the PRM0n register, be sure to stop the timer operation.**

**3. If 16-bit timer/event counter 0n operation is enabled by specifying the rising edge or both edges for the valid edge of the TI0n0 pin or TI0n1 pin while the TI0n0 pin or TI0n1 pin is high level immediately after system reset, the rising edge is detected immediately after the rising edge or both edges is specified. Be careful when pulling up the TI0n0 pin or TI0n1 pin. However, the rising edge is not detected when operation is enabled after it has been stopped.**

**4. When the P33 and P35 pins are used as the valid edges of TI000 and TI010, and the timer output function is used, set the P34 and P32 pins as the timer output pins (TO00, TO01).**

After reset: 00H R/W Address: PRM00 FFFFF607H, PRM01 FFFFF617H

PRM0n (n = 0, 1)

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| ESn11 | ESn10 | ESn01 | ESn00 | 0 | 0 | PRMn01 | PRMn00 |

| ESn11 | ESn10 | Selection of valid edge of TI0n1 |
|---|---|---|
| 0 | 0 | Falling edge |
| 0 | 1 | Rising edge |
| 1 | 0 | Setting prohibited |
| 1 | 1 | Both rising and falling edges |

| ESn01 | ESn00 | Selection of valid edge of TI0n0 |
|---|---|---|
| 0 | 0 | Falling edge |
| 0 | 1 | Rising edge |
| 1 | 0 | Setting prohibited |
| 1 | 1 | Both rising and falling edges |

**(5) Selector operation control register 1 (SELCNT1)**

The SELCNT1 register sets the count clock of 16-bit timer/event counter 0n.

The SELCNT1 register can be read or written in 8-bit or 1-bit units.

After reset, SELCNT1 is cleared to 00H.

The SELCNT1 register is set in combination with the PRM0n.PRMn01 and PRM0n.PRMn00 bits. Refer to **8.3 (6) Count clock setting for 16-bit timer/event counter 0n** for details.

After reset: 00H R/W Address: FFFFF30AH

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| SELCNT1 | 0 | 0 | 0 | 0 | 0 | 0 | ISEL11 | ISEL10 |

**(6) Count clock setting for 16-bit timer/event counter 0n**

The count clock for 16-bit timer/event counter 0n is set by using the PRM0n.PRMn01, PRM0n.PRMn00, and SELCNT1.ISEL1n bits in combination.

**(a) Count clock for 16-bit timer/event counter 00**

| SELCNT1 Register | PRM00 Register | | Selection of Count Clock[Note 1] | | | |
|---|---|---|---|---|---|---|
| ISEL10 Bit | PRM001 Bit | PRM000 Bit | Count Clock | $f_{XX}$ = 20 MHz | $f_{XX}$ = 16 MHz | $f_{XX}$ = 10 MHz |
| 0 | 0 | 0 | $f_{XX}/2$ | 100 ns | 125 ns | 200 ns |
| 0 | 0 | 1 | $f_{XX}/4$ | 200 ns | 250 ns | 400 ns |
| 0 | 1 | 0 | $f_{XX}/8$ | 400 ns | 500 ns | 800 ns |
| 0 | 1 | 1 | Valid edge of TI000[Note 2] | – | – | – |
| 1 | 0 | 0 | $f_{XX}/32$ | 1.6 $\mu$s | 2.0 $\mu$s | 3.2 $\mu$s |
| 1 | 0 | 1 | $f_{XX}/64$ | 3.2 $\mu$s | 4.0 $\mu$s | 6.4 $\mu$s |
| 1 | 1 | 0 | $f_{XX}/128$ | 6.4 $\mu$s | 8.0 $\mu$s | 12.8 $\mu$s |
| 1 | 1 | 1 | Setting prohibited | | | |

**Notes 1.** When the internal clock is selected, set so as to satisfy the following conditions:

$V_{DD}$ = REGC = 4.0 to 5.5 V: Count clock ≤ 10 MHz

$V_{DD}$ = 4.0 to 5.5 V, REGC = Capacity: Count clock ≤ 5 MHz

$V_{DD}$ = REGC = 2.7 to 4.0 V: Count clock ≤ 5 MHz

**2.** The external clock requires a pulse longer than two cycles of the internal clock ($f_{XX}/4$).

**(b) Count clock for 16-bit timer/event counter 01**

| SELCNT1 Register | PRM01 Register | | Selection of Count Clock[Note 1] | | | |
|---|---|---|---|---|---|---|
| ISEL11 Bit | PRM011 Bit | PRM010 Bit | Count Clock | $f_{XX}$ = 20 MHz | $f_{XX}$ = 16 MHz | $f_{XX}$ = 10 MHz |
| 0 | 0 | 0 | $f_{XX}$ | Setting prohibited | Setting prohibited | 100 ns |
| 0 | 0 | 1 | $f_{XX}/4$ | 200 ns | 250 ns | 400 ns |
| 0 | 1 | 0 | INTWT | – | – | – |
| 0 | 1 | 1 | Valid edge of TI010[Note 2] | – | – | – |
| 1 | 0 | 0 | $f_{XX}/2$ | 100 ns | 125 ns | 200 ns |
| 1 | 0 | 1 | $f_{XX}/8$ | 400 ns | 500 ns | 800 ns |
| 1 | 1 | 0 | $f_{XX}/16$ | 800 ns | 1.0 $\mu$s | 1.6 $\mu$s |
| 1 | 1 | 1 | Setting prohibited | | | |

**Notes 1.** When the internal clock is selected, set so as to satisfy the following conditions:

$V_{DD}$ = REGC = 4.0 to 5.5 V: Count clock $\leq$ 10 MHz

$V_{DD}$ = 4.0 to 5.5 V, REGC = Capacity: Count clock $\leq$ 5 MHz

$V_{DD}$ = REGC = 2.7 to 4.0 V: Count clock $\leq$ 5 MHz

**2.** The external clock requires a pulse longer than two cycles of the internal clock ($f_{XX}/4$).

## 8.4 Operation

### 8.4.1 Operation as interval timer

16-bit timer/event counter 0n can be made to operate as an interval timer by setting the TMC0n register and the CRC0n register as shown in Figure 8-2.

Setting procedure

The basic operation setting procedure is as follows.

<1> Set the count clock using the PRM0n register and the SELCNT1 register.
<2> Set the CRC0n register (refer to **Figure 8-2** for the setting value).
<3> Set any value to the CR0n0 register.
<4> Set the TMC0n register: Start operation (refer to **Figure 8-2** for the setting value).

**Caution The CR0n0 register cannot be rewritten during 16-bit timer/event counter 0n operation.**

**Remarks 1.** For the alternate-function pin settings, refer to **Table 4-14 Settings When Port Pins Are Used for Alternate Functions**.

**2.** For INTTM0n0 interrupt enable, refer to **CHAPTER 19 INTERRUPT/EXCEPTION PROCESSING FUNCTION**.

The interval timer repeatedly generates interrupts at the interval of the preset count value in the CR0n0 register.

If the count value in the TM0n register matches the value set in the CR0n0 register, an interrupt request signal (INTTM0n0) is generated at the same time that the value of the TM0n register is cleared to 0000H and counting is continued.

The count clock of 16-bit timer/event counter 0n can be selected with the PRM0n.PRM0n0, PRM0n.PRM0n1, and SELCNT1.ISEL1n bits.

**Remark** n = 0, 1

**Figure 8-2. Control Register Settings in Interval Timer Operation**

**(a) 16-bit timer mode control register 0n (TMC0n)**

| | | | | | TMC0n3 | TMC0n2 | TMC0n1 | OVF0n |
|---|---|---|---|---|---|---|---|---|
| TMC0n | 0 | 0 | 0 | 0 | 1 | 1 | 0/1[Note] | 0 |

Clears & starts upon match between TM0n and CR0n0

**(b) Capture/compare control register 0n (CRC0n)**

| | | | | | | CRC0n2 | CRC0n1 | CRC0n0 |
|---|---|---|---|---|---|---|---|---|
| CRC0n | 0 | 0 | 0 | 0 | 0 | 0/1 | 0/1 | 0 |

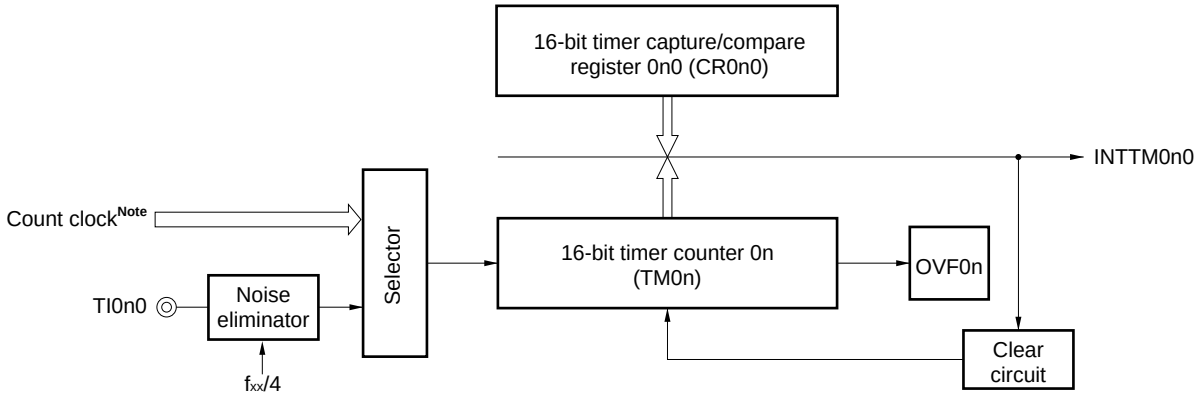
CR0n0 used as compare register

**Note** Be sure to clear the TMC0n1 bit to 0 when the TO0n pin and TI0n0 pin are used alternately.

**Remarks 1.** 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 **8.3 (1) 16-bit timer mode control register 0n (TMC0n)** and **8.3 (2) Capture/compare control register 0n (CRC0n)**.

**2.** n = 0, 1

**Figure 8-3. Configuration of Interval Timer**

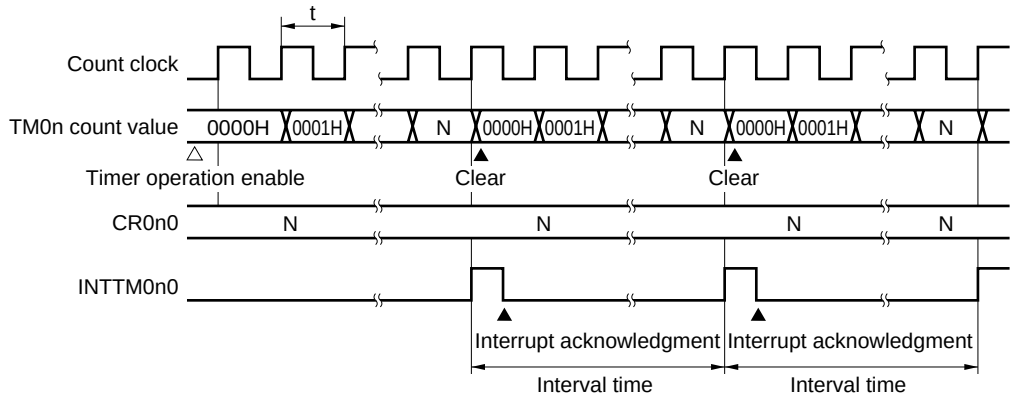



**Note** Set with the PRM0n register and SELCNT1 register.

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

**2.** n = 0, 1

**Figure 8-4. Timing of Interval Timer Operation**




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

**2.** n = 0, 1

### 8.4.2 PPG output operation

16-bit timer/event counter 0n can be used for PPG (Programmable Pulse Generator) output by setting the TMC0n register and the CRC0n register as shown in Figure 8-5.

Setting procedure

The basic operation setting procedure is as follows.

<1> Set the CRC0n register (refer to **Figure 8-5** for the setting value).
<2> Set any value as a cycle to the CR0n0 register.
<3> Set any value as a duty to the CR0n1 register.
<4> Set the TOC0n register (refer to **Figure 8-5** for the setting value).
<5> Set the count clock using the PRM0n register and SELCNT1 register.
<6> Set the TMC0n register: Start operation (refer to **Figure 8-5** for the setting value).

**Caution To change the duty value (CR0n1 register) during operation, refer to Remark 2 in Figure 8-7 PPG Output Operation Timing.**

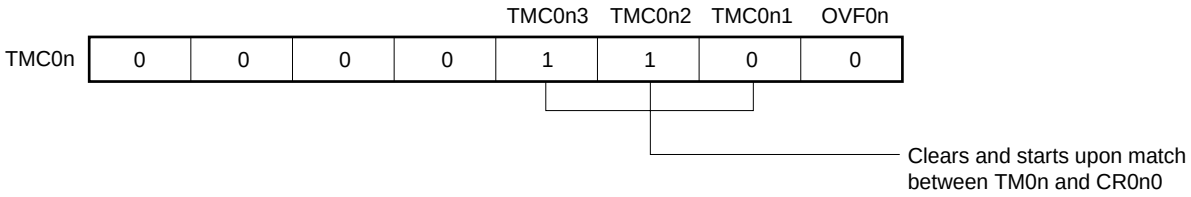
**Remarks 1.** For the alternate-function pin (TO0n) settings, refer to **Table 4-14 Settings When Port Pins Are Used for Alternate Functions**.

**2.** For INTTM0n0 interrupt enable, refer to **CHAPTER 19 INTERRUPT/EXCEPTION PROCESSING FUNCTION**.

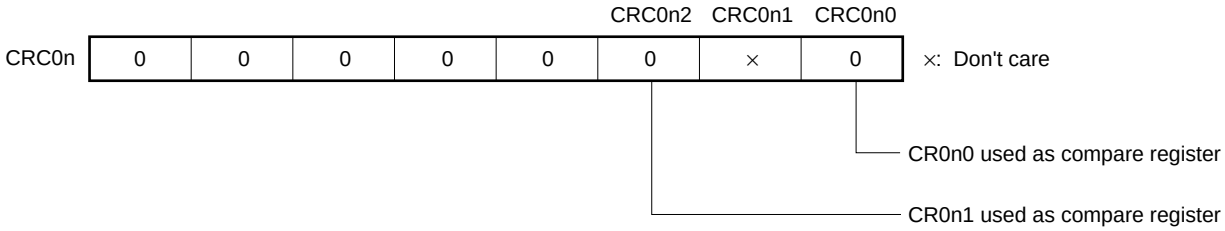
The PPG output function outputs a rectangular wave from the TO0n pin with the cycle specified by the count value set in advance to the CR0n0 register and the pulse width specified by the count value set in advance to the CR0n1 register.

**Figure 8-5. Control Register Settings in PPG Output Operation**

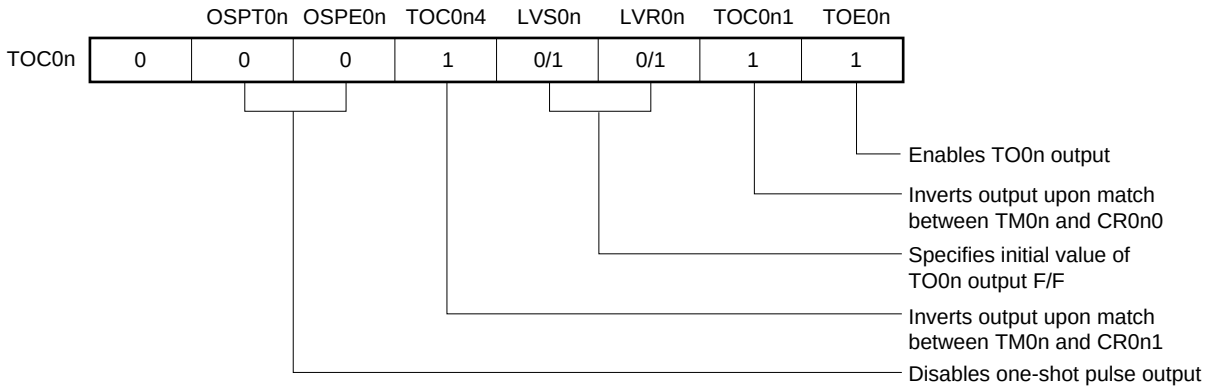
**(a) 16-bit timer mode control register 0n (TMC0n)**



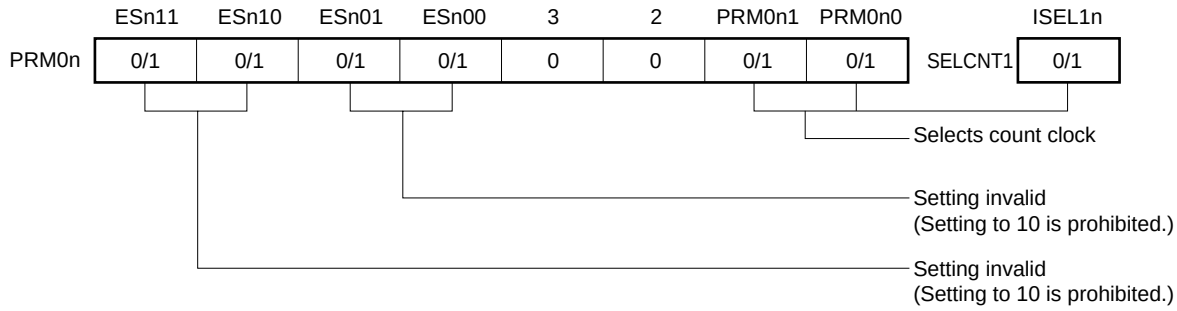

**(b) Capture/compare control register 0n (CRC0n)**




**(c) 16-bit timer output control register 0n (TOC0n)**




**(d) Prescaler mode register 0n (PRM0n) and selector operation control register 1 (SELCNT1)**

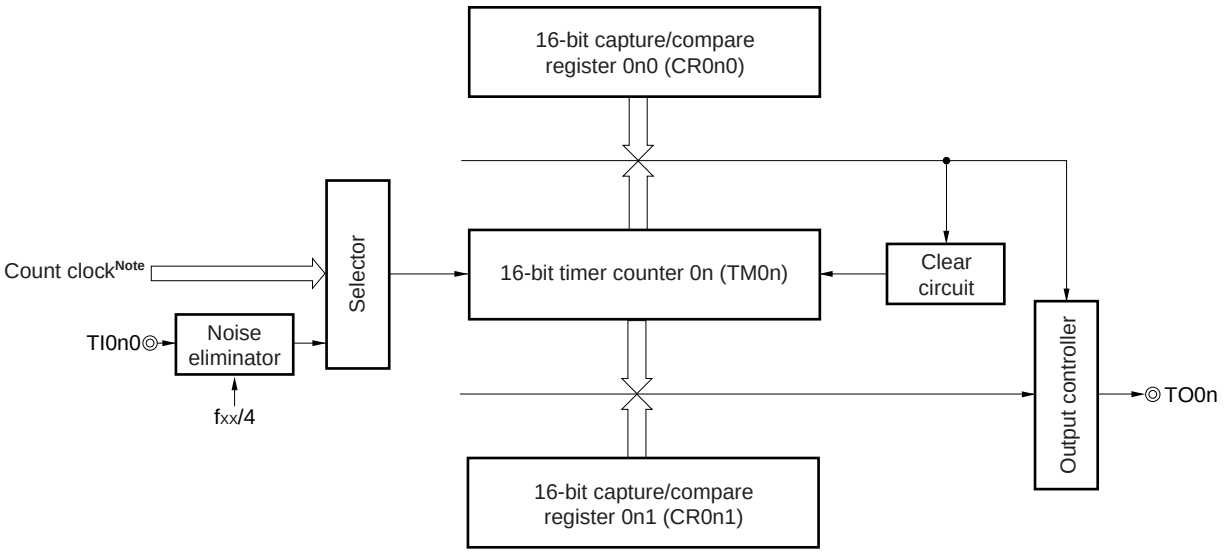



**Cautions 1. Make sure that 0000H ≤ CR0n1 < CR0n0 ≤ FFFFH is set to the CR0n0 register and CR0n1 register.**

**2. The cycle of the pulse generated by PPG output is (CR0n0 setting value + 1).**
**The duty factor is (CR0n1 setting value + 1) / (CR0n0 setting value + 1).**

**Remark** n = 0, 1

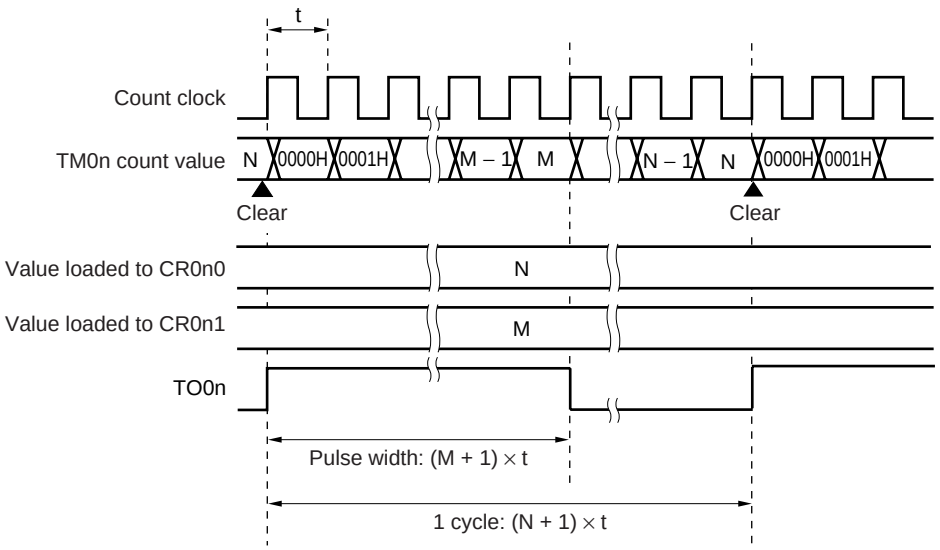
**Figure 8-6. Configuration of PPG Output**




**Note** The count clock is set with the PRM0n register and SELCNT1 register.

**Remark** n = 0, 1

**Figure 8-7. PPG Output Operation Timing**




**Caution The CR0n0 register cannot be rewritten during 16-bit timer/event counter 0n operation.**

**Remarks 1.** 0000H ≤ M < N ≤ FFFFH

**2.** Change the pulse width during 16-bit timer/event counter 0n operation (rewrite CR0n1 register) as follows in a PPG output operation.

<1> Disable the timer output inversion operation based on a match of the TM0n and CR0n1 registers (TOC0n4 bit = 0).
<2> Disable the INTTM0n1 interrupt (TM0MKn1 bit =1).
<3> Rewrite the CR0n1 register.
<4> Wait for a cycle of the TM0n register count clock.
<5> Enable the timer output inversion operation based on a match of the TM0n and CR0n1 registers (TOC0n4 bit = 1).
<6> Clear the interrupt request flag of INTTM0n1 (TM0IFn1 bit = 0).
<7> Enable the INTTM0n1 interrupt (TM0MKn1 bit = 0).

**3.** n = 0, 1

### 8.4.3 Pulse width measurement

The TM0n register can be used to measure the pulse widths of the signals input to the TI0n0 and TI0n1 pins.

Measurement can be carried out with 16-bit timer/event counter 0n used in the free-running timer mode or by restarting the timer in synchronization with the edge of the signal input to the TI0n0 pin.

When an interrupt is generated, read the valid capture register value. After confirming the TMC0n.OVF0n flag, clear it (0) by software and measure the pulse width.

Setting procedure

The basic operation setting procedure is as follows.

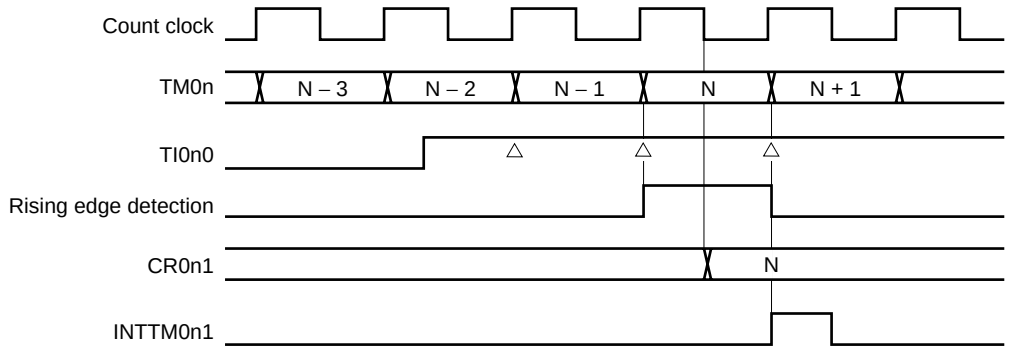
<1> Set the CRC0n register (refer to **Figures 8-9**, **8-12**, **8-14**, and **8-16** for the setting value).
<2> Set the count clock using the PRM0n register and SELCNT1 register.
<3> Set the TMC0n register: Start operation (refer to **Figures 8-9**, **8-12**, **8-14**, and **8-16** for the setting value).

**Caution When using two capture registers, set the TI0n0 and TI0n1 pins.**

**Remarks 1.** For the alternate-function pin (TI0n0, TI0n1) settings, refer to **Table 4-14 Settings When Port Pins Are Used for Alternate Functions**.

**2.** For INTTM0n0 and INTTM0n1 interrupt enable, refer to **CHAPTER 19 INTERRUPT/EXCEPTION PROCESSING FUNCTION**.

**Figure 8-8. CR0n1 Capture Operation with Rising Edge Specified**




**Remarks 1.** n = 0, 1

**2.** The valid edge is detected through sampling at a count clock cycle selected with the PRM0n register and SELCNT1 register, and the capture operation is not performed until the valid edge is detected twice. As a result, noise with a short pulse width can be eliminated.

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

If the edge specified by the PRM0n register is input to the TI0n0 pin when 16-bit timer/event counter 0n is operated in the free-running timer mode (refer to **Figure 8-9**), the value of the TM0n register is loaded to the CR0n1 register and an external interrupt request signal (INTTM0n1) is generated.

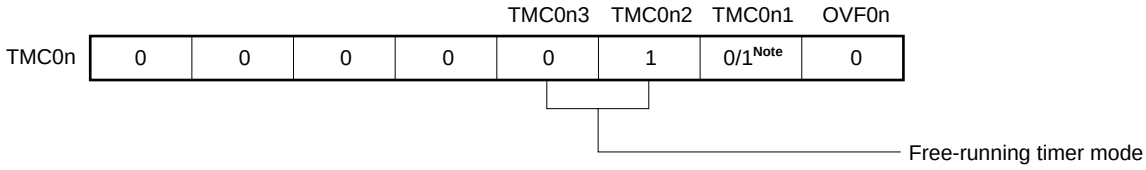
The valid edge is specified by the PRM0n.ESn00 and PRM0n.ESn01 bits. The rising edge, falling edge, or both the rising and falling edges can be selected.

The valid edge is detected through sampling at a count clock cycle selected with the PRM0n register and SELCNT1 register, and the capture operation is not performed until the valid edge is detected twice. As a result, noise with a short pulse width can be eliminated.

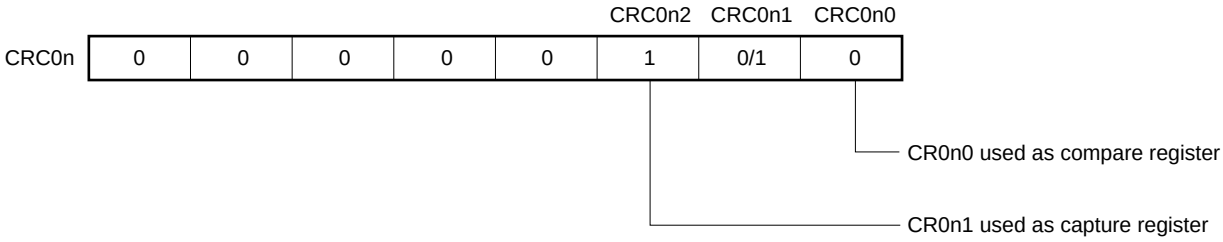
**Remark** n = 0, 1

**Figure 8-9. Control Register Settings for Pulse Width Measurement with Free-Running Timer Operation and One Capture Register (When TI0n0 Pin and CR0n1 Register Are Used)**

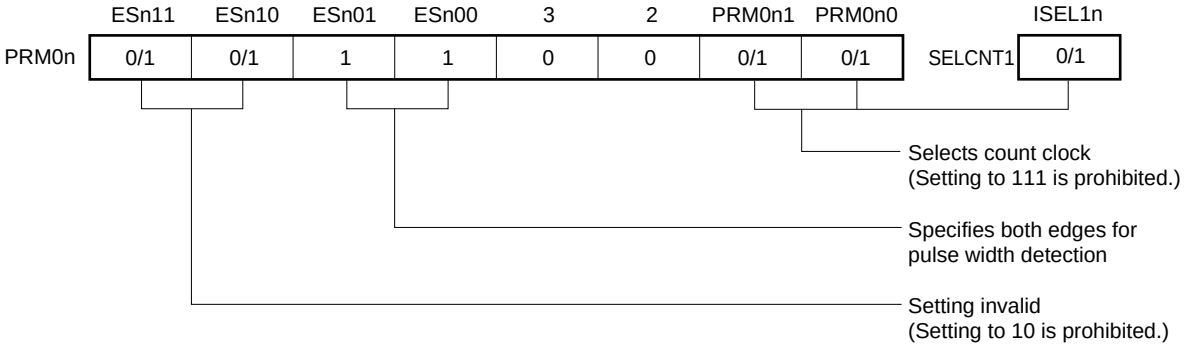
**(a) 16-bit timer mode control register 0n (TMC0n)**




**(b) Capture/compare control register 0n (CRC0n)**




**(c) Prescaler mode register 0n (PRM0n) and selector operation control register 1 (SELCNT1)**

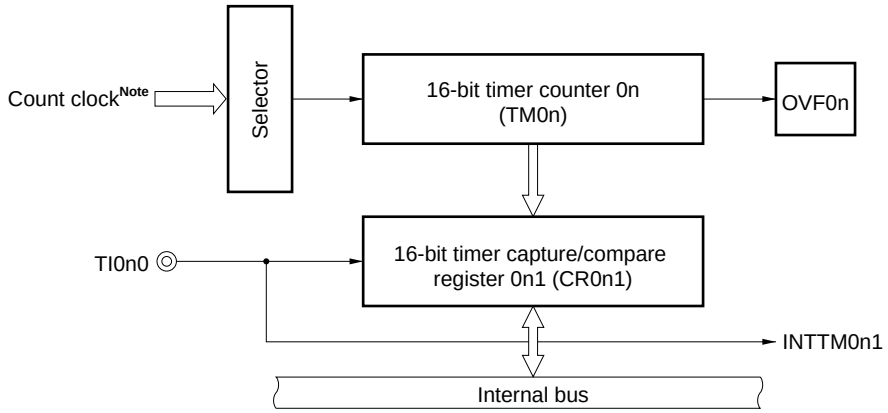



**Note** Be sure to clear the TMC0n1 bit to 0 when the TO0n pin and TI0n0 pin are used alternately.

**Remarks 1.** 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 **8.3 (1) 16-bit timer mode control register 0n (TMC0n)** and **8.3 (2) Capture/compare control register 0n (CRC0n)**.

**2.** n = 0, 1

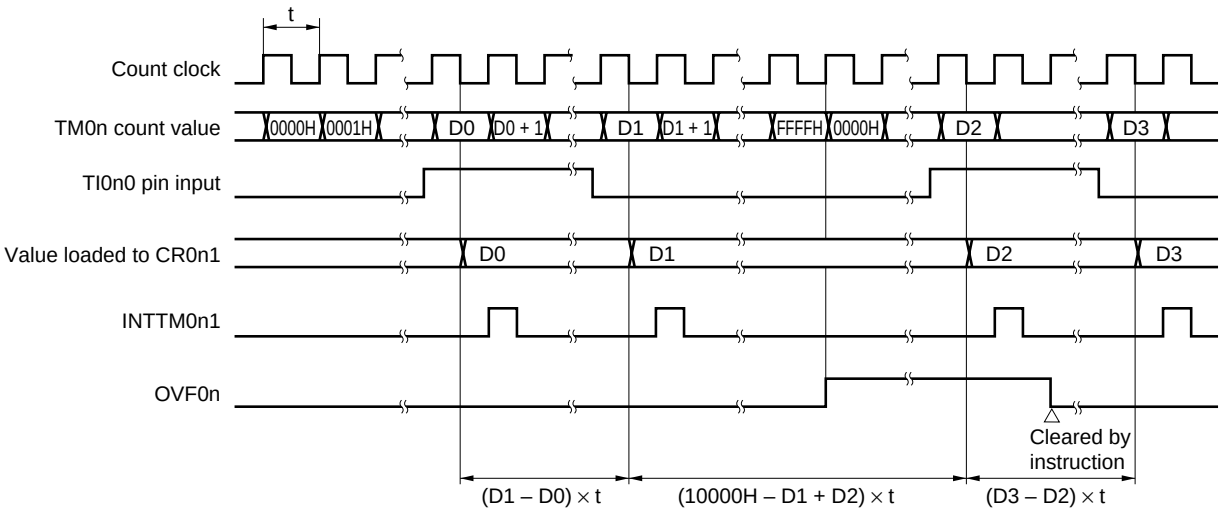
**Figure 8-10. Configuration for Pulse Width Measurement with Free-Running Timer Operation**




**Note** The count clock is set with the PRM0n register and SELCNT1 register.

**Remark** n = 0, 1

**Figure 8-11. Timing of Pulse Width Measurement with Free-Running Timer Operation and One Capture Register (with Both Edges Specified)**




**Remark** n = 0, 1

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

The pulse widths of two signals respectively input to the TI0n0 pin and the TI0n1 pin can be simultaneously measured when 16-bit timer/event counter 0n is used in the free-running timer mode (refer to **Figure 8-12**).

When the edge specified by the PRM0n.ESn00 and PRM0n.ESn01 bits is input to the TI0n0 pin, the value of the TM0n register is loaded to the CR0n1 register and an external interrupt request signal (INTTM0n1) is generated.

When the edge specified by the PRM0n.ESn10 and PRM0n.ESn11 bits is input to the TI0n1 pin, the value of the TM0n register is loaded to the CR0n0 register and an external interrupt request signal (INTTM0n0) is generated.

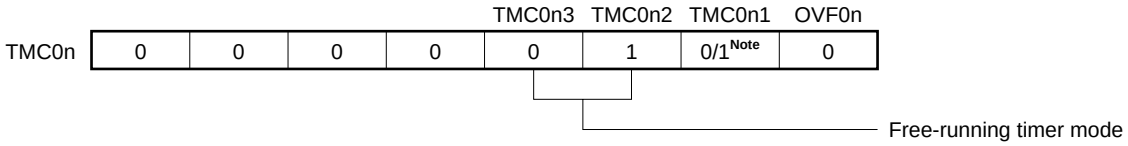
The edges of the TI0n0 and TI0n1 pins are specified by the PRM0n.ESn00 and PRM0n.ESn01 bits and the PRM0n.ESn10 and PRM0n.ESn11 bits, respectively. Specify both rising and falling edges.

The valid edge of the TI0n0 pin is detected through sampling at the count clock cycle selected with the PRM0n register and SELCNT1 register, and the capture operation is not performed until the valid level is detected twice. As a result, noise with a short pulse width can be eliminated.

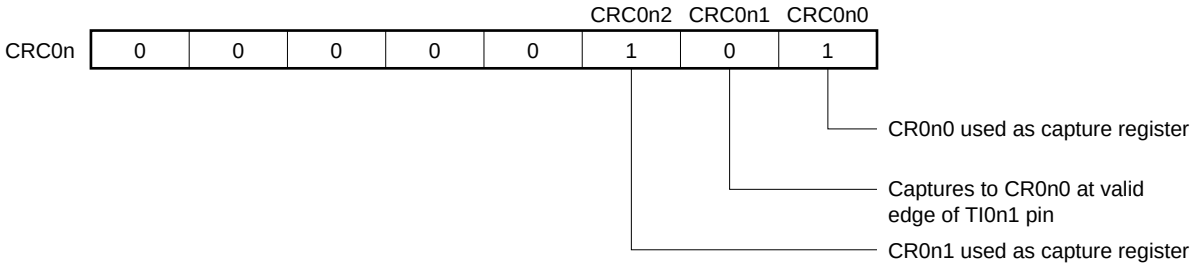
**Remark** n = 0, 1

**Figure 8-12. Control Register Settings for Measurement of Two Pulse Widths with Free-Running Timer Operation**

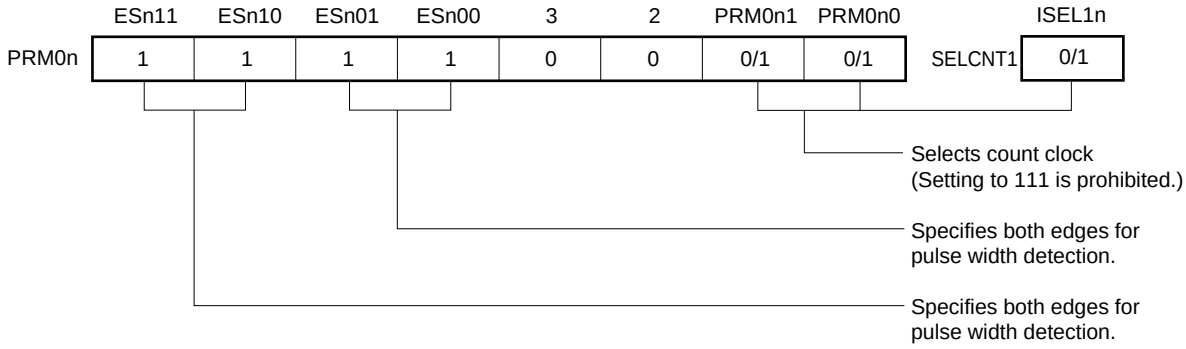
**(a) 16-bit timer mode control register 0n (TMC0n)**




**(b) Capture/compare control register 0n (CRC0n)**




**(c) Prescaler mode register 0n (PRM0n) and selector operation control register 1 (SELCNT1)**



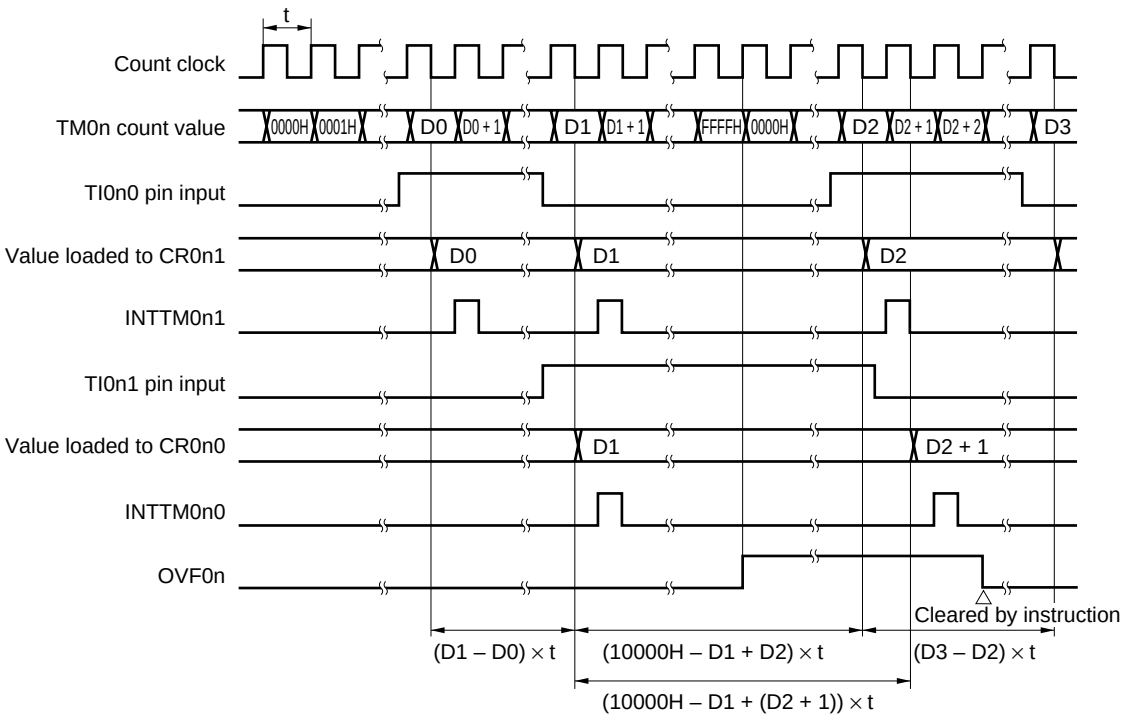

**Note** Be sure to clear the TMC0n1 bit to 0 when the TO0n pin and TI0n0 pin are used alternately.

**Remarks 1.** 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 **8.3 (1) 16-bit timer mode control register 0n (TMC0n)**.

**2.** n = 0, 1

- Capture operation (free-running timer mode)
  The following figure illustrates the operation of the capture register when the capture trigger is input.

**Figure 8-13. Timing of Pulse Width Measurement with Free-Running Timer Operation (with Both Edges Specified)**




**Remark** n = 0, 1

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

When 16-bit timer/event counter 0n is used in the free-running timer mode (refer to **Figure 8-14**), the pulse width of the signal input to the TI0n0 pin can be measured.

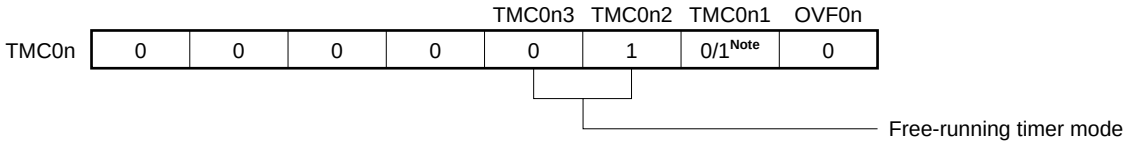
When the edge specified by the PRM0n.ESn00 and PRM0n.ESn01 bits is input to the TI0n0 pin, the value of the TM0n register is loaded to the CR0n1 register and an external interrupt request signal (INTTM0n1) is generated.

The value of the TM0n register is also loaded to the CR0n0 register when an edge inverse to the one that triggers capturing to the CR0n1 register is input.

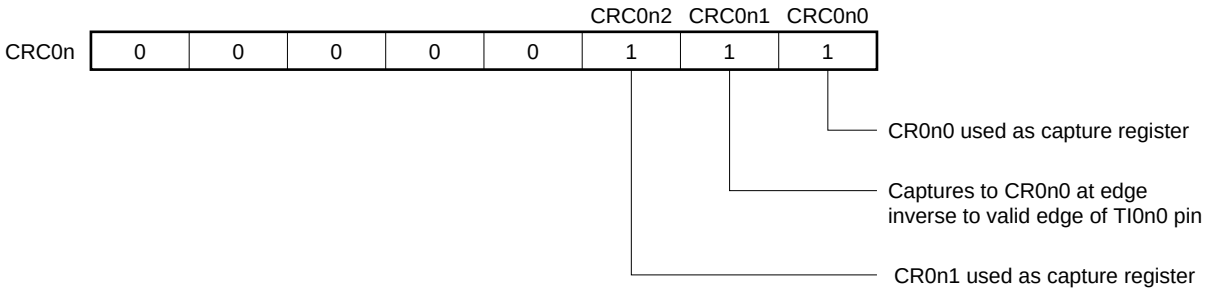
The valid edge of the TI0n0 pin is detected through sampling at a count clock cycle selected with the PRM0n register and SELCNT1 register, and the capture operation is not performed until the valid edge is detected twice. As a result, noise with a short pulse width can be eliminated.

**Figure 8-14. Control Register Settings for Pulse Width Measurement with Free-Running Timer Operation and Two Capture Registers (with Rising Edge Specified)**

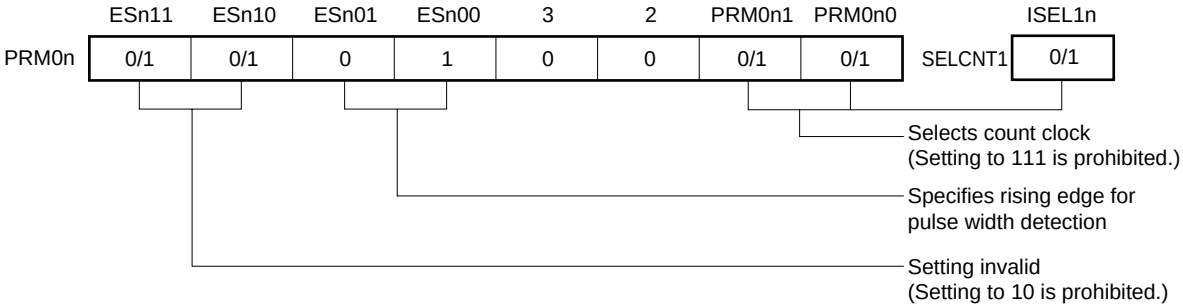
**(a) 16-bit timer mode control register 0n (TMC0n)**




**(b) Capture/compare control register 0n (CRC0n)**




**(c) Prescaler mode register 0n (PRM0n) and selector operation control register 1 (SELCNT1)**

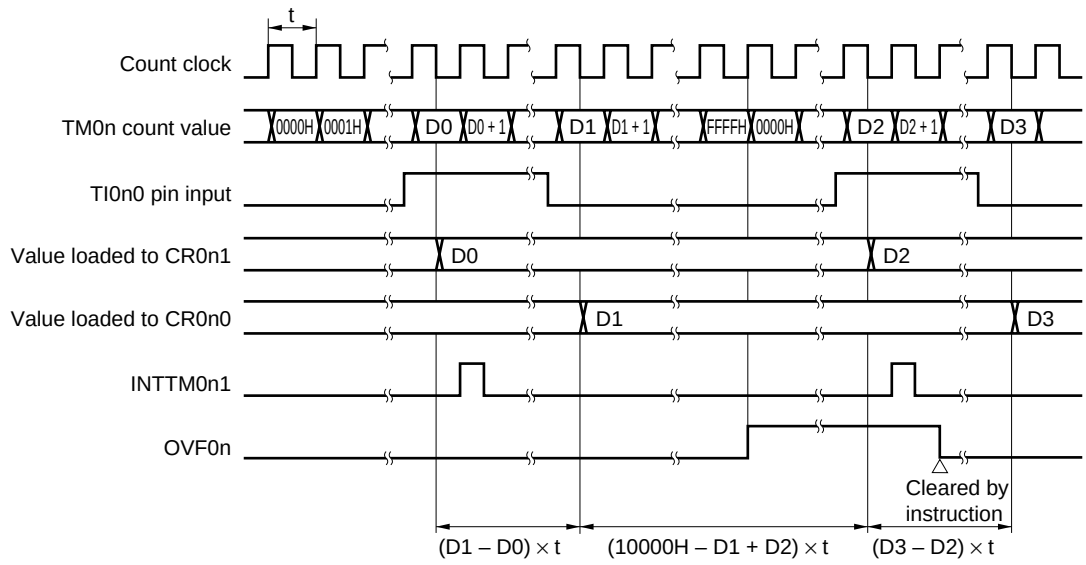



**Note** Be sure to clear the TMC0n1 bit to 0 when the TO0n pin and TI0n0 pin are used alternately.

**Remarks 1.** 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 **8.3 (1) 16-bit timer mode control register 0n (TMC0n)**.

**2.** n = 0, 1

**Figure 8-15. Timing of Pulse Width Measurement with Free-Running Timer Operation and Two Capture Registers (with Rising Edge Specified)**




**Remark** n = 0, 1

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

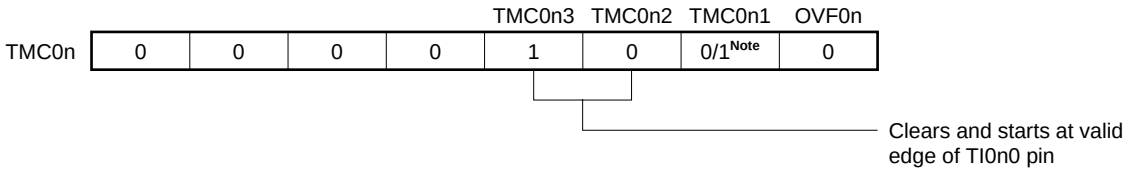
When the valid edge of the TI0n0 pin is detected, the pulse width of the signal input to the TI0n0 pin can be measured by clearing the TM0n register and then resuming counting after loading the count value of the TM0n register to the CR0n1 register (refer to **Figure 8-17**).

The edge is specified by the PRM0n.ESn00 and PRM0n.ESn01 bits. The rising or falling edge can be specified.

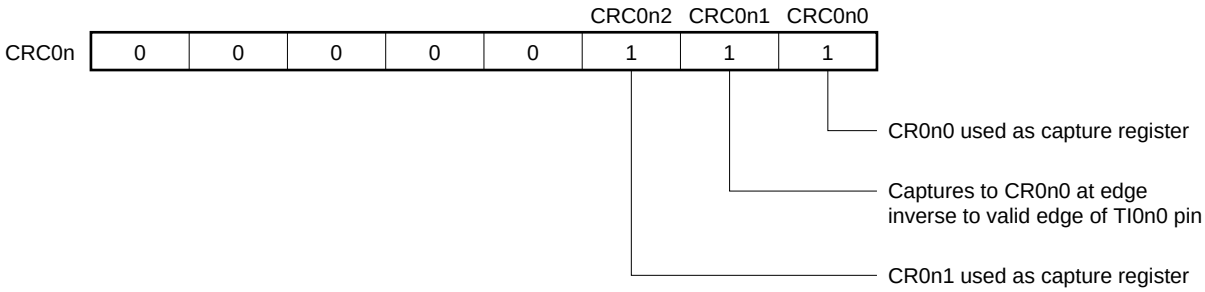
The valid edge is detected through sampling at a count clock cycle selected with the PRM0n register and SELCNT1 register, and the capture operation is not performed until the valid level is detected twice. As a result, noise with a short pulse width can be eliminated.

**Figure 8-16. Control Register Settings for Pulse Width Measurement by Restarting**

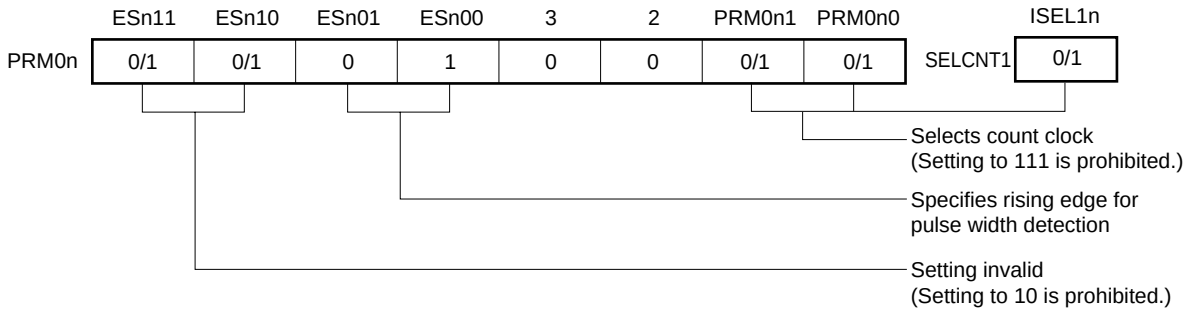
**(a) 16-bit timer mode control register 0n (TMC0n)**




**(b) Capture/compare control register 0n (CRC0n)**




**(c) Prescaler mode register 0n (PRM0n) and selector operation control register 1 (SELCNT1)**

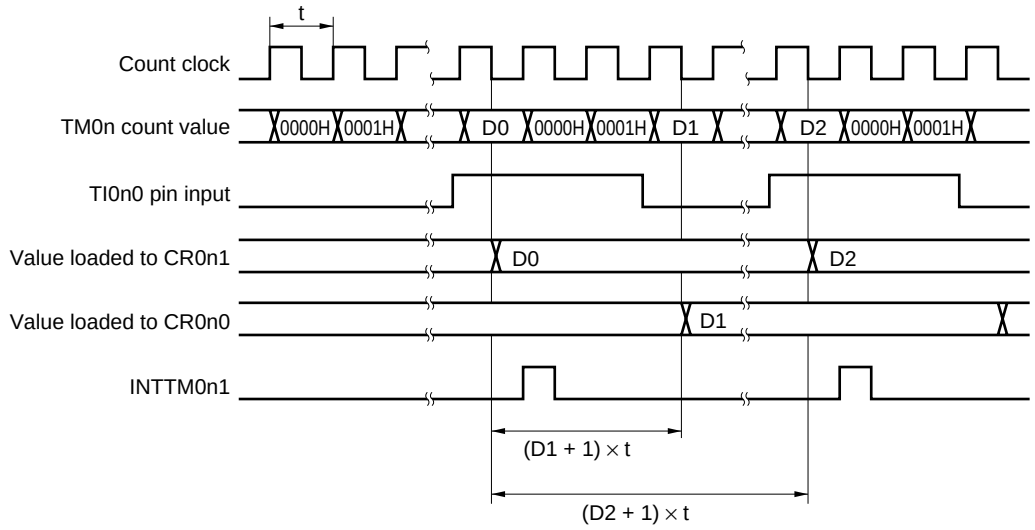



**Note** Be sure to clear the TMC0n1 bit to 0 when the TO0n pin and TI0n0 pin are used alternately.

**Remarks 1.** 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 **8.3 (1) 16-bit timer mode control register 0n (TMC0n)**.

**2.** n = 0, 1

**Figure 8-17. Timing of Pulse Width Measurement by Restarting (with Rising Edge Specified)**




**Remark** n = 0, 1

### 8.4.4 Operation as external event counter

Setting procedure

The basic operation setting procedure is as follows.

<1> Set the CRC0n register (refer to **Figure 8-18** for the setting value).
<2> Set the count clock using the PRM0n register and SELCNT1 register.
<3> Set any value (except for 0000H) to the CR0n0 register.
<4> Set the TMC0n register: Start operation (refer to **Figure 8-18** for the setting value).

**Remarks 1.** For the alternate-function pin (TI0n0) settings, refer to **Table 4-14 Settings When Port Pins Are Used for Alternate Functions**.

**2.** For INTTM0n0 interrupt enable, refer to **CHAPTER 19 INTERRUPT/EXCEPTION PROCESSING FUNCTION**.

The external event counter counts the number of clock pulses input to the TI0n0 pin from an external source by using the TM0n register.

Each time the valid edge specified by the PRM0n register has been input, the TM0n register is incremented.

When the count value of the TM0n register matches the value of the CR0n0 register, the TM0n register is cleared to 0000H and an interrupt request signal (INTTM0n0) is generated.

Set the CR0n0 register to a value other than 0000H (one-pulse count operation is not possible).

The edge is specified by the PRM0n.ESn00 and PRM0n.ESn01 bits. The rising, falling, or both the rising and falling edges can be specified.

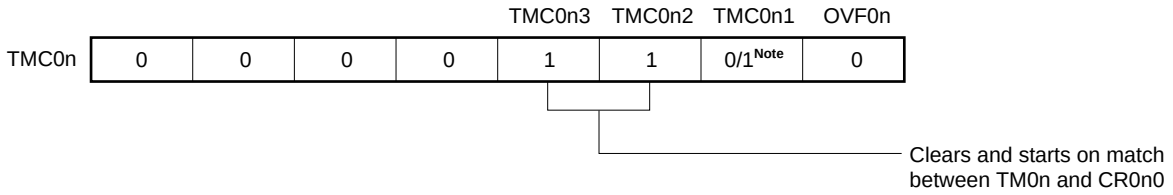
The valid edge is detected through sampling at a count clock cycle of $f_{XX}/4$, and the capture operation is not performed until the valid level is detected twice. As a result, noise with a short pulse width can be eliminated.

**Caution The value of the CR0n0 and CR0n1 registers cannot be changed during timer count operation.**

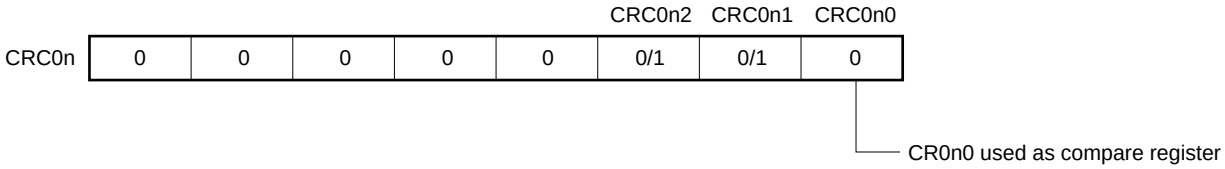
**Remark** n = 0, 1

**Figure 8-18. Control Register Settings in External Event Count Mode (with Rising Edge Specified)**

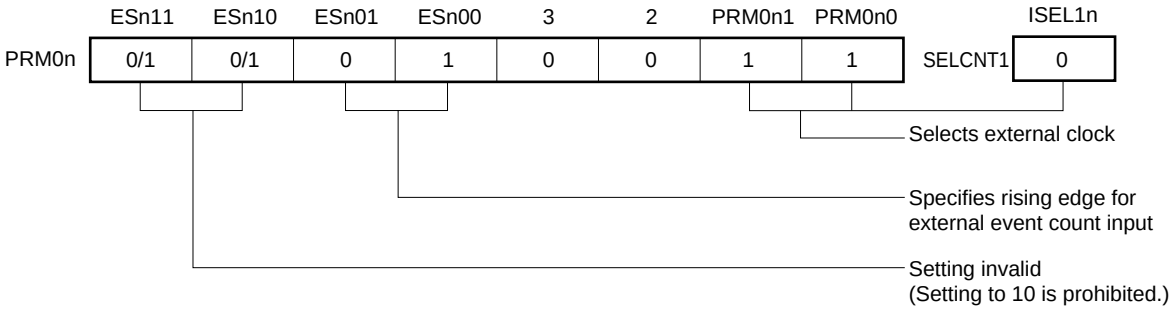
**(a) 16-bit timer mode control register 0n (TMC0n)**




**(b) Capture/compare control register 0n (CRC0n)**




**(c) Prescaler mode register 0n (PRM0n) and selector operation control register 1 (SELCNT1)**

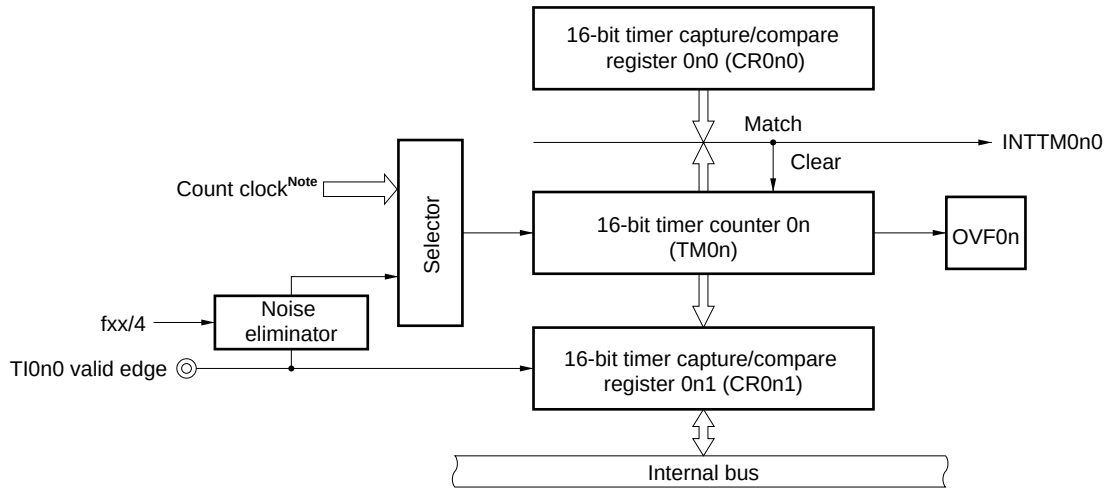



**Note** Be sure to clear the TMC0n1 bit to 0 when the TO0n pin and TI0n0 pin are used alternately.

**Remarks 1.** 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 **8.3 (1) 16-bit timer mode control register 0n (TMC0n)** and **8.3 (2) Capture/compare control register 0n (CRC0n)**.

**2.** n = 0, 1

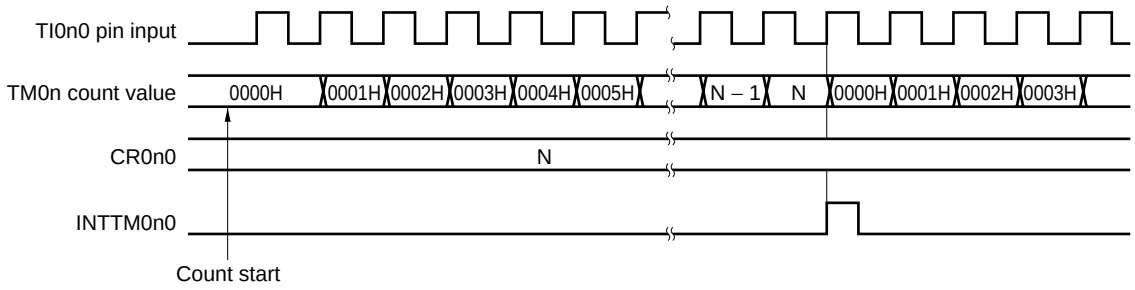
**Figure 8-19. Configuration of External Event Counter**




**Note** Set with the PRM0n register and SELCNT1 register.

**Remark** n = 0, 1

**Figure 8-20. Timing of External Event Counter Operation (with Rising Edge Specified)**




**Cautions 1. Read the TM0n register when reading the count value of the external event counter.**

**2. Counting is not possible at the first valid edge after the external event count mode is entered.**

**Remark** n = 0, 1

### 8.4.5 Square-wave output operation

Setting procedure

The basic operation setting procedure is as follows.

<1> Set the count clock using the PRM0n register and SELCNT1 register.
<2> Set the CRC0n register (refer to **Figure 8-21** for the setting value).
<3> Set the TOC0n register (refer to **Figure 8-21** for the setting value).
<4> Set any value (except for 0000H) to the CR0n0 register.
<5> Set the TMC0n register: Start operation (refer to **Figure 8-21** for the setting value).

**Remarks 1.** For the alternate-function pin (TO0n) settings, refer to **Table 4-14 Settings When Port Pins Are Used for Alternate Functions**.

**2.** For INTTM0n0 interrupt enable, refer to **CHAPTER 19 INTERRUPT/EXCEPTION PROCESSING FUNCTION**.

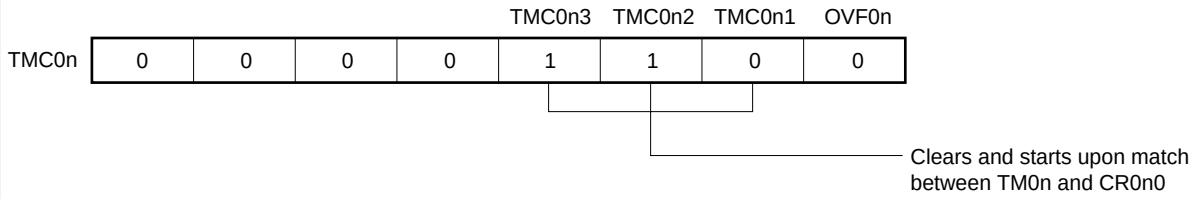
16-bit timer/event counter 0n can be used to output a square wave with any frequency at an interval specified by the count value set in advance to the CR0n0 register.

By setting the TOC0n.TOE0n and TOC0n.TOC0n1 bits to 11, the output status of the TO0n pin is inverted at an interval set in advance to the CR0n0 register. In this way, a square wave of any frequency can be output.

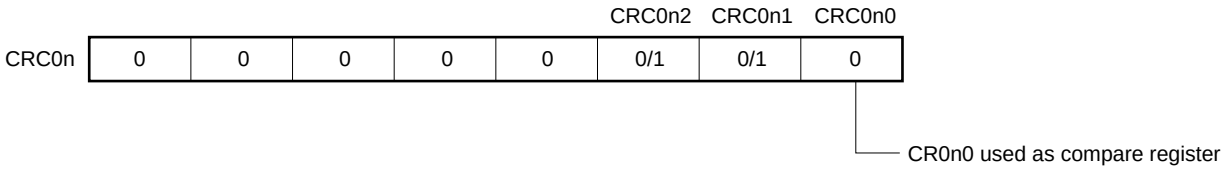
**Caution The value of the CR0n0 and CR0n1 registers cannot be changed during timer count operation.**

**Figure 8-21. Control Register Settings in Square-Wave Output Mode**

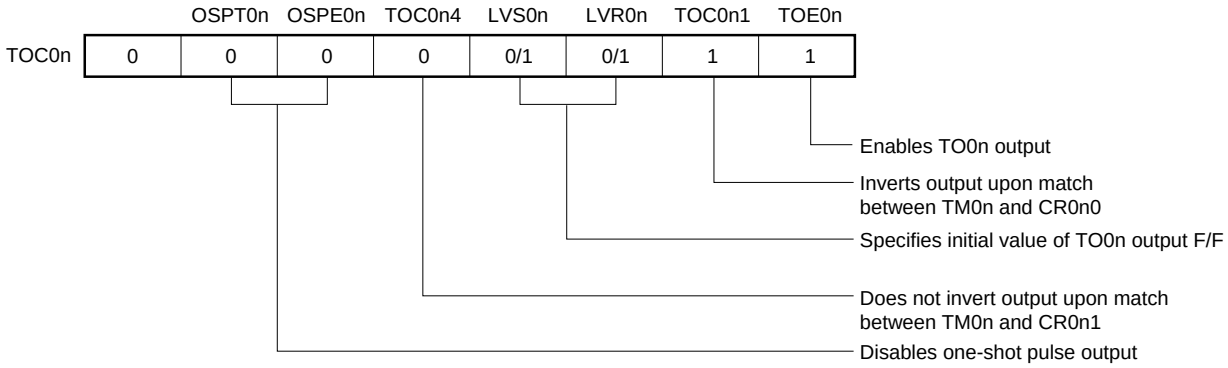
**(a) 16-bit timer mode control register 0n (TMC0n)**



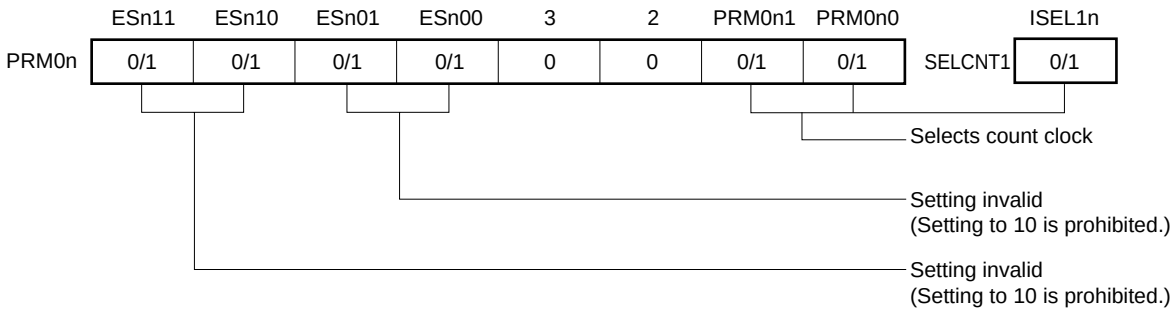

**(b) Capture/compare control register 0n (CRC0n)**




**(c) 16-bit timer output control register 0n (TOC0n)**



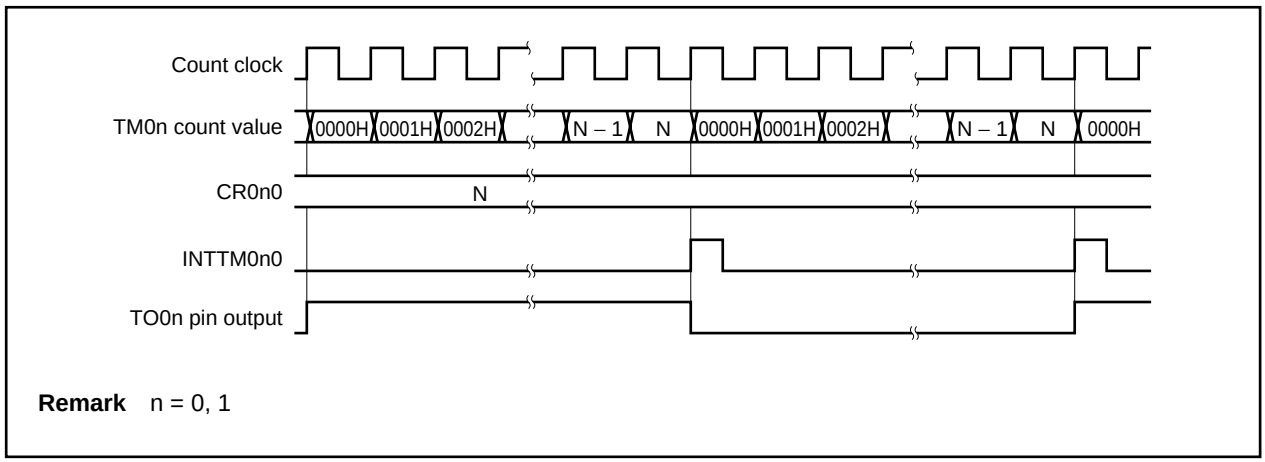

**(d) Prescaler mode register 0n (PRM0n) and selector operation control register 1 (SELCNT1)**




**Remarks 1.** For details, refer to **8.3 (2) Capture/compare control register 0n (CRC0n)** and **8.3 (3) 16-bit timer output control register 0n (TOC0n)**.

**2.** n = 0, 1

**Figure 8-22. Timing of Square-Wave Output Operation**




**Remark** n = 0, 1

### 8.4.6 One-shot pulse output operation

16-bit timer/event counter 0n can output a one-shot pulse in synchronization with a software trigger or an external trigger (TI0n0 pin input).

Setting procedure

The basic operation setting procedure is as follows.

<1> Set the count clock using the PRM0n register and SELCNT1 register.
<2> Set the CRC0n register (refer to **Figures 8-23** and **8-25** for the setting value).
<3> Set the TOC0n register (refer to **Figures 8-23** and **8-25** for the setting value).
<4> Set any value to the CR0n0 and CR0n1 registers.
<5> Set the TMC0n register: Start operation (refer to **Figures 8-23** and **8-25** for the setting value).

**Remarks 1.** For the alternate-function pin (TO0n) settings, refer to **Table 4-14 Settings When Port Pins Are Used for Alternate Functions**.

**2.** For INTTM0n0 interrupt enable, refer to **CHAPTER 19 INTERRUPT/EXCEPTION PROCESSING FUNCTION**.

**(1) One-shot pulse output with software trigger**

A one-shot pulse can be output from the TO0n pin by setting the TMC0n, CRC0n, and TOC0n registers as shown in Figure 8-23, and by setting the TOC0n.OSPT0n bit to 1 by software.

By setting the OSPT0n bit to 1, 16-bit timer/event counter 0n is cleared and started, and its output becomes active at the count value (N) set in advance to the CR0n1 register. After that, the output becomes inactive at the count value (M) set in advance to the CR0n0 register[Note].

Even after the one-shot pulse has been output, 16-bit timer/event counter 0n continues its operation. To stop 16-bit timer/event counter 0n, the TMC0n.TMC0n3 and TMC0n.TMC0n2 bits must be cleared to 00.

**Note** The case where N < M is described here. When N > M, the output becomes active with the CR0n0 register and inactive with the CR0n1 register.

**Cautions 1. Do not set the OSPT0n bit to 1 while the one-shot pulse is being output. To output the one-shot pulse again, wait until the current one-shot pulse output is completed.**

**2. The value of the CR0n0 and CR0n1 registers cannot be changed during timer count operation.**

**Remark** n = 0, 1

**Figure 8-23. Control Register Settings for One-Shot Pulse Output with Software Trigger (1/2)**

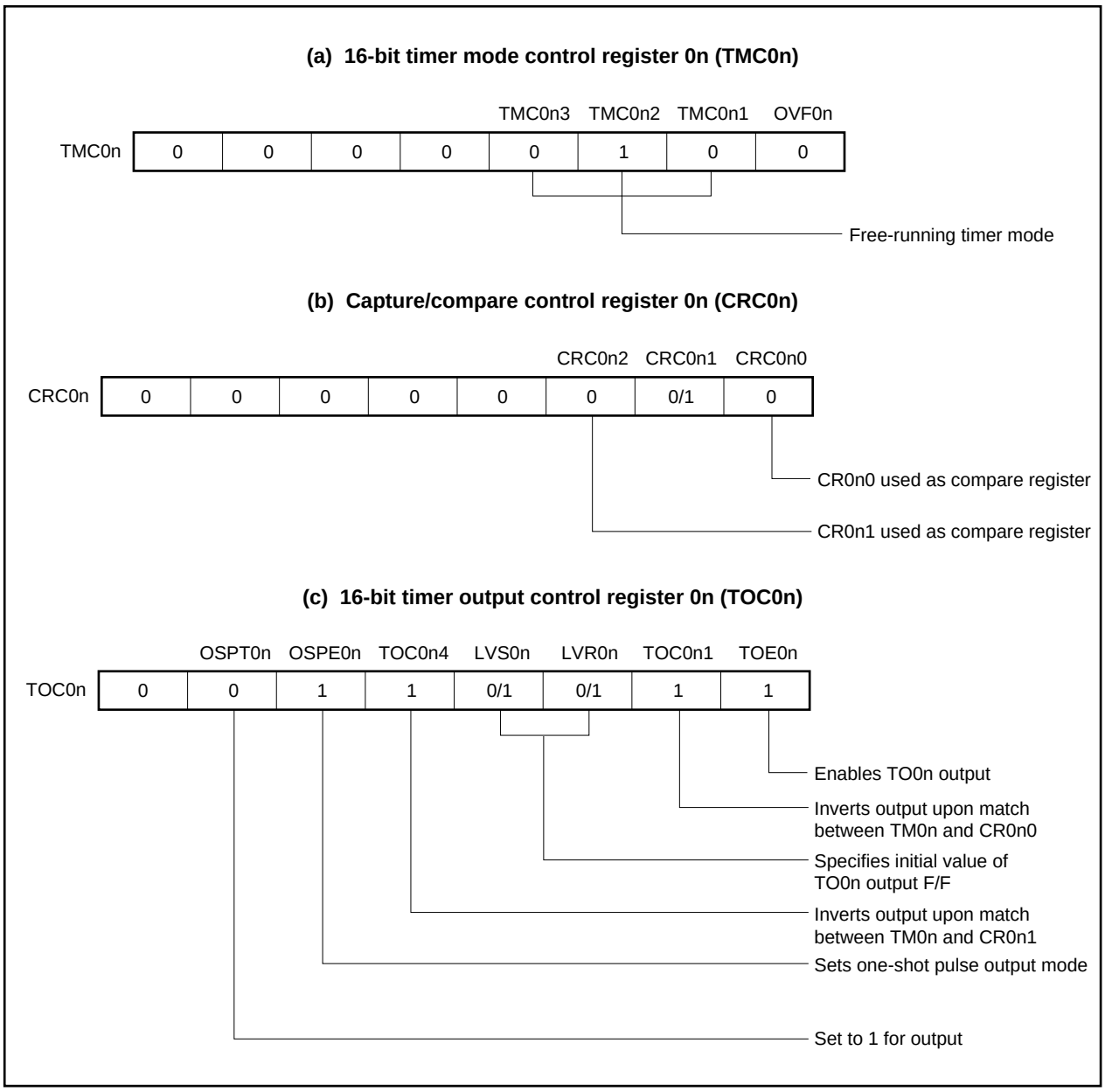
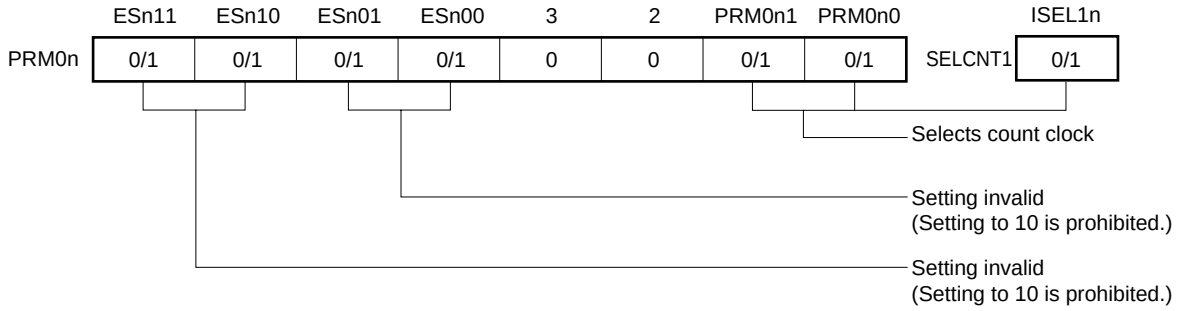


**Figure 8-23. Control Register Settings for One-Shot Pulse Output with Software Trigger (2/2)**

**(d) Prescaler mode register 0n (PRM0n) and selector operation control register 1 (SELCNT1)**

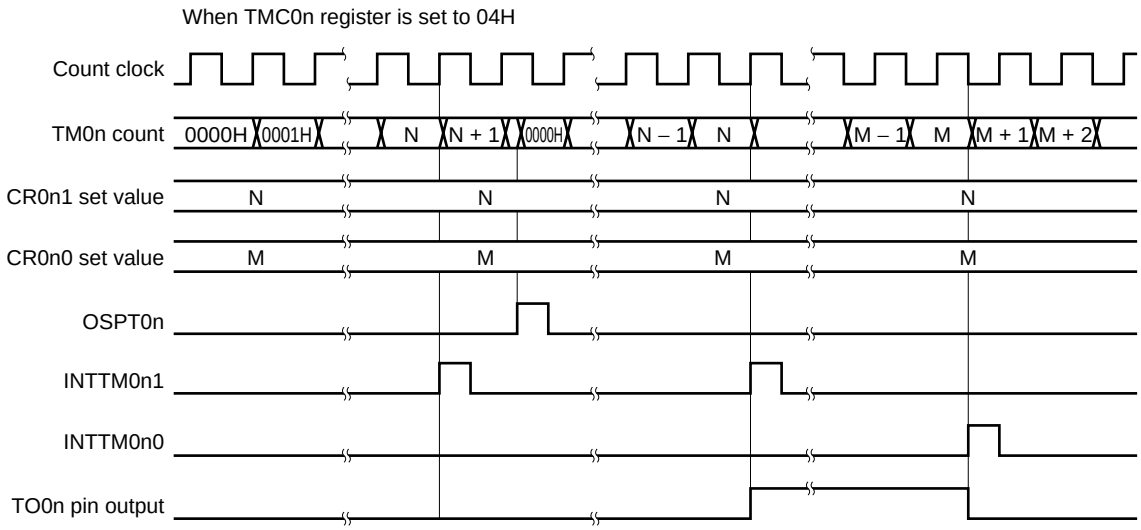



**Caution Do not set 0000H to the CR0n0 and CR0n1 registers.**

**Remarks 1.** For details, refer to **8.3 (2) Capture/compare control register 0n (CRC0n)** and **8.3 (3) 16-bit timer output control register 0n (TOC0n)**.

**2.** n = 0, 1

**Figure 8-24. Timing of One-Shot Pulse Output Operation with Software Trigger**




**Caution 16-bit timer counter 0n starts operating as soon as a value other than 00 (operation stop mode) is set to the TMC0n3 and TMC0n2 bits.**

**Remark** n = 0, 1
N < M

**(2) One-shot pulse output with external trigger**

A one-shot pulse can be output from the TO0n pin by setting the TMC0n, CRC0n, and TOC0n registers as shown in Figure 8-25, and by using the valid edge of the TI0n0 pin as an external trigger.

The valid edge of the TI0n0 pin is specified by the PRM0n.ESn00 and PRM0n.ESn01 bits. The rising, falling, or both the rising and falling edges can be specified.

When the valid edge of the TI0n0 pin is detected, 16-bit timer/event counter 0n is cleared and started, and the output becomes active at the count value set in advance to the CR0n1 register. After that, the output becomes inactive at the count value set in advance to the CR0n0 register[Note].

**Note** The case where N < M is described here. When N > M, the output becomes active with the CR0n0 register and inactive with the CR0n1 register.

**Cautions 1. Even if the external trigger is generated again while the one-shot pulse is output, it is ignored.**

**2. The value of the CR0n0 and CR0n1 registers cannot be changed during timer count operation.**

**Remark** n = 0, 1

**Figure 8-25. Control Register Settings for One-Shot Pulse Output with External Trigger (with Rising Edge Specified) (1/2)**

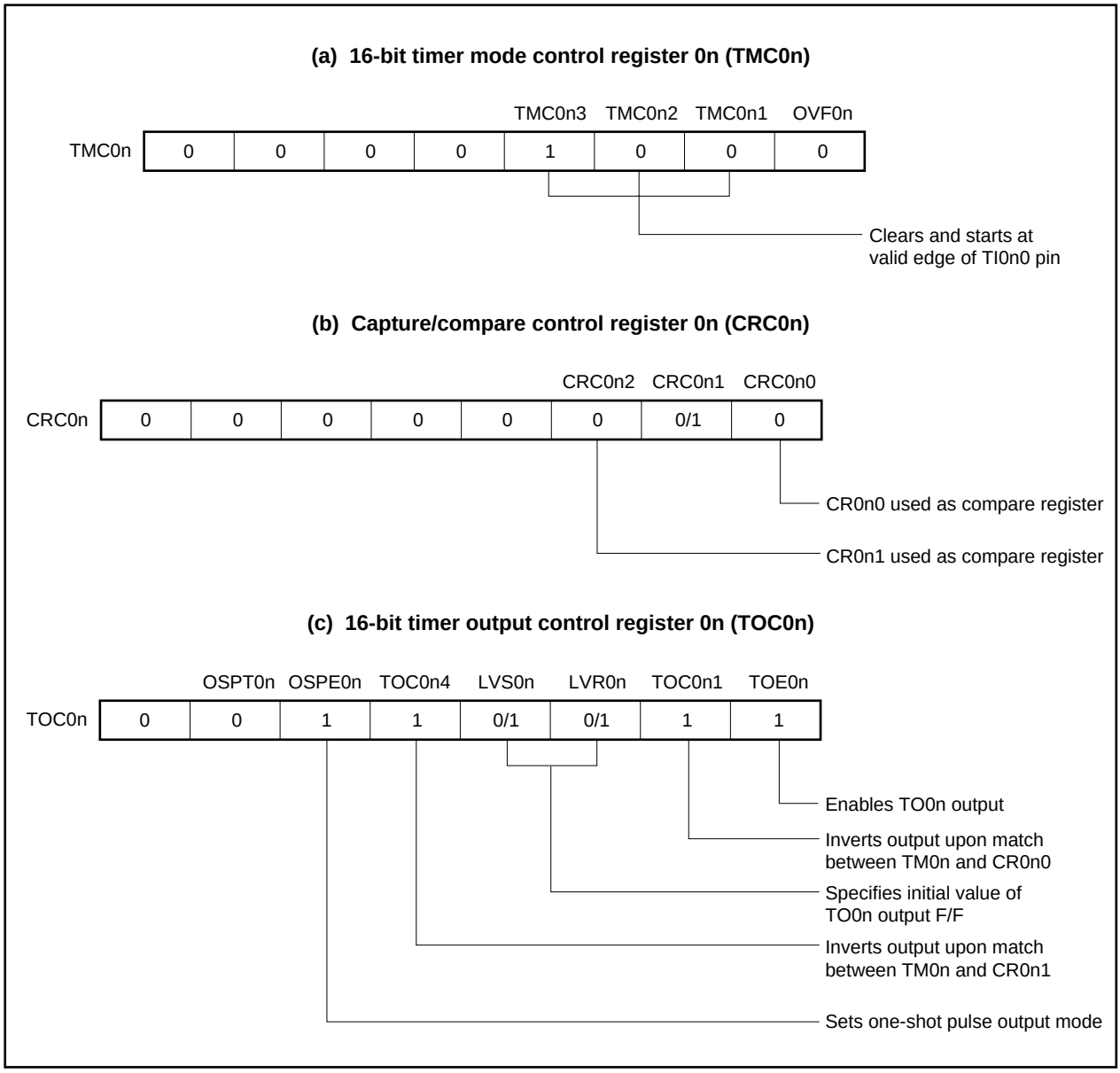
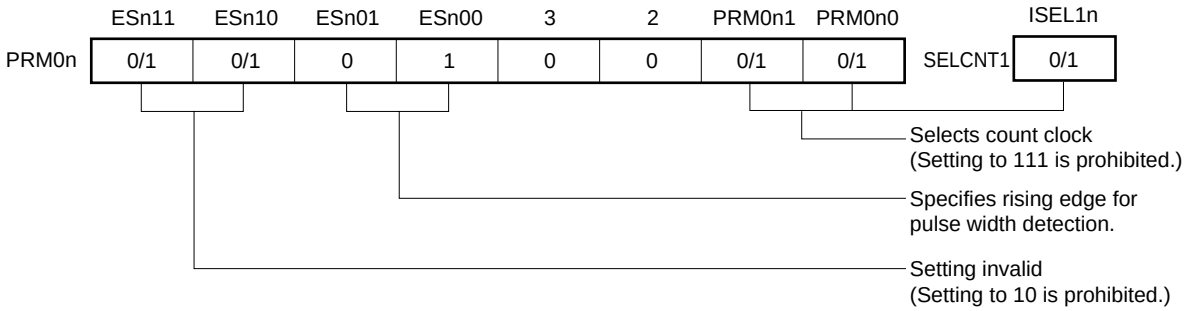


**Figure 8-25. Control Register Settings for One-Shot Pulse Output with External Trigger (with Rising Edge Specified) (2/2)**

**(d) Prescaler mode register 0n (PRM0n) and selector operation control register 1 (SELCNT1)**

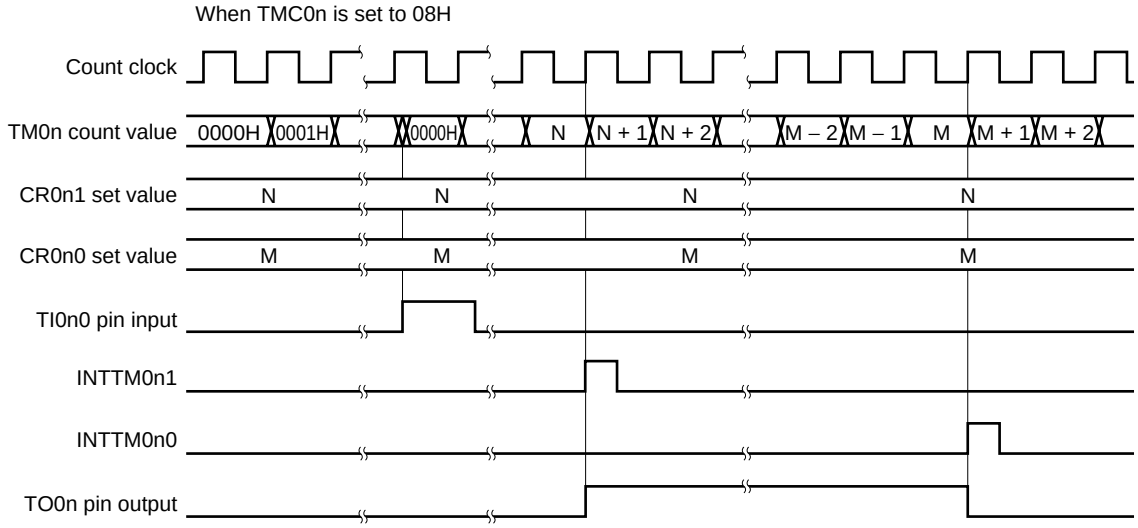



**Caution Do not set the CR0n0 and CR0n1 registers to 0000H.**

**Remarks 1.** For details, refer to **8.3 (2) Capture/compare control register 0n (CRC0n)** and **8.3 (3) 16-bit timer output control register 0n (TOC0n)**.

**2.** n = 0, 1

**Figure 8-26. Timing of One-Shot Pulse Output Operation with External Trigger (with Rising Edge Specified)**




**Caution 16-bit timer/event counter 0n starts operating as soon as a value other than 00 (operation stop mode) is set to the TMC0n2 and TMC0n3 bits.**

**Remark** n = 0, 1
N < M

### 8.4.7 Cautions

#### (1) Alternate functions of TI0n0/TO0n pins

| Channel | Pin | Alternate Function | Remarks |
|---|---|---|---|
| TM00 | TI000 | P33/TO00/TIP00/TOP00 | Shares the pin with TO00. |
| | TI001 | P34/TO00/TIP01/TOP01 | Shares the pin with TO00. |
| | TO00 | P33/TI000/TIP00/TOP00 | Assigned to two pins, P33 and P34. |
| | | P34/TI001/TIP01/TOP01 | |
| TM01 | TI010 | P35/TO01 | Shares the pin with TO01. |
| | TI011 | P50/KR0/RTP00 | |
| | TO01 | P32/ASCK0/ADTRG | Assigned to two pins, P32 and P35. |
| | | P35/TI010 | |

**(a) For TM00 channel**

- When using the output of TO00 that functions alternately as P33, only a software trigger (TOC00.OSPT00 bit) can be used as the trigger in the one-shot pulse output mode. A P33/TI000 pin input signal cannot be used as the trigger since TI000 and TO00 share a pin and are used alternately.
  A TI000 pin input signal can be used as the trigger, however, when using the output of TO00 that functions alternately as P34.
- When using the output of TO00 that functions alternately as P33, the timer output inversion operation using the valid edge of the TI000 pin input cannot be performed. The valid edge cannot be input to the P33/TI000 pin since TI000 and TO00 share a pin and are used alternately. Set the TMC00.TMC001 bit to 0 in this event.
  The timer output inversion operation using the valid edge of the TI000 pin input can be performed, however, when using the output of TO00 that functions alternately as P34.

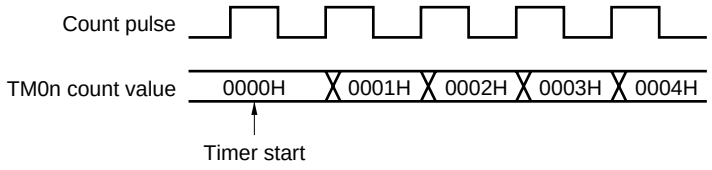
**(b) For TM01 channel**

- When using the output of TO01 that functions alternately as P35, only a software trigger (TOC01.OSPT01 bit) can be used as the trigger in the one-shot pulse output mode. A P35/TI010 pin input signal cannot be used as the trigger since TI010 and TO01 share a pin and are used alternately.
  A TI010 pin input signal can be used as the trigger, however, when using the output of TO01 that functions alternately as P32.
- When using the output of TO01 that functions alternately as P35, the timer output inversion operation using the valid edge of the TI010 pin input cannot be performed. The valid edge cannot be input to the P35/TI010 pin since TI010 and TO01 share a pin and are used alternately. Set the TMC01.TMC011 bit to 0 in this event.
  The timer output inversion operation using the valid edge of the TI010 pin input can be performed, however, when using the output of TO01 that functions alternately as P32.

**(2) Error on starting timer**

An error of up to 1 clock occurs before the match signal is generated after the timer has been started. This is because the count of the TM0n register is started asynchronously to the count pulse.

**Figure 8-27. Count Start Timing of TM0n Register**




**Remark** n = 0, 1

**(3) Setting CR0n0 and CR0n1 registers (in the mode in which clear & start occurs upon match between TM0n register and CR0n0 register)**

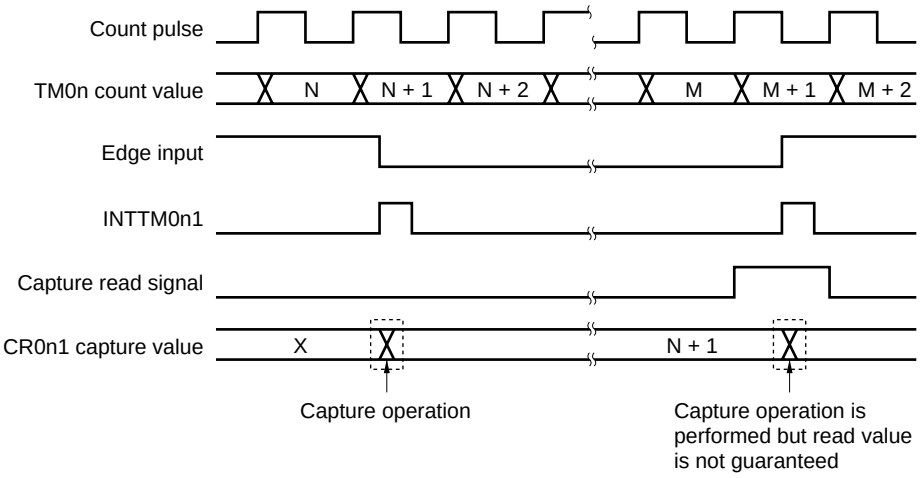
Set the CR0n0 and CR0n1 registers to a value other than 0000H (when using these registers as external event counters, one-pulse count operation is not possible).

**Remark** n = 0, 1

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

<1> If the valid edge of the TI0n0 pin is input while the CR0n1 register is read, the CR0n1 register performs capture operation, but the read value at this time is not guaranteed. However, the interrupt request signal (INTTM0n1) is generated as a result of detection of the valid edge.

**Figure 8-28. Data Hold Timing of Capture Register**




**Remark** n = 0, 1

<2> The values of the CR0n0 and CR0n1 registers are not guaranteed after 16-bit timer/event counter 0n has stopped.

**(5) Setting valid edge**

Before setting the valid edge of the TI0n0 pin, stop the timer operation by clearing the TMC0n.TMC0n2 and TMC0n.TMC0n3 bits to 00. Set the valid edge by using the PRM0n.ESn00 and PRM0n.ESn01 bits.

**Remark** n = 0, 1

**(6) Re-triggering one-shot pulse**

**(a) One-shot pulse output by software**

When a one-shot pulse is output, do not set the TOC0n.OSPT0n bit to 1. Do not output the one-shot pulse again until the INTTM0n0 signal, which occurs upon match with the CR0n0 register, or the INTTM0n1 signal, which occurs upon match with the CR0n1 register, occurs.

**Remark** n = 0, 1

**(b) One-shot pulse output with external trigger**

If the external trigger occurs again while a one-shot pulse is output, it is ignored.

### (7) Operation of OVF0n flag

#### (a) Setting of OVF0n flag

The TMC0n.OVF0n flag is set to 1 in the following case in addition to when the TM0n register overflows.

Select the mode in which clear & start occurs upon match between the TM0n register and the CR0n0 register.

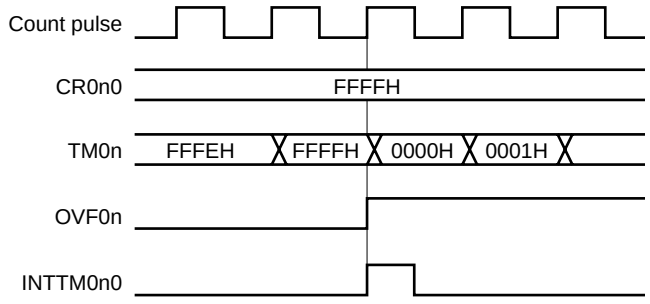
↓

Set the CR0n0 register to FFFFH

↓

When the TM0n register is cleared from FFFFH to 0000H upon match with the CR0n0 register

**Figure 8-29. Operation Timing of OVF0n Flag**




**Remark** n = 0, 1

#### (b) Clearing of OVF0n flag

After the TM0n register overflows, clearing OVF0n flag is invalid and set (1) again even if the OVF0n flag is cleared (0) before the next count clock is counted (before the TM0n register becomes 0001H).

**Remark** n = 0, 1

### (8) Timer operation

#### (a) CR0n1 register capture

Even if the TM0n register is read, the read data cannot be captured into the CR0n1 register.

#### (b) TI0n0, TI0n1 pin acknowledgment

Regardless of the CPU's operation mode, if the timer is stopped, signals input to the TI0n0 and TI0n1 pins are not acknowledged.

#### (c) One-shot pulse output

One-shot pulse output operates normally in either the free-running timer mode or the mode in which clear & start occurs on the valid edge of the TI0n0 pin. Because no overflow occurs in the mode in which clear & start occurs upon match between the TM0n register and the CR0n0 register, one-shot pulse output is not possible.

**Remark** n = 0, 1

**(9) Capture operation**

**(a) If valid edge of TI0n0 is specified for count clock**

If the valid edge of TI0n0 is specified for the count clock, the capture register that specified TI0n0 as the trigger does not operate normally.

**(b) If both rising and falling edges are selected for valid edge of TI0n0**

If both the rising and falling edges are selected for the valid edge of TI0n0, capture operation is not performed.

**(c) To ensure that signals from TI0n1 and TI0n0 are correctly captured**

For the capture trigger to capture the signals from TI0n1 and TI0n0 correctly, a pulse longer than two of the count clocks selected by the PRM0n register and SELCNT1 register is required.

**(d) Interrupt request input**

Although a capture operation is performed at the falling edge of the count clock, an interrupt request signal (INTTM0n0, INTTM0n1) is generated at the rising edge of the next count clock.

**Remark** n = 0, 1

**(10) Compare operation**

When set to the compare mode, the CR0n0 and CR0n1 registers do not perform capture operation even if a capture trigger is input.

**Caution The value of the CR0n0 register cannot be changed during timer operation. The value of the CR0n1 register cannot be changed during timer operation other than in the PPG output mode. To change the CR0n1 register in the PPG output mode, refer to 8.4.2 PPG output operation.**

**Remark** n = 0, 1

**(11) Edge detection**

**(a) Sampling clock for noise elimination**

The sampling clock for noise elimination differs depending on whether the valid edge of TI0n0 is used for the count clock or as a capture trigger. In the former case, sampling is performed using $f_{XX}/4$, and in the latter case, sampling is performed using the count clock selected by the PRM0n register and SELCNT1 register. The first capture operation does not start until the valid edges are sampled and two valid levels are detected, thus eliminating noise with a short pulse width.

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

**2.** n = 0, 1

# CHAPTER 9 8-BIT TIMER/EVENT COUNTER 5

In the V850ES/KF1+, two channels of 8-bit timer/event counter 5 are provided.

## 9.1 Functions

8-bit timer/event counter 5n has the following two modes (n = 0, 1).

- Mode using 8-bit timer/event counter alone (individual mode)
- Mode using cascade connection (16-bit resolution: cascade connection mode)

These two modes are described below.

**(1) Mode using 8-bit timer/event counter alone (individual mode)**

8-bit timer/event counter 5n operates as an 8-bit timer/event counter.
The following functions can be used.

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

**(2) Mode using cascade connection (16-bit resolution: cascade connection mode)**

8-bit timer/event counter 5n operates as a 16-bit timer/event counter by connecting the TM50 and TM51 registers in cascade. The following functions can be used.

- 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 counter 5n consists of the following hardware.

**Table 9-1. Configuration of 8-Bit Timer/Event Counter 5n**

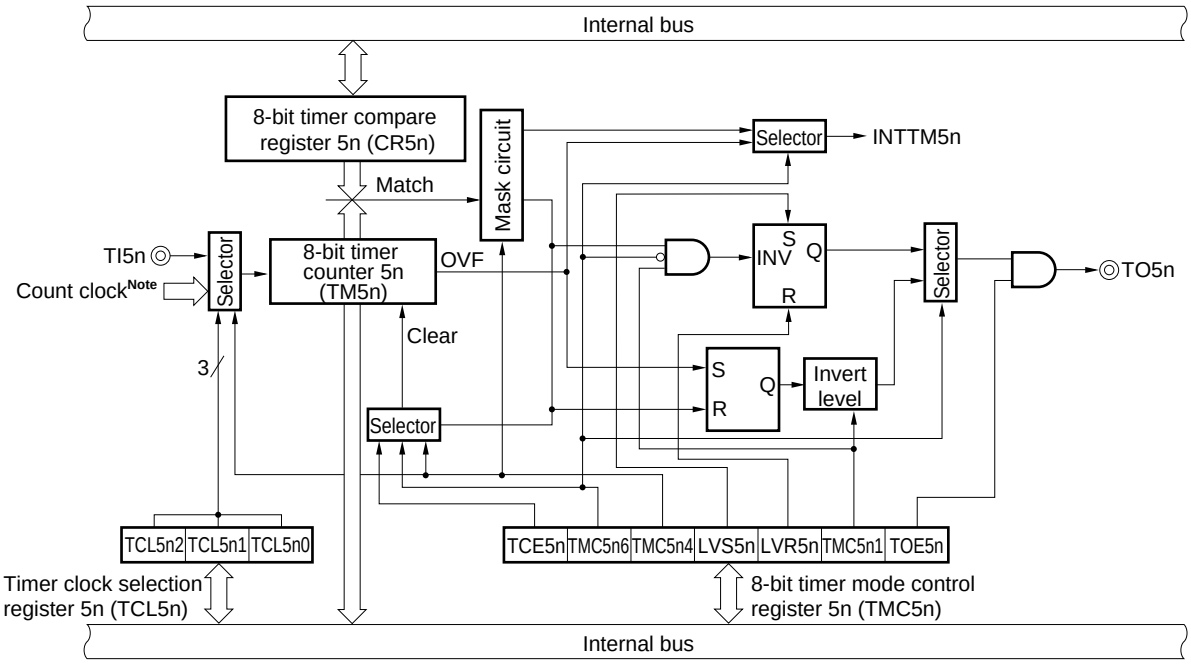
| Item | Configuration |
|---|---|
| Timer registers | 8-bit timer counters 50, 51 (TM50, TM51)<br>16-bit timer counter 5 (TM5): Only when using cascade connection |
| Registers | 8-bit timer compare registers 50, 51 (CR50, CR51)<br>16-bit timer compare register 5 (CR5): Only when using cascade connection |
| Timer output | TO50, TO51 |
| Control registers[Note] | Timer clock selection registers 50, 51 (TCL50, TCL51)<br>8-bit timer mode control registers 50, 51 (TMC50, TMC51)<br>16-bit timer mode control register 5 (TMC5): Only when using cascade connection |

**Note** When using the functions of the TI5n and TO5n pins, refer to **Table 4-14 Settings When Port Pins Are Used for Alternate Functions**.

**Remark** n = 0, 1

The block diagram of 8-bit timer/event counter 5n is shown below.

**Figure 9-1. Block Diagram of 8-Bit Timer/Event Counter 5n**




**Note** The count clock is set by the TCL5n register.

**Remark** n = 0, 1

**(1) 8-bit timer counter 5n (TM5n)**

The TM5n register is an 8-bit read-only register that counts the count pulses.

The counter is incremented in synchronization with the rising edge of the count clock.

Through cascade connection, the TM5n registers can be used as a 16-bit timer.

When using the TM50 register and the TM51 register in cascade as a 16-bit timer, these registers are read-only, in 16-bit units. Therefore, read these registers twice and compare the values, taking into consideration that the reading occurs during a count change.

After reset: 00H R Address: TM50 FFFFF5C0H, TM51 FFFFF5C1H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| TM5n (n = 0, 1) | | | | | | | | |

The count value is reset to 00H in the following cases.

<1> Reset

<2> When the TMC5n.TCE5n bit is cleared (0)

<3> The values of the TM5n register and CR5n register match in the mode in which clear & start occurs on a match between the TM5n register and the CR5n register

**Caution When connected in cascade, these registers become 0000H even when the TCE50 bit in the lowest timer (TM50) is cleared.**

**Remark** n = 0, 1

**(2) 8-bit timer compare register 5n (CR5n)**

The CR5n register can be read and written in 8-bit units.

In a mode other than the PWM mode, the value set to the CR5n register is always compared to the count value of the TM5n register, and if the two values match, an interrupt request signal (INTTM5n) is generated.

In the PWM mode, TM5n register overflow causes the TO5n pin output to change to the active level, and when the values of the TM5n register and the CR5n register match, the TO5n pin output changes to the inactive level.

The value of the CR5n register can be set in the range of 00H to FFH.

When using the TM50 register and TM51 register in cascade as a 16-bit timer, the CR50 register and CR51 register operate as 16-bit timer compare register 5 (CR5). The counter value and register value are compared in 16-bit lengths, and if they match, an interrupt request signal (INTTM50) is generated.

After reset: 00H R/W Address: CR50 FFFFF5C2H, CR51 FFFFF5C3H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CR5n | | | | | | | | |

(n = 0, 1)

**Cautions 1. In the mode in which clear & start occurs upon a match of the TM5n register and CR5n register (TMC5n.TMC5n6 bit = 0), do not write a different value to the CR5n register during the count operation.**

**2. In the PWM mode, set the CR5n register rewrite interval to three or more count clocks (clock selected with the TCL5n register).**

**3. Before changing the value of the CR5n register when using a cascade connection, be sure to stop the timer operation.**

**Remark** n = 0, 1

## 9.3 Registers

The following two registers are used to control 8-bit timer/event counter 5n.

- Timer clock selection register 5n (TCL5n)
- 8-bit timer mode control register 5n (TMC5n)

**Remark** To use the functions of the TI5n and TO5n pins, refer to **Table 4-14 Settings When Port Pins Are Used for Alternate Functions**.

**(1) Timer clock selection register 5n (TCL5n)**

The TCL5n register sets the count clock of 8-bit timer/event counter 5n and the valid edge of the TI5n pin input.

The TCL5n register can be read or written in 8-bit units.

After reset, this register is cleared to 00H.

After reset: 00H R/W Address: TCL50 FFFFF5C4H, TCL51 FFFFF5C5H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| TCL5n | 0 | 0 | 0 | 0 | 0 | TCL5n2 | TCL5n1 | TCL5n0 |

(n = 0, 1)

| TCL5n2 | TCL5n1 | TCL5n0 | Count clock selection[Note] | | |
|---|---|---|---|---|---|
| | | | Clock | $f_{XX}$ | |
| | | | | 20 MHz | 10 MHz |
| 0 | 0 | 0 | Falling edge of TI5n | – | – |
| 0 | 0 | 1 | Rising edge of TI5n | – | – |
| 0 | 1 | 0 | $f_{XX}$ | Setting prohibited | 100 ns |
| 0 | 1 | 1 | $f_{XX}/2$ | 100 ns | 200 ns |
| 1 | 0 | 0 | $f_{XX}/4$ | 200 ns | 0.4 $\mu$s |
| 1 | 0 | 1 | $f_{XX}/64$ | 3.2 $\mu$s | 6.4 $\mu$s |
| 1 | 1 | 0 | $f_{XX}/256$ | 12.8 $\mu$s | 25.6 $\mu$s |
| 1 | 1 | 1 | INTTM010 | – | – |

**Note** When the internal clock is selected, set so as to satisfy the following conditions.

REGC = $V_{DD}$ = 4.0 to 5.5 V: Count clock ≤ 10 MHz

REGC = Capacity, $V_{DD}$ = 4.0 to 5.5 V: Count clock ≤ 5 MHz

REGC = $V_{DD}$ = 2.7 to 4.0 V: Count clock ≤ 5 MHz

**Caution Before overwriting the TCL5n register with different data, stop the timer operation.**

**Remark** When the TM5n register is connected in cascade, the TCL51 register settings are invalid.

**(2) 8-bit timer mode control register 5n (TMC5n)**

The TMC5n register performs the following six settings.

- Controls counting by the TM5n register
- Selects the operation mode of the TM5n register
- Selects the individual mode or cascade connection mode
- Sets the status of the timer output flip-flop
- Controls the timer output flip-flop or selects the active level in the PWM (free-running timer) mode
- Controls timer output

The TMC5n register can be read or written in 8-bit or 1-bit units.

After reset, this register is cleared to 00H.

After reset: 00H R/W Address: TMC50 FFFFF5C6H, TMC51 FFFFF5C7H

| | <7> | 6 | 5 | 4 | 3 | 2 | 1 | <0> |
|---|---|---|---|---|---|---|---|---|
| TMC5n | TCE5n | TMC5n6 | 0 | TMC514[Note] | LVS5n | LVR5n | TMC5n1 | TOE5n |

(n = 0, 1)

| TCE5n | Control of count operation of 8-bit timer/event counter 5n |
|---|---|
| 0 | Counting is disabled after the counter is cleared to 0 (counter disabled) |
| 1 | Start count operation |

| TMC5n6 | Selection of operation mode of 8-bit timer/event counter 5n |
|---|---|
| 0 | Mode in which clear & start occurs on match between TM5n register and CR5n register |
| 1 | PWM (free-running timer) mode |

| TMC514 | Selection of individual mode or cascade connection mode for 8-bit timer/event counter 51 |
|---|---|
| 0 | Individual mode |
| 1 | Cascade connection mode (connected with 8-bit timer/event counter 50) |

| LVS5n | LVR5n | Setting of status of timer output F/F |
|---|---|---|
| 0 | 0 | Unchanged |
| 0 | 1 | Reset timer output F/F to 0 |
| 1 | 0 | Set timer output F/F to 1 |
| 1 | 1 | Setting prohibited |

| TMC5n1 | Other than PWM (free-running timer) mode (TMC5n6 bit = 0) | PWM (free-running timer) mode (TMC5n6 bit = 1) |
|---|---|---|
| | Controls timer F/F | Selects active level |
| 0 | Disable inversion operation | High active |
| 1 | Enable inversion operation | Low active |

| TOE5n | Timer output control |
|---|---|
| 0 | Disable output (TO5n pin is low level) |
| 1 | Enable output |

**Note** Bit 4 of the TMC50 register is fixed to 0.

**Cautions 1. Because the TO51 and TI51 pins are alternate functions of the same pin, only one can be used at one time.**

**2. The LVS5n and LVR5n bit settings are valid in modes other than the PWM mode.**

**3. Do not set <1> to <4> below at the same time. Set as follows.**

**<1> Set the TMC5n1, TMC5n6, and TMC514[Note] bits: Setting of operation mode**

**<2> Set the TOE5n bit for timer output enable: Timer output enable**

**<3> Set the LVS5n and LVR5n bits (Caution 2): Setting of timer output F/F**

**<4> Set the TCE5n bit**

**Remarks 1.** In the PWM mode, the PWM output is set to the inactive level by the TCE5n bit = 0.

**2.** When the LVS5n and LVR5n bits are read, 0 is read.

**3.** The values of the TMC5n6, LVS5n, LVR5n, TMC5n1, and TOE5n bits are reflected to the TO5n output regardless of the TCE5n bit value.

## 9.4 Operation

### 9.4.1 Operation as interval timer

8-bit timer/event counter 5n operates as an interval timer that repeatedly generates interrupts at the interval of the count value preset in the CR5n register. If the count value in the TM5n register matches the value set in the CR5n register, the value of the TM5n register is cleared to 00H and counting is continued, and at the same time, an interrupt request signal (INTTM5n) is generated.

Setting method

<1> Set each register.
- TCL5n register: Selects the count clock (t).
- CR5n register: Compare value (N)
- TMC5n register: Stops count operation and selects the mode in which clear & start occurs on a match between the TM5n register and CR5n register (TMC5n register = 0000xx00B, ×: don't care).

<2> When the TMC5n.TCE5n bit is set to 1, the count operation starts.

<3> When the values of the TM5n register and CR5n register match, the INTTM5n signal is generated (TM5n register is cleared to 00H).

<4> Then, the INTTM5n signal is repeatedly generated at the same interval. To stop counting, set the TCE5n bit = 0.

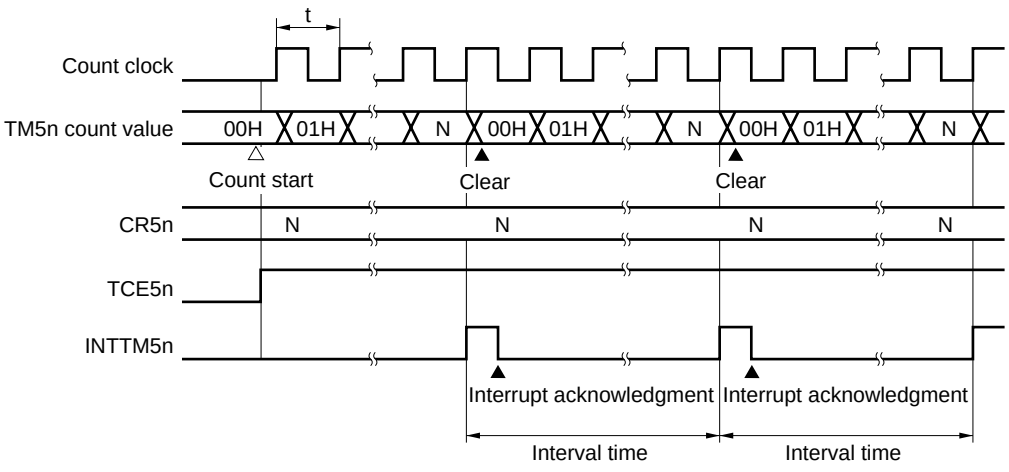
Interval time = (N + 1) × t: N = 00H to FFH

**Caution During interval timer operation, do not rewrite the value of the CR5n register.**

**Remark** n = 0, 1

**Figure 9-2. Timing of Interval Timer Operation (1/2)**

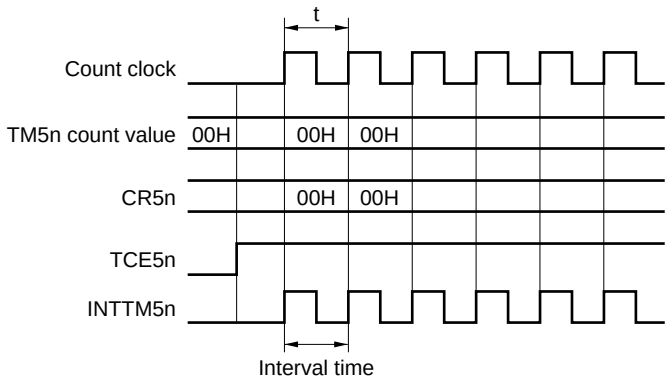
Basic operation




**Remark** n = 0, 1

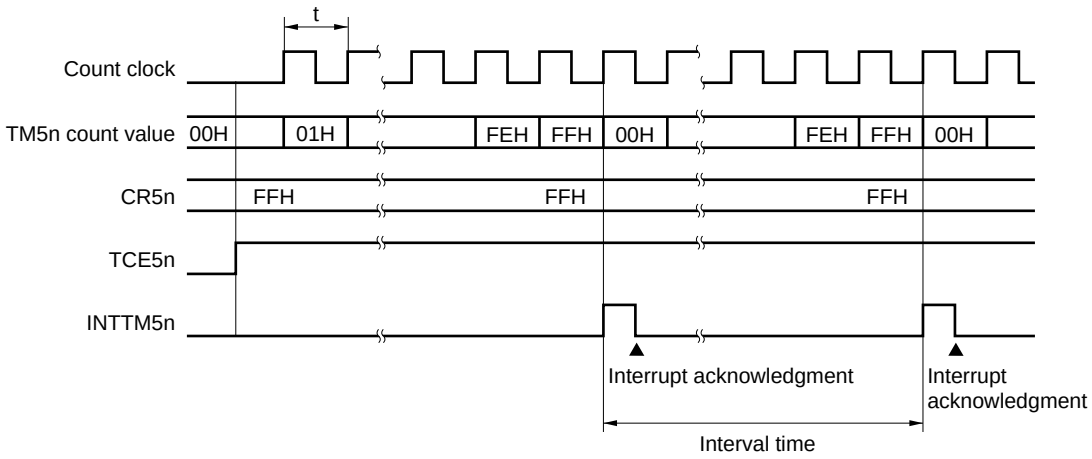
**Figure 9-2. Timing of Interval Timer Operation (2/2)**

When CR5n register = 00H




**Remark** n = 0, 1

When CR5n register = FFH




**Remark** n = 0, 1

### 9.4.2 Operation as external event counter

The external event counter counts the number of clock pulses input to the TI5n pin from an external source by using the TM5n register.

Each time the valid edge specified by the TCL5n register is input to the TI5n pin, the TM5n register is incremented. Either the rising edge or the falling edge can be specified as the valid edge.

When the count value of the TM5n register matches the value of the CR5n register, the TM5n register is cleared to 0 and an interrupt request signal (INTTM5n) is generated.

Setting method

<1> Set each register.

- TCL5n register: Selects the TI5n pin input edge.
  Falling edge of TI5n pin → TCL5n register = 00H
  Rising edge of TI5n pin → TCL5n register = 01H
- CR5n register: Compare value (N)
- TMC5n register: Stops count operation, selects the mode in which clear & start occurs on a match between the TM5n register and CR5n register, disables timer output F/F inversion operation, and disables timer output.
  (TMC5n register = 0000xx00B, ×: don't care)
- For the alternate-function pin settings, refer to **Table 4-14 Settings When Port Pins Are Used for Alternate Functions.**

<2> When the TMC5n.TCE5n bit is set to 1, the counter counts the number of pulses input from the TI5n pin.

<3> When the values of the TM5n register and CR5n register match, the INTTM5n signal is generated (TM5n register is cleared to 00H).

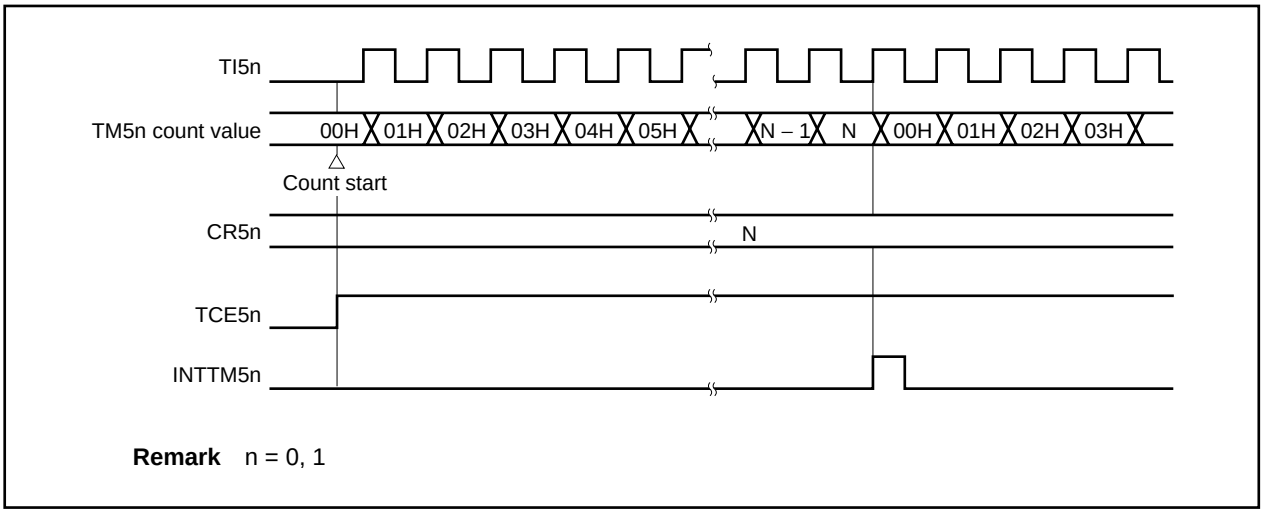
<4> Then, the INTTM5n signal is generated each time the values of the TM5n register and CR5n register match.

INTTM5n signal is generated when the valid edge is input to the TI5n pin N + 1 times: N = 00H to FFH

**Caution During external event counter operation, do not rewrite the value of the CR5n register.**

**Remark** n = 0, 1

**Figure 9-3. Timing of External Event Counter Operation (with Rising Edge Specified)**




**Remark** n = 0, 1

### 9.4.3 Square-wave output operation

A square wave with any frequency can be output at an interval determined by the value preset in the CR5n register.

By setting the TMC5n.TOE5n bit to 1, the output status of the TO5n pin is inverted at an interval determined by the count value preset in the CR5n register. In this way, a square wave of any frequency can be output (duty = 50%) (n = 0, 1).

Setting method

<1> Set each register.

- TCL5n register: Selects the count clock (t).
- CR5n register: Compare value (N)
- TMC5n register: Stops count operation, selects the mode in which clear & start occurs on a match between the TM5n register and CR5n register, sets initial value of timer output, enables timer output F/F inversion operation, and enables timer output.
  (TMC5n register = 00001011B or 00000111B)
- For the alternate-function pin settings, refer to **Table 4-14 Settings When Port Pins Are Used for Alternate Functions.**

<2> When the TMC5n.TCE5n bit is set to 1, counting starts.

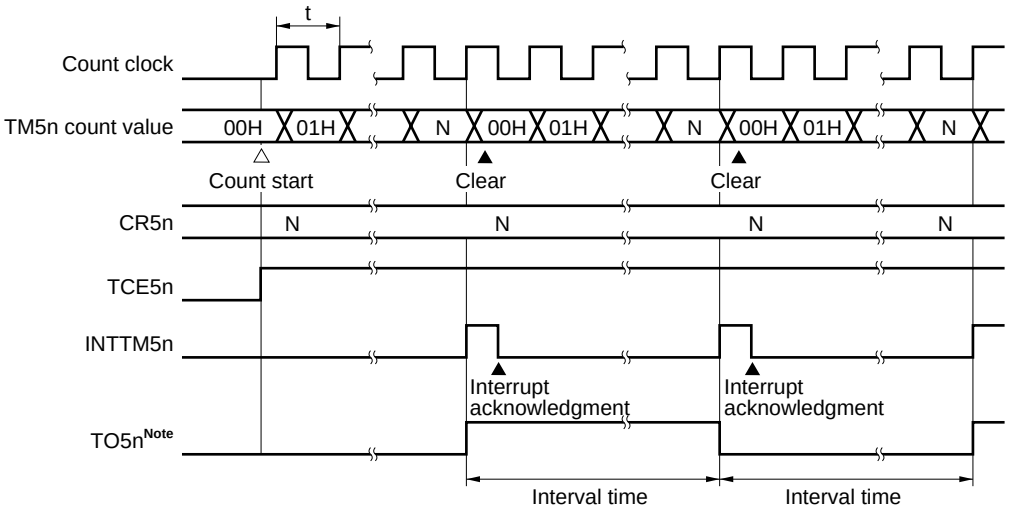
<3> When the values of the TM5n register and CR5n register match, the timer output F/F is inverted. Moreover, the INTTM5n signal is generated and the TM5n register is cleared to 00H.

<4> Then, the timer output F/F is inverted during the same interval and a square wave is output from the TO5n pin.

Frequency = 1/2t(N + 1): N = 00H to FFH

**Caution Do not rewrite the value of the CR5n register during square-wave output.**

**Figure 9-4. Timing of Square-Wave Output Operation**




**Note** The initial value of the TO5n pin output can be set using the TMC5n.LVS5n and TMC5n.LVR5n bits.

**Remark** n = 0, 1

### 9.4.4 8-bit PWM output operation

By setting the TMC5n.TMC5n6 bit to 1, 8-bit timer/event counter 5n performs PWM output.

Pulses with a duty factor determined by the value set in the CR5n register are output from the TO5n pin.

Set the width of the active level of the PWM pulse in the CR5n register. The active level can be selected using the TMC5n.TMC5n1 bit.

The count clock can be selected using the TCL5n register.

PWM output can be enabled/disabled by the TMC5n.TOE5n bit.

**Caution The CR5n register rewrite interval must be three or more operation clocks (set by the TCL5n register).**

Use method

<1> Set each register.

- TCL5n register: Selects the count clock (t).
- CR5n register: Compare value (N)
- TMC5n register: Stops count operation, selects PWM mode, and leave timer output F/F unchanged, sets active level, and enables timer output.
  (TMC5n register = 01000001B or 01000011B)
- For the alternate-function pin settings, refer to **Table 4-14 Settings When Port Pins Are Used for Alternate Functions.**

<2> When the TMC5n.TCE5n bit is set to 1, counting starts.

PWM output operation

<1> When counting starts, PWM output (output from the TO5n pin) outputs the inactive level until an overflow occurs.

<2> When an overflow occurs, the active level set by setting method <1> is output. The active level is output until the value of the CR5n register and the count value of the TM5n register match. An interrupt request signal (INTTM5n) is generated.

<3> When the value of the CR5n register and the count value of the TM5n register match, the inactive level is output and continues to be output until an overflow occurs again.

<4> Then, steps <2> and <3> are repeated until counting is stopped.

<5> When counting is stopped by clearing TCE5n bit to 0, PWM output becomes inactive.

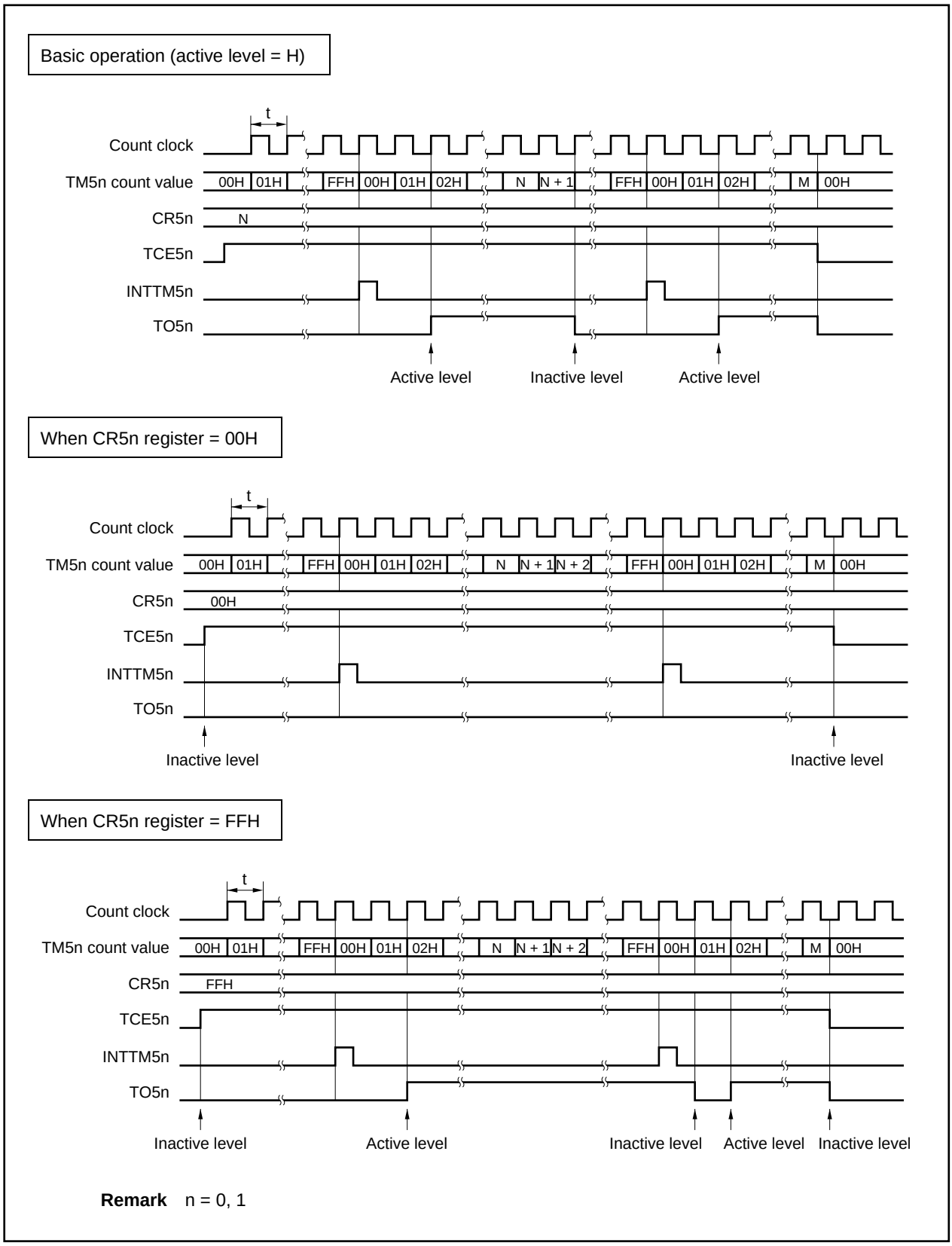
Cycle = 256t, active level width = Nt, duty = N/256: N = 00H to FFH

**Remarks 1.** n = 0, 1

**2.** For the detailed timing, refer to **Figure 9-5 Timing of PWM Output Operation** and **Figure 9-6 Timing of Operation Based on CR5n Register Transitions**.

**(a) Basic operation of PWM output**

**Figure 9-5. Timing of PWM Output Operation**



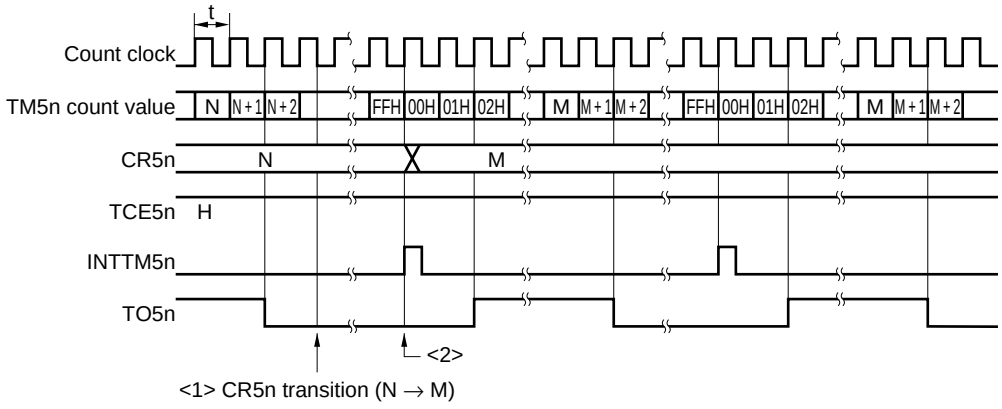

**Remark** n = 0, 1

**(b) Operation based on CR5n register transitions**

**Figure 9-6. Timing of Operation Based on CR5n Register Transitions**

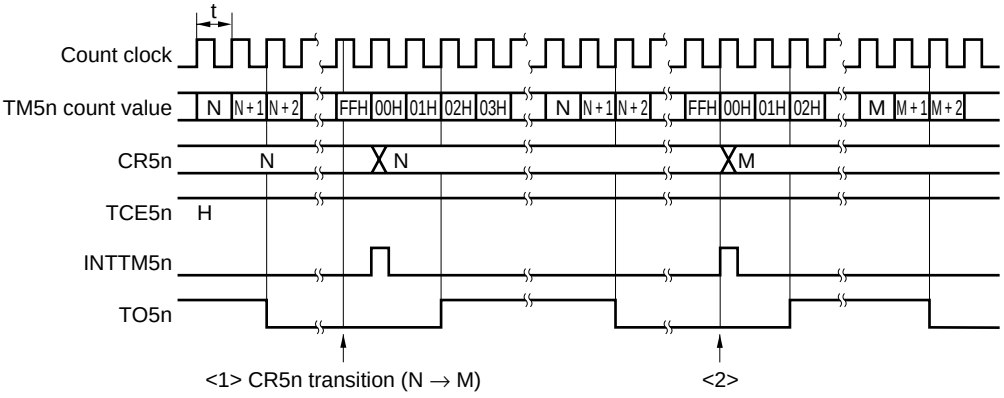
When the value of the CR5n register changes from N to M before the rising edge of the FFH clock

→ The value of the CR5n register is transferred at the overflow that occurs immediately after.




When the value of the CR5n register changes from N to M after the rising edge of the FFH clock

→ The value of the CR5n register is transferred at the second overflow.




**Caution In the case of read from the CR5n register between <1> and <2>, the value that is actually used differs (Read value: M; Actual value of CR5n register: N).**

**Remark** n = 0, 1

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

The 16-bit resolution timer/event counter mode is selected by setting the TMC51.TMC514 bit to 1.

8-bit timer/event counter 5n operates as an interval timer by repeatedly generating interrupts using the count value preset in 16-bit timer compare register 5 (CR5) as the interval.

Setting method

<1> Set each register.

- TCL50 register: Selects the count clock (t)
  (The TCL51 register does not need to be set in cascade connection)
- CR50 register: Compare value (N) ... Lower 8 bits (settable from 00H to FFH)
- CR51 register: Compare value (N) ... Higher 8 bits (settable from 00H to FFH)
- TMC50, TMC51 registers: Selects the mode in which clear & start occurs on a match between TM5 register and CR5 register (×: don't care)
  (TMC50 register = 0000xx00B
  TMC51 register = 0001xx00B)

<2> Set the TMC51.TCE51 bit to 1. Then set the TMC50.TCE50 bit to 1 to start the count operation.

<3> When the values of the TM5 register and CR5 register connected in cascade match, the INTTM50 signal is generated (the TM5 register is cleared to 0000H).

<4> The INTTM50 signal is then generated repeatedly at the same interval.

Interval time = (N + 1) × t: N = 0000H to FFFFH

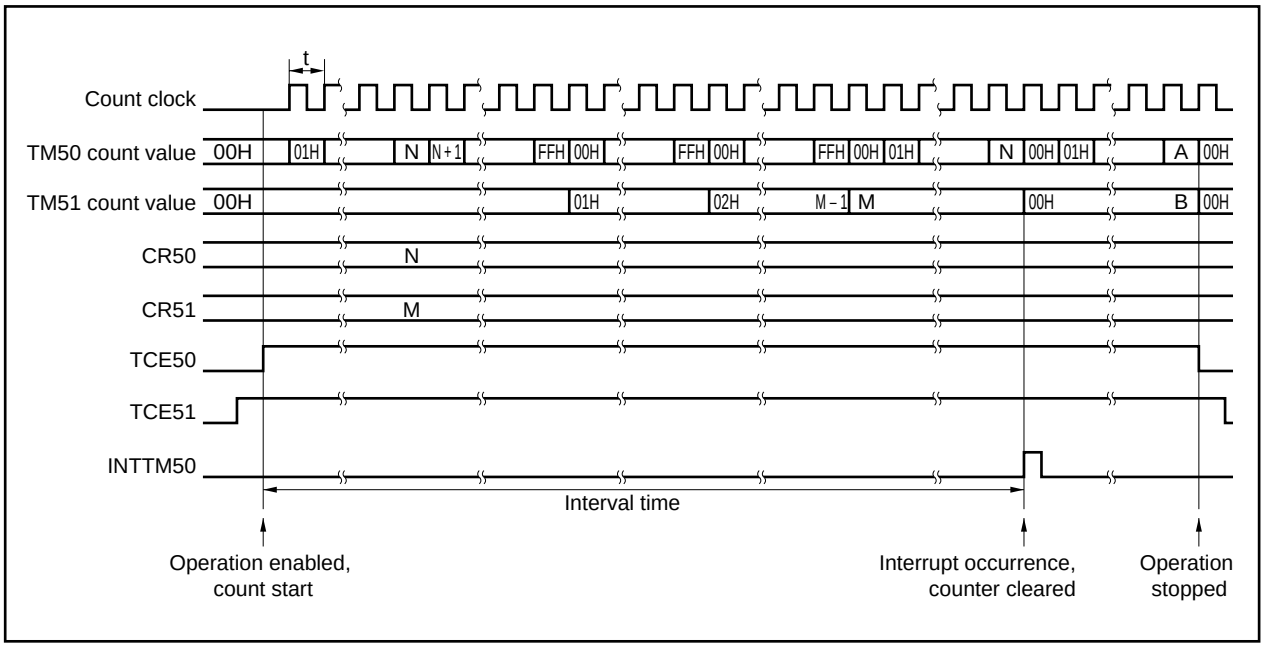
**Cautions 1. To write using 8-bit access during cascade connection, set the TCE51 bit to 1 at operation start and then set the TCE50 bit to 1. When operation is stopped, clear the TCE50 bit to 0 and then clear the TCE51 bit to 0.**

**2. During cascade connection, TI50 pin input, TO50 pin output, and the INTTM50 signal are used. Do not use TI51 pin input, TO51 pin output, and the INTTM51 signal; mask them instead (for details, refer to CHAPTER 19 INTERRUPT/EXCEPTION PROCESSING FUNCTION). Clear the LVS51, LVR51, TMC511, and TOE51 bits to 0.**

**3. Do not change the value of the CR5 register during timer operation.**

Figure 9-7 shows a timing example of the cascade connection mode with 16-bit resolution.

**Figure 9-7. Cascade Connection Mode with 16-Bit Resolution**



### 9.4.6 Operation as external event counter (16 bits)

The 16-bit resolution timer/event counter mode is selected by setting the TMC51.TMC514 bit to 1.

The external event counter counts the number of clock pulses input to the TI50 pin from an external source using 16-bit timer counter 5 (TM5).

Setting method

<1> Set each register.

- TCL50 register: Selects the TI50 pin input edge.
  (The TCL51 register does not have to be set during cascade connection.)
  Falling edge of TI50 pin → TCL50 register = 00H
  Rising edge of TI50 pin → TCL50 register = 01H
- CR50 register: Compare value (N) ... Lower 8 bits (settable from 00H to FFH)
- CR51 register: Compare value (N) ... Higher 8 bits (settable from 00H to FFH)
- TMC50, TMC51 registers: Stops count operation, selects the clear & start mode entered on a match between the TM5 register and CR5 register, disables timer output F/F inversion, and disables timer output.
  (×: don't care)
  (TMC50 register = 0000xx00B
  TMC51 register = 0001xx00B)
- For the alternate-function pin settings, refer to **Table 4-14 Settings When Port Pins Are Used for Alternate Functions.**

<2> Set the TMC51.TCE51 bit to 1. Then set the TMC50.TCE50 bit to 1 and count the number of pulses input from the TI50 pin.

<3> When the values of the TM5 register and CR5 register connected in cascade match, the INTTM50 signal is generated (the TM5 register is cleared to 0000H).

<4> The INTTM50 signal is then generated each time the values of the TM5 register and CR5 register match.

INTTM50 signal is generated when the valid edge is input to the TI50 pin N + 1 times: N = 0000H to FFFFH

**Cautions 1. During external event counter operation, do not rewrite the value of the CR5n register.**

**2. To write using 8-bit access during cascade connection, set the TCE51 bit to 1 and then set the TCE50 bit to 1. When operation is stopped, clear the TCE50 bit to 0 and then clear the TCE51 bit to 0 (n = 0, 1).**

**3. During cascade connection, TI50 pin input and the INTTM50 signal are used. Do not use TI51 pin input, TO51 pin output, and the INTTM51 signal; mask them instead (for details, refer to CHAPTER 19 INTERRUPT/EXCEPTION PROCESSING FUNCTION). Clear the LVS51, LVR51, TMC511, and TOE51 bits to 0.**

**4. Do not change the value of the CR5 register during external event counter operation.**

### 9.4.7 Square-wave output operation (16-bit resolution)

The 16-bit resolution timer/event counter mode is selected by setting the TMC51.TMC514 bit to 1.

8-bit timer/event counter 5n outputs a square wave of any frequency using the interval preset in 16-bit timer compare register 5 (CR5).

Setting method

<1> Set each register.

- TCL50 register: Selects the count clock (t)
  (The TCL51 register does not have to be set in cascade connection)
- CR50 register: Compare value (N) ... Lower 8 bits (settable from 00H to FFH)
- CR51 register: Compare value (N) ... Higher 8 bits (settable from 00H to FFH)
- TMC50, TMC51 registers: Stops count operation, selects the mode in which clear & start occurs on a match between the TM5 register and CR5 register.

| LVS50 | LVR50 | Timer Output F/F Status Settings |
|---|---|---|
| 1 | 0 | High-level output |
| 0 | 1 | Low-level output |

Enables timer output F/F inversion, and enables timer output.
(TMC50 register = 00001011B or 00000111B
TMC51 register = 00010000B)

- For the alternate-function pin settings, refer to **Table 4-14 Settings When Port Pins Are Used for Alternate Functions.**

<2> Set the TMC51.TCE51 bit to 1. Then set the TMC50.TCE50 bit to 1 to start the count operation.

<3> When the values of the TM5 register and the CR5 register connected in cascade match, the TO50 timer output F/F is inverted. Moreover, the INTTM50 signal is generated and the TM5 register is cleared to 0000H.

<4> Then, the timer output F/F is inverted during the same interval and a square wave is output from the TO50 pin.

Frequency = 1/2t(N + 1): N = 0000H to FFFFH

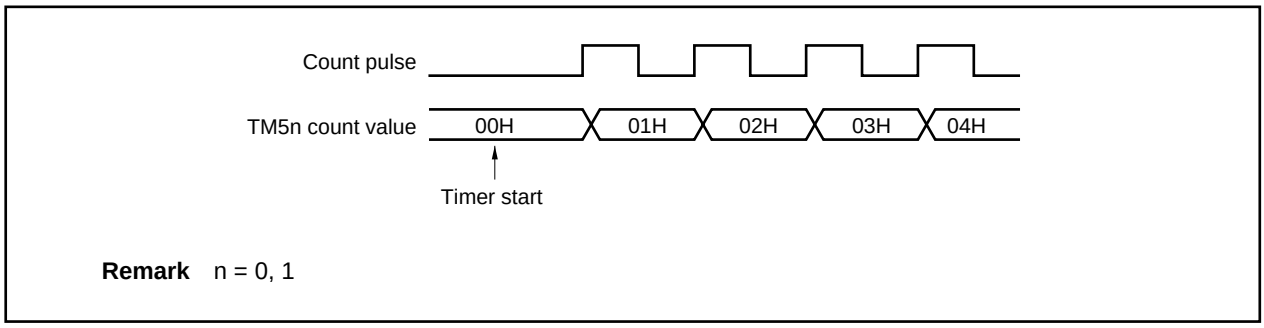
**Caution Do not write a different value to the CR5 register during operation.**

### 9.4.8 Cautions

**(1) Error on starting timer**

An error of up to 1 clock occurs before the match signal is generated after the timer has been started. This is because the TM5n register is started asynchronously to the count pulse.

**Figure 9-8. Count Start Timing of TM5n Register**




**Remark** n = 0, 1

# CHAPTER 10 8-BIT TIMER H

In the V850ES/KF1+, two channels of 8-bit timer H are provided.

## 10.1 Functions

8-bit timer Hn has the following functions.

- Interval timer
- PWM output
- Square wave output
- Carrier generator mode

**Remark** n = 0, 1

## 10.2 Configuration

8-bit timer Hn consists of the following hardware.

**Table 10-1. Configuration of 8-Bit Timer Hn**

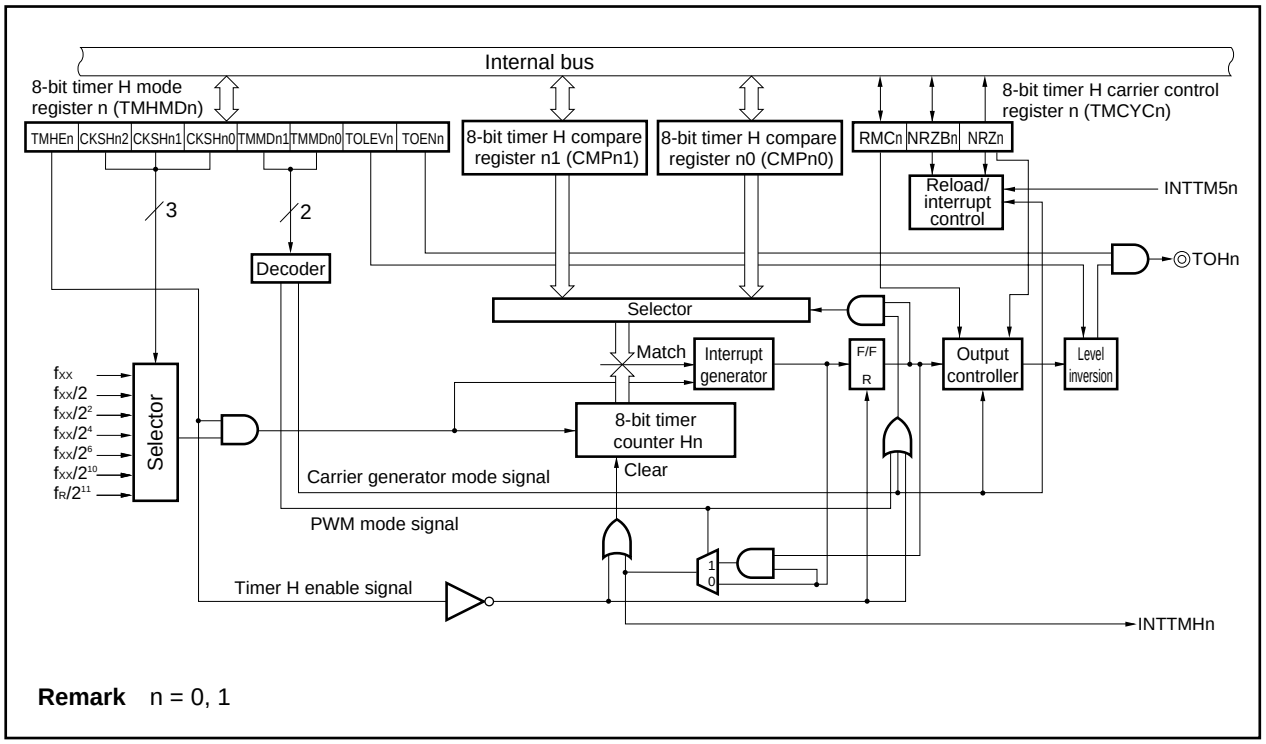
| Item | Configuration |
|---|---|
| Timer registers | 8-bit timer counter Hn: 1 each |
| Registers | 8-bit timer H compare register n0 (CMPn0): 1 each<br>8-bit timer H compare register n1 (CMPn1): 1 each |
| Timer outputs | 1 each (TOHn pin) |
| Control registers[Note] | 8-bit timer H mode register n (TMHMDn)<br>8-bit timer H carrier control register n (TMCYCn) |

**Note** To use the TOHn pin function, refer to **Table 4-14 Settings When Port Pins Are Used for Alternate Functions**.

**Remark** n = 0, 1

The block diagram of 8-bit timer Hn is shown below.

**Figure 10-1. Block Diagram of 8-Bit Timer Hn**




**Remark** n = 0, 1

**(1) 8-bit timer H compare register n0 (CMPn0)**

The CMPn0 register can be read or written in 8-bit units.

After reset, CMPn0 is cleared to 00H.

After reset: 00H R/W Address: CMP00 FFFFF582H, CMP10 FFFFF592H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CMPn0 | | | | | | | | |

(n = 0, 1)

**Caution Rewriting the CMPn0 register during timer count operation is prohibited.**

**(2) 8-bit timer H compare register n1 (CMPn1)**

The CMPn1 register can be read or written in 8-bit units.

After reset, CMPn1 is cleared to 00H.

After reset: 00H R/W Address: CMP01 FFFFF583H, CMP11 FFFFF593H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CMPn1 | | | | | | | | |

(n = 0, 1)

The CMPn1 register can be rewritten during timer count operation.

In the carrier generator mode, after the CMPn1 register is set, if the count value of 8-bit timer counter Hn and the set value of the CMPn1 register match, an interrupt request signal (INTTMHn) is generated. At the same time, the value of 8-bit timer counter Hn is cleared to 00H.

If the set value of the CMPn1 register is rewritten during timer operation, the reload timing is when the count value of 8-bit timer counter Hn and the set value of the CMPn1 register match. If the transfer timing and write to the CMPn1 register by software conflict, transfer is not performed.

**Caution In the PWM output mode and carrier generator mode, be sure to set the CMPn1 register when starting the timer count operation (TMHMDn.TMHEn bit = 1) after the timer count operation was stopped (TMHEn bit = 0) (be sure to set again even if setting the same value to the CMPn1 register).**

## 10.3 Registers

The registers that control 8-bit timer Hn are as follows.

- 8-bit timer H mode register n (TMHMDn)
- 8-bit timer H carrier control register n (TMCYCn)

**Remarks 1.** To use the TOHn pin function, refer to **Table 4-14 Settings When Port Pins Are Used for Alternate Functions**.

**2.** n = 0, 1

**(1) 8-bit timer H mode register n (TMHMDn)**

The TMHMDn register controls the mode of 8-bit timer Hn.

The TMHMDn register can be read or written in 8-bit or 1-bit units.

After reset, TMHMDn is cleared to 00H.

**Remark** n = 0, 1

### (a) 8-bit timer H mode register 0 (TMHMD0)

After reset: 00H R/W Address: FFFFF580H

| | <7> | 6 | 5 | 4 | 3 | 2 | <1> | <0> |
|---|---|---|---|---|---|---|---|---|
| TMHMD0 | TMHE0 | CKSH02 | CKSH01 | CKSH00 | TMMD01 | TMMD00 | TOLEV0 | TOEN0 |

| TMHE0 | 8-bit timer H0 operation enable |
|---|---|
| 0 | Stop timer count operation (8-bit timer counter H0 = 00H) |
| 1 | Enable timer count operation (Counting starts when clock is input) |

| CKSH02 | CKSH01 | CKSH00 | Selection of count clock | | | |
|---|---|---|---|---|---|---|
| | | | Count clock[Note] | $f_{XX}$ = 20 MHz | $f_{XX}$ = 16.0 MHz | $f_{XX}$ = 10.0 MHz |
| 0 | 0 | 0 | $f_{XX}$ | Setting prohibited | Setting prohibited | 100 ns |
| 0 | 0 | 1 | $f_{XX}/2$ | 100 ns | 125 ns | 200 ns |
| 0 | 1 | 0 | $f_{XX}/4$ | 200 ns | 250 ns | 400 ns |
| 0 | 1 | 1 | $f_{XX}/16$ | 800 ns | 1 $\mu$s | 1.6 $\mu$s |
| 1 | 0 | 0 | $f_{XX}/64$ | 1.6 $\mu$s | 4 $\mu$s | 6.4 $\mu$s |
| 1 | 0 | 1 | $f_{XX}/1024$ | 51.2 $\mu$s | 64 $\mu$s | 102.4 $\mu$s |
| Other than above | | | Setting prohibited | | | |

| TMMD01 | TMMD00 | 8-bit timer H0 operation mode |
|---|---|---|
| 0 | 0 | Interval timer mode |
| 0 | 1 | Carrier generator mode |
| 1 | 0 | PWM output mode |
| 1 | 1 | Setting prohibited |

| TOLEV0 | Timer output level control (default) |
|---|---|
| 0 | Low level |
| 1 | High level |

| TOEN0 | Timer output control |
|---|---|
| 0 | Disable output |
| 1 | Enable output |

**Note** Set so as to satisfy the following conditions.

$V_{DD}$ = REGC = 4.0 to 5.5 V: Count clock $\leq$ 10 MHz

$V_{DD}$ = 4.0 to 5.5 V, REGC = Capacity: Count clock $\leq$ 5 MHz

$V_{DD}$ = REGC = 2.7 to 4.0 V: Count clock $\leq$ 5 MHz

**Cautions 1. When the TMHE0 bit = 1, setting bits other than those of the TMHMD0 register is prohibited.**

**2. In the PWM output mode and carrier generator mode, be sure to set the CMP01 register when starting the timer count operation (TMHE0 bit = 1) after the timer count operation was stopped (TMHE0 bit = 0) (be sure to set again even if setting the same value to the CMP01 register).**

**3. When using the carrier generator mode, set 8-bit timer H0 count clock frequency to six times 8-bit timer/event counter 50 count clock frequency or higher.**

### (b) 8-bit timer H mode register 1 (TMHMD1)

After reset: 00H R/W Address: FFFFF590H

| | <7> | 6 | 5 | 4 | 3 | 2 | <1> | <0> |
|---|---|---|---|---|---|---|---|---|
| TMHMD1 | TMHE1 | CKSH12 | CKSH11 | CKSH10 | TMMD11 | TMMD10 | TOLEV1 | TOEN1 |

| TMHE1 | 8-bit timer H1 operation enable |
|---|---|
| 0 | Stop timer count operation (8-bit timer counter H1 = 00H) |
| 1 | Enable timer count operation (Counting starts when clock is input) |

| CKSH12 | CKSH11 | CKSH10 | Selection of count clock | | | |
|---|---|---|---|---|---|---|
| | | | Count clock[Note] | $f_{XX}$ = 20.0 MHz | $f_{XX}$ = 16.0 MHz | $f_{XX}$ = 10.0 MHz |
| 0 | 0 | 0 | $f_{XX}$ | Setting prohibited | Setting prohibited | 100 ns |
| 0 | 0 | 1 | $f_{XX}/2$ | 100 ns | 125 ns | 200 ns |
| 0 | 1 | 0 | $f_{XX}/4$ | 200 ns | 250 ns | 400 ns |
| 0 | 1 | 1 | $f_{XX}/16$ | 800 ns | 1 $\mu$s | 1.6 $\mu$s |
| 1 | 0 | 0 | $f_{XX}/64$ | 1.6 $\mu$s | 4 $\mu$s | 6.4 $\mu$s |
| 1 | 0 | 1 | $f_R/2048$ | | | |
| Other than above | | | Setting prohibited | | | |

| TMMD11 | TMMD10 | 8-bit timer H1 operation mode |
|---|---|---|
| 0 | 0 | Interval timer mode |
| 0 | 1 | Carrier generator mode |
| 1 | 0 | PWM output mode |
| 1 | 1 | Setting prohibited |

| TOLEV1 | Timer output level control (default) |
|---|---|
| 0 | Low level |
| 1 | High level |

| TOEN1 | Timer output control |
|---|---|
| 0 | Disable output |
| 1 | Enable output |

**Note** Set so as to satisfy the following conditions.

$V_{DD}$ = REGC = 4.0 to 5.5 V: Count clock ≤ 10 MHz

$V_{DD}$ = 4.0 to 5.5 V, REGC = Capacity: Count clock ≤ 5 MHz

$V_{DD}$ = REGC = 2.7 to 4.0 V: Count clock ≤ 5 MHz

**Cautions 1. When the TMHE1 bit = 1, setting bits other than those of the TMHMD1 register is prohibited.**

**2. In the PWM output mode and carrier generator mode, be sure to set the CMP11 register when starting the timer count operation (TMHE1 bit = 1) after the timer count operation was stopped (TMHE1 bit = 0) (be sure to set again even if setting the same value to the CMP11 register).**

**3. When using the carrier generator mode, set 8-bit timer H1 count clock frequency to six times 8-bit timer/event counter 51 count clock frequency or higher.**

**(2) 8-bit timer H carrier control register n (TMCYCn)**

This register controls the 8-bit timer Hn remote control output and carrier pulse output status.

The TMCYCn register can be read or written in 8-bit or 1-bit units, but the NRZn bit is a read-only bit.

After reset, TMCYCn is cleared to 00H.

After reset: 00H R/W Address: TMCYC0 FFFFF581H, TMCYC1 FFFFF591H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | <0> |
|---|---|---|---|---|---|---|---|---|
| TMCYCn | 0 | 0 | 0 | 0 | 0 | RMCn | NRZBn | NRZn |

(n = 0, 1)

| RMCn | NRZBn | Remote control output |
|---|---|---|
| 0 | 0 | Low-level output |
| 0 | 1 | High-level output |
| 1 | 0 | Low-level output |
| 1 | 1 | Carrier pulse output |

| NRZn | Carrier pulse output status flag |
|---|---|
| 0 | Carrier output disabled status (low-level status) |
| 1 | Carrier output enable status |

## 10.4 Operation

### 10.4.1 Operation as interval timer/square wave output

When the count value of 8-bit timer counter Hn and the set value of the CMPn0 register match, an interrupt request signal (INTTMHn) is generated and 8-bit timer counter Hn is cleared to 00H.

The CMPn1 register cannot be used in the interval timer mode. Even if the CMPn1 register is set, this has no effect on the timer output because matches between 8-bit timer counter Hn and the CMPn1 register are not detected.

A square wave of the desired frequency (duty = 50%) is output from the TOHn pin, by setting the TMHMDn.TOENn bit to 1.

**(1) Usage method**

The INTTMHn signal is repeatedly generated in the same interval.

<1> Set each register.

**Figure 10-2. Register Settings in Interval Timer Mode**

(i) 8-bit timer H mode register n (TMHMDn) settings

| | TMHEn | CKSHn2 | CKSHn1 | CKSHn0 | TMMDn1 | TMMDn0 | TOLEVn | TOENn |
|---|---|---|---|---|---|---|---|---|
| TMHMDn | 0 | 0/1 | 0/1 | 0/1 | 0 | 0 | 0/1 | 0/1 |

- TOENn → Sets timer output
- TOLEVn → Sets timer output level inversion
- TMMDn0 → Sets interval timer mode
- CKSHn2, CKSHn1, CKSHn0 → Selects count clock ($f_{CNT}$)
- TMHEn → Stops count operation

(ii) CMPn0 register settings

- Compare value (N)

<2> When the TMHEn bit is set to 1, counting starts.

<3> When the count value of 8-bit timer counter Hn and the set value of the CMPn0 register match, the INTTMHn signal is generated and 8-bit timer counter Hn is cleared to 00H.

$$\text{Interval time} = (N + 1)/f_{CNT}$$

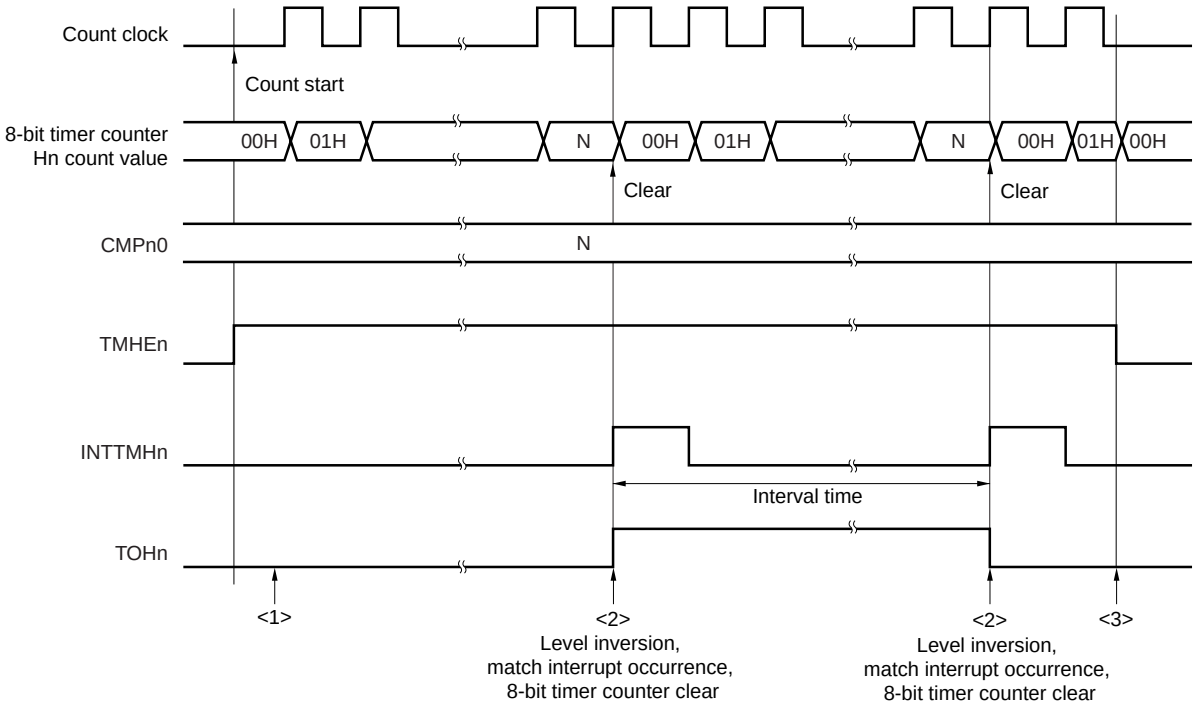
<4> Then, the INTTMHn signal is generated in the same interval. To stop the count operation, clear the TMHEn bit to 0.

**(2) Timing chart**

The timing in the interval timer mode is as follows.

**Figure 10-3. Timing of Interval Timer/Square Wave Output Operation (1/2)**

Basic operation



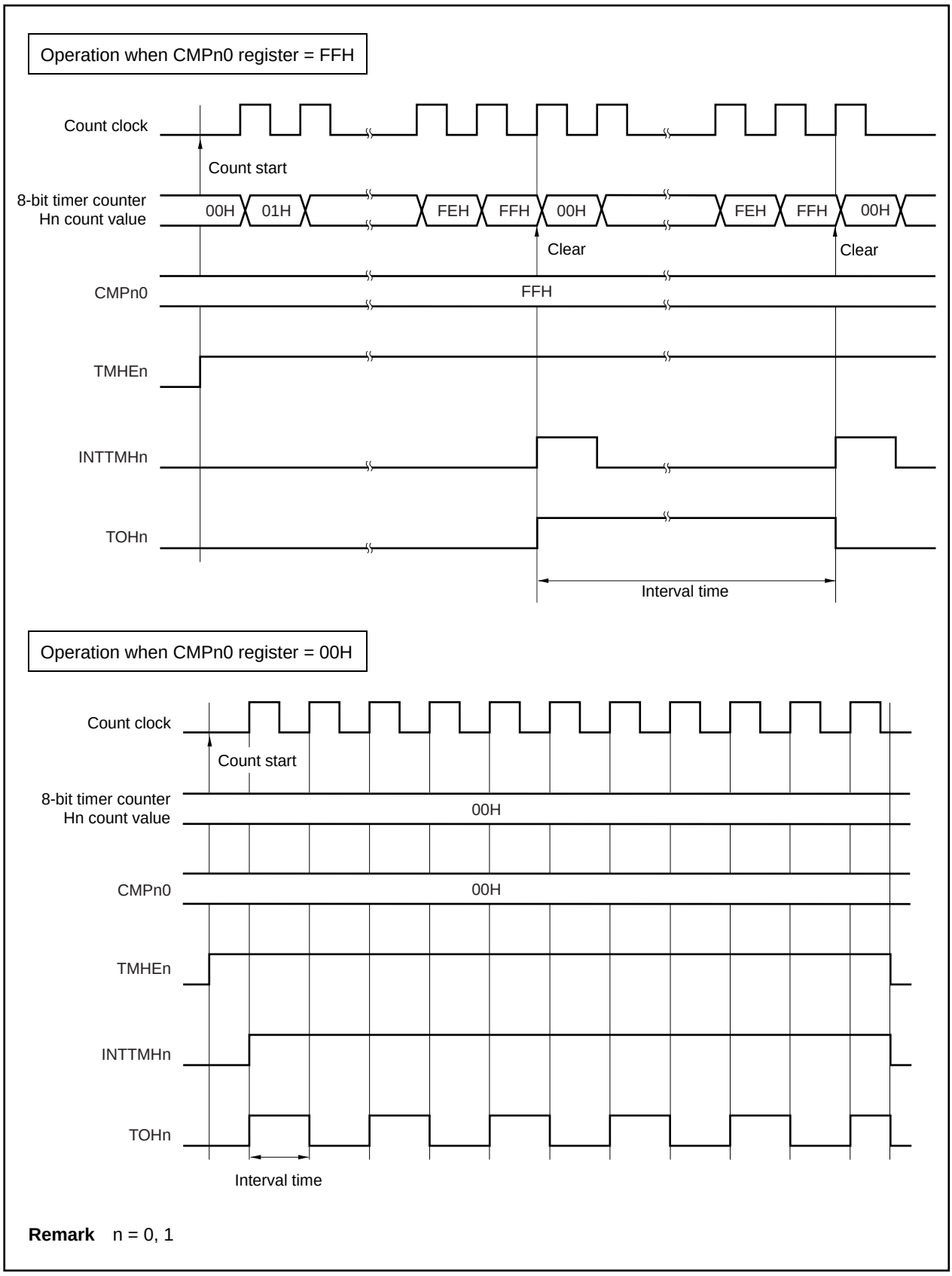

<1> When the TMHEn bit is set to 1, the count operation is enabled. The count clock starts counting no more than one clock after operation has been enabled.

<2> When the count value of 8-bit timer counter Hn and the set value of the CMPn0 register match, the value of 8-bit timer counter Hn is cleared, the TOHn output level is inverted, and the INTTMHn signal is output.

<3> The INTTMHn signal and TOHn output become inactive when the TMHEn bit is cleared to 0 during 8-bit timer Hn operation. If the level is already inactive, it remains unchanged.

**Remark** n = 0, 1

**Figure 10-3. Timing of Interval Timer/Square Wave Output Operation (2/2)**




**Remark** n = 0, 1

### 10.4.2 PWM output mode operation

In the PWM output mode, a pulse of any duty and cycle can be output.

The CMPn0 register controls the timer output (TOHn) cycle. Rewriting the CMPn0 register during timer operation is prohibited.

The CMPn1 register controls the timer output (TOHn) duty. The CMPn1 register can be rewritten during timer operation.

The operation in the PWM output mode is as follows.

After timer counting starts, when the count value of 8-bit timer counter Hn and the set value of the CMPn0 register match, the TOHn output becomes active and 8-bit timer counter Hn is cleared to 00H. When the count value of 8-bit timer counter Hn and the set value of the CMPn1 register match, TOHn output becomes inactive.

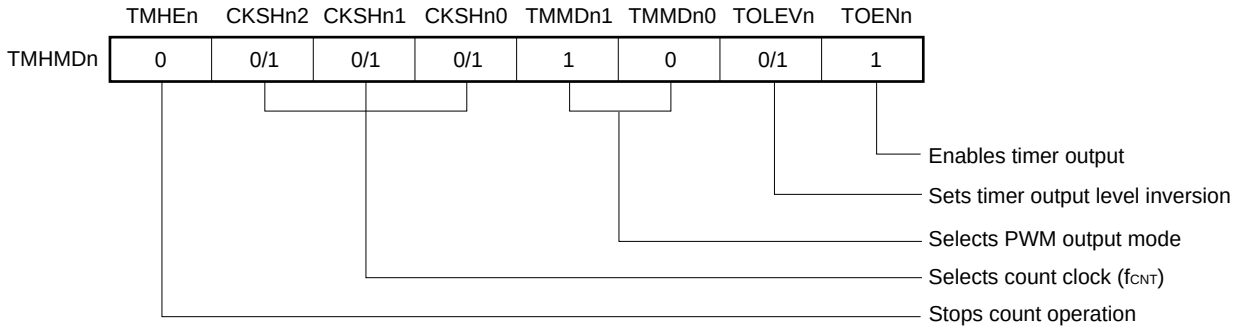
#### (1) Usage method

In the PWM output mode, a pulse of any duty and cycle can be output.

<1> Set each register.

**Figure 10-4. Register Settings in PWM Output Mode**

(i) 8-bit timer H mode register n (TMHMDn) settings




(ii) CMPn0 register setting

- Compare value (N): Sets cycle

(iii) CMPn1 register setting

- Compare value (M): Sets duty

**Remarks 1.** n = 0, 1

**2.** 00H ≤ CMPn1 (M) < CMPn0 (N) ≤ FFH

<2> When the TMHEn bit is set to 1, counting starts.

<3> After the count operation is enabled, the first compare register to be compared is the CMPn0 register. When the count value of 8-bit timer counter Hn and the set value of the CMPn0 register match, 8-bit timer counter Hn is cleared, an interrupt request signal (INTTMHn) is generated, and the TOHn output becomes active. At the same time, the register that is compared with 8-bit timer counter Hn changes from the CMPn0 register to the CMPn1 register.

<4> When the count value of 8-bit timer counter Hn and the set value of the CMPn1 register match, the TOHn output becomes inactive, and at the same time the register that is compared with 8-bit timer counter Hn changes from the CMPn1 register to the CMPn0 register. At this time, 8-bit timer counter Hn is not cleared and the INTTMHn signal is not generated.

<5> A pulse of any duty can be obtained through the repetition of steps <3> and <4> above.

<6> To stop the count operation, clear the TMHEn bit to 0.

Designating the set value of the CMPn0 register as (N), the set value of the CMPn1 register as (M), and the count clock frequency as $f_{CNT}$, the PWM pulse output cycle and duty are as follows.

PWM pulse output cycle = $(N + 1)/f_{CNT}$
Duty = inactive width: Active width = (M + 1) : (N + 1)

**Cautions 1. In the PWM output mode, three operating clocks (signal selected by CKSHn0 to CKSHn2 bits) are required for actual transfer of the new value to the register after the CMPn1 register has been rewritten.**

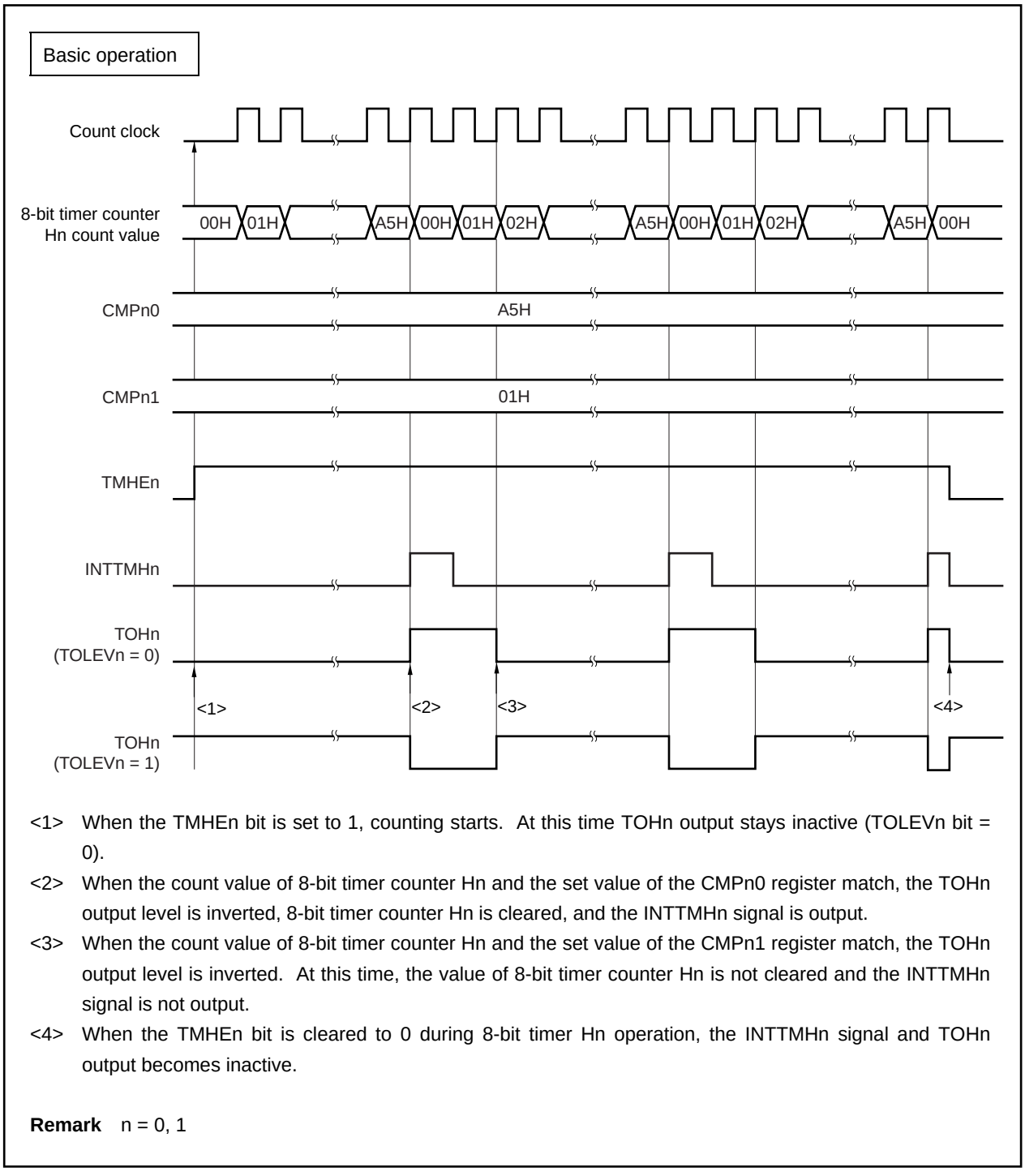
**2. Be sure to set the CMPn1 register when starting the timer count operation (TMHEn bit = 1) after the timer count operation was stopped (TMHEn bit = 0) (be sure to set again even if setting the same value to the CMPn1 register).**

**(2) Timing chart**

The operation timing in the PWM output mode is as follows.

**Caution The set value (M) of the CMPn1 register and the set value (N) of the CMPn0 register must always be set within the following range.**
**00H ≤ CMPn1 (M) < CMPn0 (N) ≤ FFH**

**Figure 10-5. Operation Timing in PWM Output Mode (1/4)**




<1> When the TMHEn bit is set to 1, counting starts. At this time TOHn output stays inactive (TOLEVn bit = 0).

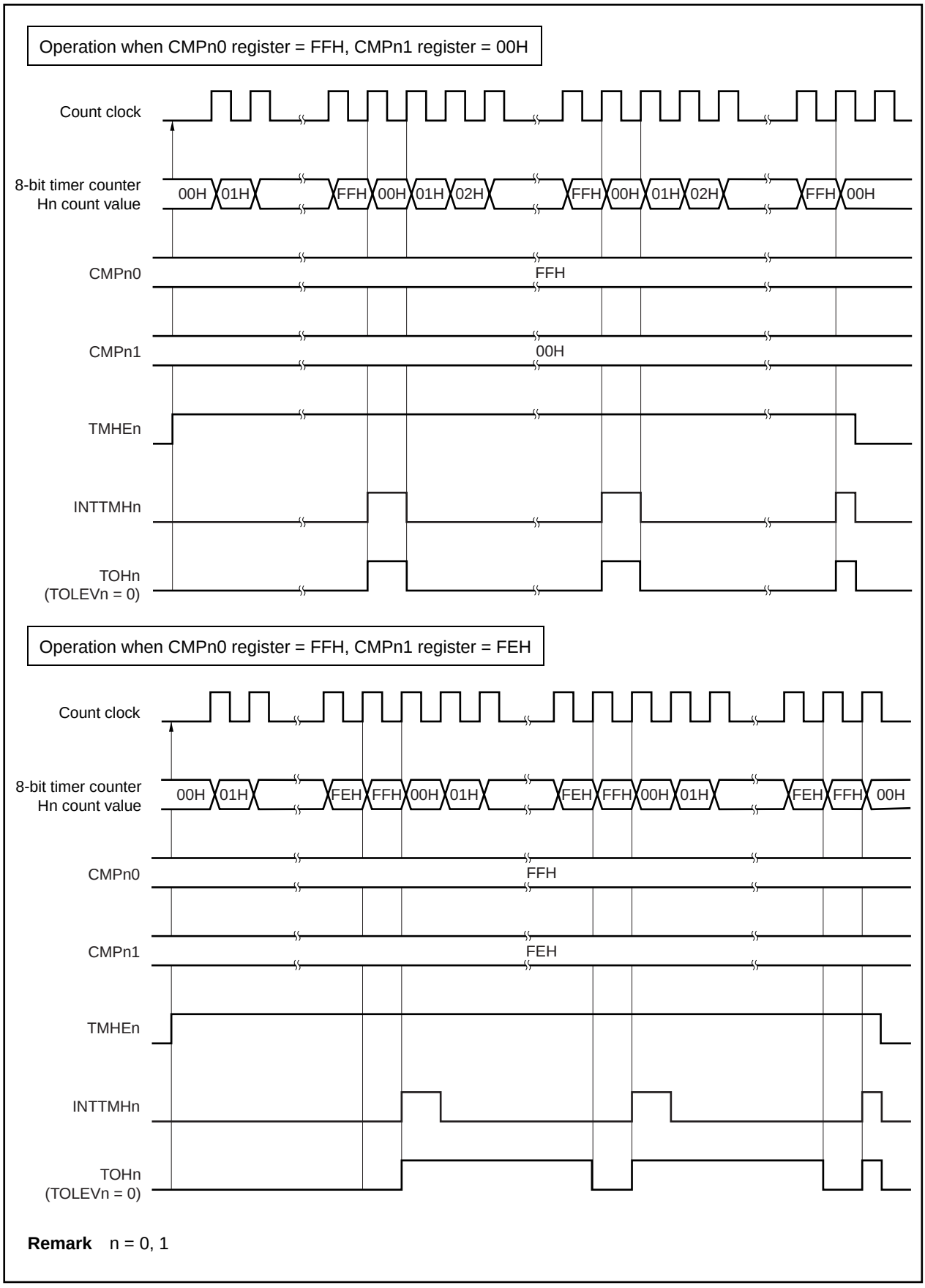
<2> When the count value of 8-bit timer counter Hn and the set value of the CMPn0 register match, the TOHn output level is inverted, 8-bit timer counter Hn is cleared, and the INTTMHn signal is output.

<3> When the count value of 8-bit timer counter Hn and the set value of the CMPn1 register match, the TOHn output level is inverted. At this time, the value of 8-bit timer counter Hn is not cleared and the INTTMHn signal is not output.

<4> When the TMHEn bit is cleared to 0 during 8-bit timer Hn operation, the INTTMHn signal and TOHn output becomes inactive.

**Remark** n = 0, 1

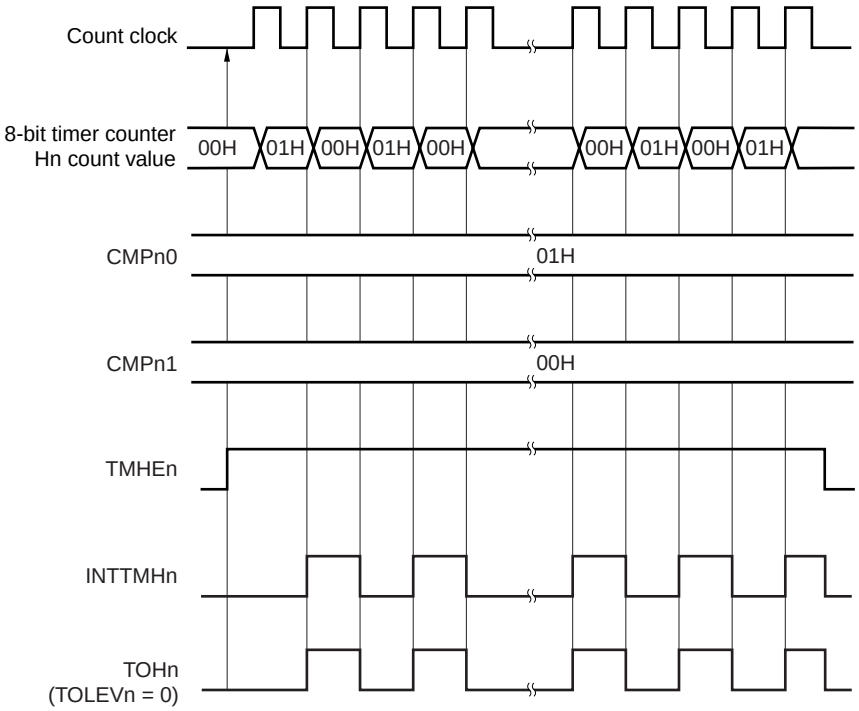
**Figure 10-5. Operation Timing in PWM Output Mode (2/4)**




**Remark** n = 0, 1

**Figure 10-5. Operation Timing in PWM Output Mode (3/4)**

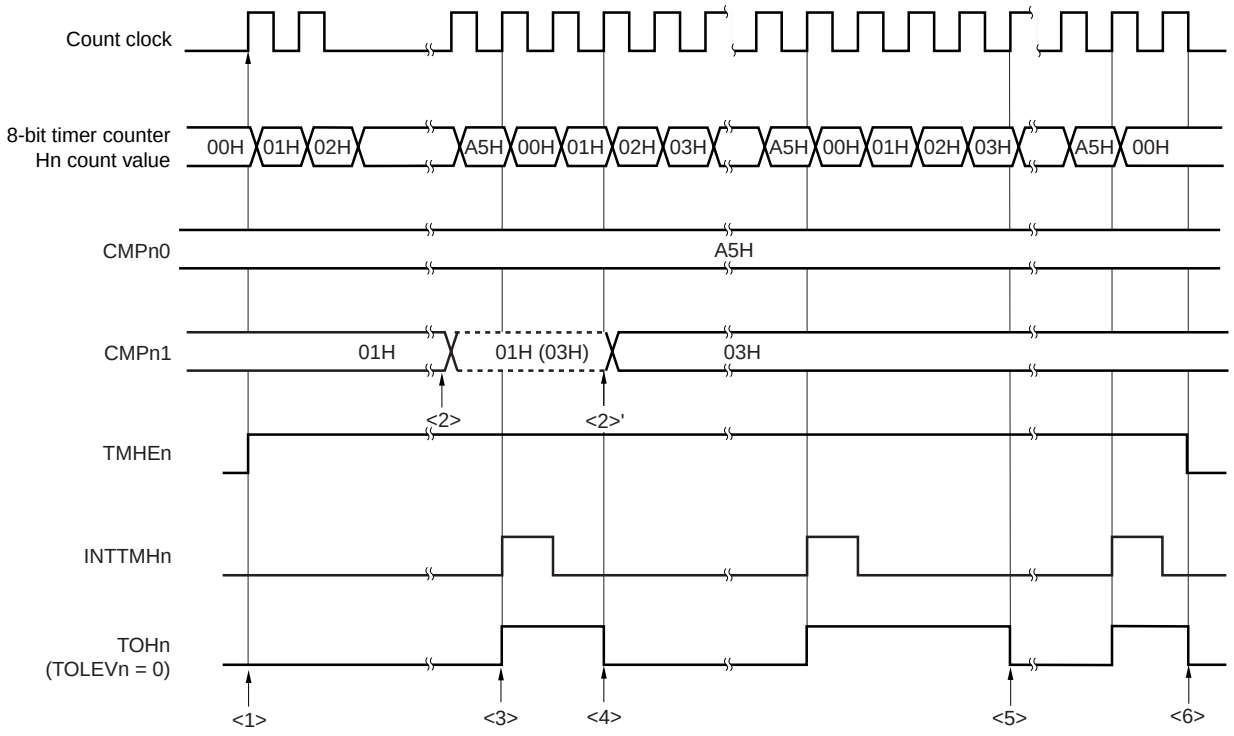
Operation when CMPn0 register = 01H, CMPn1 register = 00H




**Remark** n = 0, 1

**Figure 10-5. Operation Timing in PWM Output Mode (4/4)**

Operation based on CMPn1 register transitions (CMPn1 register = 01H → 03H, CMPn0 register = A5H)




<1> When the TMHEn bit is set to 1, counting starts. At this time, the TOHn output remains inactive (TOLEVn bit = 0).

<2> The set value of the CMPn1 register can be changed during count operation. This operation is asynchronous to the count clock.

<3> When the count value of 8-bit timer counter Hn and the set value of the CMPn0 register match, 8-bit timer counter Hn is cleared, the TOHn output becomes active, and the INTTMHn signal is generated.

<4> Even if the value of the CMPn1 register is changed, that value is latched and not transferred to the register. When the count value of 8-bit timer counter Hn and the set value of the CMPn1 register prior to the change match, the changed value is transferred to the CMPn1 register and the value of the CMPn1 register is changed (<2>').

However, three or more count clocks are required from the time the value of the CMPn1 register is changed until it is transferred to the register. Even if a match signal is generated within three count clocks, the changed value cannot be transferred to the register.

<5> When the count value of 8-bit timer counter Hn matches the changed set value of the CMPn1 register, the TOHn output becomes inactive. 8-bit timer counter Hn is not cleared and the INTTMHn signal is not generated.

<6> When the TMHEn bit is cleared to 0 during 8-bit timer Hn operation, the INTTMHn signal and TOHn output become inactive.

### 10.4.3 Carrier generator mode operation

The carrier clock generated by 8-bit timer Hn is output using the cycle set with 8-bit timer/event counter 5n.

In the carrier generator mode, 8-bit timer/event counter 5n is used to control the extent to which the carrier pulse of 8-bit timer Hn is output, and the carrier pulse is output from the TOHn output.

#### (1) Carrier generation

In the carrier generator mode, the CMPn0 register generates a waveform with the low-level width of the carrier pulse and the CMPn1 register generates a waveform with the high-level width of the carrier pulse.

During 8-bit timer Hn operation, the CMPn1 register can be rewritten, but rewriting of the CMPn0 register is prohibited.

#### (2) Carrier output control

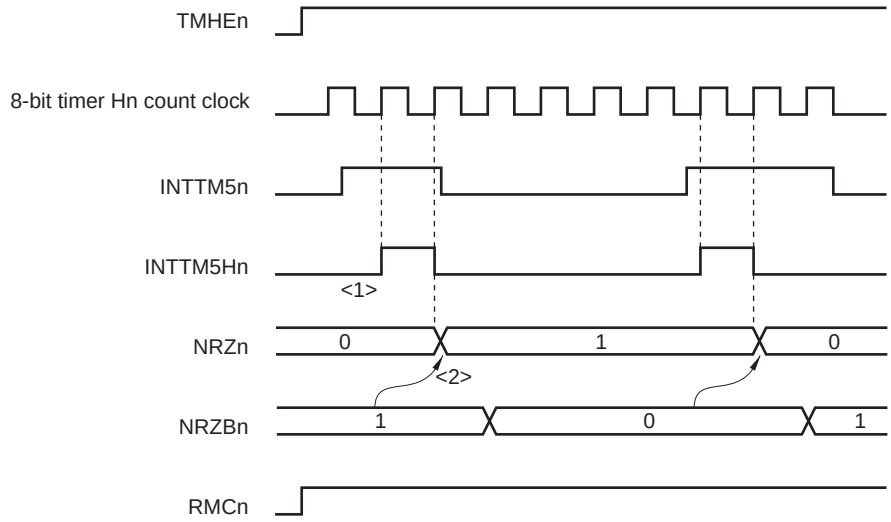
Carrier output control is performed with the interrupt request signal (INTTM5n) of 8-bit timer/event counter 5n and the TMCYCn.NRZBn and TMCYCn.RMCn bits. The output relationships are as follows.

| RMCn Bit | NRZBn Bit | Output |
|---|---|---|
| 0 | 0 | Low level output |
| 0 | 1 | High level output |
| 1 | 0 | Low level output |
| 1 | 1 | Carrier pulse output |

**Remark** n = 0, 1

To control carrier pulse output during count operation, the TMCYCn.NRZn and TMCYCn.NRZBn bits have a master and slave bit configuration. The NRZn bit is read-only while the NRZBn bit can be read and written. The INTTM5n signal is synchronized with the 8-bit timer Hn clock and output as the INTTM5Hn signal. The INTTM5Hn signal becomes the data transfer signal of the NRZn bit and the value of the NRZBn bit is transferred to the NRZn bit. The transfer timing from the NRZBn bit to the NRZn bit is as follows.

**Figure 10-6. Transfer Timing**




<1> The INTTM5n signal is synchronized with the count clock of 8-bit timer Hn and is output as the INTTM5Hn signal.

<2> The value of the NRZBn bit is transferred to the NRZn bit at the second clock from the rising edge of the INTTM5Hn signal.

**Cautions 1. Do not rewrite the NRZBn bit again until at least the second clock after it has been rewritten, or else transfer from the NRZBn bit to the NRZn bit is not guaranteed.**

**2. When using 8-bit timer/event counter 5n in the carrier generator mode, an interrupt occurs at the timing of <1>. An interrupt occurs at a different timing when it is used in other than the carrier generator mode.**

**Remark** n = 0, 1

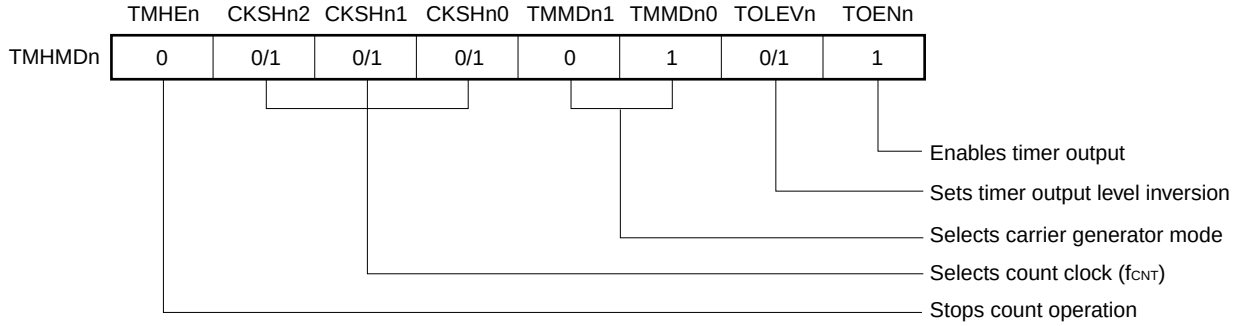
**(3) Usage method**

Any carrier clock can be output from the TOHn pin.

<1> Set each register.

**Figure 10-7. Register Settings in Carrier Generator Mode**

- 8-bit timer H mode register n (TMHMDn)




- CMPn0 register: Compare value
- CMPn1 register: Compare value
- TMCYCn register: RMCn = 1 ... Remote control output enable bit
  NRZBn = 0/1 ... Carrier output enable bit
- TCL5n, TMC5n registers: Refer to **9.3 Registers**.

**Remark** n = 0, 1

<2> When the TMHEn bit is set to 1, 8-bit timer Hn count operation starts.

<3> When the TMC5n.TCE5n bit is set to 1, 8-bit timer/event counter 5n count operation starts.

<4> After the count operation is enabled, the first compare register to be compared is the CMPn0 register. When the count value of 8-bit timer counter Hn and the set value of the CMPn0 register match, the INTTMHn signal is generated, 8-bit timer counter Hn is cleared, and at the same time, the register that is compared with 8-bit timer counter Hn changes from the CMPn0 register to the CMPn1 register.

<5> When the count value of 8-bit timer counter Hn and the set value of the CMPn1 register match, the INTTMHn signal is generated, 8-bit timer counter Hn is cleared, and at the same time, the register that is compared with 8-bit timer counter Hn changes from the CMPn1 register to the CMPn0 register.

<6> The carrier clock is obtained through the repetition of steps <4> and <5> above.

<7> The INTTM5n signal is synchronized with 8-bit timer Hn and output as the INTTM5Hn signal. This signal becomes the data transfer signal of the NRZBn bit and the value of the NRZBn bit is transferred to the NRZn bit.

<8> When the NRZn bit becomes high level, the carrier clock is output from the TOHn pin.

<9> Any carrier clock can be obtained through the repetition of the above steps. To stop the count operation, clear the TMHEn bit to 0.

Designating the set value of the CMPn0 register as (N), the set value of the CMPn1 register as (M), and the count clock frequency as $f_{CNT}$, the carrier clock output cycle and duty are as follows.

| |
|---|
| Carrier clock output cycle = (N + M + 2)/$f_{CNT}$<br>Duty = High level width: Carrier clock output width = (M + 1) : (N + M + 2) |

**Caution Be sure to set the CMPn1 register when starting the timer count operation (TMHEn bit = 1) after the timer count operation was stopped (TMHEn bit = 0) (be sure to set again even if setting the same value to the CMPn1 register).**

**(4) Timing chart**

The carrier output control timing is as follows.

**Cautions 1. Set the values of the CMPn0 and CMPn1 registers in the range of 01H to FFH.**

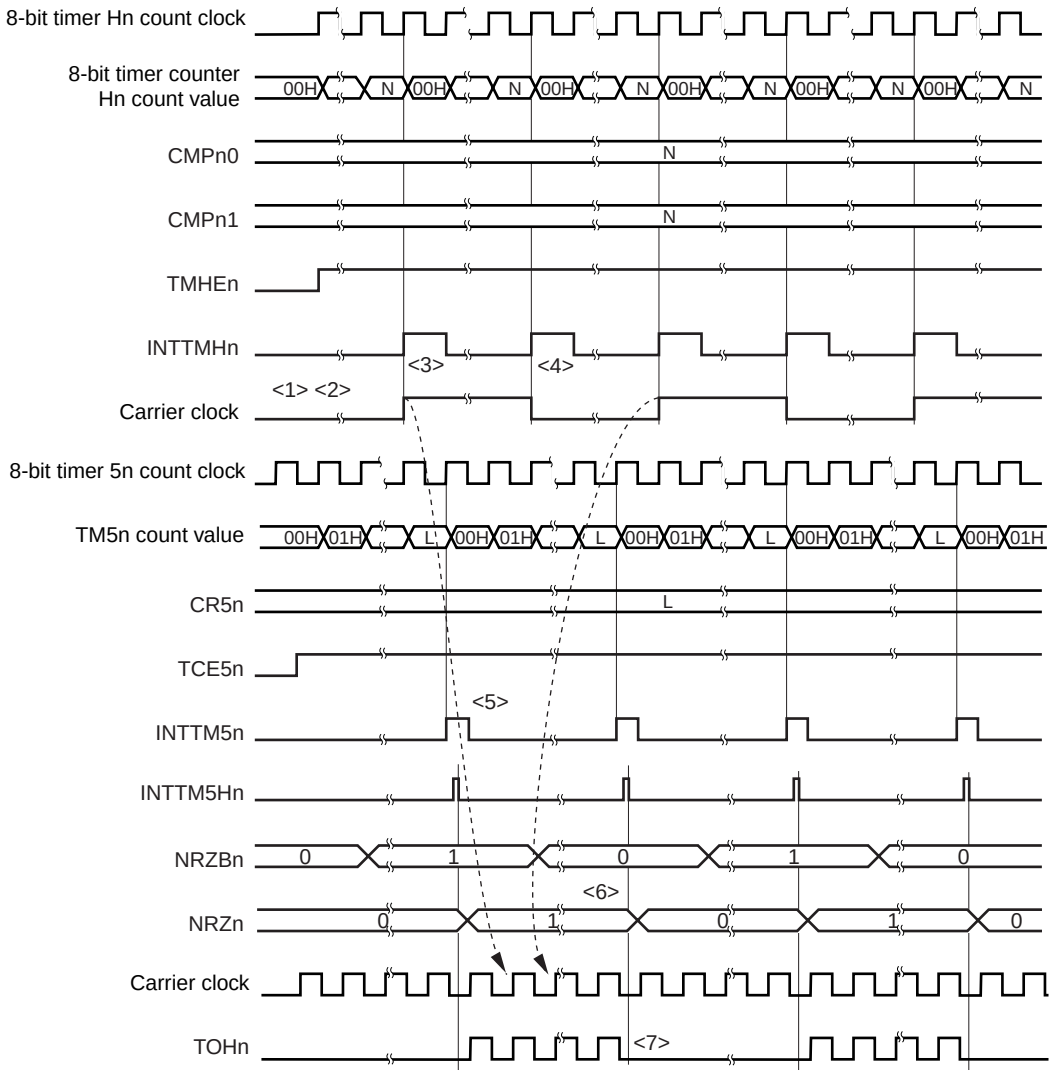
**2. In the carrier generator mode, three operating clocks (signal selected by the TMHMDn.CKSHn0 to TMHMDn.CKSHn2 bits) are required for actual transfer of the new value to the register after the CMPn1 register has been rewritten.**

**3. Be sure to perform the TMCYCn.RMCn bit setting before the start of the count operation.**

**4. When using the carrier generator mode, set the 8-bit timer Hn count clock frequency to six times the 8-bit timer/event counter 5n count clock frequency or higher.**

**Figure 10-8. Carrier Generator Mode (1/3)**

Operation when CMPn0 register = N, CMPn1 register = N is set




<1> When the TMHEn bit = 0 and the TCE5n bit = 0, the operation of 8-bit timer Hn is stopped.

<2> When the TMHEn bit is set to 1, 8-bit timer Hn starts counting. The carrier clock is maintained inactive at this time.

<3> When the count value of 8-bit timer counter Hn and the set value of the CMPn0 register match, the first INTTMHn signal is generated, the carrier clock signal is inverted, and the register that is compared with 8-bit timer counter Hn changes from the CMPn0 register to the CMPn1 register. 8-bit timer counter Hn is cleared to 00H.

<4> When the count value of 8-bit timer counter Hn and the set value of the CMPn1 register match, the INTTMHn signal is generated, the carrier clock signal is inverted, and the register that is compared with 8-bit timer counter Hn changes from the CMPn1 register to the CMPn0 register. 8-bit timer counter Hn is cleared to 00H. A carrier clock with a duty of 50% is generated through the repetition of steps <3> and <4>.

<5> The INTTM5n signal is synchronized with 8-bit timer Hn and output as the INTTM5Hn signal.

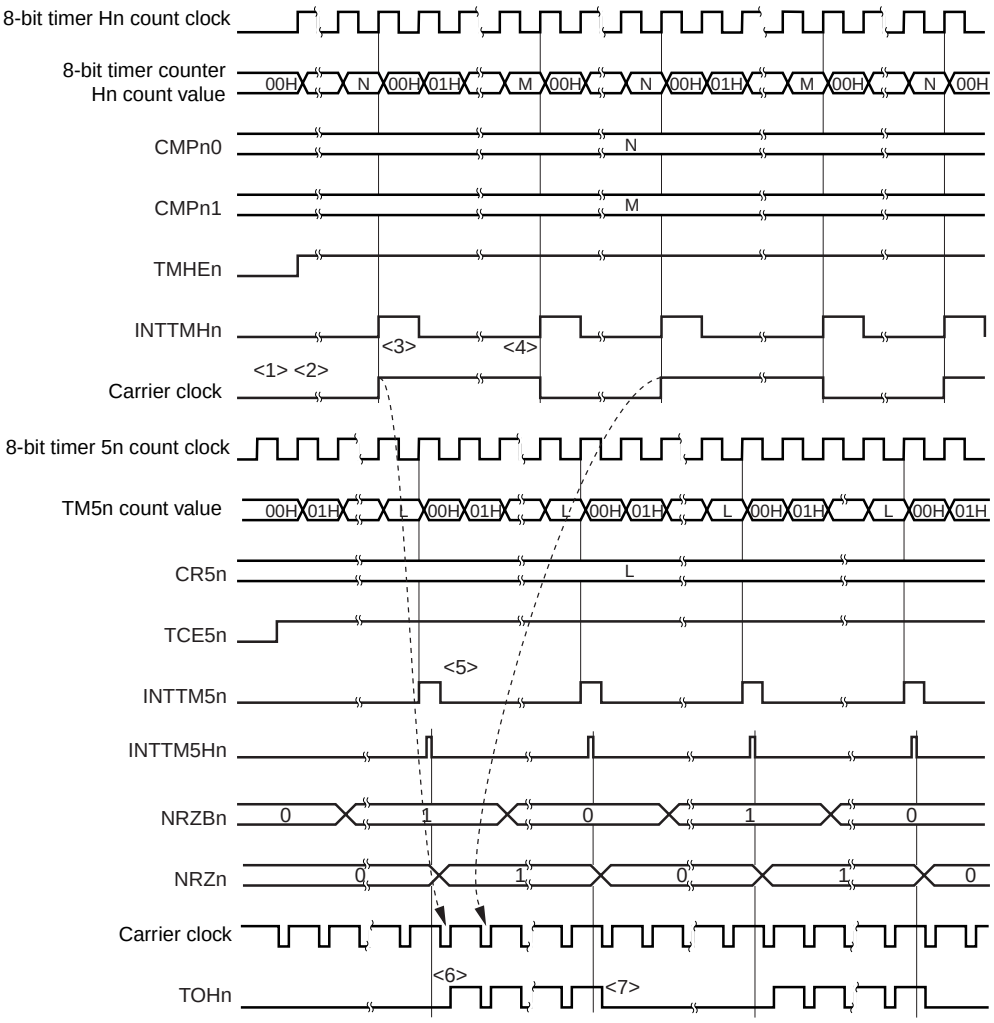
<6> The INTTM5Hn signal becomes the data transfer signal of the NRZBn bit, and the value of the NRZBn bit is transferred to the NRZn bit.

<7> The TOHn output is made low level by clearing the NRZn bit to 0.

**Remark** n = 0, 1

**Figure 10-8. Carrier Generator Mode (2/3)**

Operation when CMPn0 register = N, CMPn1 register = M is set




<1> When the TMHEn bit = 0 and the TCE5n bit = 0, the operation of 8-bit timer Hn is stopped.

<2> When the TMHEn bit is set to 1, 8-bit timer Hn starts counting. The carrier clock is maintained inactive at this time.

<3> When the count value of 8-bit timer counter Hn and the set value of the CMPn0 register match, the first INTTMHn signal is generated, the carrier clock signal is inverted, and the register that is compared with 8-bit timer counter Hn changes from the CMPn0 register to the CMPn1 register. 8-bit timer counter Hn is cleared to 00H.

<4> When the count value of 8-bit timer counter Hn and the set value of the CMPn1 register match, the INTTMHn signal is generated, the carrier clock signal is inverted, and the register that is compared with 8-bit timer counter Hn changes from the CMPn1 register to the CMPn0 register. 8-bit timer counter Hn is cleared to 00H. A carrier clock with a fixed duty (other than 50%) is generated through the repetition of steps <3> and <4>.

<5> The INTTM5n signal is synchronized with 8-bit timer Hn and output as the INTTM5Hn signal.

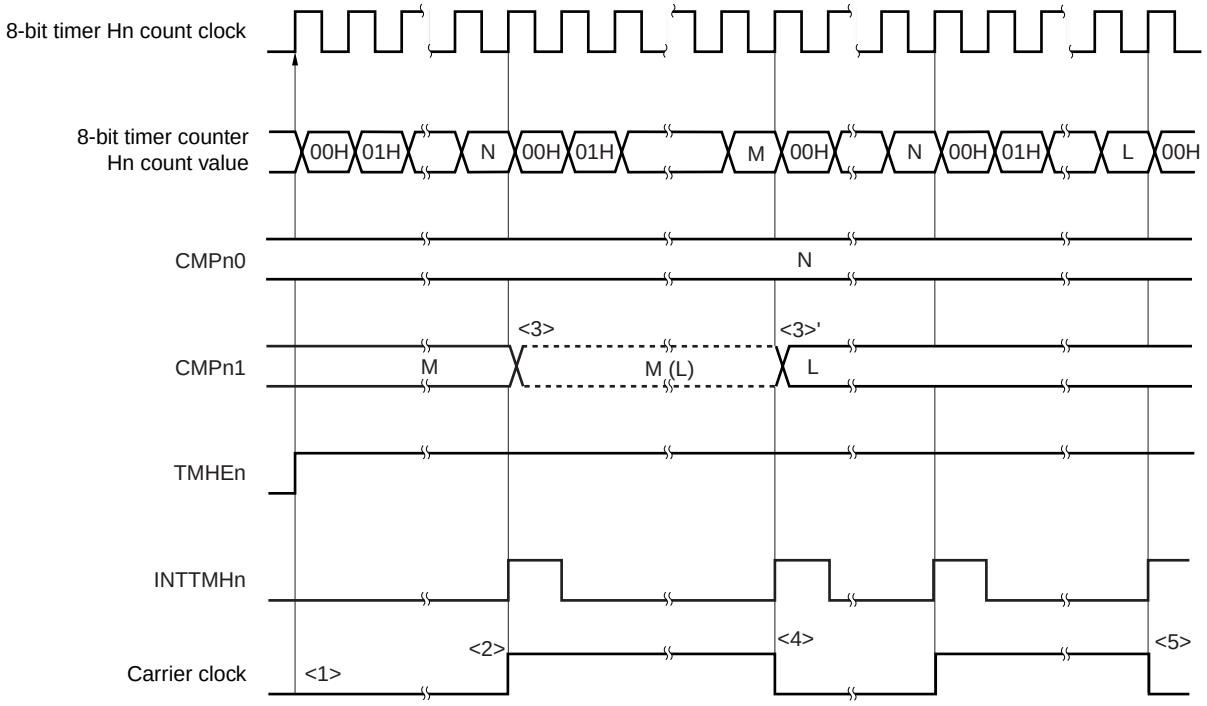
<6> The carrier is output from the rising edge of the first carrier clock by setting the NRZn bit to 1.

<7> By clearing the NRZn bit to 0, the TOHn output is also maintained high level while the carrier clock is high level, and does not change to low level (the high level width of the carrier waveform is guaranteed through steps <6> and <7>).

**Remark** n = 0, 1

**Figure 10-8. Carrier Generator Mode (3/3)**

Operation based on CMPn1 register transitions




<1> When the TMHEn bit is set to 1, counting starts. The carrier clock is maintained inactive at this time.

<2> When the count value of 8-bit timer counter Hn and the set value of the CMPn0 register match, 8-bit timer counter Hn is cleared to 00H and the INTTMHn signal is output.

<3> The CMPn1 register can be rewritten during 8-bit timer Hn operation, but the changed value (L) is latched. The value of the CMPn1 register is changed when the count value of 8-bit timer counter Hn and the value of the CMPn1 register prior to the change (M) match (<3>').

<4> When the count value of 8-bit timer counter Hn and the value (M) of the CMPn1 register match, the INTTMHn signal is output, the carrier signal is inverted, and 8-bit timer counter Hn is cleared to 00H.

<5> The timing at which the count value of 8-bit timer counter Hn and the set value of the CMPn1 register match again is the changed value (L).

**Remark** n = 0, 1

# CHAPTER 11 INTERVAL TIMER, WATCH TIMER

The V850ES/KF1+ includes interval timer BRG and a watch timer. Interval timer BRG can also be used as the source clock of the watch timer. The watch timer can also be used as interval timer WT.

Two interval timer channels and one watch timer channel can be used at the same time.

## 11.1 Interval Timer BRG

### 11.1.1 Functions

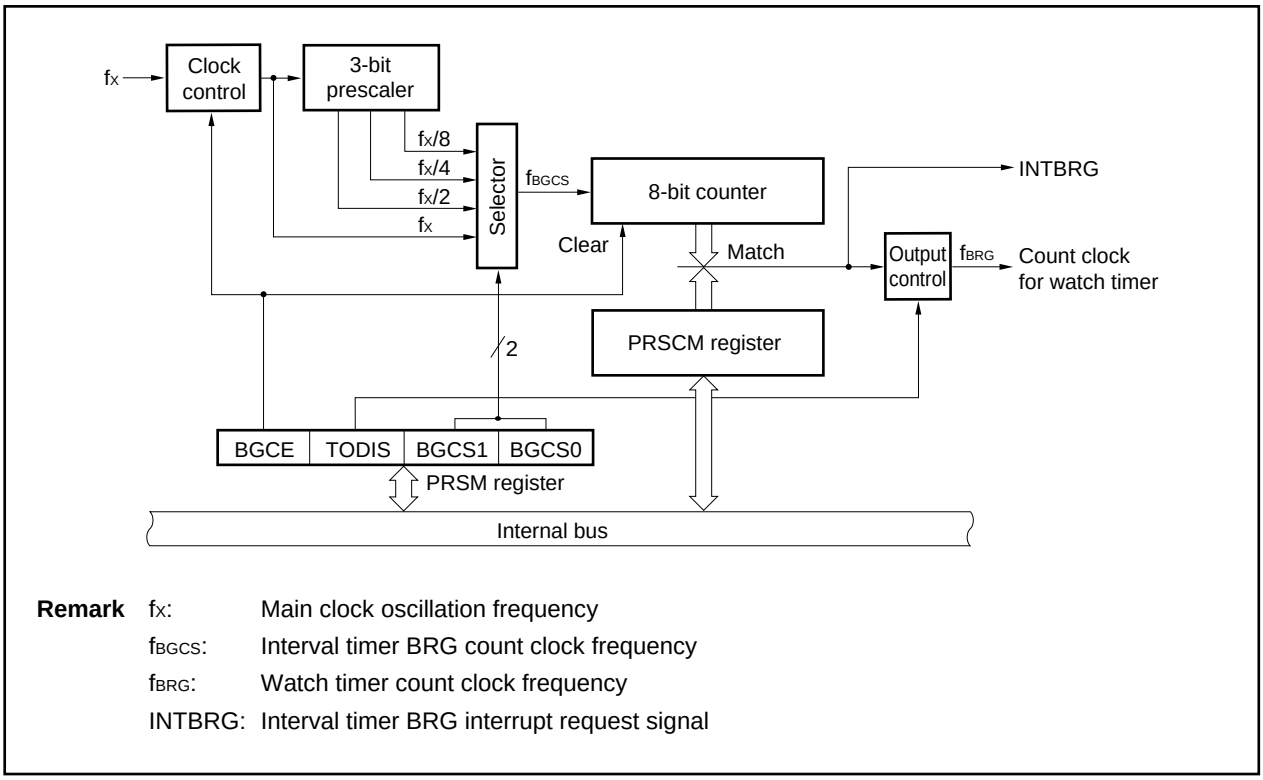
Interval timer BRG has the following functions.

- Interval timer BRG: An interrupt request signal (INTBRG) is generated at a specified interval.
- Generation of count clock for watch timer: When the main clock is used as the count clock for the watch timer, a count clock ($f_{BRG}$) is generated.

### 11.1.2 Configuration

The following shows the block diagram of interval timer BRG.

**Figure 11-1. Block Diagram of Interval Timer BRG**




**Remark** $f_X$: Main clock oscillation frequency
$f_{BGCS}$: Interval timer BRG count clock frequency
$f_{BRG}$: Watch timer count clock frequency
INTBRG: Interval timer BRG interrupt request signal

**(1) Clock control**

The clock control controls supply/stop of the operation clock ($f_X$) of interval timer BRG.

**(2) 3-bit prescaler**

The 3-bit prescaler divides $f_X$ to generate $f_X/2$, $f_X/4$, and $f_X/8$.

**(3) Selector**

The selector selects the count clock ($f_{BGCS}$) for interval timer BRG from $f_X$, $f_X/2$, $f_X/4$, and $f_X/8$.

**(4) 8-bit counter**

The 8-bit counter counts the count clock ($f_{BGCS}$).

**(5) Output control**

The output control controls supply of the count clock ($f_{BRG}$) for the watch timer.

**(6) PRSCM register**

The PRSCM register is an 8-bit compare register that sets the interval time.

**(7) PRSM register**

The PRSM register controls the operation of interval timer BRG, the selector, and clock supply to the watch timer.

### 11.1.3 Registers

Interval timer BRG includes the following registers.

#### (1) Interval timer BRG mode register (PRSM)

PRSM controls the operation of interval timer BRG, selection of count clock, and clock supply to the watch timer.

This register can be read or written in 8-bit or 1-bit units.

After reset, PRSM is cleared to 00H.

After reset: 00H R/W Address: FFFFF8B0H

| | 7 | 6 | 5 | <4> | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PRSM | 0 | 0 | 0 | BGCE | 0 | TODIS | BGCS1 | BGCS0 |

| BGCE | Control of interval timer operation |
|---|---|
| 0 | Operation stopped, 8-bit counter cleared to 01H |
| 1 | Operate |

| TODIS | Control of clock supply for watch timer |
|---|---|
| 0 | Clock for watch timer supplied |
| 1 | Clock for watch timer not supplied |

| BGCS1 | BGCS0 | Selection of input clock ($f_{BGCS}$)[Note] | | | |
|---|---|---|---|---|---|
| | | | 10 MHz | 5 MHz | 4 MHz |
| 0 | 0 | $f_X$ | 100 ns | 200 ns | 250 ns |
| 0 | 1 | $f_X/2$ | 200 ns | 400 ns | 500 ns |
| 1 | 0 | $f_X/4$ | 400 ns | 800 ns | 1 $\mu$s |
| 1 | 1 | $f_X/8$ | 800 ns | 1.6 $\mu$s | 2 $\mu$s |

**Note** Set these bits so that the following conditions are satisfied.

$V_{DD}$ = 4.0 to 5.5 V: $f_{BGCS} \leq$ 10 MHz

$V_{DD}$ = 2.7 to 4.0 V: $f_{BGCS} \leq$ 5 MHz

**Cautions 1. Do not change the values of the TODIS, BGCS1, and BGCS0 bits while interval timer BRG is operating (BGCE bit = 1). Set the TODIS, BGCS1, and BGCS0 bits before setting (1) the BGCE bit.**

**2. When the BGCE bit is cleared (to 0), the 8-bit counter is cleared.**

**(2) Interval timer BRG compare register (PRSCM)**

PRSCM is an 8-bit compare register.

This register can be read or written in 8-bit units.

After reset, PRSCM is cleared to 00H.

After reset: 00H R/W Address: FFFFF8B1H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PRSCM | PRSCM7 | PRSCM6 | PRSCM5 | PRSCM4 | PRSCM3 | PRSCM2 | PRSCM1 | PRSCM0 |

**Caution Do not rewrite the PRSCM register while interval timer BRG is operating (PRSM.BGCE bit = 1). Set the PRSCM register before setting (1) the BGCE bit.**

### 11.1.4 Operation

**(1) Operation of interval timer BRG**

Set the count clock by using the PRSM.BGCS1 and PRSM.BGCS0 bits and the 8-bit compare value by using the PRSCM register.

When the PRSM.BGCE bit is set (1), interval timer BRG starts operating.

Each time the count value of the 8-bit counter and the set value in the PRSCM register match, an interrupt request signal (INTBRG) is generated. At the same time, the 8-bit counter is cleared to 00H and counting is continued.

The interval time can be obtained from the following equation.

Interval time = $2^m \times N/f_X$

**Remark** m: Divided value (set value in the BGCS1 and BGCS0 bits) = 0 to 3
N: Set value in PRSCM register = 1 to 256 (when the set value in the PRSCM register is 00H, N = 256)
$f_X$: Main clock oscillation frequency

**(2) Count clock supply for watch timer**

Set the count clock by using the PRSM.BGCS1 and PRSM.BGCS0 bits and the 8-bit compare value by using the PRSCM register, so that the count clock frequency ($f_{BRG}$) of the watch timer is 32.768 kHz. Clear (0) the PRSM.TODIS bit at the same time.

When the PRSM.BGCE bit is set (1), $f_{BRG}$ is supplied to the watch timer.

$f_{BRG}$ is obtained from the following equation.

$f_{BRG} = f_X/(2^{m+1} \times N)$

To set $f_{BRG}$ to 32.768 kHz, perform the following calculation to set the BGCS1 and BGCS0 bits and the PRSCM register.

<1> Set N = $f_X$/65,536 (round off the decimal) to set m = 0.
<2> If N is even, N = N/2 and m = m + 1
<3> Repeat step <2> until N is odd or m = 3
<4> Set N to the PRSCM register and m to the BGCS1 and BGCS0 bits.

Example: When $f_X$ = 4.00 MHz

<1> N = 4,000,000/65,536 = 61 (round off the decimal), m = 0
<2>, <3> Since N is odd, the values remain as N = 61, m = 0
<4> The set value in the PRSCM register: 3DH (61), the set values in the BGCS1 and BGCS0 bits: 00

**Remark** m: Divided value (set value in the BGCS1 and BGCS0 bits) = 0 to 3
N: Set value in PRSCM register = 1 to 256 (when the set value in the PRSCM register is 00H, N = 256)
$f_X$: Main clock oscillation frequency

## 11.2 Watch Timer

### 11.2.1 Functions

The watch timer has the following functions.

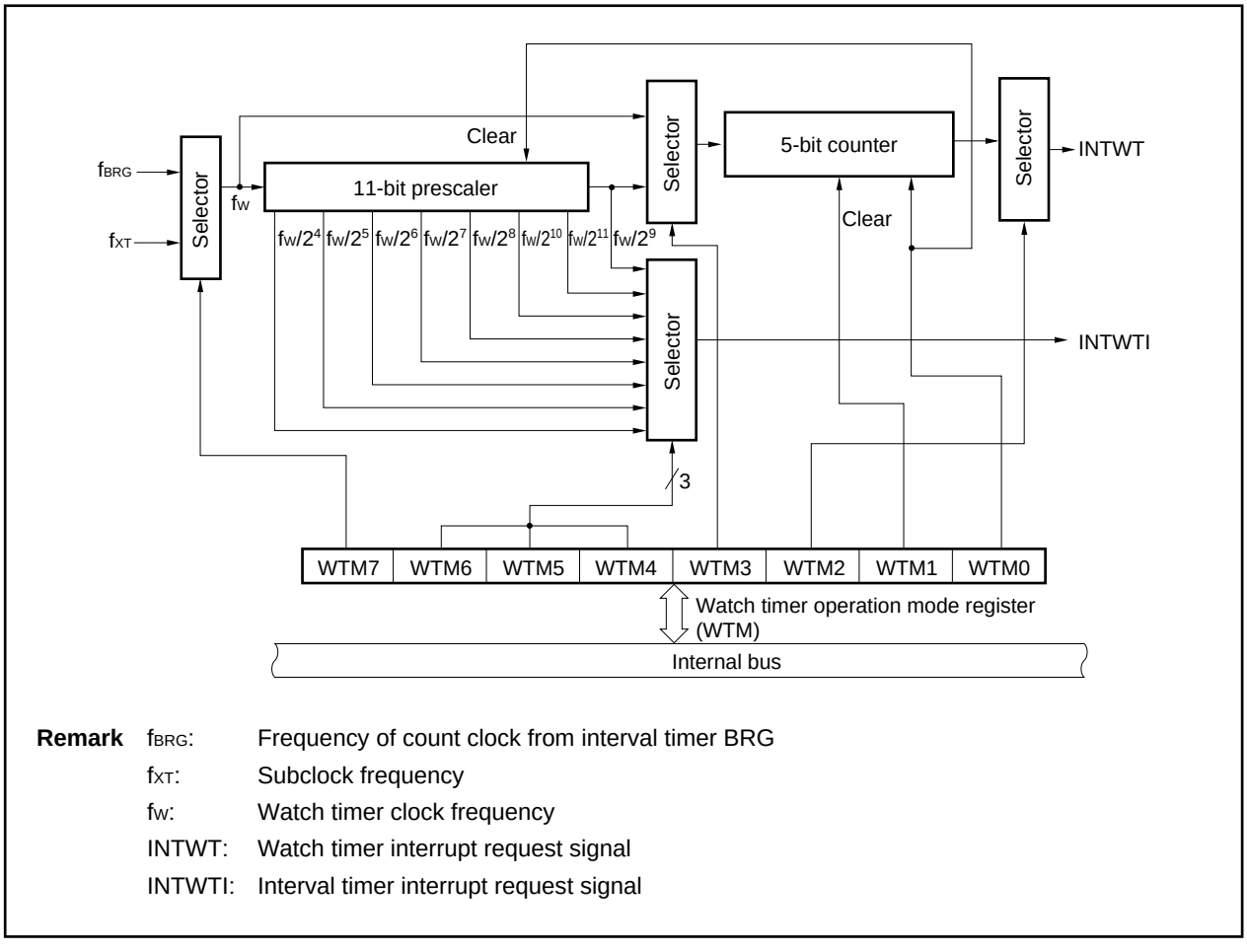
- Watch timer: An interrupt request signal (INTWT) is generated at time intervals of 0.5 or 0.25 seconds by using the main clock or subclock.
- Interval timer: An interrupt request signal (INTWTI) is generated at the preset time interval.

The watch timer and interval timer functions can be used at the same time.

### 11.2.2 Configuration

The following shows the block diagram of the watch timer.

**Figure 11-2. Block Diagram of Watch Timer**




**Remark** $f_{BRG}$: Frequency of count clock from interval timer BRG
$f_{XT}$: Subclock frequency
$f_W$: Watch timer clock frequency
INTWT: Watch timer interrupt request signal
INTWTI: Interval timer interrupt request signal

**(1) 11-bit prescaler**

The 11-bit prescaler generates a clock of $f_W/2^4$ to $f_W/2^{11}$ by dividing $f_W$.

**(2) 5-bit counter**

The 5-bit counter generates the watch timer interrupt request signal (INTWT) at intervals of $2^4/f_W$, $2^5/f_W$, $2^{13}/f_W$, or $2^{14}/f_W$ by counting $f_W$ or $f_W/2^9$.

**(3) Selectors**

The watch timer has the following four selectors.

- Selector that selects the main clock (the clock from interval timer BRG ($f_{BRG}$) or the subclock ($f_{XT}$)) as the clock for the watch timer.
- Selector that selects $f_W$ or $f_W/2^9$ as the count clock frequency of the 5-bit counter
- Selector that selects $2^4/f_W$ or $2^{13}/f_W$, or $2^5/f_W$ or $2^{14}/f_W$ as the INTWT signal generation time interval.
- Selector that selects the generation time interval of the interval timer WT interrupt request signal (INTWTI) from $2^4/f_W$ to $2^{11}/f_W$.

**(4) 8-bit counter**

The 8-bit counter counts the count clock ($f_{BGCS}$).

**(5) WTM register**

The WTM register is an 8-bit register that controls the operation of the watch timer/interval timer WT and sets the interval of interrupt request signal generation.

### 11.2.3 Register

The watch timer includes the following register.

**(1) Watch timer operation mode register (WTM)**

This register enables or disables the count clock and operation of the watch timer, sets the interval time of the 11-bit prescaler, controls the operation of the 5-bit counter, and sets the time of watch timer interrupt request signal (INTWT) generation.

The WTM register can be read or written in 8-bit or 1-bit units.

After reset, WTM is cleared to 00H.

After reset: 00H R/W Address: FFFFF680H

| | 7 | 6 | 5 | 4 | 3 | 2 | <1> | <0> |
|---|---|---|---|---|---|---|---|---|
| WTM | WTM7 | WTM6 | WTM5 | WTM4 | WTM3 | WTM2 | WTM1 | WTM0 |

| WTM7 | WTM6 | WTM5 | WTM4 | Selection of interval time of prescaler |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | $2^4/f_W$ (488 $\mu$s: $f_W = f_{XT}$) |
| 0 | 0 | 0 | 1 | $2^5/f_W$ (977 $\mu$s: $f_W = f_{XT}$) |
| 0 | 0 | 1 | 0 | $2^6/f_W$ (1.95 ms: $f_W = f_{XT}$) |
| 0 | 0 | 1 | 1 | $2^7/f_W$ (3.91 ms: $f_W = f_{XT}$) |
| 0 | 1 | 0 | 0 | $2^8/f_W$ (7.81 ms: $f_W = f_{XT}$) |
| 0 | 1 | 0 | 1 | $2^9/f_W$ (15.6 ms: $f_W = f_{XT}$) |
| 0 | 1 | 1 | 0 | $2^{10}/f_W$ (31.3 ms: $f_W = f_{XT}$) |
| 0 | 1 | 1 | 1 | $2^{11}/f_W$ (62.5 ms: $f_W = f_{XT}$) |
| 1 | 0 | 0 | 0 | $2^4/f_W$ (488 $\mu$s: $f_W = f_{BRG}$) |
| 1 | 0 | 0 | 1 | $2^5/f_W$ (977 $\mu$s: $f_W = f_{BRG}$) |
| 1 | 0 | 1 | 0 | $2^6/f_W$ (1.95 ms: $f_W = f_{BRG}$) |
| 1 | 0 | 1 | 1 | $2^7/f_W$ (3.91 ms: $f_W = f_{BRG}$) |
| 1 | 1 | 0 | 0 | $2^8/f_W$ (7.81 ms: $f_W = f_{BRG}$) |
| 1 | 1 | 0 | 1 | $2^9/f_W$ (15.6 ms: $f_W = f_{BRG}$) |
| 1 | 1 | 1 | 0 | $2^{10}/f_W$ (31.3 ms: $f_W = f_{BRG}$) |
| 1 | 1 | 1 | 1 | $2^{11}/f_W$ (62.5 ms: $f_W = f_{BRG}$) |

| WTM7 | WTM3 | WTM2 | Selection of set time of watch flag |
|---|---|---|---|
| 0 | 0 | 0 | $2^{14}/f_W$ (0.5 s: $f_W = f_{XT}$) |
| 0 | 0 | 1 | $2^{13}/f_W$ (0.25 s: $f_W = f_{XT}$) |
| 0 | 1 | 0 | $2^5/f_W$ (977 $\mu$s: $f_W = f_{XT}$) |
| 0 | 1 | 1 | $2^4/f_W$ (488 $\mu$s: $f_W = f_{XT}$) |
| 1 | 0 | 0 | $2^{14}/f_W$ (0.5 s: $f_W = f_{BRG}$) |
| 1 | 0 | 1 | $2^{13}/f_W$ (0.25 s: $f_W = f_{BRG}$) |
| 1 | 1 | 0 | $2^5/f_W$ (977 $\mu$s: $f_W = f_{BRG}$) |
| 1 | 1 | 1 | $2^4/f_W$ (488 $\mu$s: $f_W = f_{BRG}$) |

| WTM1 | Control of 5-bit counter operation |
|---|---|
| 0 | Clear after operation stops |
| 1 | Start |

| WTM0 | Watch timer operation enable |
|---|---|
| 0 | Stop operation (clear both prescaler and 5-bit counter) |
| 1 | Enable operation |

**Caution Rewrite the WTM2 to WTM7 bits while both the WTM0 and WTM1 bits are 0.**

**Remarks 1.** $f_W$: Watch timer clock frequency

**2.** Values in parentheses apply when $f_W$ = 32.768 kHz

### 11.2.4 Operation

#### (1) Operation as watch timer

The watch timer generates an interrupt request at fixed time intervals.

The watch timer operates using time intervals of 0.25 or 0.5 seconds with the subclock (32.768 kHz).

The count operation starts when the WTM.WTM0 and WTM.WTM1 bits are set to 11. When these bits are cleared to 00, the 11-bit prescaler and 5-bit counter are cleared and the count operation stops.

The 5-bit counter can be cleared to synchronize the time by clearing the WTM1 bit to 0 when the watch timer and interval timer WT operate simultaneously. At this time, an error of up to 15.6 ms may occur in the watch timer, but interval timer WT is not affected.

#### (2) Operation as interval timer

The watch timer can also be used as an interval timer that repeatedly generates an interrupt request signal (INTWTI) at intervals specified by a count value set in advance.

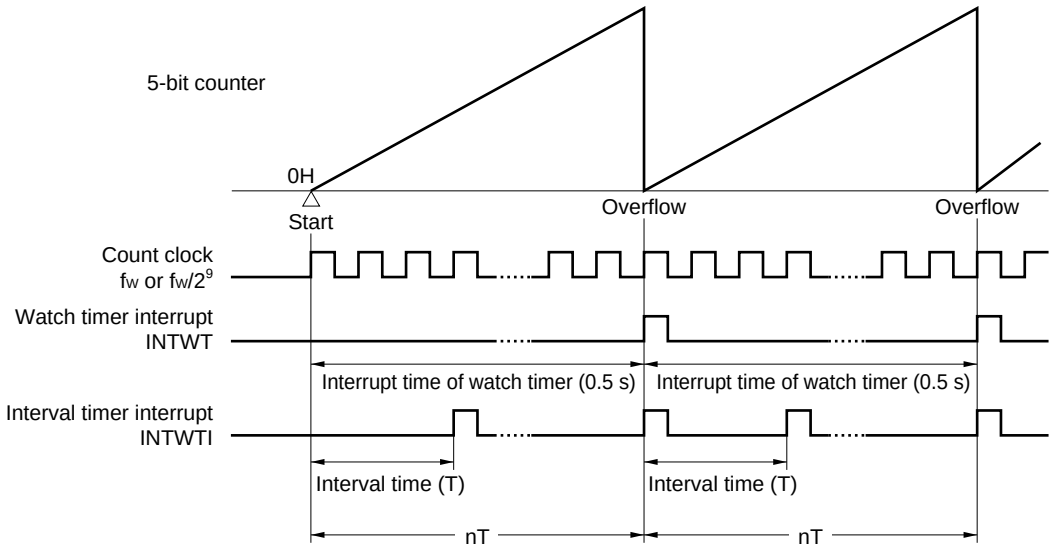
The interval time can be selected by the WTM.WTM4 to WTM.WTM7 bits.

**Table 11-1. Interval Time of Interval Timer**

| WTM7 | WTM6 | WTM5 | WTM4 | Interval Time | |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | $2^4 \times 1/f_W$ | 488 $\mu$s (operating at $f_W = f_{XT}$ = 32.768 kHz) |
| 0 | 0 | 0 | 1 | $2^5 \times 1/f_W$ | 977 $\mu$s (operating at $f_W = f_{XT}$ = 32.768 kHz) |
| 0 | 0 | 1 | 0 | $2^6 \times 1/f_W$ | 1.95 ms (operating at $f_W = f_{XT}$ = 32.768 kHz) |
| 0 | 0 | 1 | 1 | $2^7 \times 1/f_W$ | 3.91 ms (operating at $f_W = f_{XT}$ = 32.768 kHz) |
| 0 | 1 | 0 | 0 | $2^8 \times 1/f_W$ | 7.81 ms (operating at $f_W = f_{XT}$ = 32.768 kHz) |
| 0 | 1 | 0 | 1 | $2^9 \times 1/f_W$ | 15.6 ms (operating at $f_W = f_{XT}$ = 32.768 kHz) |
| 0 | 1 | 1 | 0 | $2^{10} \times 1/f_W$ | 31.3 ms (operating at $f_W = f_{XT}$ = 32.768 kHz) |
| 0 | 1 | 1 | 1 | $2^{11} \times 1/f_W$ | 62.5 ms (operating at $f_W = f_{XT}$ = 32.768 kHz) |
| 1 | 0 | 0 | 0 | $2^4 \times 1/f_W$ | 488 $\mu$s (operating at $f_W = f_{BRG}$ = 32.768 kHz) |
| 1 | 0 | 0 | 1 | $2^5 \times 1/f_W$ | 977 $\mu$s (operating at $f_W = f_{BRG}$ = 32.768 kHz) |
| 1 | 0 | 1 | 0 | $2^6 \times 1/f_W$ | 1.95 ms (operating at $f_W = f_{BRG}$ = 32.768 kHz) |
| 1 | 0 | 1 | 1 | $2^7 \times 1/f_W$ | 3.91 ms (operating at $f_W = f_{BRG}$ = 32.768 kHz) |
| 1 | 1 | 0 | 0 | $2^8 \times 1/f_W$ | 7.81 ms (operating at $f_W = f_{BRG}$ = 32.768 kHz) |
| 1 | 1 | 0 | 1 | $2^9 \times 1/f_W$ | 15.6 ms (operating at $f_W = f_{BRG}$ = 32.768 kHz) |
| 1 | 1 | 1 | 0 | $2^{10} \times 1/f_W$ | 31.3 ms (operating at $f_W = f_{BRG}$ = 32.768 kHz) |
| 1 | 1 | 1 | 1 | $2^{11} \times 1/f_W$ | 62.5 ms (operating at $f_W = f_{BRG}$ = 32.768 kHz) |

**Remark** $f_W$: Watch timer clock frequency

**Figure 11-3. Operation Timing of Watch Timer/Interval Timer**




**Remarks 1.** Assuming that the interrupt time of the watch timer is set to 0.5 seconds.

**2.** fw: Watch timer clock frequency

Values in parentheses apply when count clock fw = 32.768 kHz.

n: Number of interval timer WT operations

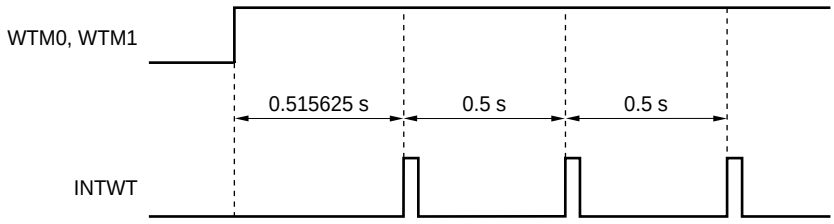
## 11.3 Cautions

**(1) Operation as watch timer**

Some time is required before the first watch timer interrupt request signal (INTWT) is generated after operation is enabled (WTM.WTM1 and WTM.WTM0 bits = 11).

**Figure 11-4. Example of Generation of Watch Timer Interrupt Request Signal (INTWT) (When Interrupt Period = 0.5 s)**

It takes 0.515625 (max.) seconds for the first INTWT signal to be generated ($2^9 \times 1/32768 = 0.015625$ (max.) seconds longer). An INTWT signal is then generated every 0.5 seconds.



**(2) When watch timer and interval timer BRG operate simultaneously**

When using the subclock as the count clock for the watch timer, the interval time of interval timer BRG can be set to any value. Changing the interval time does not affect the watch timer (before changing the interval time, stop operation).

When using the main clock as the count clock for the watch timer, set the interval time of interval timer BRG to approximately 65,536 Hz. Do not change this value.

**(3) When interval timer BRG and interval timer WT operate simultaneously**

When using the subclock as the count clock for interval timer WT, the interval times of interval timers BRG and WT can be set to any values. They can also be changed later (before changing the value, stop operation).

When using the main clock as the count clock for interval timer WT, the interval time of interval timer BRG can be set to any value, but cannot be changed later (it can be changed only when interval timer WT stops operation). The interval time of interval timer WT can be set to $\times 2^5$ to $\times 2^{12}$ of the set value of interval timer BRG. It can also be changed later.

**(4) When watch timer and interval timer WT operate simultaneously**

The interval time of interval timer WT can be set to a value between 488 $\mu$s and 62.5 ms. It cannot be changed later.

Do not stop interval timer WT (clear (0) the WTM.WTM0 bit) while the watch timer is operating. If the WTM0 bit is set (1) after it had been cleared (0), the watch timer will have a discrepancy of up to 0.5 or 0.25 seconds.

**(5) When watch timer, interval timer BRG, and interval timer WT operate simultaneously**

When using the subclock as the count clock for the watch timer, the interval times of interval timers BRG and WT can be set to any values. The interval time of interval timer BRG can be changed later (before changing the value, stop operation).

When using the main clock as the count clock for the watch timer, set the interval time of interval timer BRG to approximately 65,536 kHz. It cannot be changed later. The interval time of interval timer WT can be set to a value between 488 $\mu$s and 62.5 ms. It cannot be changed later.

Do not stop interval timer BRG (clear (0) the PRSM.BGCE bit) or interval timer WT (clear (0) the WTM.WTM0 bit) while the watch timer is operating.

# CHAPTER 12 WATCHDOG TIMER FUNCTIONS

## 12.1 Watchdog Timer 1

### 12.1.1 Functions

Watchdog timer 1 has the following operation modes.

- Watchdog timer
- Interval timer

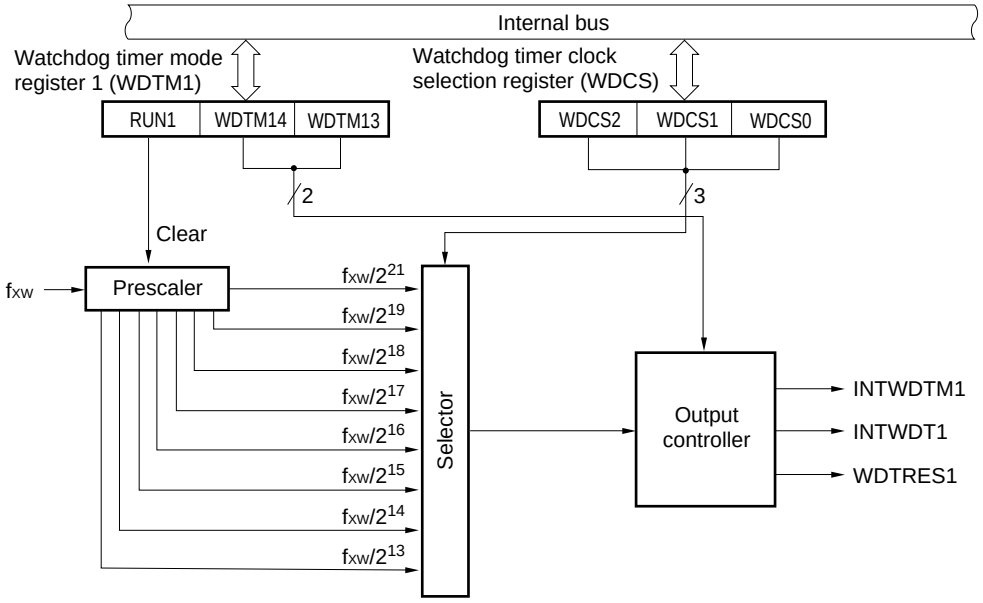
The following functions are realized from the above-listed operation modes.

- Generation of non-maskable interrupt request signal (INTWDT1) upon overflow of watchdog timer 1[Note]
- Generation of system reset signal (WDTRES1) upon overflow of watchdog timer 1
- Generation of maskable interrupt request signal (INTWDTM1) upon overflow of interval timer

**Note** For non-maskable interrupt servicing due to non-maskable interrupt request signal (INTWDT1, INTWDT2), refer to **19.10 Cautions**.

**Remark** Select whether to use watchdog timer 1 in the watchdog timer 1 mode or the interval timer mode with the WDTM1 register.

**Figure 12-1. Block Diagram of Watchdog Timer 1**




**Remark** INTWDTM1: Request signal for maskable interrupt through watchdog timer 1 overflow
INTWDT1: Request signal for non-maskable interrupt through watchdog timer 1 overflow
WDTRES1: Reset signal through watchdog timer 1 overflow
$f_{XW} = f_X$: Watchdog timer 1 clock frequency

### 12.1.2 Configuration

Watchdog timer 1 consists of the following hardware.

**Table 12-1. Configuration of Watchdog Timer 1**

| Item | Configuration |
|---|---|
| Control registers | Watchdog timer clock selection register (WDCS)<br>Watchdog timer mode register 1 (WDTM1) |

### 12.1.3 Registers

The registers that control watchdog timer 1 are as follows.

- Watchdog timer clock selection register (WDCS)
- Watchdog timer mode register 1 (WDTM1)

#### (1) Watchdog timer clock selection register (WDCS)

This register sets the overflow time of watchdog timer 1 and the interval timer.

The WDCS register can be read or written in 8-bit or 1-bit units.

After reset, WDCS is cleared to 00H.

After reset: 00H R/W Address: FFFFF6C1H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| WDCS | 0 | 0 | 0 | 0 | 0 | WDCS2 | WDCS1 | WDCS0 |

| WDCS2 | WDCS1 | WDCS0 | Overflow time of watchdog timer 1/interval timer | | | |
|---|---|---|---|---|---|---|
| | | | | $f_{XW}$ | | |
| | | | | 4 MHz | 5 MHz | 10 MHz |
| 0 | 0 | 0 | $2^{13}/f_{XW}$ | 2.048 ms | 1.638 ms | 0.819 ms |
| 0 | 0 | 1 | $2^{14}/f_{XW}$ | 4.096 ms | 3.277 ms | 1.638 ms |
| 0 | 1 | 0 | $2^{15}/f_{XW}$ | 8.192 ms | 6.554 ms | 3.277 ms |
| 0 | 1 | 1 | $2^{16}/f_{XW}$ | 16.38 ms | 13.11 ms | 6.554 ms |
| 1 | 0 | 0 | $2^{17}/f_{XW}$ | 32.77 ms | 26.21 ms | 13.11 ms |
| 1 | 0 | 1 | $2^{18}/f_{XW}$ | 65.54 ms | 52.43 ms | 26.2 ms |
| 1 | 1 | 0 | $2^{19}/f_{XW}$ | 131.1 ms | 104.9 ms | 52.43 ms |
| 1 | 1 | 1 | $2^{21}/f_{XW}$ | 524.3 ms | 419.4 ms | 209.7 ms |

**Remark** $f_{XW} = f_X$: Watchdog timer 1 clock frequency

**(2) Watchdog timer mode register 1 (WDTM1)**

This register sets the watchdog timer 1 operation mode and enables/disables count operations.

This register is a special register that can be written only in a special sequence (refer to **3.4.7 Special registers**).

The WDTM1 register can be read or written in 8-bit or 1-bit units.

After reset, WDTM1 is cleared to 00H.

**Caution When the main clock is stopped and the CPU is operating on the subclock, do not access the WDTM1 register using an access method that causes a wait.**
**For details, refer to 3.4.8 (2).**

After reset: 00H R/W Address: FFFFF6C2H

| | <7> | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| WDTM1 | RUN1 | 0 | 0 | WDTM14 | WDTM13 | 0 | 0 | 0 |

| RUN1 | Selection of operation mode of watchdog timer 1[Note 1] |
|---|---|
| 0 | Stop counting |
| 1 | Clear counter and start counting |

| WDTM14 | WDTM13 | Selection of operation mode of watchdog timer 1[Note 2] |
|---|---|---|
| 0 | 0 | Interval timer mode<br>(Upon overflow, maskable interrupt INTWDTM1 is generated.) |
| 0 | 1 | |
| 1 | 0 | Watchdog timer mode 1[Note 3]<br>(Upon overflow, non-maskable interrupt INTWDT1 is generated.) |
| 1 | 1 | Watchdog timer mode 2<br>(Upon overflow, reset operation WDTRES1 is started.) |

**Notes 1.** Once the RUN1 bit is set (to 1), it cannot be cleared (to 0) by software.
Therefore, when counting is started, it cannot be stopped except by reset.

**2.** Once the WDTM13 and WDTM14 bits are set (to 1), they cannot be cleared (to 0) by software and can be cleared only by reset.

**3.** For non-maskable interrupt servicing due to non-maskable interrupt request signal (INTWDT1), refer to **19.10 Cautions**.

### 12.1.4 Operation

**(1) Operation as watchdog timer 1**

Watchdog timer 1 operation to detect a program loop is selected by setting the WDTM1.WDTM14 bit to 1.

The count clock (program loop detection time interval) of watchdog timer 1 can be selected using the WDCS.WDCS0 to WDCS.WDCS2 bits. The count operation is started by setting the WDTM1.RUN1 bit to 1. When, after the count operation is started, the RUN1 bit is again set to 1 within the set program loop detection time interval, watchdog timer 1 is cleared and the count operation starts again.

If the program loop detection time is exceeded without RUN1 bit being set to 1, a reset signal (WDTRES1) or a non-maskable interrupt request signal (INTWDT1) is generated depending on the value of the WDTM1.WDTM13 bit.

The count operation of watchdog timer 1 stops in the STOP mode and IDLE mode. Set the RUN1 bit to 1 before the STOP mode or IDLE mode is entered in order to clear watchdog timer 1.

Because watchdog timer 1 operates in the HALT mode, make sure that an overflow will not occur during HALT.

**Cautions 1. When the subclock is selected for the CPU clock, the count operation of watchdog timer 1 is stopped (the value of watchdog timer 1 is maintained).**

**2. For non-maskable interrupt servicing due to the INTWDT1 signal, refer to 19.10 Cautions.**

**Table 12-2. Program Loop Detection Time of Watchdog Timer 1**

| Clock | Program Loop Detection Time | | |
|---|---|---|---|
| | $f_{XW}$ = 4 MHz | $f_{XW}$ = 5 MHz | $f_{XW}$ = 10 MHz |
| $2^{13}/f_{XW}$ | 2.048 ms | 1.638 ms | 0.819 ms |
| $2^{14}/f_{XW}$ | 4.096 ms | 3.277 ms | 1.683 ms |
| $2^{15}/f_{XW}$ | 8.192 ms | 6.554 ms | 3.277 ms |
| $2^{16}/f_{XW}$ | 16.38 ms | 13.11 ms | 6.554 ms |
| $2^{17}/f_{XW}$ | 32.77 ms | 26.21 ms | 13.11 ms |
| $2^{18}/f_{XW}$ | 65.54 ms | 52.43 ms | 26.21 ms |
| $2^{19}/f_{XW}$ | 131.1 ms | 104.9 ms | 52.43 ms |
| $2^{21}/f_{XW}$ | 524.3 ms | 419.4 ms | 209.7 ms |

**Remark** $f_{XW} = f_X$: Watchdog timer 1 clock frequency

**(2) Operation as interval timer**

Watchdog timer 1 can be made to operate as an interval timer that repeatedly generates interrupts using the count value set in advance as the interval, by clearing the WDTM1.WDTM14 bit to 0.

When watchdog timer 1 operates as an interval timer, the interrupt mask flag (WDTMK) and priority specification flags (WDTPR0 to WDTPR2) of the WDTIC register are valid and maskable interrupt request signals (INTWDTM1) can be generated. The default priority of the INTWDTM1 signal is set to the highest level among the maskable interrupt request signals.

The interval timer continues to operate in the HALT mode, but it stops operating in the STOP mode and the IDLE mode.

**Cautions 1. Once the WDTM14 bit is set to 1 (thereby selecting the watchdog timer 1 mode), the interval timer mode is not entered as long as reset is not performed.**

**2. When the subclock is selected for the CPU clock, the count operation of the watchdog timer 1 stops (the value of the watchdog timer is maintained).**

**Table 12-3. Interval Time of Interval Timer**

| Clock | Interval Time | | |
|---|---|---|---|
| | $f_{XW}$ = 4 MHz | $f_{XW}$ = 5 MHz | $f_{XW}$ = 10 MHz |
| $2^{13}/f_{XW}$ | 2.048 ms | 1.638 ms | 0.819 ms |
| $2^{14}/f_{XW}$ | 4.096 ms | 3.277 ms | 1.638 ms |
| $2^{15}/f_{XW}$ | 8.192 ms | 6.554 ms | 3.277 ms |
| $2^{16}/f_{XW}$ | 16.38 ms | 13.11 ms | 6.554 ms |
| $2^{17}/f_{XW}$ | 32.77 ms | 26.21 ms | 13.11 ms |
| $2^{18}/f_{XW}$ | 65.54 ms | 52.43 ms | 26.21 ms |
| $2^{19}/f_{XW}$ | 131.1 ms | 104.9 ms | 52.43 ms |
| $2^{21}/f_{XW}$ | 524.3 ms | 419.4 ms | 209.7 ms |

**Remark** $f_{XW} = f_X$: Watchdog timer 1 clock frequency

## 12.2 Watchdog Timer 2

### 12.2.1 Functions

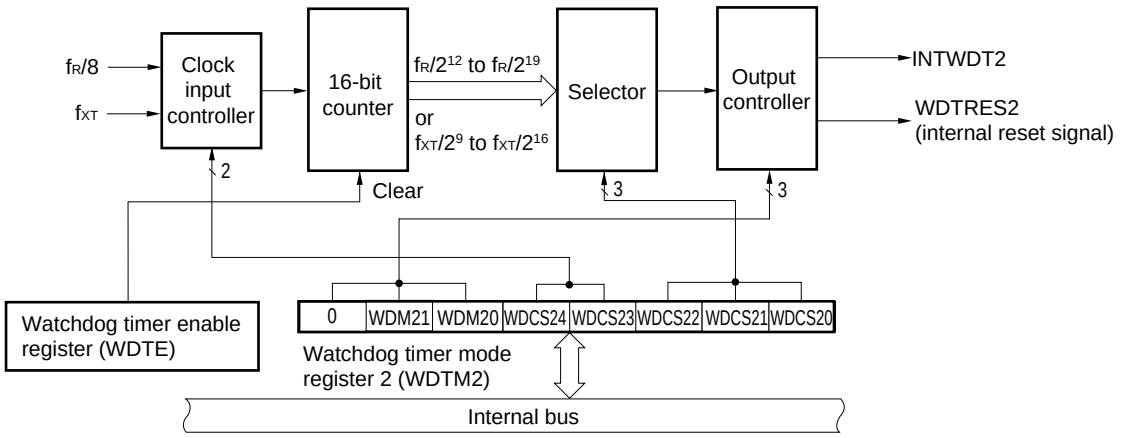
Watchdog timer 2 has the following functions.

- Default start watchdog timer[Note 1]
  → Reset mode: Reset operation upon overflow of watchdog timer 2 (generation of WDTRES2 signal)
  → Non-maskable interrupt request mode: NMI operation upon overflow of watchdog timer 2 (generation of INTWDT2 signal)[Note 2]
- Input selectable from Ring-OSC clock and subclock as the source clock

**Notes 1.** Watchdog timer 2 automatically starts in the reset mode following reset release.
When watchdog timer 2 is not used, either stop its operation before reset is executed through this function, or clear once watchdog timer 2 and stop it within the next interval time.
Also, write to the WDTM2 register for verification purposes only once, even if the default settings (reset mode, interval time: $f_{XX}/2^{25}$) need not be changed.

**2.** For non-maskable interrupt servicing due to a non-maskable interrupt request signal (INTWDT2), refer to **19.10 Cautions**.

**Figure 12-2. Block Diagram of Watchdog Timer 2**




**Remark** $f_R$: Ring-OSC clock frequency
$f_{XT}$: Subclock frequency
INTWDT2: Non-maskable interrupt request signal through watchdog timer 2
WDTRES2: Watchdog timer 2 reset signal

### 12.2.2 Configuration

Watchdog timer 2 consists of the following hardware.

**Table 12-4. Configuration of Watchdog Timer 2**

| Item | Configuration |
|---|---|
| Control registers | Watchdog timer mode register 2 (WDTM2)<br>Watchdog timer enable register (WDTE) |

### 12.2.3 Registers

**(1) Watchdog timer mode register 2 (WDTM2)**

This register sets the overflow time and operation clock of watchdog timer 2.

The WDTM2 register can be read or written in 8-bit units. This register can be read any number of times, but it can be written only once following reset release.

After reset, WDTM2 is set to 67H.

**Caution When the main clock is stopped and the CPU is operating on the subclock, do not access the WDTM2 register using an access method that causes a wait.**
**For details, refer to 3.4.8 (2).**

After reset: 67H R/W Address: FFFFF6D0H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| WDTM2 | 0 | WDM21 | WDM20 | WDCS24 | WDCS23 | WDCS22 | WDCS21 | WDCS20 |

| WDM21 | WDM20 | Selection of operation mode of watchdog timer 2 |
|---|---|---|
| 0 | 0 | Stops operation |
| 0 | 1 | Non-maskable interrupt request mode (generation of INTWDT2) |
| 1 | – | Reset mode (generation of WDTRES2) |

**Cautions 1. To stop the operation of watchdog timer 2, write “1FH” to the WDTM2 register.**
**2. For details about bits WDCS0 to WDCS4, refer to Table 12-5 Watchdog Timer 2 Clock Selection.**
**3. If the WDTM2 register is written twice after a reset, an overflow signal is forcibly output.**

**Table 12-5. Watchdog Timer 2 Clock Selection**

| WDCS24 | WDCS23 | WDCS22 | WDCS21 | WDCS20 | Selected Clock | Program Loop Detection Time |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | $2^{12}/f_R$ | 17.1 ms ($f_R$ = 240 kHz (TYP.)) |
| 0 | 0 | 0 | 0 | 1 | $2^{13}/f_R$ | 34.1 ms ($f_R$ = 240 kHz (TYP.)) |
| 0 | 0 | 0 | 1 | 0 | $2^{14}/f_R$ | 68.2 ms ($f_R$ = 240 kHz (TYP.)) |
| 0 | 0 | 0 | 1 | 1 | $2^{15}/f_R$ | 136.5 ms ($f_R$ = 240 kHz (TYP.)) |
| 0 | 0 | 1 | 0 | 0 | $2^{16}/f_R$ | 273.1 ms ($f_R$ = 240 kHz (TYP.)) |
| 0 | 0 | 1 | 0 | 1 | $2^{17}/f_R$ | 546.1 ms ($f_R$ = 240 kHz (TYP.)) |
| 0 | 0 | 1 | 1 | 0 | $2^{18}/f_R$ | 1092.3 ms ($f_R$ = 240 kHz (TYP.)) |
| 0 | 0 | 1 | 1 | 1 | $2^{19}/f_R$ | 2184.5 ms ($f_R$ = 240 kHz (TYP.)) |
| 0 | 1 | 0 | 0 | 0 | $2^{9}/f_{XT}$ | 15.625 ms ($f_{XT}$ = 32.768 kHz) |
| 0 | 1 | 0 | 0 | 1 | $2^{10}/f_{XT}$ | 31.25 ms ($f_{XT}$ = 32.768 kHz) |
| 0 | 1 | 0 | 1 | 0 | $2^{11}/f_{XT}$ | 62.5 ms ($f_{XT}$ = 32.768 kHz) |
| 0 | 1 | 0 | 1 | 1 | $2^{12}/f_{XT}$ | 125 ms ($f_{XT}$ = 32.768 kHz) |
| 0 | 1 | 1 | 0 | 0 | $2^{13}/f_{XT}$ | 250 ms ($f_{XT}$ = 32.768 kHz) |
| 0 | 1 | 1 | 0 | 1 | $2^{14}/f_{XT}$ | 500 ms ($f_{XT}$ = 32.768 kHz) |
| 0 | 1 | 1 | 1 | 0 | $2^{15}/f_{XT}$ | 1000 ms ($f_{XT}$ = 32.768 kHz) |
| 0 | 1 | 1 | 1 | 1 | $2^{16}/f_{XT}$ | 2000 ms ($f_{XT}$ = 32.768 kHz) |
| 1 | × | × | × | × | Operation stopped | |

**(2) Watchdog timer enable register (WDTE)**

The counter of watchdog timer 2 is cleared and counting restarted by writing “ACH” to the WDTE register.

The WDTE register can be read or written in 8-bit units.

After reset, WDTE is set to 9AH.

After reset: 9AH R/W Address: FFFFF6D1H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| WDTE | | | | | | | | |

**Cautions 1. When a value other than “ACH” is written to the WDTE register, an overflow signal is forcibly output.**

**2. When a 1-bit memory manipulation instruction is executed for the WDTE register, an overflow signal is forcibly output.**

**3. The read value of the WDTE register is always “9AH” (value that differs from written value “ACH”).**

### 12.2.4 Operation

Watchdog timer 2 automatically starts in the reset mode following reset release.

The WDTM2 register can be written to only once following reset through byte access. To use watchdog timer 2, write the operation mode and the interval time to the WDTM2 register using 8-bit memory manipulation instructions. After this is done, the operation of watchdog timer 2 cannot be stopped.

The watchdog timer 2 program loop detection time interval can be selected by the WDTM2.WDCS24 to WDTM2.WDCS20 bits. Writing ACH to the WDTE register clears the counter of watchdog timer 2 and starts the count operation again. After the count operation starts, write ACH to the WDTE register within the set program loop detection time interval.

If the program loop detection time is exceeded without ACH being written to the WDTE register, a reset signal (WDTRES2) or non-maskable interrupt request signal (INTWDT2) is generated depending on the set value of the WDTM2.WDM21 and WDTM2.WDM20 bits.

To not use watchdog timer 2, write 1FH to the WDTM2 register.

For non-maskable interrupt servicing when the non-maskable interrupt request mode is set, refer to **19.10 Cautions**.

Because watchdog timer 2 operates in the HALT/IDLE/STOP mode, exercise care that the timer does not overflow in the HALT/IDLE/STOP mode.

# CHAPTER 13 REAL-TIME OUTPUT FUNCTION (RTO)

## 13.1 Function

The real-time output function (RTO) transfers preset data to the RTBL0 and RTBH0 registers, and then transfers this data with hardware to an external device via the real-time output latches, upon occurrence of a timer interrupt. The pins through which the data is output to an external device constitute a port called a real-time output port.

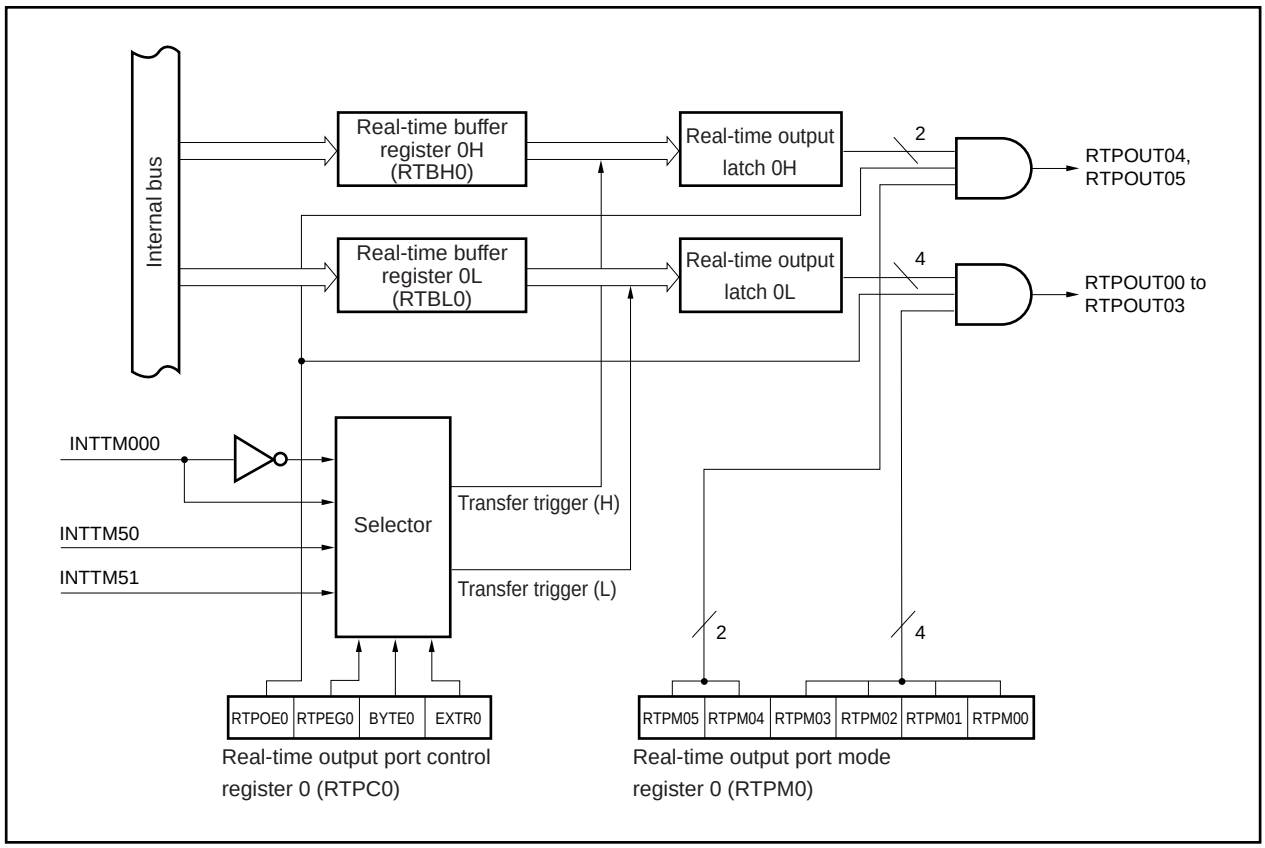
Because RTO can output signal without jitter, it is suitable for controlling a stepping motor.

In the V850ES/KF1+, a 6-bit real-time output port channel is provided.

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

The block diagram of RTO is shown below.

**Figure 13-1. Block Diagram of RTO**



## 13.2 Configuration

RTO consists of the following hardware.

**Table 13-1. Configuration of RTO**

| Item | Configuration |
|---|---|
| Registers | Real-time output buffer register 0 (RTBL0, RTBH0) |
| Control registers | Real-time output port mode register 0 (RTPM0)<br>Real-time output port control register 0 (RTPC0) |

**(1) Real-time output buffer register 0 (RTBL0, RTBH0)**

RTBL0 and RTBH0 are 4-bit registers that hold output data in advance.

These registers are mapped to independent addresses in the peripheral I/O register area.

They can be read or written in 8-bit or 1-bit units.

If an operation mode of 4 bits × 1 channel or 2 bits × 1 channel is specified (RTPC0.BYTE0 bit = 0), data can be individually set to the RTBL0 and RTBH0 registers. The data of both these registers can be read at once by specifying the address of either of these registers.

If an operation mode of 6 bits × 1 channel is specified (BYTE0 bit = 1), 8-bit data can be set to both the RTBL0 and RTBH0 registers by writing the data to either of these registers. Moreover, the data of both these registers can be read at once by specifying the address of either of these registers.

Table 13-2 shows the operation when the RTBL0 and RTBH0 registers are manipulated.

After reset: 00H R/W Address: RTBL0 FFFFF6E0H, RTBH0 FFFFF6E2H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RTBL0 | | | | | RTBL03 | RTBL02 | RTBL01 | RTBL00 |
| RTBH0 | 0 | 0 | RTBH05 | RTBH04 | | | | |

**Cautions 1. When writing to bits 6 and 7 of the RTBH0 register, always write 0.**

**2. When the main clock is stopped and the CPU is operating on the subclock, do not access the RTBL0 and RTBH0 registers using an access method that causes a wait. For details, refer to 3.4.8 (2).**

**Table 13-2. Operation During Manipulation of RTBL0 and RTBH0 Registers**

| Operation Mode | Register to Be Manipulated | Read | | Write[Note] | |
|---|---|---|---|---|---|
| | | Higher 4 Bits | Lower 4 Bits | Higher 4 Bits | Lower 4 Bits |
| 4 bits × 1 channel, 2 bits × 1 channel | RTBL0 | RTBH0 | RTBL0 | Invalid | RTBL0 |
| | RTBH0 | RTBH0 | RTBL0 | RTBH0 | Invalid |
| 6 bits × 1 channel | RTBL0 | RTBH0 | RTBL0 | RTBH0 | RTBL0 |
| | RTBH0 | RTBH0 | RTBL0 | RTBH0 | RTBL0 |

**Note** After setting the real-time output port, set output data to the RTBL0 and RTBH0 registers by the time a real-time output trigger is generated.

## 13.3 Registers

RTO is controlled using the following two types of registers.

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

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

This register selects the real-time output port mode or port mode in 1-bit units.

The RTPM0 register can be read or written in 8-bit or 1-bit units.

After reset, RTPM0 is cleared to 00H.

After reset: 00H R/W Address: FFFFF6E4H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RTPM0 | 0 | 0 | RTPM05 | RTPM04 | RTPM03 | RTPM02 | RTPM01 | RTPM00 |

| RTPM0m | Control of real-time output port (m = 0 to 5) |
|---|---|
| 0 | Real-time output disabled |
| 1 | Real-time output enabled |

**Cautions 1. To reflect real-time output signals (RTPOUT00 to RTPOUT05) to the pins (RTP00 to RTP05), set them to the real-time output port with the PMC5 and PFC5 registers.**

**2. By enabling real-time output operation (RTPC0.RTPOE0 bit = 1), the bits specified as real-time output enabled perform real-time output, and the bits specified as real-time output disabled output 0.**

**3. If real-time output is disabled (RTPOE0 bit = 0), real-time output signals (RTPOUT00 to RTPOUT05) all output 0, regardless of the RTPM0 register setting.**

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

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

The relationship between the operation mode and output trigger of the real-time output port is as shown in Table 13-3.

The RTPC0 register can be read or written in 8-bit or 1-bit units.

After reset, RTPC0 is cleared to 00H.

After reset: 00H R/W Address: FFFFF6E5H

| | <7> | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RTPC0 | RTPOE0 | RTPEG0 | BYTE0 | EXTR0[Note 1] | 0 | 0 | 0 | 0 |

| RTPOE0 | Control of real-time output operation |
|---|---|
| 0 | Disables operation[Note 2] |
| 1 | Enables operation |

| RTPEG0 | Valid edge of INTTM000 signal |
|---|---|
| 0 | Falling edge[Note 3] |
| 1 | Rising edge |

| BYTE0 | Specification of channel configuration for real-time output |
|---|---|
| 0 | 4 bits × 1 channel, 2 bits × 1 channel |
| 1 | 6 bits × 1 channel |

**Notes 1.** For the EXTR0 bit, refer to **Table 13-3**.

**2.** When real-time output operation is disabled (RTPOE0 bit = 0), real-time output signals (RTPOUT00 to RTPOUT05) all output 0.

**3.** The INTTM000 signal is output for 1 clock of the count clock selected with 16-bit timer/event counter 00.

**Caution Perform the settings for the RTPEG0, BYTE0, and EXTR0 bits only when the RTPOE0 bit = 0.**

**Table 13-3. Operation Modes and Output Triggers of Real-Time Output Port**

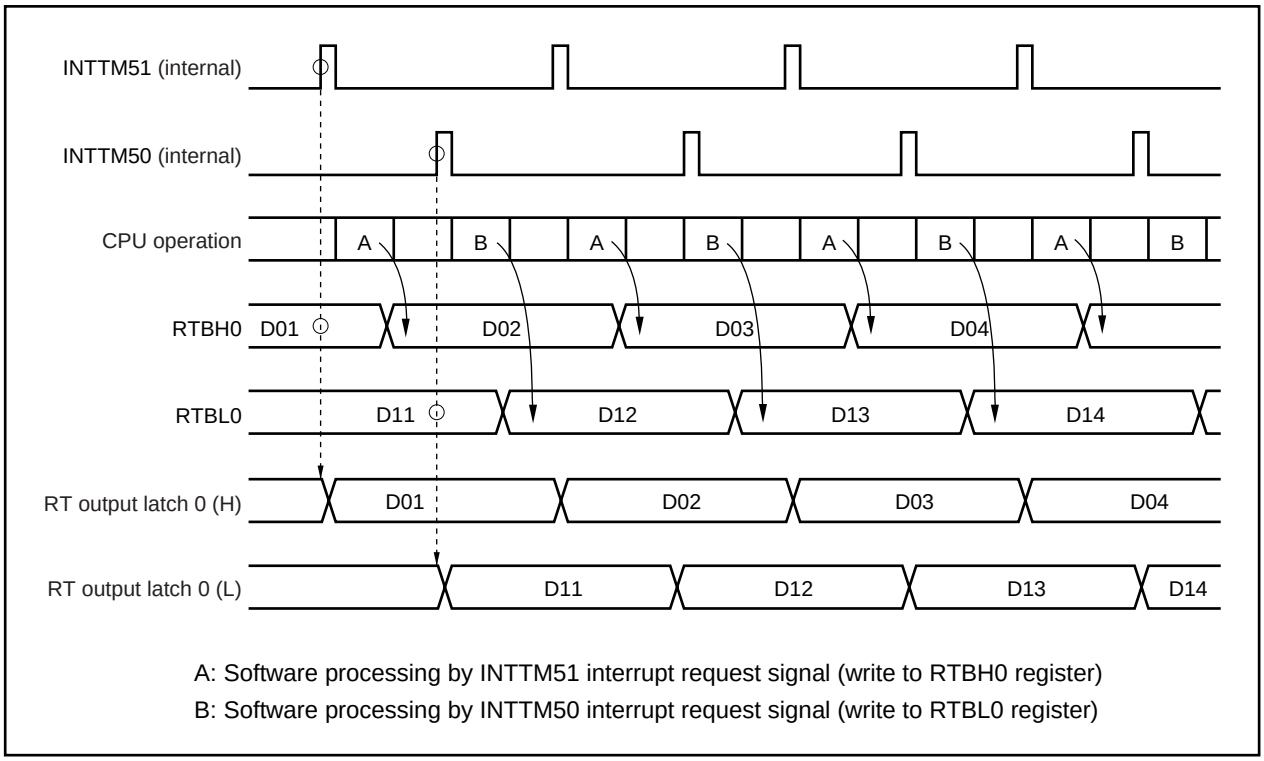
| BYTE0 | EXTR0 | Operation Mode | RTBH0 (RTP04, RTP05) | RTBL0 (RTP00 to RTP03) |
|---|---|---|---|---|
| 0 | 0 | 4 bits × 1 channel, 2 bits × 1 channel | INTTM51 | INTTM50 |
| | 1 | | INTTM50 | INTTM000 |
| 1 | 0 | 6 bits × 1 channel | INTTM50 | |
| | 1 | | INTTM000 | |

## 13.4 Operation

If the real-time output operation is enabled by setting the RTPC0.RTPOE0 bit to 1, the data of the RTBH0 and RTBL0 registers is transferred to the real-time output latch in synchronization with the generation of the selected transfer trigger (set by the RTPC0.EXTR0 and RTPC0.BYTE0 bits). Of the transferred data, only the data of the bits specified as real-time output enabled by the RTPM0 register is output from bits RTPOUT00 to RTPOUT05. The bits specified as real-time output disabled by the RTPM0 register output 0.

If the real-time output operation is disabled by clearing the RTPOE0 bit to 0, the RTPOUT00 to RTPOUT05 signals output 0 regardless of the setting of the RTPM0 register.

**Figure 13-2. Example of Operation Timing of RTO0 (When EXTR0 and BYTE0 Bits = 00)**




A: Software processing by INTTM51 interrupt request signal (write to RTBH0 register)
B: Software processing by INTTM50 interrupt request signal (write to RTBL0 register)

**Remark** For the operation during standby, refer to **CHAPTER 21 STANDBY FUNCTION**.

## 13.5 Usage

(1) Disable real-time output.
Clear the RTPC0.RTPOE0 bit to 0.

(2) Perform initialization as follows.
- Specify the real-time output port mode or port mode in 1-bit units.
  Set the RTPM0 register.
- Channel configuration: Select the trigger and valid edge.
  Set the RTPC0.EXTR0, RTPC0.BYTE0, and RTPC0.RTPEG0 bits.
- Set the initial values to the RTBH0 and RTBL0 registers[Note 1].

(3) Enable real-time output.
Set the RTPOE0 bit to 1.

(4) Set the next output value to the RTBH0 and RTBL0 registers by the time the selected transfer trigger is generated[Note 2].

(5) Set the next real-time output value to the RTBH0 and RTBL0 registers through interrupt servicing corresponding to the selected trigger.

**Notes 1.** If write to the RTBH0 and RTBL0 registers is performed when the RTPOE0 bit = 0, that value is transferred to real-time output latches 0H and 0L, respectively.

**2.** Even if write is performed to the RTBH0 and RTBL0 registers when the RTPOE0 bit = 1, data transfer to real-time output latches 0H and 0L is not performed.

**Caution To reflect the real-time output signals (RTPOUT00 to RTPOUT05) to the pins, set the real-time output ports (RTP00 to RTP05) with the PMC5 and PFC5 registers.**

## 13.6 Cautions

(1) Prevent the following conflicts by software.

- Conflict between real-time output disable/enable switching (RTPOE0 bit) and selected real-time output trigger
- Conflict between write to the RTBH0 and RTBL0 registers in the real-time output enabled status and the selected real-time output trigger.

(2) Before performing initialization, disable real-time output (RTPOE0 bit = 0).

(3) Once real-time output has been disabled (RTPOE0 bit = 0), be sure to initialize the RTBH0 and RTBL0 registers before enabling real-time output again (RTPOE0 bit = 0 → 1).

## 13.7 Security Function

A circuit that sets the pin outputs to high impedance as a security function for when malfunctions of a stepping motor controlled by RTO occur is provided on chip. It forcibly resets the pins allocated to RTP00 to RTP05 via external interrupt INTP0 pin edge detection, placing them in the high-impedance state.

The ports (P50 to P55 pins) placed in high impedance by INTP0[Note 1] pin are initialized[Note 2], so settings for these ports must be performed again.

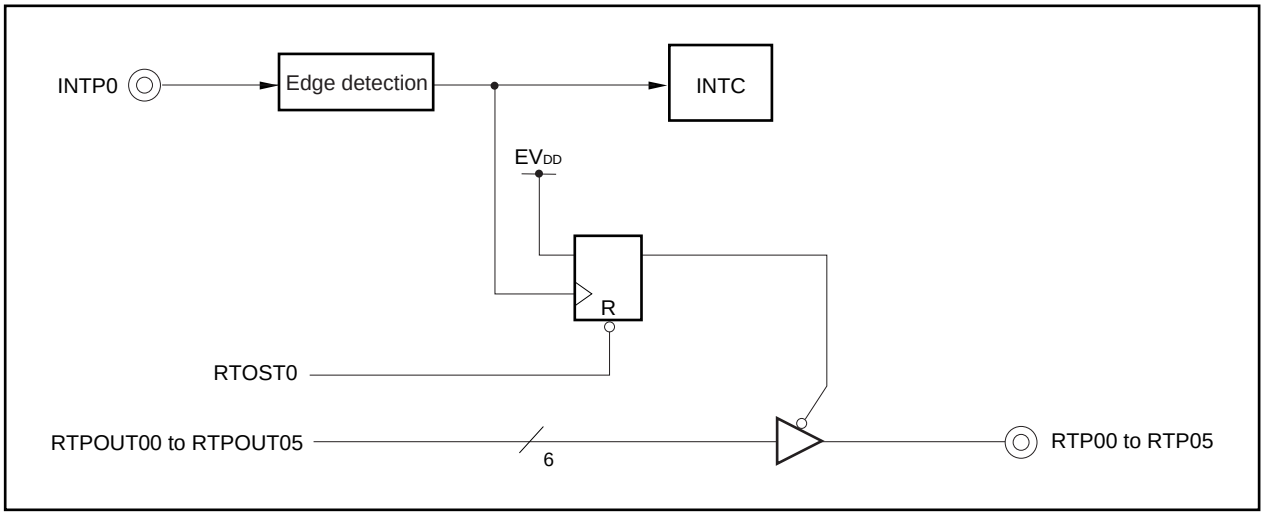
**Notes 1.** Regardless of the port settings, P50 to P55 pins are all placed in high impedance via the INTP0 pin.

**2.** The bits that are initialized are all the bits corresponding to P50 to P55 pins of the following registers.

- P5 register
- PM5 register
- PMC5 register
- PU5 register
- PFC5 register
- PF5 register

The block diagram of the security function is shown below.

**Figure 13-3. Block Diagram of Security Function**




This function is set with the PLLCTL.RTOST0 bit.

**(1) PLL control register (PLLCTL)**

The PLLCTL register is an 8-bit register that controls the RTO security function and PLL.

This register can be read or written in 8-bit or 1-bit units.

After reset, PLLCTL is set to 01H.

After reset: 01H R/W Address: FFFFF806H

| | 7 | 6 | 5 | 4 | 3 | <2> | <1> | <0> |
|---|---|---|---|---|---|---|---|---|
| PLLCTL | 0 | 0 | 0 | 0 | 0 | RTOST0 | SELPLL Note | PLLON Note |

| RTOST0 | Control of RTP00 to RTP05 security function |
|---|---|
| 0 | INTP0 pin is not used as trigger for security function |
| 1 | INTP0 pin is used as trigger for security function |

**Note** For details on the SELPLL and PLLON bits, refer to **CHAPTER 6 CLOCK GENERATION FUNCTION**.

**Cautions 1. Before outputting a value to the real-time output ports (RTP00 to RTP05), select the INTP0 pin interrupt edge detection and then set the RTOST0 bit.**

**2. To set again the ports (P50 to P55 pins) as real-time output ports after placing them in high impedance via the INTP0 pin, first cancel the security function.**

**[Procedure to set ports again]**

**<1> Cancel the security function and enable port setting by clearing the RTOST0 bit to 0.**

**<2> Set the RTOST0 bit to 1 (only if required).**

**<3> Set again as real-time output port.**

**3. Be sure to clear bits 4 to 7 to 0. Changing bit 3 does not affect the operation.**

# CHAPTER 14 A/D CONVERTER

## 14.1 Overview

The A/D converter converts analog input signals into digital values and has an 8-channel (ANI0 to ANI7) configuration.

The A/D converter has the following functions.

- ○ Operating voltage ($AV_{REF0}$): 2.7 to 5.5 V
- ○ Successive approximation method 10-bit A/D converter
- ○ Analog input pin: 8
- ○ Trigger mode:
  - Software trigger mode
  - Timer trigger mode (INTTM010)
  - External trigger mode (ADTRG pin)
- ○ Operation mode
  - Select mode
  - Scan mode
- ○ A/D conversion time:
  - Normal mode:
    14 to 100 $\mu$s @ 4.0 V ≤ $AV_{REF0}$ ≤ 5.5 V
    17 to 100 $\mu$s @ 2.7 V ≤ $AV_{REF0}$ < 4.0 V
  - High-speed mode:
    3 to 100 $\mu$s @ 4.5 V ≤ $AV_{REF0}$ ≤ 5.5 V
    4.8 to 100 $\mu$s @ 4.0 V ≤ $AV_{REF0}$ < 4.5 V
    6 to 100 $\mu$s @ 2.85 V ≤ $AV_{REF0}$ < 4.0 V
    14 to 100 $\mu$s @ 2.7 V ≤ $AV_{REF0}$ < 2.85 V
- ○ Power fail detection function

## 14.2 Functions

**(1) 10-bit resolution A/D conversion**

1 analog input channel is selected from the ANI0 to ANI7 pins, and an A/D conversion operation with resolution of 10 bits is repeatedly executed. Every time A/D conversion is completed, an interrupt request signal (INTAD) is generated.

**(2) Power fail detection function**

This is a function to detect low voltage in a battery. The results of A/D conversion (the value in the ADCRH register) and the PFT register are compared, and INTAD signal is generated only when the comparison conditions match.

## 14.3 Configuration

The A/D converter consists of the following hardware.

**Figure 14-1. Block Diagram of A/D Converter**

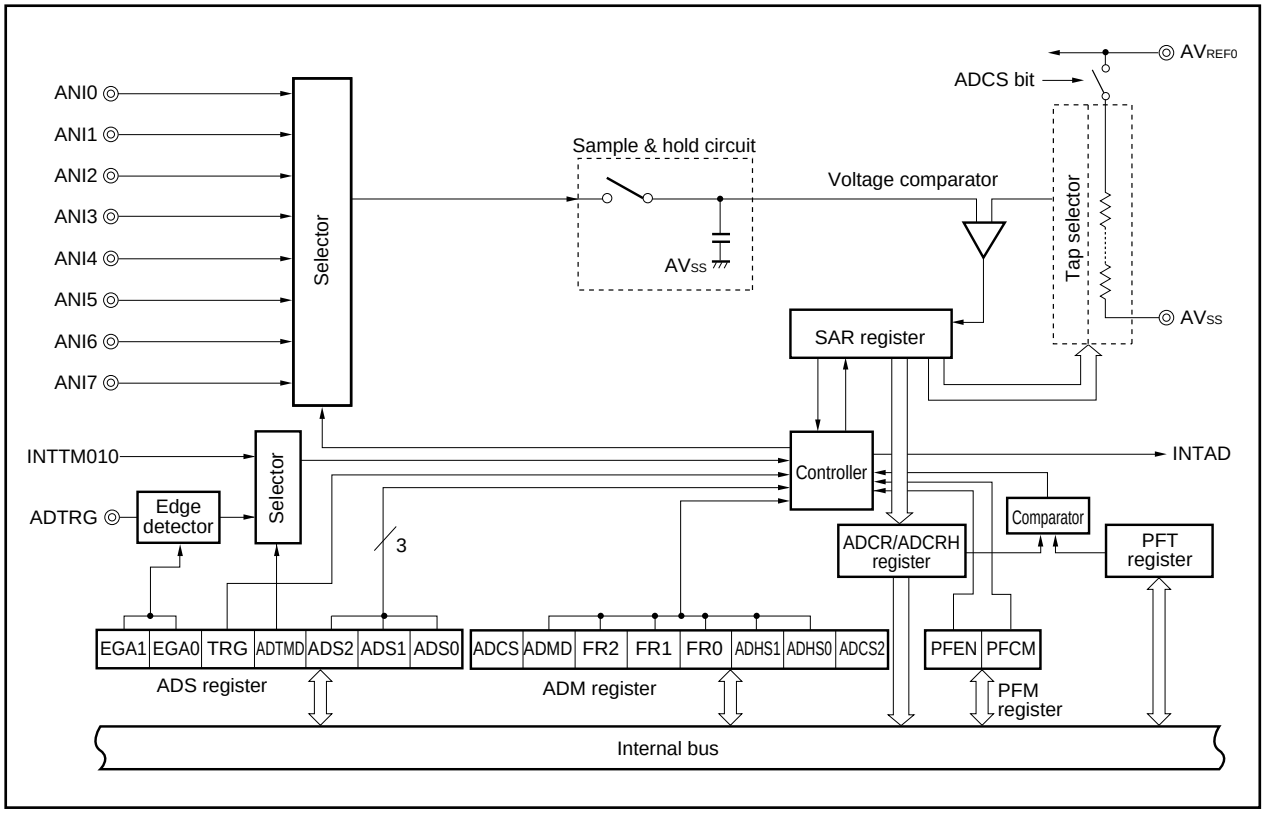



**Table 14-1. Registers of A/D Converter Used by Software**

| Item | Configuration |
|---|---|
| Registers | A/D conversion result register (ADCR)<br>A/D conversion result register H (ADCRH): Only higher 8 bits can be read<br>Power fail comparison threshold register (PFT)<br>A/D converter mode register (ADM)<br>Analog input channel specification register (ADS)<br>Power fail comparison mode register (PFM) |

**(1) ANI0 to ANI7 pins**

These are analog input pins for the 8 channels of the A/D converter. They are used to input analog signals to be converted into digital signals. Pins other than those selected as analog input by the ADS register can be used as input ports.

**(2) Sample & hold circuit**

The sample & hold circuit samples the analog input signals selected by the input circuit and sends the sampled data to the voltage comparator. This circuit holds the sampled analog input voltage during A/D conversion.

**(3) Series resistor string**

The series resistor string is connected between $AV_{REF0}$ and $AV_{SS}$ and generates a voltage for comparison with the analog input signal.

**(4) Voltage comparator**

The voltage comparator compares the value that is sampled and held with the output voltage of the series resistor string.

**(5) Successive approximation register (SAR)**

This register compares the sampled analog voltage value with the voltage value from the series resistor string, and converts the comparison result starting from the most significant bit (MSB).

When the least significant bit (LSB) has been converted to a digital value (end of A/D conversion), the contents of the SAR register are transferred to the ADCR register.

The SAR register cannot be read or written directly.

**(6) A/D conversion result register (ADCR), A/D conversion result register H (ADCRH)**

Each time A/D conversion ends, the conversion results are loaded from the successive approximation register and the results of A/D conversion are held in the higher 10 bits of this register (the lower 6 bits are fixed to 0).

**(7) Controller**

The controller compares the A/D conversion results (the value of the ADCRH register) with the value of the PFT register when A/D conversion ends or the power fail detection function is used. It generates INTAD signal only when the comparison conditions match.

**(8) $AV_{REF0}$ pin**

This is the analog power supply pin/reference voltage input pin of the A/D converter. Always use the same potential as the $V_{DD}$ pin even when not using the A/D converter.

The signals input to the ANI0 to ANI7 pins are converted into digital signals based on the voltage applied across $AV_{REF0}$ and $AV_{SS}$.

**(9) $AV_{SS}$ pin**

This is the ground potential pin of the A/D converter. Always use the same potential as the $V_{SS}$ pin even when not using the A/D converter.

**(10) A/D converter mode register (ADM)**

This register sets the conversion time of the analog input to be converted to a digital signal and the conversion operation start/stop.

**(11) Analog input channel specification register (ADS)**

This register specifies the input port for the analog voltage to be converted to a digital signal.

**(12) Power fail comparison mode register (PFM)**

This register sets the power fail detection mode.

**(13) Power fail comparison threshold register (PFT)**

This register sets the threshold to be compared with the ADCR register.

## 14.4 Registers

The A/D converter is controlled by the following registers.

- A/D converter mode register (ADM)
- Analog input channel specification register (ADS)
- Power fail comparison mode register (PFM)
- Power fail comparison threshold register (PFT)
- A/D conversion result register, A/D conversion result register H (ADCR, ADCRH)

**(1) A/D converter mode register (ADM)**

This register sets the conversion time of the analog input signal to be converted into a digital signal as well as conversion start and stop.

The ADM register can be read or written in 8-bit or 1-bit units.

After reset, ADM is cleared to 00H.

After reset: 00H R/W Address: FFFFF200H

| | <7> | 6 | 5 | 4 | 3 | 2 | 1 | <0> |
|---|---|---|---|---|---|---|---|---|
| ADM | ADCS | ADMD | FR2[Note 1] | FR1[Note 1] | FR0[Note 1] | ADHS1[Note 1] | ADHS0[Note 1] | ADCS2 |

| ADCS | Control of A/D conversion operation |
|---|---|
| 0 | Conversion operation stopped |
| 1 | Conversion operation enabled |

| ADMD | Control of operation mode |
|---|---|
| 0 | Select mode |
| 1 | Scan mode |

| ADHS1 | Selection of 5 V A/D conversion time mode ($AV_{REF0} \geq 4.5$ V) |
|---|---|
| 0 | Normal mode |
| 1 | High-speed mode (valid only when $AV_{REF0} \geq 4.5$ V) |

| ADHS0 | Selection of 3 V A/D conversion time mode ($AV_{REF0} \geq 2.7$ or 2.85 V) |
|---|---|
| 0 | Normal mode |
| 1 | High-speed mode (valid only when $AV_{REF0} \geq 2.7$ or 2.85 V) |

| ADCS2 | Control of reference voltage generator for boosting[Note 2] |
|---|---|
| 0 | Reference voltage generator operation stopped |
| 1 | Reference voltage generator operation enabled |

**Notes 1.** For details of the FR2 to FR0 bits and the A/D conversion, refer to **Table 14-2 A/D Conversion Time**.

**2.** The operation of the reference voltage generator for boosting is controlled by the ADCS2 bit and it takes 1 or 14 $\mu$s after operation is started until it is stabilized. Therefore, if the ADCS bit is set to 1 (A/D conversion is started) at least 1 or 14 $\mu$s after the ADCS2 bit was set to 1 (reference voltage generator for boosting is on), the first conversion result is valid.

**Cautions 1. Changing bits FR2 to FR0, ADHS1, and ADHS0 while the ADCS bit = 1 is prohibited (write access to the ADM register is enabled and rewriting of bits FR2 to FR0, ADHS1, and ADHS0 is prohibited).**

**2. Setting ADHS1 and ADHS0 bits to 11 is prohibited.**

**3. When the main clock is stopped and the CPU is operating on the subclock, do not access the ADM register using an access method that causes a wait.**
**For details, refer to 3.4.8 (2).**

**Remark** $f_{XX}$: Main clock frequency

**Table 14-2. A/D Conversion Time**

| ADHS1 | ADHS0 | FR2 | FR1 | FR0 | | A/D Conversion Time ($\mu$s) | | | | Conversion Time Mode |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | 20 MHz@ $AV_{REF0} \geq 4.5$ V | 16 MHz@ $AV_{REF0} \geq 4.0$ V | 8 MHz@ $AV_{REF0} \geq 2.85$ V | 8 MHz@ $AV_{REF0} \geq 2.7$ V | |
| 0 | 0 | 0 | 0 | 0 | 288/$f_{XX}$ | 14.4 | 18.0 | 36.0 | 36.0 | Normal mode $AV_{REF0} \geq 2.7$ V |
| 0 | 0 | 0 | 0 | 1 | 240/$f_{XX}$ | Setting prohibited | 15.0 | 30.0 | 30.0 | |
| 0 | 0 | 0 | 1 | 0 | 192/$f_{XX}$ | Setting prohibited | Setting prohibited | 24.0 | 24.0 | |
| 0 | 0 | 0 | 1 | 1 | Setting prohibited | | | | | |
| 0 | 0 | 1 | 0 | 0 | 144/$f_{XX}$ | Setting prohibited | Setting prohibited | 18.0 | 18.0 | Normal mode $AV_{REF0} \geq 2.7$ V |
| 0 | 0 | 1 | 0 | 1 | 120/$f_{XX}$ | Setting prohibited | Setting prohibited | Setting prohibited | Setting prohibited | |
| 0 | 0 | 1 | 1 | 0 | 96/$f_{XX}$ | Setting prohibited | Setting prohibited | Setting prohibited | Setting prohibited | |
| 0 | 0 | 1 | 1 | 1 | Setting prohibited | | | | | |
| 0 | 1 | 0 | 0 | 0 | 96/$f_{XX}$ | 4.8 | 6.0 | 12.0 | Setting prohibited | High-speed mode $AV_{REF0} \geq 2.85$ V |
| 0 | 1 | 0 | 0 | 1 | 72/$f_{XX}$ | Setting prohibited | Setting prohibited | 9.0 | Setting prohibited | |
| 0 | 1 | 0 | 1 | 0 | 48/$f_{XX}$ | Setting prohibited | Setting prohibited | 6.0 | Setting prohibited | |
| 0 | 1 | 0 | 1 | 1 | 24/$f_{XX}$ | Setting prohibited | Setting prohibited | Setting prohibited | Setting prohibited | |
| 0 | 1 | 1 | 0 | 0 | 224/$f_{XX}$ | 11.2 | 14.0 | 28.0 | 28.0 | High-speed mode $AV_{REF0} \geq 2.7$ V |
| 0 | 1 | 1 | 0 | 1 | 168/$f_{XX}$ | Setting prohibited | 10.5 | 21.0 | 21.0 | |
| 0 | 1 | 1 | 1 | 0 | 112/$f_{XX}$ | Setting prohibited | Setting prohibited | Setting prohibited | Setting prohibited | |
| 0 | 1 | 1 | 1 | 1 | 56/$f_{XX}$ | Setting prohibited | Setting prohibited | Setting prohibited | Setting prohibited | |
| 1 | 0 | 0 | 0 | 0 | 72/$f_{XX}$ | 3.6 | Setting prohibited | Setting prohibited | Setting prohibited | High-speed mode $AV_{REF0} \geq 4.5$ V |
| 1 | 0 | 0 | 0 | 1 | 54/$f_{XX}$ | Setting prohibited | Setting prohibited | Setting prohibited | Setting prohibited | |
| 1 | 0 | 0 | 1 | 0 | 36/$f_{XX}$ | Setting prohibited | Setting prohibited | Setting prohibited | Setting prohibited | |
| 1 | 0 | 0 | 1 | 1 | 18/$f_{XX}$ | Setting prohibited | Setting prohibited | Setting prohibited | Setting prohibited | |
| 1 | 0 | 1 | × | × | Setting prohibited | | | | | |
| 1 | 1 | × | × | × | Setting prohibited | | | | | |

**(a) Controlling reference voltage generator for boosting**

When the ADCS2 bit = 0, power to the A/D converter drops. The converter requires a setup time of 14 $\mu$s (normal mode: ADHS1 and ADHS0 bits = 00) or 1 $\mu$s (high-speed mode: ADHS1 and ADHS0 bits = 11) or more after the ADCS2 bit has been set to 1.

Therefore, the result of A/D conversion becomes valid from the first result by setting the ADCS bit to 1 at least 14 or 1 $\mu$s after the ADCS2 bit has been set to 1.

**Table 14-3. Setting of ADCS Bit and ADCS2 Bit**

| ADCS | ADCS2 | A/D Conversion Operation |
|---|---|---|
| 0 | 0 | Stopped status (DC power consumption path does not exist) |
| 0 | 1 | Conversion standby mode (only the reference voltage generator for boosting consumes power) |
| 1 | 0 | Conversion mode (reference voltage generator stops operation[Note 1]) |
| 1 | 1 | Conversion mode (reference voltage generator is operating[Note 2]) |

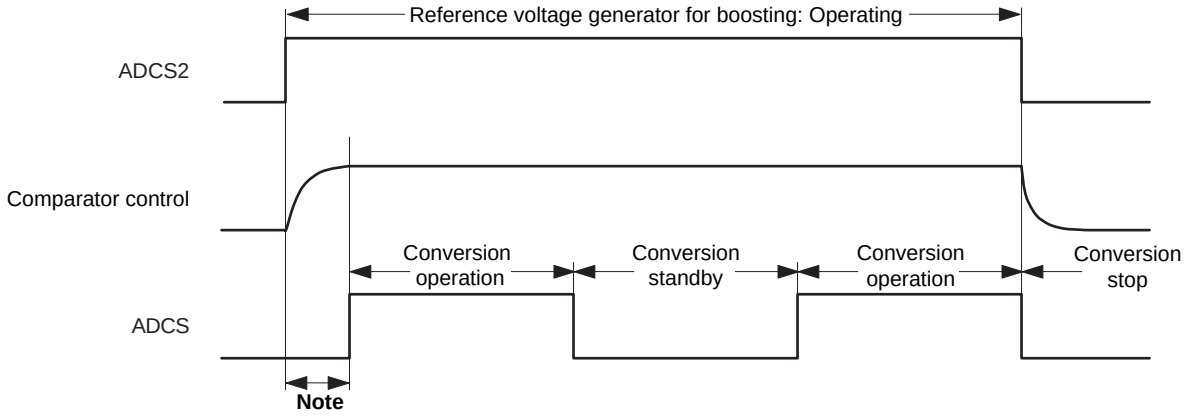
**Notes 1.** If the ADCS and ADCS2 bits are changed from 00B to 10B, the reference voltage generator for boosting automatically turns on. If the ADCS bit is cleared to 0 while the ADCS2 bit is 0, the voltage generator automatically turns off. In the software trigger mode (ADS.TRG bit = 0), use of the first A/D conversion result is prohibited.

In the hardware trigger mode (TRG bit = 1), use the A/D conversion result only if A/D conversion is started after the lapse of the oscillation stabilization time of the reference voltage generator for boosting.

**2.** If the ADCS and ADCS2 bits are changed from 00B to 11B, the reference voltage generator for boosting automatically turns on. If the ADCS bit is cleared to 0 while the ADCS2 bit is 1, the voltage generator stays on. In the software trigger mode (TRG bit = 0), use of the first A/D conversion result is prohibited.

In the hardware trigger mode (TRG bit = 1), use the A/D conversion result only if A/D conversion is started after the lapse of the oscillation stabilization time of the reference voltage generator for boosting.

**Figure 14-2. Operation Sequence**




**Note** 1 or 14 $\mu$s or more are required for the operation of the reference voltage generator for boosting between when the ADCS2 bit is set (1) and when the ADCS bit is set (1).

**(2) Analog input channel specification register (ADS)**

This register specifies the analog voltage input port for A/D conversion.

The ADS register can be read or written in 8-bit or 1-bit units.

After reset, ADS is cleared to 00H.

After reset: 00H R/W Address: FFFFF201H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ADS | EGA1[Note 1] | EGA0[Note 1] | TRG | ADTMD[Note 2] | 0 | ADS2 | ADS1 | ADS0 |

| EGA1[Note 1] | EGA0[Note 1] | Specification of external trigger signal (ADTRG) edge |
|---|---|---|
| 0 | 0 | No edge detection |
| 0 | 1 | Falling edge |
| 1 | 0 | Rising edge |
| 1 | 1 | Both rising and falling edges |

| TRG | Trigger mode selection |
|---|---|
| 0 | Software trigger mode |
| 1 | Hardware trigger mode |

| ADTMD[Note 2] | Specification of hardware trigger mode |
|---|---|
| 0 | External trigger (ADTRG pin input) |
| 1 | Timer trigger (INTTM010 signal generated) |

| ADS2 | ADS1 | ADS0 | Specification of analog input channel | |
|---|---|---|---|---|
| | | | Select mode | Scan mode |
| 0 | 0 | 0 | ANI0 | ANI0 |
| 0 | 0 | 1 | ANI1 | ANI0, ANI1 |
| 0 | 1 | 0 | ANI2 | ANI0 to ANI2 |
| 0 | 1 | 1 | ANI3 | ANI0 to ANI3 |
| 1 | 0 | 0 | ANI4 | ANI0 to ANI4 |
| 1 | 0 | 1 | ANI5 | ANI0 to ANI5 |
| 1 | 1 | 0 | ANI6 | ANI0 to ANI6 |
| 1 | 1 | 1 | ANI7 | ANI0 to ANI7 |

**Notes 1.** The EGA1 and EGA0 bits are valid only when the hardware trigger mode (TRG bit = 1) and external trigger mode (ADTRG pin input: ADTMD bit = 1) are selected.

**2.** The ADTMD bit is valid only when the hardware trigger mode (TRG bit = 1) is selected.

**Cautions 1. When the main clock is stopped and the CPU is operating on the subclock, do not access the ADS register using an access method that causes a wait. For details, refer to 3.4.8 (2).**

**2. Be sure to clear bit 3 to 0.**

**(3) A/D conversion result register, A/D conversion result register H (ADCR, ADCRH)**

The ADCR and ADCRH registers store the A/D conversion results.

These registers are read-only in 16-bit or 8-bit units. However, specify the ADCR register for 16-bit access, and the ADCRH register for 8-bit access. In the ADCR register, the 10 bits of conversion results are read in the higher 10 bits and 0 is read in the lower 6 bits. In the ADCRH register, the higher 8 bits of the conversion results are read.

After reset, these registers are undefined.

After reset: Undefined R Address: FFFFF204H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADCR | AD9 | AD8 | AD7 | AD6 | AD5 | AD4 | AD3 | AD2 | AD1 | AD0 | 0 | 0 | 0 | 0 | 0 | 0 |

After reset: Undefined R Address: FFFFF205H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ADCRH | AD9 | AD8 | AD7 | AD6 | AD5 | AD4 | AD3 | AD2 |

**Caution When the main clock is stopped and the CPU is operating on the subclock, do not access the ADCR and ADCRH registers using an access method that causes a wait. For details, refer to 3.4.8 (2).**

The following shows the relationship between the analog input voltage input to the analog input pins (ANI0 to ANI7) and A/D conversion results (ADCR register).

$$SAR = INT\left(\frac{V_{IN}}{AV_{REF0}} \times 1024 + 0.5\right)$$

$$ADCR^{Note} = SAR \times 64$$

Or,

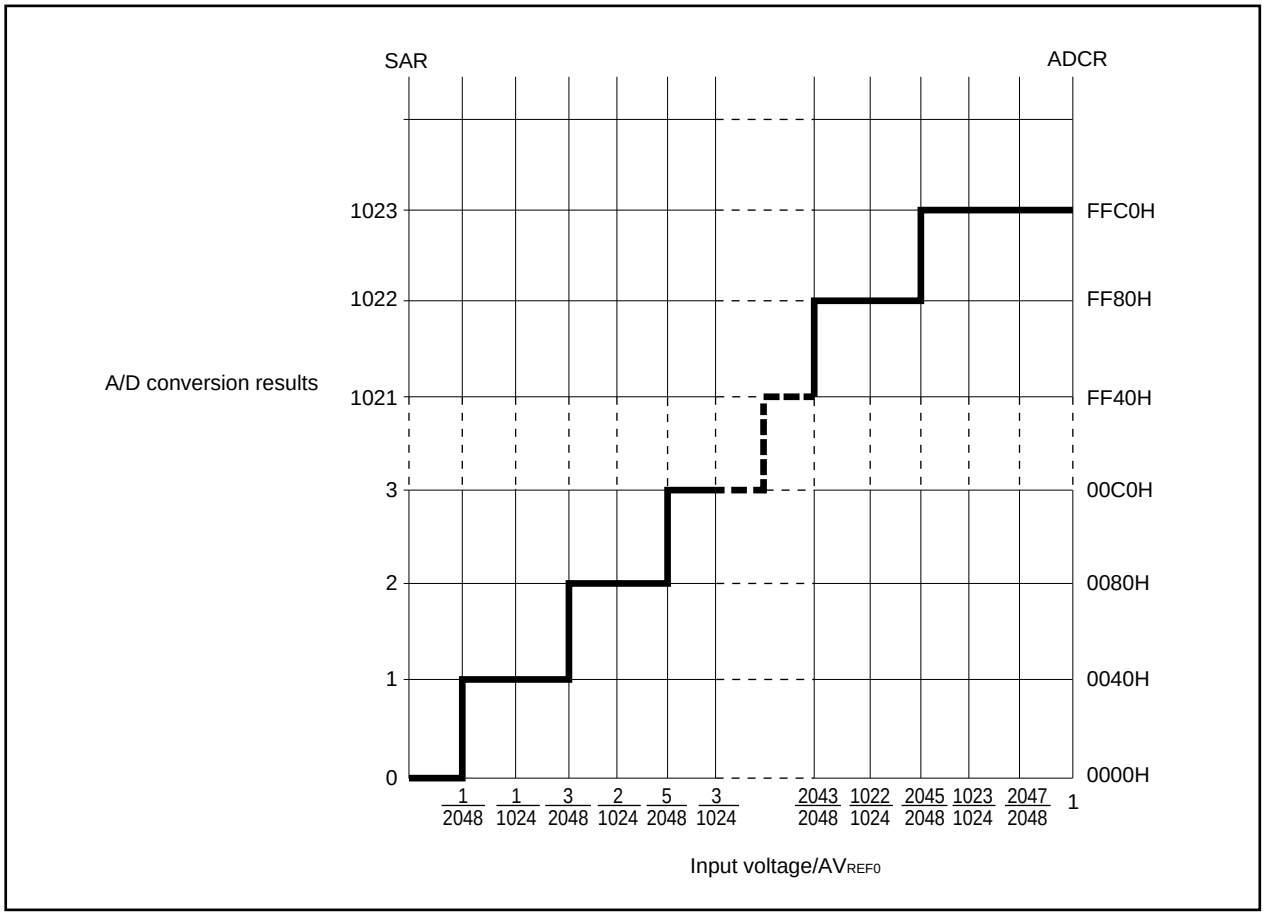
$$(SAR - 0.5) \times \frac{AV_{REF0}}{1024} \leq V_{IN} < (SAR + 0.5) \times \frac{AV_{REF0}}{1024}$$

INT ( ): Function that returns the integer part of the value in parentheses
$V_{IN}$: Analog input voltage
$AV_{REF0}$: Voltage of $AV_{REF0}$ pin
ADCR: Value in the ADCR register

**Note** The lower 6 bits of the ADCR register are fixed to 0.

The following shows the relationship between the analog input voltage and A/D conversion results.

**Figure 14-3. Relationship Between Analog Input Voltage and A/D Conversion Results**



**(4) Power fail comparison mode register (PFM)**

This register sets the power fail detection mode.

The PFM register compares the value in the PFT register with the value of the ADCRH register.

The PFM register can be read or written in 8-bit or 1-bit units.

After reset, PFM is cleared to 00H.

After reset: 00H R/W Address: FFFFF202H

| | <7> | <6> | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PFM | PFEN | PFCM | 0 | 0 | 0 | 0 | 0 | 0 |

| PFEN | Selection of power fail comparison enable/disable |
|---|---|
| 0 | Power fail comparison disabled |
| 1 | Power fail comparison enabled |

| PFCM | Selection of power fail comparison mode |
|---|---|
| 0 | Interrupt request signal (INTAD) generated when ADCR ≥ PFT |
| 1 | Interrupt request signal (INTAD) generated when ADCR < PFT |

**Caution When the main clock is stopped and the CPU is operating on the subclock, do not access the PFM register using an access method that causes a wait. For details, refer to 3.4.8 (2).**

**(5) Power fail comparison threshold register (PFT)**

The PFT register sets the comparison value in the power fail detection mode.

The 8-bit data set in the PFT register is compared with the value of the ADCRH register.

The PFT register can be read or written in 8-bit units.

After reset, PFT is cleared to 00H.

After reset: 00H R/W Address: FFFFF203H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| PFT | | | | | | | | |

**Caution When the main clock is stopped and the CPU is operating on the subclock, do not access the PFT register using an access method that causes a wait. For details, refer to 3.4.8 (2).**

## 14.5 Operation

### 14.5.1 Basic operation

<1> Select the channel whose analog signal is to be converted into a digital signal using the ADS register.
Set the ADM.ADHS1 or ADM.ADHS0 bit.

<2> Set the ADM.ADCS2 bit to 1 and wait 1 or 14 $\mu$s or longer.

<3> Set the ADM.ADCS bit to 1 to start A/D conversion.
(Steps <4> to <10> are executed by hardware.)

<4> The sample & hold circuit samples the voltage input to the selected analog input channel.

<5> After sampling for a specific time, the sample & hold circuit enters the hold status and holds the input analog voltage until it has been converted into a digital signal.

<6> Set bit 9 of the successive approximation register (SAR) to 1. The tap selector sets the voltage tap of the series resistor string to (1/2) $\times$ $AV_{REF0}$.

<7> The voltage comparator compares the voltage difference between the voltage tap of the series resistor string and the analog input voltage. If the analog input voltage is greater than (1/2) $\times$ $AV_{REF0}$, the MSB of the SAR register remains set to 1. If the analog input voltage is less than (1/2) $\times$ $AV_{REF0}$, the MSB is cleared to 0.

<8> Next, bit 8 of the SAR register is automatically set to 1 and the next comparison starts. Depending on the previously determined value of bit 9, the voltage tap of the series resistor string is selected as follows.

- Bit 9 = 1: (3/4) $\times$ $AV_{REF0}$
- Bit 9 = 0: (1/4) $\times$ $AV_{REF0}$

The analog input voltage is compared with one of these voltage taps and bit 8 of the SAR register is manipulated as follows depending on the result of the comparison.

Analog input voltage $\geq$ voltage tap: Bit 8 = 1
Analog input voltage $\leq$ voltage tap: Bit 8 = 0

<9> The above steps are repeated until bit 0 of the SAR register has been manipulated.

<10> When comparison of all 10 bits of the SAR register has been completed, the valid digital value remains in the SAR register, and the value of the SAR register is transferred and latched to the ADCR register.
At the same time, an A/D conversion end interrupt request signal (INTAD) is generated.

<11> Repeat steps <4> to <10> until the ADCS bit is cleared to 0.
For another A/D conversion, start at <3>. However, when operating the A/D converter with the ADCS2 bit cleared to 0, start at <2>.

### 14.5.2 Trigger modes

The V850ES/KF1+ has the following three trigger modes that set the A/D conversion start timing. These trigger modes are set by the ADS register.

- Software trigger mode
- External trigger mode (hardware trigger mode)
- Timer trigger mode (hardware trigger mode)

**(1) Software trigger mode**

This mode is used to start A/D conversion by setting the ADM.ADCS bit to 1 while the ADS.TRG bit is 0.

Conversion is repeatedly performed as long as the ADCS bit is not cleared to 0 after completion of A/D conversion.

If the ADM, ADS, PFM, or PFT register is written during conversion, A/D conversion is aborted and started again from the beginning.

**(2) External trigger mode (hardware trigger mode)**

This is the status in which the ADS.TRG bit is set to 1 and ADS.ADTMD bit is cleared to 0. This mode is used to start A/D conversion by detecting an external trigger (ADTRG) after the ADCS bit has been set to 1.

The A/D converter waits for the external trigger (ADTRG) after the ADCS bit is set to 1.

The valid edge of the signal input to the ADTRG pin is specified by using the ADS.EGA1 and ADS.EGA0 bits.

When the specified valid edge is detected, A/D conversion is started.

When A/D conversion is completed, the A/D converter waits for the external trigger (ADTRG) again.

If a valid edge is input to the ADTRG pin during A/D conversion, A/D conversion is aborted and started again from the beginning.

If the ADM, ADS, PFM, or PFT register is written during conversion, A/D conversion is aborted and the A/D converter waits for an external trigger (ADTRG).

**(3) Timer trigger mode (hardware trigger mode)**

This mode is used to start A/D conversion by detecting a timer trigger (INTTM010) after the ADCS bit has been set to 1 with the TGR bit = 1 and ADTMD bit = 1.

The A/D converter waits for the timer trigger (INTTM010) after the ADCS bit is set to 1.

When the INTTM010 signal is generated, A/D conversion is started.

When A/D conversion is completed, the A/D converter waits for the timer trigger (INTTM010) again.

If the INTTM010 signal is generated during A/D conversion, A/D conversion is aborted and started again from the beginning.

If the ADM, ADS, PFM, or PFT register is written during conversion, A/D conversion is aborted and the A/D converter waits for a timer trigger (INTTM010).

### 14.5.3 Operation modes

The following two operation modes are available. These operation modes are set by the ADM register.

- Select mode
- Scan mode

#### (1) Select mode

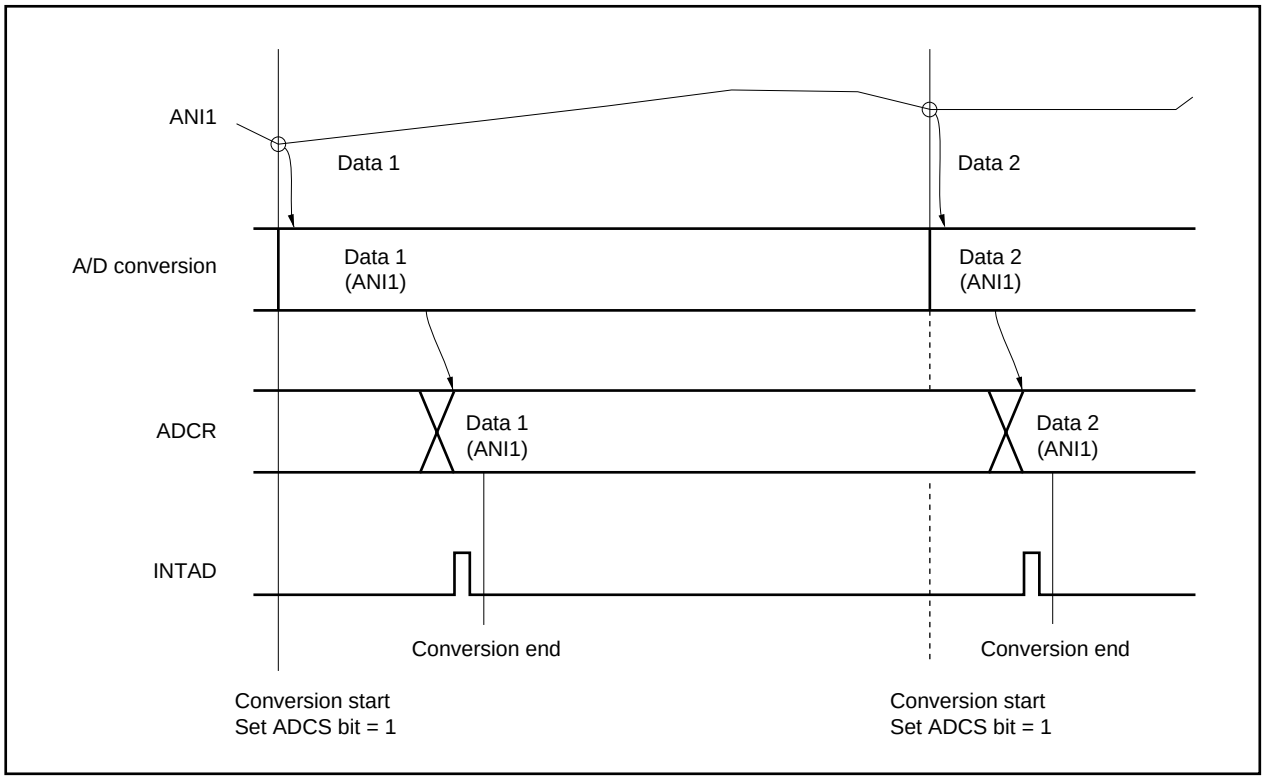
One input analog signal specified by the ADS register while the ADM.ADMD bit = 0 is converted. When conversion is complete, the result of conversion is stored in the ADCR register.

At the same time, the A/D conversion end interrupt request signal (INTAD) is generated. However, the INTAD signal may or may not be generated depending on setting of the PFM and PFT registers. For details, refer to **14.5.4 Power fail detection function**.

If anything is written to the ADM, ADS, PFM, and PFT registers during conversion, A/D conversion is aborted. In the software trigger mode, A/D conversion is started from the beginning again. In the hardware trigger mode, the A/D converter waits for a trigger.

If the trigger is detected during conversion in hardware trigger mode, A/D conversion is aborted and started again from the beginning.

**Figure 14-4. Example of Select Mode Operation Timing (ADS.ADS2 to ADS.ADS0 Bits = 0001B)**



**(2) Scan mode**

In this mode, the analog signals specified by the ADS register and input from the ANI0 pin while the ADM.ADMD bit = 1 are sequentially selected and converted.

When conversion of one analog input signal is complete, the conversion result is stored in the ADCR register and, at the same time, the A/D conversion end interrupt request signal (INTAD) is generated.

The A/D conversion results of all the analog input signals are stored in the ADCR register. It is therefore recommended to save the contents of the ADCR register to RAM once A/D conversion of one analog input signal has been completed.

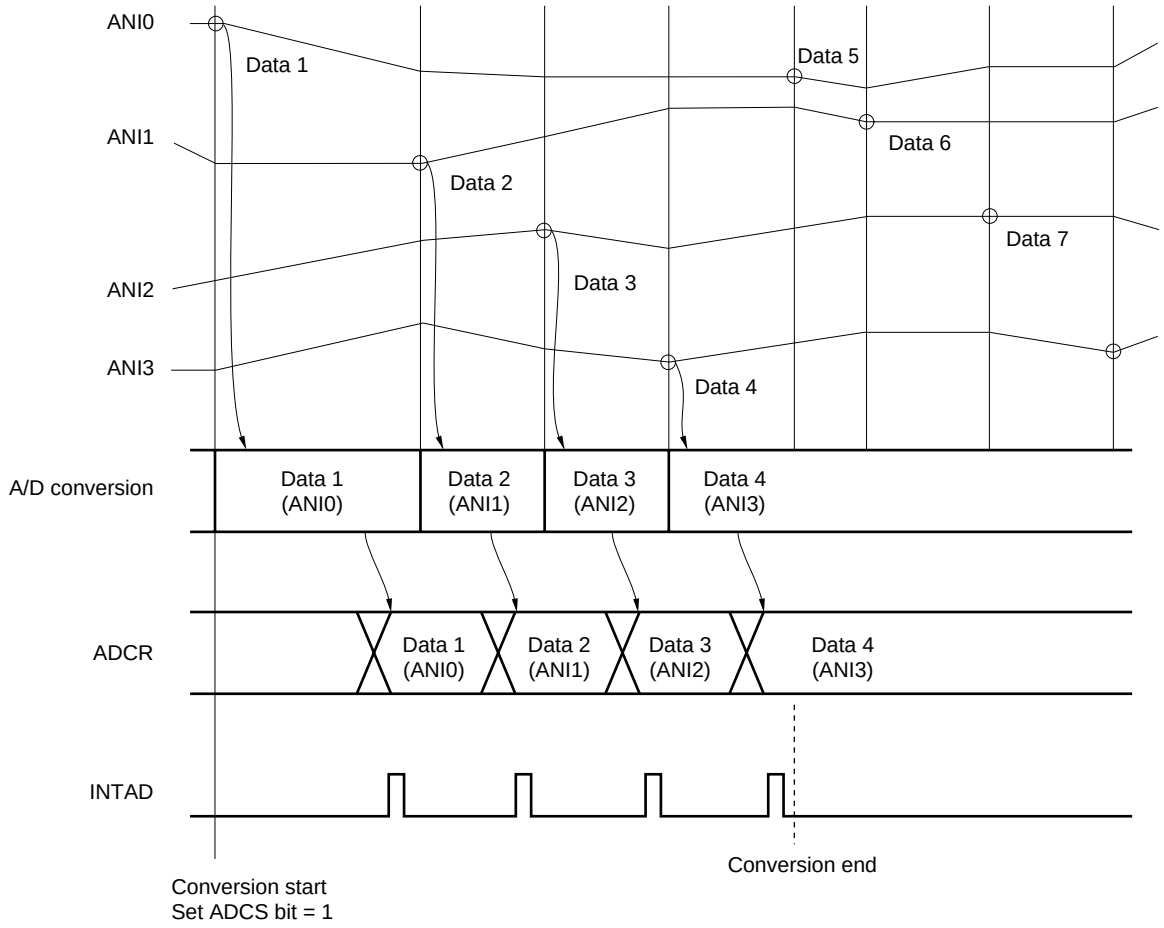
In the hardware trigger mode (ADS.TRG bit = 1), the A/D converter waits for a trigger after it has completed A/D conversion of the analog signals specified by the ADS register and input from the ANI0 pin.

If anything is written to the ADM, ADS, PFM, and PFT registers during conversion, A/D conversion is aborted. In the software trigger mode, A/D conversion is started from the beginning again. In the hardware trigger mode, the A/D converter waits for a trigger. Conversion starts again from the ANI0 pin.

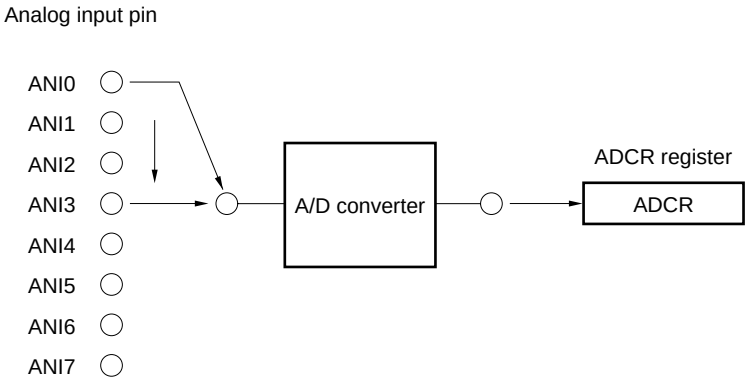
If the trigger is detected during conversion in hardware trigger mode, A/D conversion is aborted and started again from the beginning (ANI0 pin).

**Figure 14-5. Example of Scan Mode Operation Timing (ADS.ADS2 to ADS.ADS0 Bits = 0011B)**

**(a) Timing example**




**(b) Block diagram**

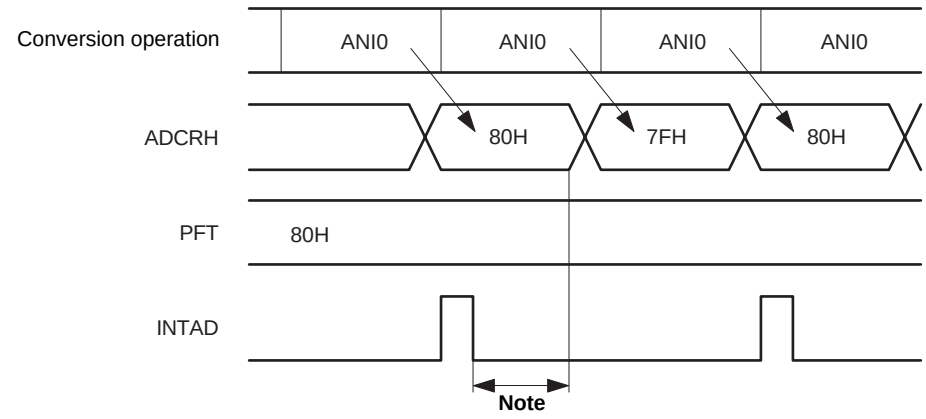


### 14.5.4 Power fail detection function

The conversion end interrupt request signal (INTAD) can be controlled as follows using the PFM and PFT registers.

- If the PFM.PFEN bit = 0, the INTAD signal is generated each time conversion ends.
- If the PFEN bit = 1 and the PFM.PFCM bit = 0, the conversion result (ADCRH register) and the value of the PFT register are compared when conversion ends, and the INTAD signal is generated only if ADCRH ≥ PFT.
- If the PFEN and PFCM bits = 1, the conversion result and the value of the PFT register are compared when conversion ends, and the INTAD signal is generated only if ADCRH < PFT.
- Because, when the PFEN bit = 1, the conversion result is overwritten after the INTAD signal has been generated, unless the conversion result is read by the time the next conversion ends, in some cases it may appear as if the actual operation differs from the operation described above (refer to **Figure 14-6**).

**Figure 14-6. Power Fail Detection Function (PFCM Bit = 0)**




**Note** If reading is not performed during this interval, the conversion result changes to the next conversion result.

### 14.5.5 Setting method

The following describes how to set registers.

**(1) When using the A/D converter for A/D conversion**

<1> Set (1) the ADM.ADCS2 bit.

<2> Select the channel and conversion time by setting the ADS.ADS2 to ADS.ADS0 bits and the ADM.ADHS1, ADM.ADHS0, and ADM.FR2 to ADM.FR0 bits.

<3> Set (1) the ADM.ADCS bit.

<4> Transfer the A/D conversion data to the ADCR register.

<5> An interrupt request signal (INTAD) is generated.

<Changing the channel>

<6> Change the channel by setting the ADS2 to ADS0 bits.

<7> Transfer the A/D conversion data to the ADCR register.

<8> The INTAD signal is generated.

<Ending A/D conversion>

<9> Clear (0) the ADCS bit.

<10> Clear (0) the ADCS2 bit.

**Cautions 1. The time taken from <1> to <3> must be 1 or 14 μs or longer.**

**2. Steps <1> and <2> may be reversed.**

**3. Step <1> may be omitted. However, if omitted, do not use the first conversion result after <3>.**

**4. The time taken from <4> to <7> is different from the conversion time set by the ADHS1, ADHS0, and FR2 to FR0 bits.**
**The time taken for <6> and <7> is the conversion time set by the ADHS1, ADHS0, and FR2 to FR0 bits.**

**(2) When using the A/D converter for the power fail detection function**

<1> Set (1) the PFM.PFEN bit.

<2> Set the power fail comparison conditions by using the PFM.PFCM bit.

<3> Set (1) the ADM.ADCS2 bit.

<4> Select the channel and conversion time by setting the ADS.ADS2 to ADS.ADS0 bits and the ADM.ADHS1, ADM.ADHS0, and ADM.FR2 to ADM.FR0 bits.

<5> Set the threshold value in the PFT register.

<6> Set (1) the ADM.ADCS bit.

<7> Transfer the A/D conversion data to the ADCR register.

<8> Compare the ADCRH register with the PFT register. An interrupt request signal (INTAD) is generated when the conditions match.

<Changing the channel>

<9> Change the channel by setting the ADS2 to ADS0 bits.

<10> Transfer the A/D conversion data to the ADCR register.

<11> The ADCRH register is compared with the PFT register. When the conditions match, an INTAD signal is generated.

<Ending A/D conversion>

<12> Clear (0) the ADCS bit.

<13> Clear (0) the ADCS2 bit.

**Remark** If the operation of the power fail detection function is enabled, all the A/D conversion results are compared, regardless of whether the select mode or scan mode is set.

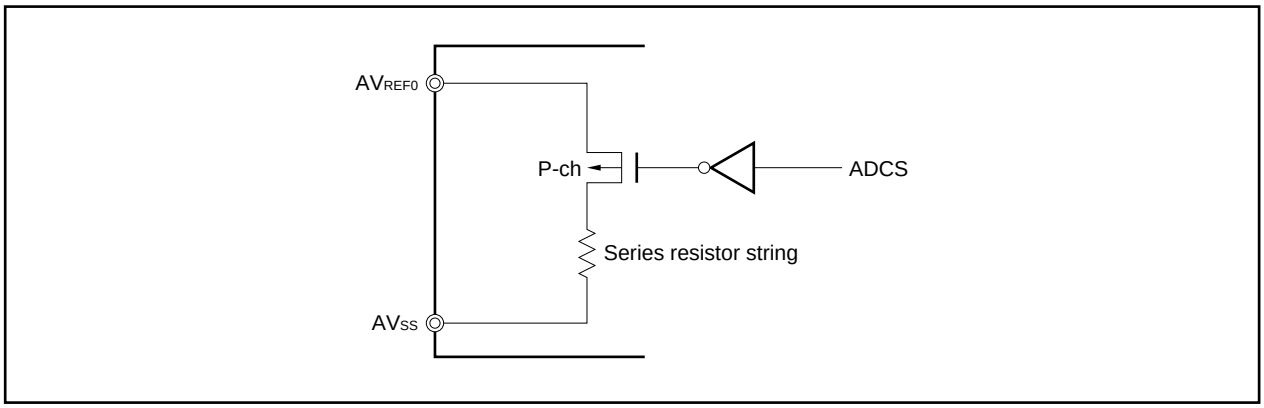
## 14.6 Cautions

**(1) Power consumption in standby mode**

The operation of the A/D converter stops in the standby mode. At this time, the power consumption can be reduced by stopping the conversion operation (the ADM.ADCS bit = 0).

Figure 14-7 shows an example of how to reduce the power consumption in the standby mode.

**Figure 14-7. Example of How to Reduce Power Consumption in Standby Mode**




**(2) Input range of ANI0 to ANI7 pins**

Use the A/D converter with the ANI0 to ANI7 pin input voltages within the specified range. If a voltage of $AV_{REF0}$ or higher or $AV_{SS}$ or lower (even if within the absolute maximum ratings) is input to these pins, the conversion value of the channel is undefined. Also, this may affect the conversion value of other channels.

**(3) Conflicting operations**

(a) Conflict between writing to the ADCR register and reading from ADCR register upon the end of conversion

Reading the ADCR register takes precedence. After the register has been read, a new conversion result is written to the ADCR register.

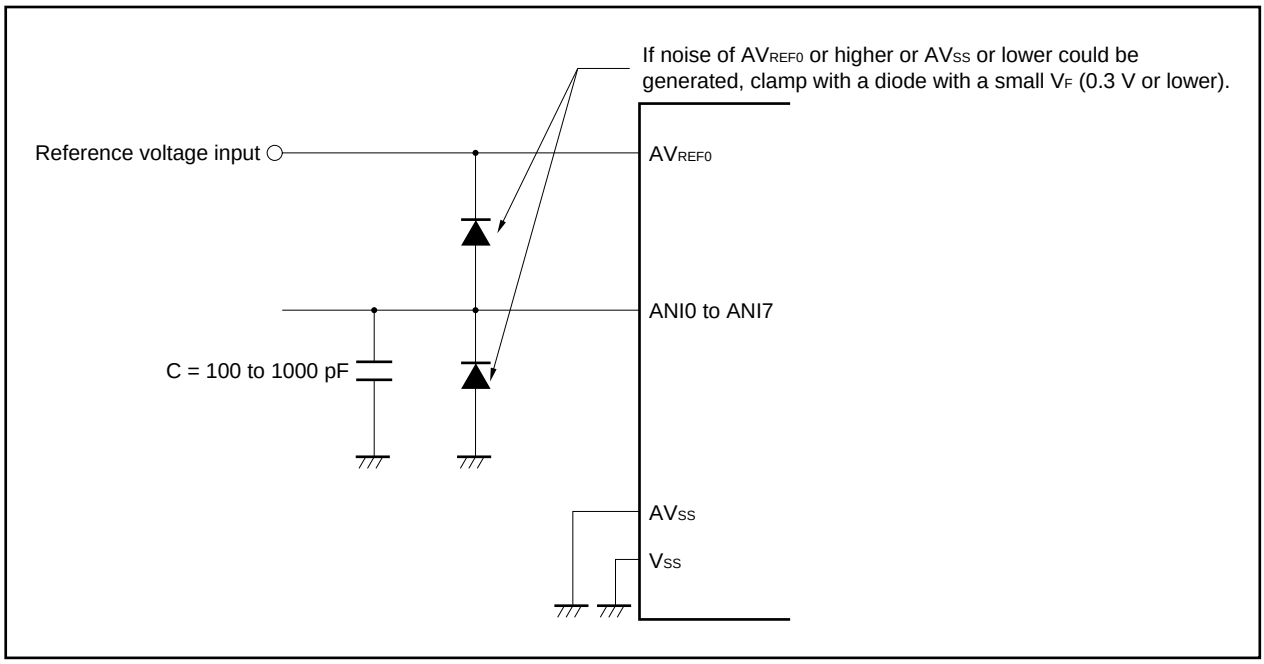
(b) Conflict between writing to the ADCR register and writing to the ADM register or writing to the ADS register upon the end of conversion

Writing to the ADM register or ADS register takes precedence. The ADCR register is not written, and neither is the conversion end interrupt request signal (INTAD) generated.

**(4) Measures against noise**

To keep a resolution of 10 bits, be aware of noise on the $AV_{REF0}$ and ANI0 to ANI7 pins. The higher the output impedance of the analog input source, the greater the effect of noise. Therefore, it is recommended to connect external capacitors as shown in Figure 14-8 to reduce noise.

**Figure 14-8. Handling of Analog Input Pins**




**(5) ANI0/P70 to ANI7/P77 pins**

The analog input pins (ANI0 to ANI7) function alternately as input port pins (P70 to P77).

When performing A/D conversion by selecting any of the ANI0 to ANI7 pins, do not execute an input instruction to port 7 during conversion. This may decrease the conversion resolution.

If digital pulses are applied to the pin adjacent to the pin subject to A/D conversion, the value of the A/D conversion may differ from the expected value because of coupling noise. Therefore, do not apply pulses to the pin adjacent to the pin subject to A/D conversion.

**(6) Input impedance of $AV_{REF0}$ pin**

A series resistor string of tens of kΩ is connected between the $AV_{REF0}$ pin and $AV_{SS}$ pin.

Therefore, if the output impedance of the reference voltage source is high, this will result in a series connection to the series resistor string between the $AV_{REF0}$ pin and $AV_{SS}$ pin, resulting in a large reference voltage error.

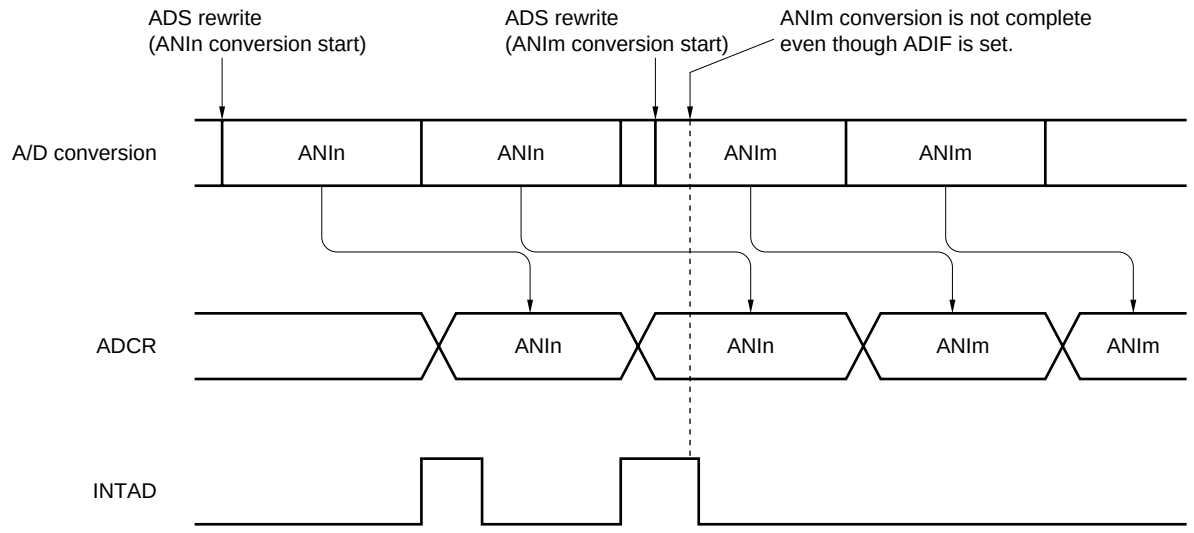
**(7) Interrupt request flag (ADIC.ADIF bit)**

Even when the ADS register is changed, the ADIF bit is not cleared (0).

Therefore, if the analog input pin is changed during A/D conversion, the ADIF bit may be set (1) because A/D conversion of the previous analog input pin ends immediately before the ADS register is rewritten. In a such case, note that if the ADIF bit is read immediately after the ADS register has been rewritten, the ADIF bit is set (1) even though A/D conversion of the analog input pin after the change has not been completed.

When stopping A/D conversion once and resuming it, clear the ADIF bit (0) before resuming A/D conversion.

**Figure 14-9. A/D Conversion End Interrupt Request Occurrence Timing**




**Remark** n = 0 to 7
m = 0 to 7

**(8) Conversion results immediately after A/D conversion start**

If the ADM.ADCS bit is set to 1 within 1 or 14 $\mu$s after the ADM.ADCS2 bit has been set to 1, or if the ADCS bit is set to 1 with the ADCS2 bit cleared to 0, the converted value immediately after the A/D conversion operation has started may not satisfy the rating. Take appropriate measures such as polling the A/D conversion end interrupt request signal (INTAD) and discarding the first conversion result.

**(9) Reading A/D conversion result register (ADCR)**

When the ADM or ADS register has been written, the contents of the ADCR register may become undefined. When the conversion operation is complete, read the conversion results before writing to the ADM or ADS register. A correct conversion result may not be able to be read at a timing other than the above.

When the CPU is operating on the subclock and main clock oscillation ($f_X$) is stopped, do not read the ADCR register. For details, refer to **3.4.8 (2)**.

**(10) A/D converter sampling time and A/D conversion start delay time**

The A/D converter sampling time differs depending on the set value of the ADM register. A delay time exists until actual sampling is started after A/D converter operation is enabled.

When using a set in which the A/D conversion time must be strictly observed, care is required for the contents shown in Figure 14-10 and Table 14-4.

**Figure 14-10. Timing of A/D Converter Sampling and A/D Conversion Start Delay**

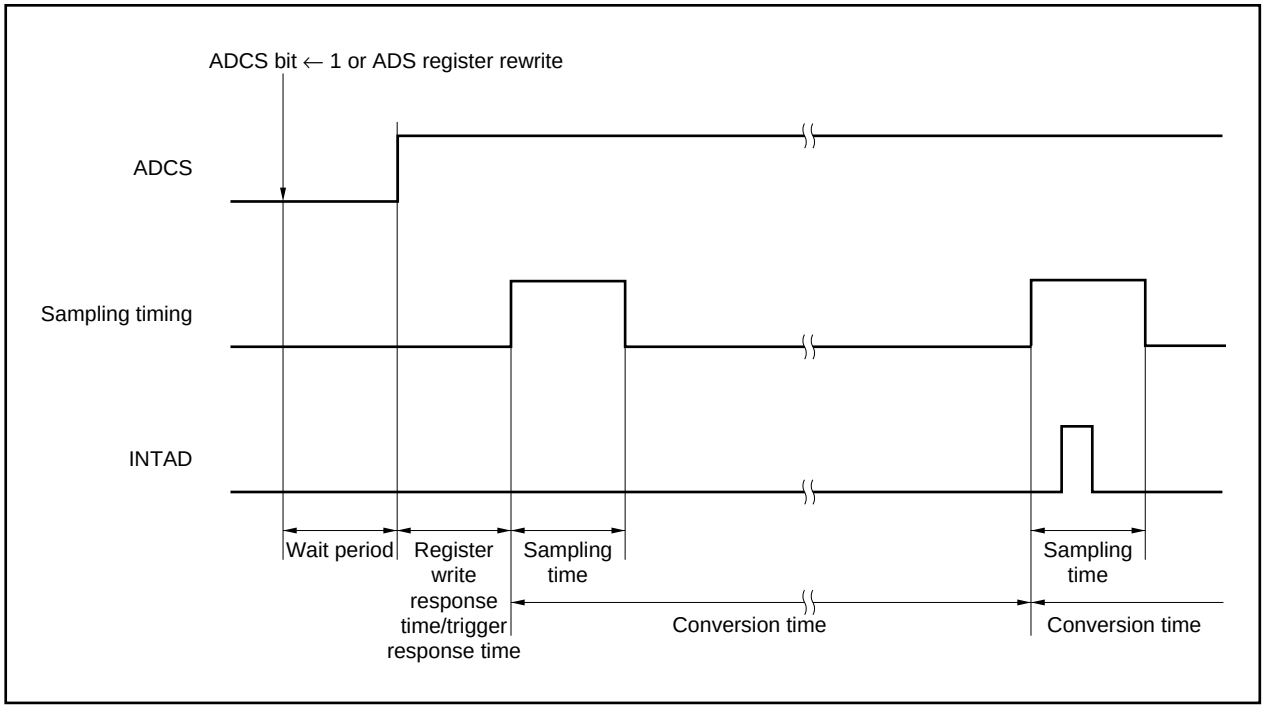


**Table 14-4. A/D Converter Conversion Time**

| ADHS1 | ADHS0 | FR2 | FR1 | FR0 | Conversion Time | Sampling Time | Register Write Response Time[Note] | | Trigger Response Time[Note] | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | MIN. | MAX. | MIN. | MAX. |
| 0 | 0 | 0 | 0 | 0 | 288/$f_{XX}$ | 176/$f_{XX}$ | 11/$f_{XX}$ | 12/$f_{XX}$ | 7/$f_{XX}$ | 8/$f_{XX}$ |
| 0 | 0 | 0 | 0 | 1 | 240/$f_{XX}$ | 176/$f_{XX}$ | 11/$f_{XX}$ | 12/$f_{XX}$ | 7/$f_{XX}$ | 8/$f_{XX}$ |
| 0 | 0 | 0 | 1 | 0 | 192/$f_{XX}$ | 132/$f_{XX}$ | 10/$f_{XX}$ | 11/$f_{XX}$ | 6/$f_{XX}$ | 7/$f_{XX}$ |
| 0 | 0 | 1 | 0 | 0 | 144/$f_{XX}$ | 88/$f_{XX}$ | 9/$f_{XX}$ | 10/$f_{XX}$ | 5/$f_{XX}$ | 6/$f_{XX}$ |
| 0 | 0 | 1 | 0 | 1 | 120/$f_{XX}$ | 88/$f_{XX}$ | 9/$f_{XX}$ | 10/$f_{XX}$ | 5/$f_{XX}$ | 6/$f_{XX}$ |
| 0 | 0 | 1 | 1 | 0 | 96/$f_{XX}$ | 48/$f_{XX}$ | 11/$f_{XX}$ | 12/$f_{XX}$ | 7/$f_{XX}$ | 8/$f_{XX}$ |
| 0 | 1 | 0 | 0 | 0 | 96/$f_{XX}$ | 48/$f_{XX}$ | 11/$f_{XX}$ | 12/$f_{XX}$ | 7/$f_{XX}$ | 8/$f_{XX}$ |
| 0 | 1 | 0 | 0 | 1 | 72/$f_{XX}$ | 36/$f_{XX}$ | 10/$f_{XX}$ | 11/$f_{XX}$ | 6/$f_{XX}$ | 7/$f_{XX}$ |
| 0 | 1 | 0 | 1 | 0 | 48/$f_{XX}$ | 24/$f_{XX}$ | 9/$f_{XX}$ | 10/$f_{XX}$ | 5/$f_{XX}$ | 6/$f_{XX}$ |
| 0 | 1 | 0 | 1 | 1 | 24/$f_{XX}$ | 12/$f_{XX}$ | 8/$f_{XX}$ | 9/$f_{XX}$ | 4/$f_{XX}$ | 5/$f_{XX}$ |
| 0 | 1 | 1 | 0 | 0 | 224/$f_{XX}$ | 176/$f_{XX}$ | 11/$f_{XX}$ | 12/$f_{XX}$ | 7/$f_{XX}$ | 8/$f_{XX}$ |
| 0 | 1 | 1 | 0 | 1 | 168/$f_{XX}$ | 132/$f_{XX}$ | 10/$f_{XX}$ | 11/$f_{XX}$ | 6/$f_{XX}$ | 7/$f_{XX}$ |
| 0 | 1 | 1 | 1 | 0 | 112/$f_{XX}$ | 88/$f_{XX}$ | 9/$f_{XX}$ | 10/$f_{XX}$ | 5/$f_{XX}$ | 6/$f_{XX}$ |
| 0 | 1 | 1 | 1 | 1 | 56/$f_{XX}$ | 44/$f_{XX}$ | 8/$f_{XX}$ | 9/$f_{XX}$ | 4/$f_{XX}$ | 5/$f_{XX}$ |
| 1 | 0 | 0 | 0 | 0 | 72/$f_{XX}$ | 24/$f_{XX}$ | 11/$f_{XX}$ | 12/$f_{XX}$ | 7/$f_{XX}$ | 8/$f_{XX}$ |
| 1 | 0 | 0 | 0 | 1 | 54/$f_{XX}$ | 18/$f_{XX}$ | 10/$f_{XX}$ | 11/$f_{XX}$ | 6/$f_{XX}$ | 7/$f_{XX}$ |
| 1 | 0 | 0 | 1 | 0 | 36/$f_{XX}$ | 12/$f_{XX}$ | 9/$f_{XX}$ | 10/$f_{XX}$ | 5/$f_{XX}$ | 6/$f_{XX}$ |
| 1 | 0 | 0 | 1 | 1 | 18/$f_{XX}$ | 6/$f_{XX}$ | 8/$f_{XX}$ | 9/$f_{XX}$ | 4/$f_{XX}$ | 5/$f_{XX}$ |
| Other than above | | | | | Setting prohibited | – | – | – | – | – |

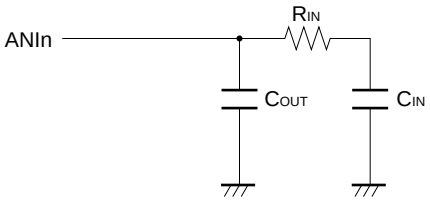
**Note** Each response time is the time after the wait period. For the wait function, refer to **3.4.8 (2) Access to special on-chip peripheral I/O register**.

**Remark** $f_{XX}$: Main clock frequency

**(11) Internal equivalent circuit**

The following shows the equivalent circuit of the analog input block.

**Figure 14-11. Internal Equivalent Circuit of ANIn Pin**




| $AV_{REF0}$ | $R_{IN}$ | $C_{OUT}$ | $C_{IN}$ |
|---|---|---|---|
| 4.5 V | 3 kΩ | 8 pF | 15 pF |
| 2.7 V | 60 kΩ | 8 pF | 15 pF |

**Remarks 1.** The above values are reference values.

**2.** n = 0 to 7

## 14.7 How to Read A/D Converter Characteristics Table

Here, special terms unique to the A/D converter are explained.

**(1) Resolution**

This is the minimum analog input voltage that can be identified. That is, the percentage of the analog input voltage per bit of digital output is called 1 LSB (Least Significant Bit). The percentage of 1 LSB with respect to the full scale is expressed by %FSR (Full Scale Range). %FSR indicates the ratio of analog input voltage that can be converted as a percentage, and is always represented by the following formula regardless of the resolution.

1 %FSR = (Max. value of analog input voltage that can be converted − Min. value of analog input voltage that can be converted)/100
= ($AV_{REF0}$ – 0)/100
= $AV_{REF0}$/100

1 LSB is as follows when the resolution is 10 bits.

1 LSB = $1/2^{10}$ = 1/1024
= 0.098 %FSR

Accuracy has no relation to resolution, but is determined by overall error.

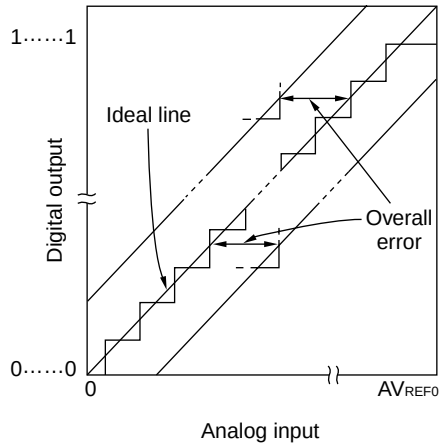
**(2) Overall error**

This shows the maximum error value between the actual measured value and the theoretical value.

Zero-scale error, full-scale error, linearity error and errors that are combinations of these express the overall error.

Note that the quantization error is not included in the overall error in the characteristics table.

**Figure 14-12. Overall Error**

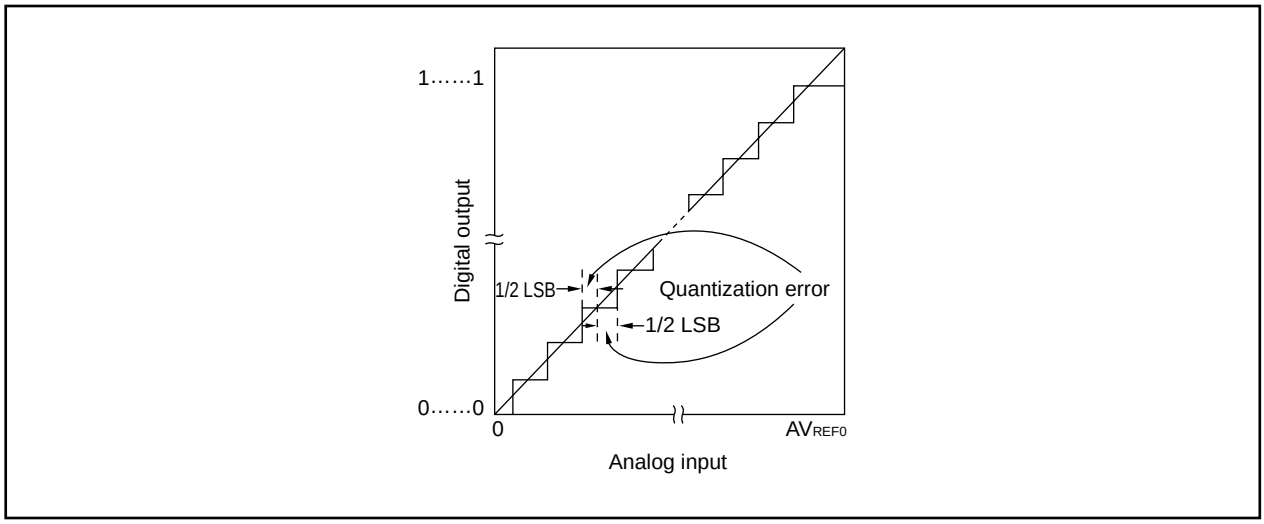


**(3) Quantization error**

When analog values are converted to digital values, a ±1/2 LSB error naturally occurs. In an A/D converter, an analog input voltage in a range of ±1/2 LSB is converted to the same digital code, so a 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 in the characteristics table.

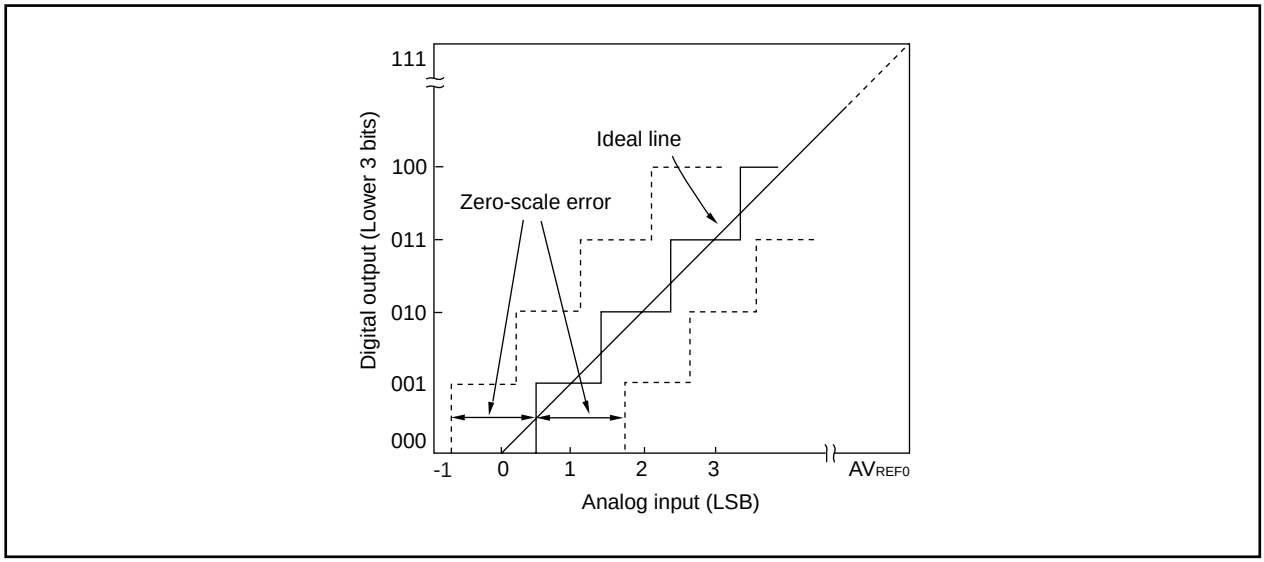
**Figure 14-13. Quantization Error**




**(4) Zero-scale error**

This shows the difference between the actual measurement value of the analog input voltage and the theoretical value (1/2 LSB) when the digital output changes from 0……000 to 0……001.

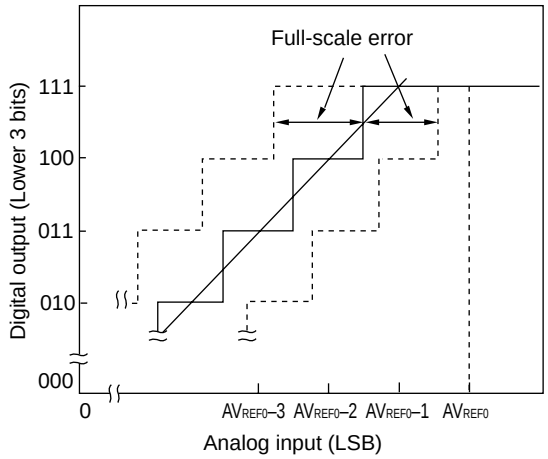
**Figure 14-14. Zero-Scale Error**



**(5) Full-scale error**

This shows the difference between the actual measurement value of the analog input voltage and the theoretical value (full scale − 3/2 LSB) when the digital output changes from 1……110 to 1……111.

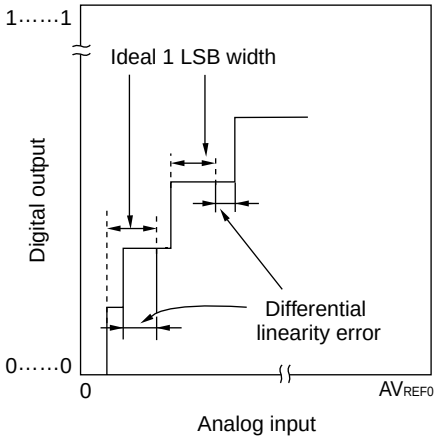
**Figure 14-15. Full-Scale Error**




**(6) Differential linearity error**

While the ideal width of code output is 1 LSB, this indicates the difference between the actual measurement value and the ideal value.

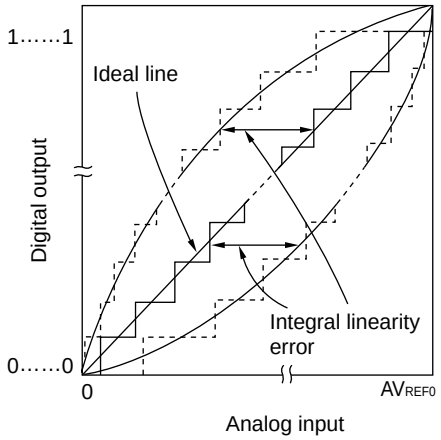
**Figure 14-16. Differential Linearity Error**



**(7) Integral linearity error**

This shows the degree to which the conversion characteristics deviate from the ideal linear relationship. It expresses the maximum value of the difference between the actual measurement value and the ideal straight line when the zero-scale error and full-scale error are 0.

**Figure 14-17. Integral Linearity Error**




**(8) Conversion time**

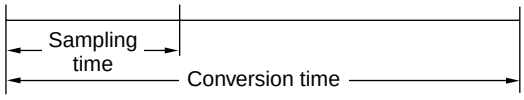
This expresses the time from when the analog input voltage was applied to the time when the digital output was obtained.

The sampling time is included in the conversion time in the characteristics table.

**(9) Sampling time**

This is the time the analog switch is turned on for the analog voltage to be sampled by the sample & hold circuit.

**Figure 14-18. Sampling Time**



# CHAPTER 15 ASYNCHRONOUS SERIAL INTERFACE (UART)

In the V850ES/KF1+, two channels of asynchronous serial interface (UART) are provided. Of these channels, UART0 supports LIN-bus.

## 15.1 Features

- Maximum transfer speed: 312.5 kbps
- Full-duplex communications
  On-chip RXBn register
  On-chip TXBn register
- Two-pin configuration[Note]
  TXDn: Transmit data output pin
  RXDn: Receive data input pin
- Reception error detection functions
  - Parity error
  - Framing error
  - Overrun error
- Interrupt sources: 3 types
  - Reception error interrupt request signal (INTSREn): Interrupt is generated according to the logical OR of the three types of reception errors
  - Reception completion interrupt request signal (INTSRn): Interrupt is generated when receive data is transferred from the receive shift register to the RXBn register after serial transfer is completed during a reception enabled state
  - Transmission completion interrupt request signal (INTSTn): Interrupt is generated when the serial transmission of transmit data (8 or 7 bits) from the transmit shift register is completed
- Character length: 7 or 8 bits
- Parity functions: Odd, even, 0, or none
- Transmission stop bits: 1 or 2 bits
- MSB-first or LSB-first transfer of data selectable (UART0 only)
- Transmit data output level inversion function (UART0 only)
- 13 to 20 bits selectable for Sync Break Field transmission (UART0 only)
- 11 bits or more identifiable for Sync Break Field reception (SBF reception flag (UART0 only))
- On-chip dedicated baud rate generator

**Note** The ASCK0 pin (external clock input) is available only for UART0.

**Remark** n = 0, 1

## 15.2 Configuration

**Table 15-1. Configuration of UARTn**

| Item | Configuration |
|---|---|
| Registers | Receive buffer register n (RXBn)<br>Transmit buffer register n (TXBn)<br>Receive shift register<br>Transmit shift register<br>Asynchronous serial interface mode register n (ASIMn)<br>Asynchronous serial interface status register n (ASISn)<br>Asynchronous serial interface transmit status register n (ASIFn)<br>LIN operation control register 0 (ASICL0) |
| Other | Reception control parity check<br>Addition of transmission control parity |

**Remark** n = 0, 1

Figure 15-1 shows the configuration of UARTn.

**(1) Asynchronous serial interface mode register n (ASIMn)**

The ASIMn register is an 8-bit register for specifying the operation of UARTn.

**(2) Asynchronous serial interface status register n (ASISn)**

The ASISn register consists of a set of flags that indicate the error contents when a reception error occurs. The various reception error flags are set (1) when a reception error occurs and are cleared (0) when the ASISn register is read.

**(3) Asynchronous serial interface transmit status register n (ASIFn)**

The ASIFn register is an 8-bit register that indicates the status when a transmit operation is performed.

This register consists of a transmit buffer data flag, which indicates the hold status of the TXBn register data, and the transmit shift register data flag, which indicates whether transmission is in progress.

**(4) LIN operation control register 0 (ASICL0)**

The ASICL0 register is an 8-bit register that controls the output format for SBF transmission/reception and transmission.

The ASICL0 register can be used only with UART0.

**(5) Reception control parity check**

The receive operation is controlled according to the contents set in the ASIMn register. A check for parity errors is also performed during a receive operation, and if an error is detected, a value corresponding to the error contents is set in the ASISn register.

**(6) Receive shift register**

This is a shift register that converts the serial data that was input to the RXDn pin to parallel data. One byte of data is received, and if a stop bit is detected, the receive data is transferred to the RXBn register.

This register cannot be directly manipulated.

**(7) Receive buffer register n (RXBn)**

The RXBn register is an 8-bit buffer register for holding receive data. When 7 characters are received, 0 is stored in the MSB.

During a reception enabled state, receive data is transferred from the receive shift register to the RXBn register, synchronized with the end of the shift-in processing of one frame.

Also, the reception completion interrupt request signal (INTSRn) is generated by the transfer of data to the RXBn register.

**(8) Transmit shift register**

This is a shift register that converts the parallel data that was transferred from the TXBn register to serial data. When one byte of data is transferred from the TXBn register, the shift register data is output from the TXDn pin.

The transmission completion interrupt request signal (INTSTn) is generated synchronized with the completion of transmission of one frame.

This register cannot be directly manipulated.

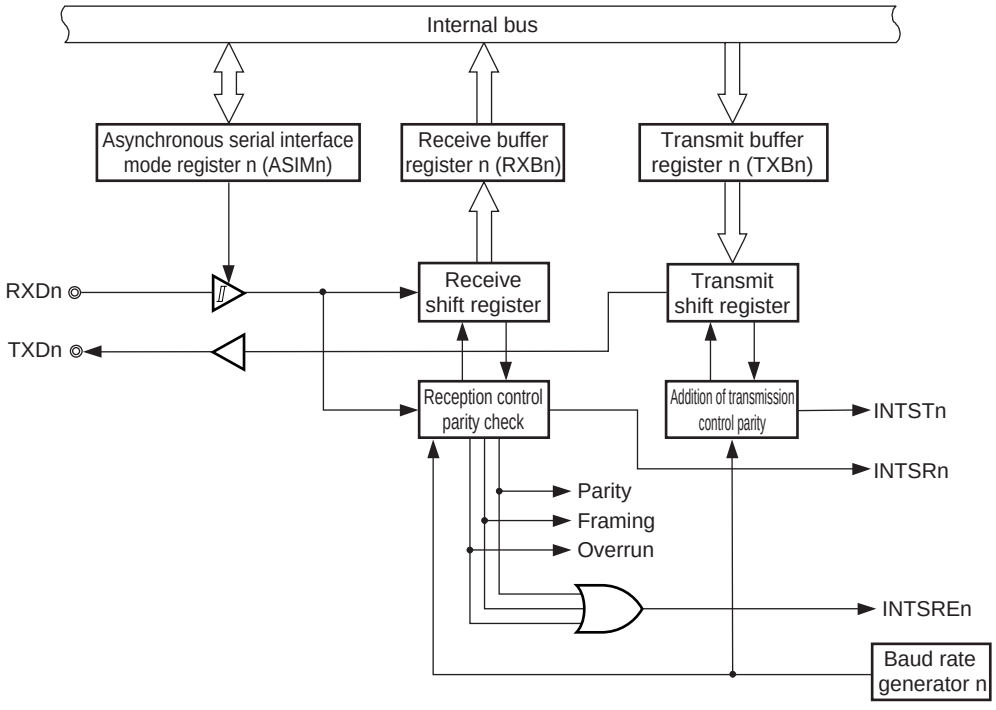
**(9) Transmit buffer register n (TXBn)**

The TXBn register is an 8-bit buffer for transmit data. A transmit operation is started by writing transmit data to the TXBn register.

**(10) Addition of transmission control parity**

A transmit operation is controlled by adding a start bit, parity bit, or stop bit to the data that is written to the TXBn register, according to the contents that were set in the ASIMn register.

**Figure 15-1. Block Diagram of UARTn**




**Remark** For the configuration of baud rate generator n, refer to **Figure 15-16**.

## 15.3 Registers

### (1) Asynchronous serial interface mode register n (ASIMn)

The ASIMn register is an 8-bit register that controls the UARTn transfer operation.

This register can be read or written in 8-bit or 1-bit units.

After reset, ASIMn is set to 01H.

**Cautions 1. When using UARTn, be sure to set the external pins related to UARTn functions to the control mode before setting the CKSRn and BRGCn registers, and then set the UARTEn bit to 1. Then set the other bits.**

**2. Set the UARTEn and RXEn bits to 1 while a high level is input to the RXDn pin. If these bits are set to 1 while a low level is input to the RXDn pin, reception will be started.**

(1/2)

After reset: 01H R/W Address: ASIM0 FFFFFA00H, ASIM1 FFFFFA10H

| | <7> | <6> | <5> | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ASIMn | UARTEn | TXEn | RXEn | PSn1 | PSn0 | CLn | SLn | ISRMn |

(n = 0, 1)

| UARTEn | Control of operating clock |
|---|---|
| 0 | Stop clock supply to UARTn. |
| 1 | Supply clock to UARTn. |

- If the UARTEn bit is cleared to 0, UARTn is asynchronously reset[Note].
- If the UARTEn bit = 0, UARTn is reset. To operate UARTn, first set the UARTEn bit to 1.
- If the UARTEn bit is cleared from 1 to 0, all the registers of UARTn are initialized. To set the UARTEn bit to 1 again, be sure to re-set the registers of UARTn.

The output of the TXDn pin goes high when transmission is disabled, regardless of the setting of the UARTEn bit.

| TXEn | Transmission enable/disable |
|---|---|
| 0 | Disable transmission |
| 1 | Enable transmission |

- Set the TXEn bit to 1 after setting the UARTEn bit to 1 at startup. Clear the UARTEn bit to 0 after clearing the TXEn bit to 0 to stop.
- To initialize the transmission unit, clear (0) the TXEn bit, and after letting 2 Clock cycles (base clock) elapse, set (1) the TXEn bit again. If the TXEn bit is not set again, initialization may not be successful. (For details about the base clock, refer to **15.6.1 (1) Base clock**.)

**Note** The ASISn, ASIFn, and RXBn registers are reset.

(2/2)

| RXEn | Reception enable/disable |
|---|---|
| 0 | Disable reception[Note] |
| 1 | Enable reception |

- Set the RXEn bit to 1 after setting the UARTEn bit to 1 at startup. Clear the UARTEn bit to 0 after clearing the RXEn bit to 0 to stop.
- To initialize the reception unit status, clear (0) the RXEn bit, and after letting 2 Clock cycles (base clock) elapse, set (1) the RXEn bit again. If the RXEn bit is not set again, initialization may not be successful. (For details about the base clock, refer to **15.6.1 (1) Base clock**.)

| PSn1 | PSn0 | Transmit operation | Receive operation |
|---|---|---|---|
| 0 | 0 | Don't output parity bit | Receive with no parity |
| 0 | 1 | Output 0 parity | Receive as 0 parity |
| 1 | 0 | Output odd parity | Judge as odd parity |
| 1 | 1 | Output even parity | Judge as even parity |

- To overwrite the PSn1 and PSn0 bits, first clear (0) the TXEn and RXEn bits.
- If "0 parity" is selected for reception, no parity judgment is performed. Therefore, no error interrupt is generated because the ASISn.PEn bit is not set.

| CLn | Specification of character length of 1 frame of transmit/receive data |
|---|---|
| 0 | 7 bits |
| 1 | 8 bits |

- To overwrite the CLn bit, first clear (0) the TXEn and RXEn bits.

| SLn | Specification of stop bit length of transmit data |
|---|---|
| 0 | 1 bit |
| 1 | 2 bits |

- To overwrite the SLn bit, first clear (0) the TXEn bit.
- Since reception is always done with a stop bit length of 1, the SLn bit setting does not affect receive operations.

| ISRMn | Enable/disable of generation of reception completion interrupt request signals when an error occurs |
|---|---|
| 0 | Generate a reception error interrupt request signal (INTSREn) as an interrupt when an error occurs.<br>In this case, no reception completion interrupt request signal (INTSRn) is generated. |
| 1 | Generate a reception completion interrupt request signal (INTSRn) as an interrupt when an error occurs.<br>In this case, no reception error interrupt request signal (INTSREn) is generated. |

- To overwrite the ISRMn bit, first clear (0) the RXEn bit.

**Note** When reception is disabled, the receive shift register does not detect a start bit. No shift-in processing or transfer processing to the RXBn register is performed, and the contents of the RXBn register are retained.

When reception is enabled, the receive shift operation starts, synchronized with the detection of the start bit, and when the reception of one frame is completed, the contents of the receive shift register are transferred to the RXBn register. A reception completion interrupt request signal (INTSRn) is also generated in synchronization with the transfer to the RXBn register.

**(2) Asynchronous serial interface status register n (ASISn)**

The ASISn register, which consists of 3 error flag bits (PEn, FEn, and OVEn), indicates the error status when UARTn reception is complete.

The ASISn register is cleared to 00H by a read operation. When a reception error occurs, the RXBn register should be read and the error flag should be cleared after the ASISn register is read.

This register is read-only in 8-bit units.

After reset, ASISn is cleared to 00H.

**Cautions 1. When the ASIMn.UARTEn bit or ASIMn.RXEn bit is cleared to 0, or when the ASISn register is read, the PEn, FEn, and OVEn bits are cleared (0).**

**2. Operation using a bit manipulation instruction is prohibited.**

**3. When the main clock is stopped and the CPU is operating on the subclock, do not access the ASISn register using an access method that causes a wait.**
**For details, refer to 3.4.8 (2).**

After reset: 00H R Address: ASIS0 FFFFFA03H, ASIS1 FFFFFA13H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ASISn | 0 | 0 | 0 | 0 | 0 | PEn | FEn | OVEn |

(n = 0, 1)

| PEn | Status flag indicating a parity error |
|---|---|
| 0 | When the UARTEn or RXEn bit is cleared to 0, or after the ASISn register has been read |
| 1 | When reception was completed, the receive data parity did not match the parity bit |
| • The operation of the PEn bit differs according to the settings of the ASIMn.PSn1 and ASIMn.PSn0 bits. | |

| FEn | Status flag indicating framing error |
|---|---|
| 0 | When the UARTEn or RXEn bit is cleared to 0, or after the ASISn register has been read |
| 1 | When reception was completed, no stop bit was detected |
| • For receive data stop bits, only the first bit is checked regardless of the stop bit length. | |

| OVEn | Status flag indicating an overrun error |
|---|---|
| 0 | When the UARTEn or RXEn bit is cleared to 0, or after the ASISn register has been read |
| 1 | When UARTn completed the next receive operation before reading receive data of the RXBn register |
| • When an overrun error occurs, the next receive data value is not written to the RXBn register and the data is discarded. | |

### (3) Asynchronous serial interface transmit status register n (ASIFn)

The ASIFn register, which consists of 2 status flag bits, indicates the status during transmission.

By writing the next data to the TXBn register after data is transferred from the TXBn register to the transmit shift register, transmit operations can be performed continuously without suspension even during an interrupt interval. When transmission is performed continuously, data should be written after referencing the TXBFn bit to prevent writing to the TXBn register by mistake.

This register is read-only in 8-bit or 1-bit units.

After reset, ASIFn is cleared to 00H.

After reset: 00H R Address: ASIF0 FFFFFA05H, ASIF1 FFFFFA15H

| | 7 | 6 | 5 | 4 | 3 | 2 | <1> | <0> |
|---|---|---|---|---|---|---|---|---|
| ASIFn | 0 | 0 | 0 | 0 | 0 | 0 | TXBFn | TXSFn |

(n = 0, 1)

| TXBFn | Transmission buffer data flag |
|---|---|
| 0 | Data to be transferred next to TXBn register does not exist (When the ASIMn.UARTEn or ASIMn.TXEn bit is cleared to 0, or when data has been transferred to the transmission shift register) |
| 1 | Data to be transferred next exists in TXBn register (Data exists in TXBn register when the TXBn register has been written to) |
| • When transmission is performed continuously, data should be written to the TXBn register after confirming that this flag is 0. If writing to TXBn register is performed when this flag is 1, transmit data cannot be guaranteed. | |

| TXSFn | Transmit shift register data flag (indicates the transmission status of UARTn) |
|---|---|
| 0 | Initial status or a waiting transmission (When the UARTEn or TXEn bit is cleared to 0, or when following transmission completion, the next data transfer from the TXBn register is not performed) |
| 1 | Transmission in progress (When data has been transferred from the TXBn register) |
| • When the transmission unit is initialized, initialization should be executed after confirming that this flag is 0 following the occurrence of a transmission completion interrupt request signal (INTSTn). If initialization is performed when this flag is 1, transmit data cannot be guaranteed. | |

**(4) Receive buffer register n (RXBn)**

The RXBn register is an 8-bit buffer register for storing parallel data that had been converted by the receive shift register.

When reception is enabled (ASIMn.RXEn bit = 1), receive data is transferred from the receive shift register to the RXBn register, synchronized with the completion of the shift-in processing of one frame. Also, a reception completion interrupt request signal (INTSRn) is generated by the transfer to the RXBn register. For information about the timing for generating this interrupt request, refer to **15.5.4 Receive operation**.

If reception is disabled (ASIMn.RXEn bit = 0), the contents of the RXBn register are retained, and no processing is performed for transferring data to the RXBn register even when the shift-in processing of one frame is completed. Also, the INTSRn signal is not generated.

When 7 bits is specified for the data length, bits 6 to 0 of the RXBn register are transferred for the receive data and the MSB (bit 7) is always 0. However, if an overrun error (ASISn.OVEn bit = 1) occurs, the receive data at that time is not transferred to the RXBn register.

The RXBn register becomes FFH when a reset is input or ASIMn.UARTEn bit = 0.

This register is read-only in 8-bit units.

After reset: FFH R Address: RXB0 FFFFFA02H, RXB1 FFFFFA12H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RXBn | RXBn7 | RXBn6 | RXBn5 | RXBn4 | RXBn3 | RXBn2 | RXBn1 | RXBn0 |

(n = 0, 1)

**(5) Transmit buffer register n (TXBn)**

The TXBn register is an 8-bit buffer register for setting transmit data.

When transmission is enabled (ASIMn.TXEn bit = 1), the transmit operation is started by writing data to TXBn register.

When transmission is disabled (TXEn bit = 0), even if data is written to the TXBn register, the value is ignored.

The TXBn register data is transferred to the transmit shift register, and a transmission completion interrupt request signal (INTSTn) is generated, synchronized with the completion of the transmission of one frame from the transmit shift register. For information about the timing for generating this interrupt request, refer to **15.5.2 Transmit operation**.

When the ASIFn.TXBFn bit = 1, writing must not be performed to the TXBn register.

This register can be read or written in 8-bit units.

After reset, TXBn is set to FFH.

After reset: FFH R/W Address: TXB0 FFFFFA04H, TXB1 FFFFFA14H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| TXBn | TXBn7 | TXBn6 | TXBn5 | TXBn4 | TXBn3 | TXBn2 | TXBn1 | TXBn0 |

(n = 0, 1)

### (6) LIN operation control register 0 (ASICL0)

The ASICL0 register is an 8-bit register that controls the output format for SBF transmission/reception and transmission.

This register can be read or written in 8-bit or 1-bit units.

After reset, ASICL0 is set to 16H.

**Caution The ASICL0 register is valid only for UART0. UART1 does not support this register.**

After reset: 16H R/W Address: FFFFFA08H

| | <7> | <6> | 5 | 4 | 3 | 2 | <1> | 0 |
|---|---|---|---|---|---|---|---|---|
| ASICL0 | SBRF0[Note] | SBRT0 | SBTT0 | SBL02 | SBL01 | SBL00 | UDIR0 | TXDLV0 |

| SBRF0[Note] | SBF reception status flag |
|---|---|
| 0 | If ASIM0.UARTE0 bit = 0 and ASIM0.RXE0 bit = 0 or if SBF reception has been completed correctly |
| 1 | SBF reception in progress |

| SBRT0 | SBF reception trigger |
|---|---|
| 0 | – |
| 1 | Reception trigger |

| SBTT0 | SBF transmission trigger |
|---|---|
| 0 | – |
| 1 | Transmission trigger |

| SBL02 | SBL01 | SBL00 | SBF transmission output width control |
|---|---|---|---|
| 1 | 0 | 1 | SBF is output with 13-bit length (default) |
| 1 | 1 | 0 | SBF is output with 14-bit length |
| 1 | 1 | 1 | SBF is output with 15-bit length |
| 0 | 0 | 0 | SBF is output with 16-bit length |
| 0 | 0 | 1 | SBF is output with 17-bit length |
| 0 | 1 | 0 | SBF is output with 18-bit length |
| 0 | 1 | 1 | SBF is output with 19-bit length |
| 1 | 0 | 0 | SBF is output with 20-bit length |

| UDIR0 | First-bit specification |
|---|---|
| 0 | MSB |
| 1 | LSB |

| TXDLV0 | Enables/disables inverting TXD0 pin output |
|---|---|
| 0 | Normal output of TXD0 pin |
| 1 | Inverted output of TXD0 pin |

**Note** The SBRF0 bit is read-only.

**(7) Selector operation control register 0 (SELCNT0)**

The SELCNT0 register is an 8-bit register that selects the TM01 capture trigger.

If SELCNT0.ISEL00 is set to 1 (RXD0 pin is selected) when LIN is used, the transfer rate for calculating the baud rate error can be checked using TM01.

This register can be read or written in 8-bit or 1-bit units.

After reset, SELCNT0 is cleared to 00H.

After reset: 00H R/W Address: FFFFF308H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| SELCNT0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ISEL00 |

| ISEL00 | Selection of TM01 capture trigger (TM010) |
|---|---|
| 0 | Select TI010 (P35) pin |
| 1 | Select RXD0 (P31) pin |

## 15.4 Interrupt Request Signals

The following three types of interrupt request signals are generated from UARTn.

- Reception error interrupt request signal (INTSREn)
- Reception completion interrupt request signal (INTSRn)
- Transmission completion interrupt request signal (INTSTn)

The default priorities among these three types of interrupt request signals are, from high to low, reception error interrupt, reception completion interrupt, and transmission completion interrupt.

**Table 15-2. Generated Interrupt Request Signals and Default Priorities**

| Interrupt Request Signal | Priority |
|---|---|
| Reception error interrupt request signal (INTSREn) | 1 |
| Reception completion interrupt request signal (INTSRn) | 2 |
| Transmission completion interrupt request signal (INTSTn) | 3 |

**(1) Reception error interrupt request signal (INTSREn)**

When reception is enabled, the INTSREn signal is generated according to the logical OR of the three types of reception errors explained for the ASISn register. Whether the INTSREn signal or the INTSRn signal is generated when an error occurs can be specified according to the ISRMn bit.

When reception is disabled, the INTSREn signal is not generated.

**(2) Reception completion interrupt request signal (INTSRn)**

When reception is enabled, the INTSRn signal is generated when data is shifted in to the receive shift register and transferred to the RXBn register.

The INTSRn signal can be generated in place of the INTSREn signal according to the ISRMn bit even when a reception error has occurred.

When reception is disabled, the INTSRn signal is not generated.

**(3) Transmission completion interrupt request signal (INTSTn)**

The INTSTn signal is generated when one frame of transmit data containing 7-bit or 8-bit characters is shifted out from the transmit shift register.

## 15.5 Operation

### 15.5.1 Data format

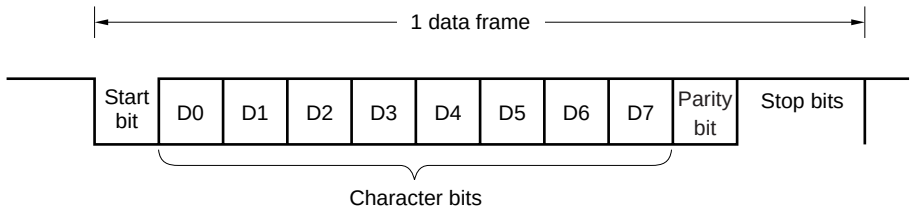
Full-duplex serial data transmission and reception can be performed.

The transmit/receive data format consists of one data frame containing a start bit, character bits, a parity bit, and stop bits as shown in Figure 15-2.

The character bit length within one data frame, the type of parity, and the stop bit length are specified according to the ASIMn register.

Also, data is transferred LSB first.

**Figure 15-2. Format of UARTn Transmit/Receive Data**




- Start bit ··· 1 bit
- Character bits ··· 7 bits or 8 bits
- Parity bit ··· Even parity, odd parity, 0 parity, or no parity
- Stop bits ··· 1 bit or 2 bits

### 15.5.2 Transmit operation

When the ASIMn.UARTEn bit is set to 1, a high level is output from the TXDn pin.

Then, when the ASIMn.TXEn bit is set to 1, transmission is enabled, and the transmit operation is started by writing transmit data to the TXBn register.

**(1) Transmission enabled state**

This state is set by the TXEn bit.

- TXEn bit = 1: Transmission enabled state
- TXEn bit = 0: Transmission disabled state

Since UARTn does not have a CTS (transmission enabled signal) input pin, a port should be used to confirm whether the destination is in a reception enabled state.

**(2) Starting a transmit operation**

In the transmission enabled state, a transmit operation is started by writing transmit data to the TXBn register. When a transmit operation is started, the data in the TXBn register is transferred to the transmit shift register. Then, the transmit shift register outputs data to the TXDn pin (the transmit data is transferred sequentially starting with the start bit). The start bit, parity bit, and stop bits are added automatically.

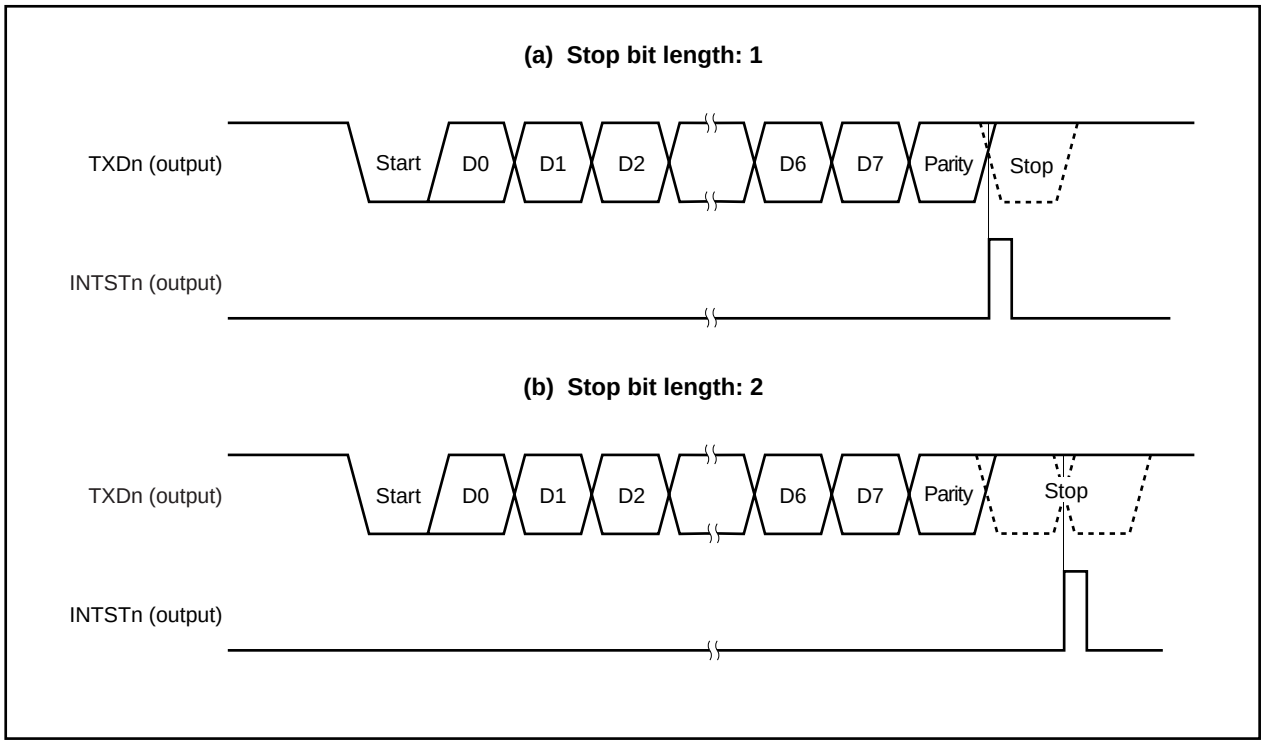
**(3) Transmission interrupt**

When the transmit shift register becomes empty, a transmission completion interrupt request signal (INTSTn) is generated. The timing for generating the INTSTn signal differs according to the specification of the stop bit length. The INTSTn signal is generated at the same time that the last stop bit is output.

If the data to be transmitted next has not been written to the TXBn register, the transmit operation is suspended.

**Caution Normally, when the transmit shift register becomes empty, the INTSTn signal is generated. However, the INTSTn signal is not generated if the transmit shift register becomes empty due to reset.**

**Figure 15-3. UARTn Transmission Completion Interrupt Timing**



### 15.5.3 Continuous transmission operation

UARTn can write the next transmit data to the TXBn register at the timing that the transmit shift register starts the shift operation. This enables an efficient transmission rate to be realized by continuously transmitting data even during the transmission completion interrupt service after the transmission of one data frame. In addition, reading the ASIFn.TXSFn bit after the occurrence of a transmission completion interrupt request signal (INTSTn) enables the TXBn register to be efficiently written twice (2 bytes) without waiting for the transmission of 1 data frame.

When continuous transmission is performed, data should be written after referencing the ASIFn register to confirm the transmission status and whether or not data can be written to the TXBn register.

**Caution The values of the ASIFn.TXBFn and ASIFn.TXSFn bits change 10 → 11 → 01 in continuous transmission. Therefore, do not confirm the status based on the combination of the TXBFn and TXSFn bits.**
**Read only the TXBFn bit during continuous transmission.**

| TXBFn | Whether or Not Writing to TXBn Register Is Enabled |
|---|---|
| 0 | Writing is enabled |
| 1 | Writing is not enabled |

**Caution When transmission is performed continuously, write the first transmit data (first byte) to the TXBn register and confirm that the TXBFn bit is 0, and then write the next transmit data (second byte) to the TXBn register. If writing to the TXBn register is performed when the TXBFn bit is 1, transmit data cannot be guaranteed.**

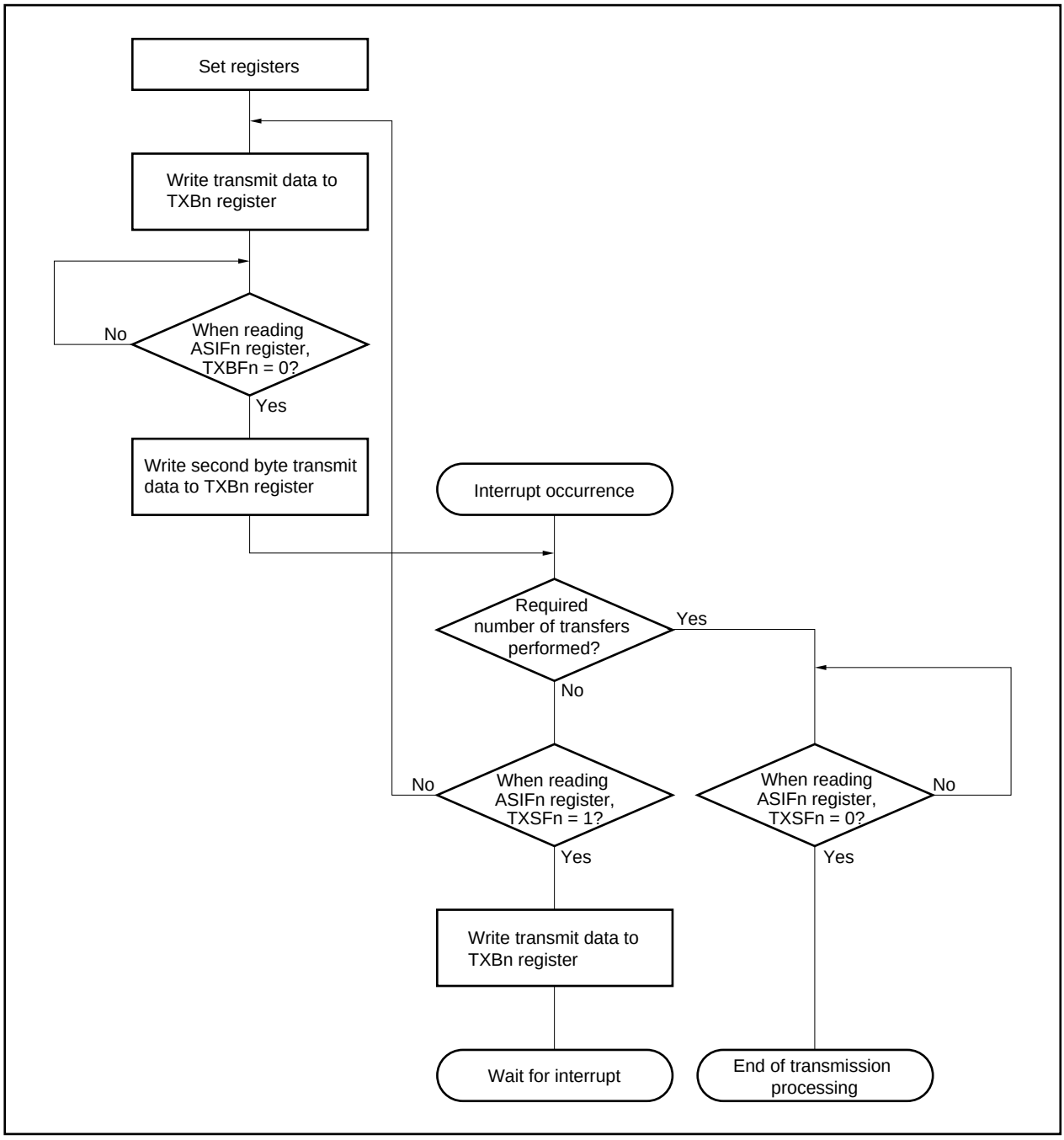
The communication status can be confirmed by referring to the TXSFn bit.

| TXSFn | Transmission Status |
|---|---|
| 0 | Transmission is completed. |
| 1 | Under transmission. |

**Cautions 1. When initializing the transmission unit when continuous transmission is completed, confirm that the TXSFn bit is 0 after the occurrence of the transmission completion interrupt, and then execute initialization. If initialization is performed when the TXSFn bit is 1, transmit data cannot be guaranteed.**

**2. While transmission is being performed continuously, an overrun error may occur if the next transmission is completed before the INTSTn interrupt servicing following the transmission of 1 data frame is executed. An overrun error can be detected by embedding a program that can count the number of transmit data and referencing the TXSFn bit.**

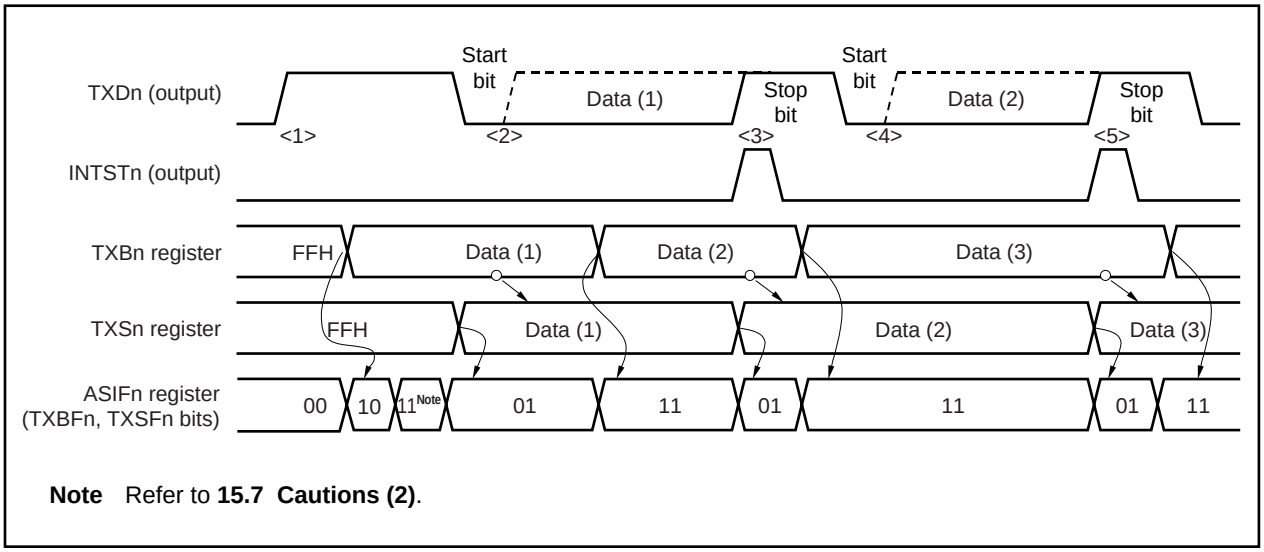
**Figure 15-4. Continuous Transmission Processing Flow**



**(1) Starting procedure**

The procedure to start continuous transmission is shown below.

**Figure 15-5. Continuous Transmission Starting Procedure**




**Note** Refer to **15.7 Cautions (2)**.

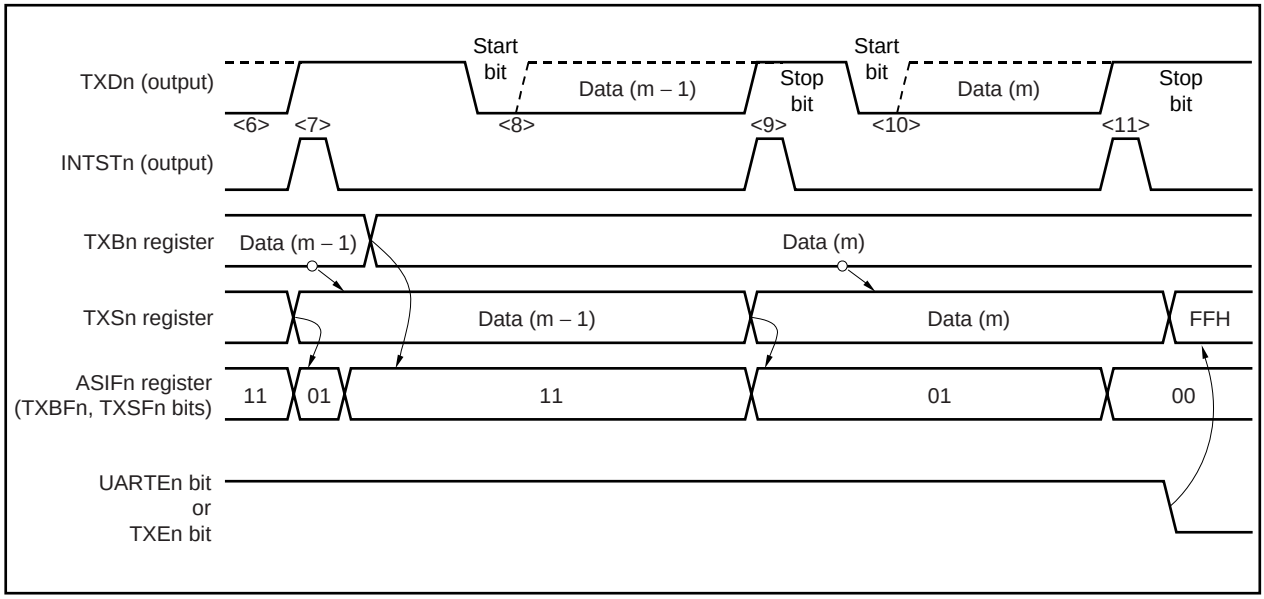
| Transmission Starting Procedure | Internal Operation | ASIFn Register | |
|---|---|---|---|
| | | TXBFn | TXSFn |
| • Set transmission mode | <1> Start transmission unit | 0 | 0 |
| • Write data (1) → | | 1 | 0 |
| | <2> Generate start bit | 1 | 1 Note |
| | | 0 | 1 |
| | Start data (1) transmission → | 0 | 1 |
| • Read ASIFn register (confirm that TXBFn bit = 0) ← | | 0 | 1 |
| • Write data (2) → | | 1 | 1 |
| | <<Transmission in progress>> | | |
| | <3> INTSTn interrupt occurs → | 0 | 1 |
| • Read ASIFn register (confirm that TXBFn bit = 0) ← | | 0 | 1 |
| • Write data (3) → | | 1 | 1 |
| | <4> Generate start bit | | |
| | Start data (2) transmission | | |
| | <<Transmission in progress>> | | |
| | <5> INTSTn interrupt occurs → | 0 | 1 |
| • Read ASIFn register (confirm that TXBFn bit = 0) ← | | 0 | 1 |
| • Write data (4) → | | 1 | 1 |

**Note** Refer to **15.7 Cautions (2)**.

**(2) Ending procedure**

The procedure for ending continuous transmission is shown below.

**Figure 15-6. Continuous Transmission End Procedure**




| Transmission End Procedure | Internal Operation | ASIFn Register | |
|---|---|---|---|
| | | TXBFn | TXSFn |
| | <6> Transmission of data (m – 2) is in progress | 1 | 1 |
| | <7> INTSTn interrupt occurs → | 0 | 1 |
| • Read ASIFn register (confirm that TXBFn bit = 0) ← | | 0 | 1 |
| • Write data (m) → | | 1 | 1 |
| | <8> Generate start bit<br>Start data (m – 1) transmission<br><<Transmission in progress>> | | |
| | <9> INTSTn interrupt occurs → | 0 | 1 |
| • Read ASIFn register (confirm that TXSFn bit = 1) ← | | 0 | 1 |
| There is no write data | | | |
| | <10> Generate start bit<br>Start data (m) transmission<br><<Transmission in progress>> | | |
| | <11> Generate INTSTn interrupt → | 0 | 0 |
| • Read ASIFn register (confirm that TXSFn bit = 0) ← | | 0 | 0 |
| • Clear (0) the UARTEn bit or TXEn bit | Initialize internal circuits | | |

### 15.5.4 Receive operation

The awaiting reception state is set by setting the ASIMn.UARTEn bit to 1 and then setting the ASIMn.RXEn bit to 1. To start the receive operation, start sampling at the falling edge when the falling of the RXDn pin is detected. If the RXDn pin is low level at a start bit sampling point, the start bit is recognized. When the receive operation begins, serial data is stored sequentially in the receive shift register according to the baud rate that was set. A reception completion interrupt request signal (INTSRn) is generated each time the reception of one frame of data is completed. Normally, the receive data is transferred from the RXBn register to memory by this interrupt servicing.

**(1) Reception enabled state**

The receive operation is set to the reception enabled state by setting the RXEn bit to 1.

- RXEn bit = 1: Reception enabled state
- RXEn bit = 0: Reception disabled state

In receive disabled state, the reception hardware stands by in the initial state. At this time, the contents of the RXBn register are retained, and no reception completion interrupt or reception error interrupt is generated.

**(2) Starting a receive operation**

A receive operation is started by the detection of a start bit.

The RXDn pin is sampled using the serial clock from baud rate generator n (BRGn).

**(3) Reception completion interrupt**

When the RXEn bit = 1 and the reception of one frame of data is completed (the stop bit is detected), the INTSRn signal is generated and the receive data within the receive shift register is transferred to the RXBn register at the same time.

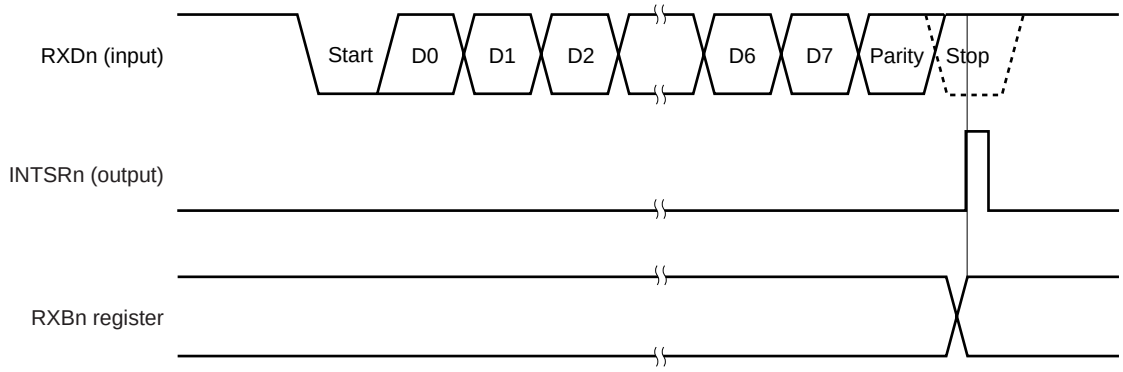
Also, if an overrun error (ASISn.OVEn bit = 1) occurs, the receive data at that time is not transferred to the RXBn register, and either the INTSRn signal or a reception error interrupt request signal (INTSREn) is generated according to the ASIMn.ISRMn bit setting.

Even if a parity error (ASISn.PEn bit = 1) or framing error (ASISn.FEn bit = 1) occurs during a reception operation, the receive operation continues until stop bit is received, and after reception is completed, either the INTSRn signal or the INTSREn signal is generated according to the ISRMn bit setting (the receive data within the receive shift register is transferred to the RXBn register).

If the RXEn bit is cleared (0) during a receive operation, the receive operation is immediately stopped. The contents of the RXBn register and the ASISn register at this time do not change, and the INTSRn signal or the INTSREn signal is not generated.

The INTSRn signal or the INTSREn signal is not generated when the RXEn bit = 0 (reception is disabled).

**Figure 15-7. UARTn Reception Completion Interrupt Timing**




**Cautions 1. Be sure to read the RXBn register even when a reception error occurs. If the RXBn register is not read, an overrun error will occur at the next data reception and the reception error status will continue infinitely.**

**2. Reception is always performed assuming a stop bit length of 1.**
**A second stop bit is ignored.**

### 15.5.5 Reception error

The three types of errors that can occur during a receive operation are a parity error, framing error, and overrun error. As a result of data reception, the various flags of the ASISn register are set (1), and a reception error interrupt request signal (INTSREn) or a reception completion interrupt request signal (INTSRn) is generated at the same time. The ASIMn.ISRMn bit specifies whether the INTSREn signal or the INTSRn signal is generated.

The type of error that occurred during reception can be detected by reading the contents of the ASISn register during the INTSREn or INTSRn interrupt servicing.

The contents of the ASISn register are cleared (0) by reading the ASISn register.

**Table 15-3. Reception Error Causes**

| Error Flag | Reception Error | Cause |
|---|---|---|
| PEn | Parity error | The parity specification during transmission did not match the parity of the reception data |
| FEn | Framing error | No stop bit was detected |
| OVEn | Overrun error | The reception of the next data was completed before data was read from the RXBn register |

**(1) Separation of reception error interrupt request signal**

A reception error interrupt request signal can be separated from the INTSRn signal and generated as the INTSREn signal by clearing the ISRMn bit to 0.

**Figure 15-8. When Reception Error Interrupt Request Signal Is Separated from INTSRn Signal (ISRMn Bit = 0)**

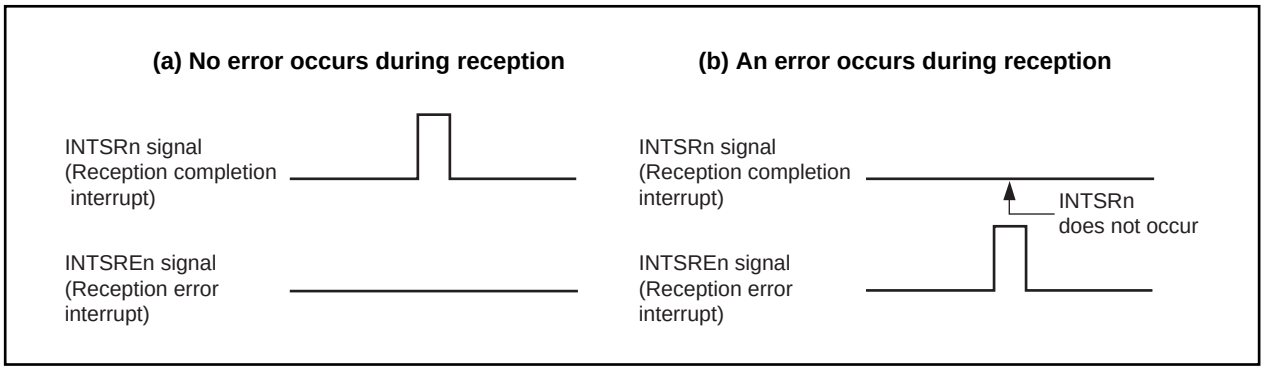
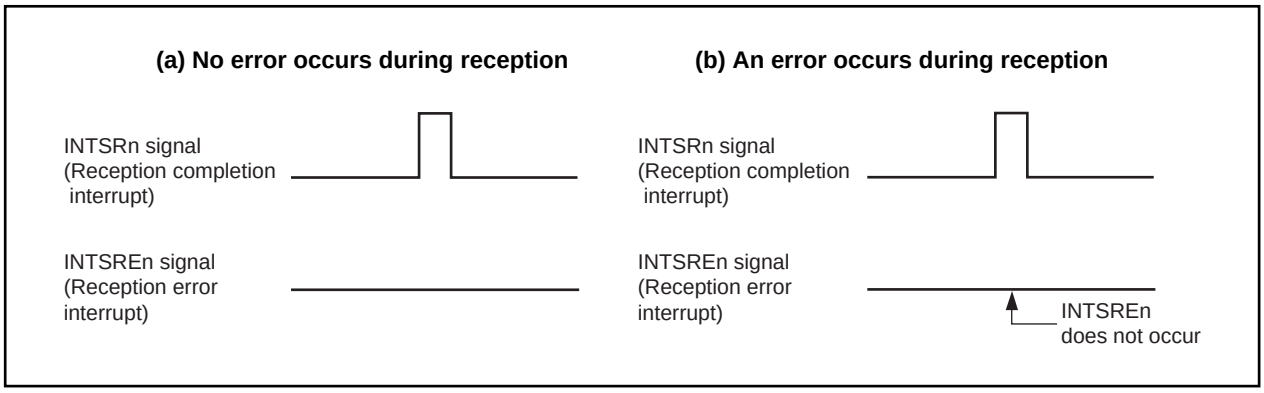



**Figure 15-9. When Reception Error Interrupt Request Signal Is Included in INTSRn Signal (ISRMn Bit = 1)**



### 15.5.6 Parity types and corresponding operation

A parity bit is used to detect a bit error in communication data. Normally, the same type of parity bit is used on the transmission and reception sides.

**(1) Even parity**

**(i) During transmission**

The parity bit is controlled so that the number of bits with the value “1” within the transmit data including the parity bit is even. The parity bit value is as follows.

- If the number of bits with the value “1” within the transmit data is odd: 1
- If the number of bits with the value “1” within the transmit data is even: 0

**(ii) During reception**

The number of bits with the value “1” within the receive data including the parity bit is counted, and a parity error is generated if this number is odd.

**(2) Odd parity**

**(i) During transmission**

In contrast to even parity, the parity bit is controlled so that the number of bits with the value “1” within the transmit data including the parity bit is odd. The parity bit value is as follows.

- If the number of bits with the value “1” within the transmit data is odd: 0
- If the number of bits with the value “1” within the transmit data is even: 1

**(ii) During reception**

The number of bits with the value “1” within the receive data including the parity bit is counted, and a parity error is generated if this number is even.

**(3) 0 parity**

During transmission the parity bit is set to “0” regardless of the transmit data.

During reception, no parity bit check is performed. Therefore, no parity error is generated regardless of whether the parity bit is “0” or “1”.

**(4) No parity**

No parity bit is added to the transmit data.

During reception, the receive operation is performed as if there were no parity bit. Since there is no parity bit, no parity error is generated.

### 15.5.7 Receive data noise filter

The RXDn signal is sampled at the rising edge of the prescaler output base clock ($f_{UCLK}$). If the same sampling value is obtained twice, the match detector output changes, and this output is sampled as input data. Therefore, data not exceeding one clock width is judged to be noise and is not delivered to the internal circuit (refer to **Figure 15-11**). Refer to **15.6.1 (1) Base clock** regarding the base clock.

Also, since the circuit is configured as shown in Figure 15-10, internal processing during a receive operation is delayed by up to 2 clocks according to the external signal status.

**Figure 15-10. Noise Filter Circuit**

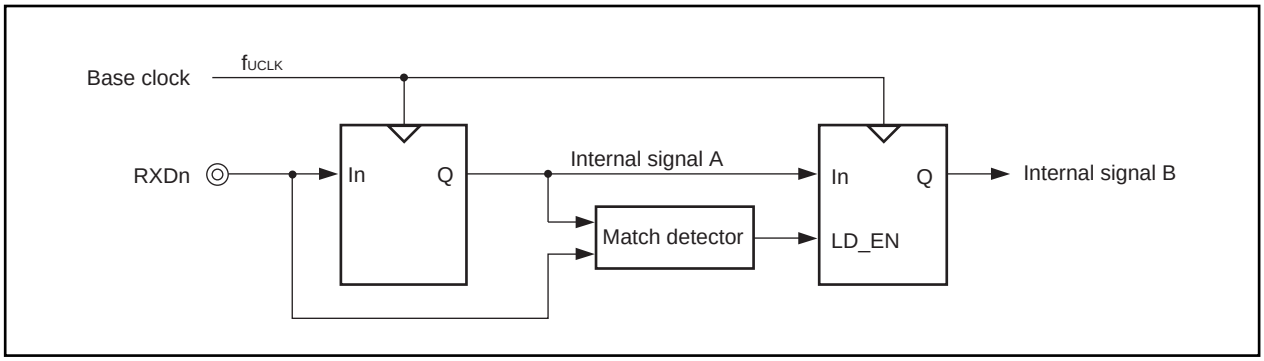
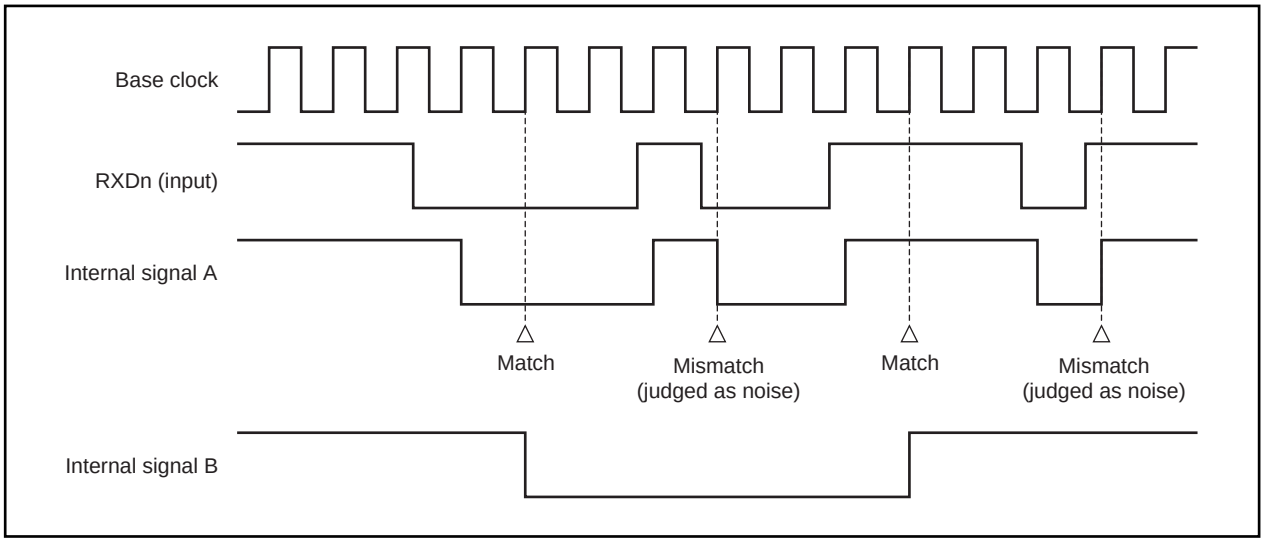



**Figure 15-11. Timing of RXDn Signal Judged as Noise**



### 15.5.8 SBF transmission/reception (UART0 only)

The UART0 of the V850ES/KF1+ has an SBF (Sync Break Field) transmission/reception control function to enable use of the LIN function.

**Remark** LIN stands for Local Interconnect Network and is a low-speed (1 to 20 kbps) serial communication protocol intended to aid the cost reduction of an automotive network.

LIN communication is single-master communication, and up to 15 slaves can be connected to one master.

The LIN slaves are used to control the switches, actuators, and sensors, and these are connected to the LIN master via the LIN network.

Normally, the LIN master is connected to a network such as CAN (Controller Area Network).

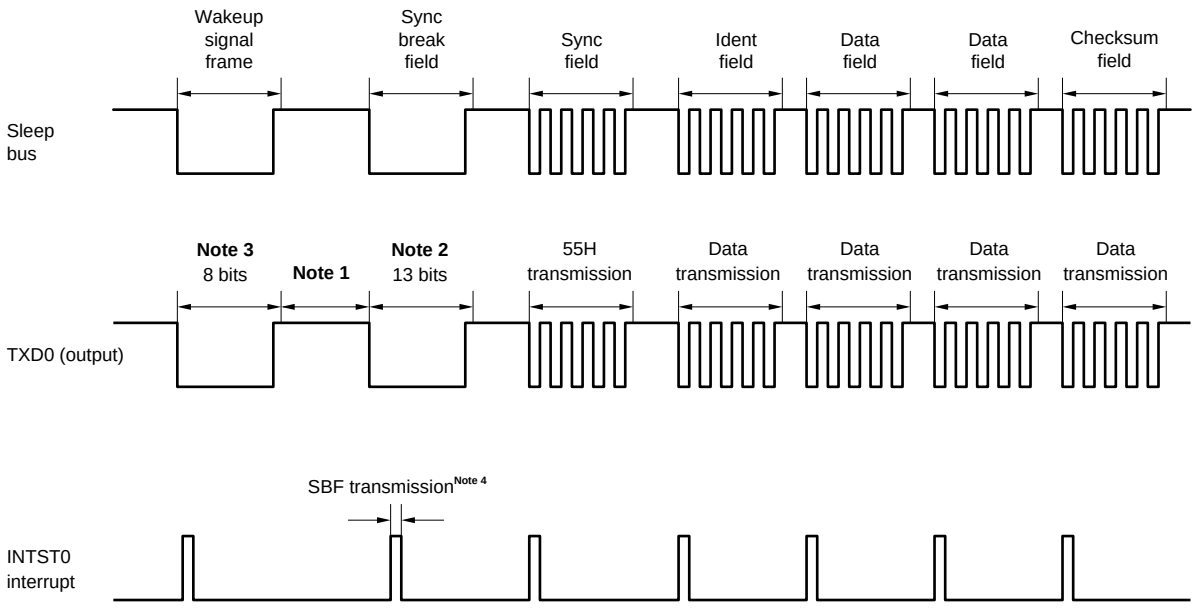
In addition, the LIN bus uses a single-wire method and is connected to the nodes via a transceiver that complies with ISO9141.

In the LIN protocol, the master transmits a frame with baud rate information and the slave receives it and corrects the baud rate error. Therefore, communication is possible when the baud rate error in the slave is ±15% or less.

#### (1) SBF transmission/reception format

Figures 15-12 and 15-13 outline the transmission and reception manipulations of LIN.

**Figure 15-12. LIN Transmission Manipulation Outline**



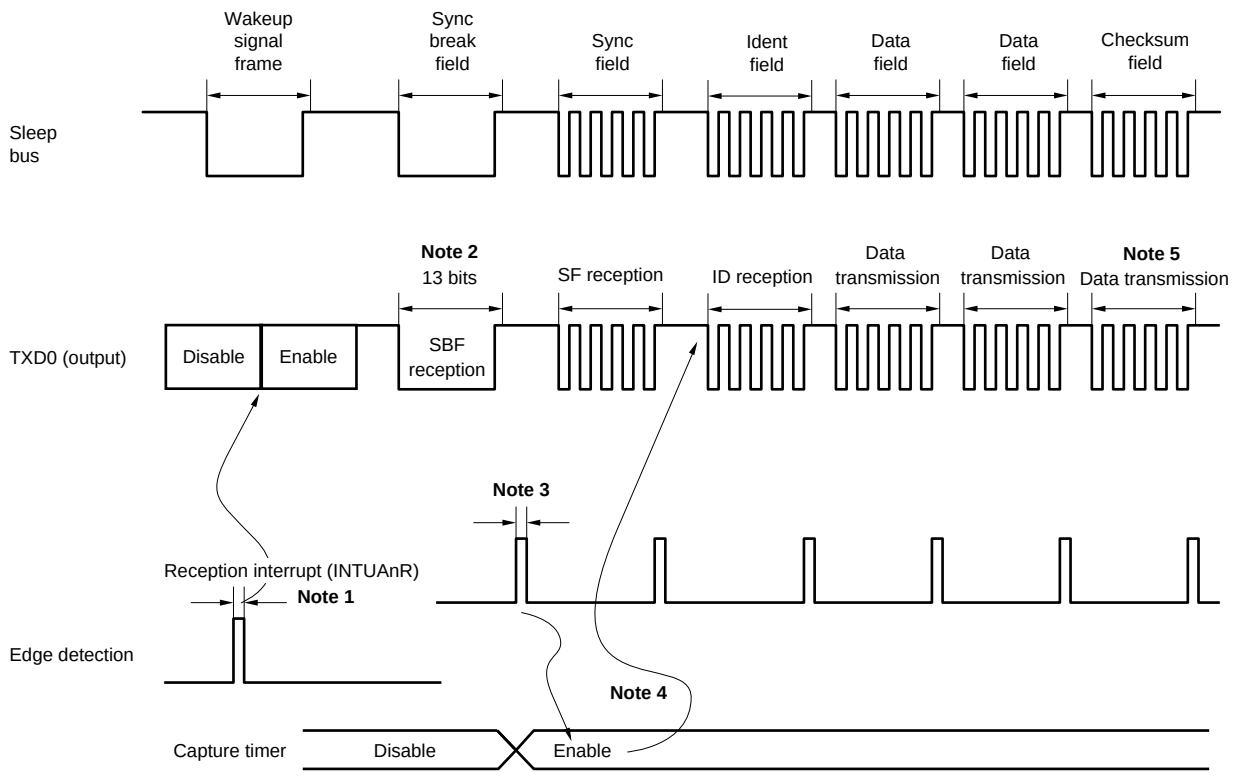

**Notes 1.** The interval between each field is controlled by software.

**2.** SBF output is performed by hardware. The output width is the bit length set by the ASICL0.SBL02 to ASICL0.SBL00 bits. If even finer output width adjustments are required, such adjustments can be performed using the value of BRG (refer to **15.6 Dedicated Baud Rate Generator n (BRGn)**).

**3.** 80H transfer in the 8-bit mode is substituted for the wakeup signal frame.

**4.** A transmission completion interrupt request signal (INTST0) is output at the start of each transmission. The INTST0 signal is also output at the start of each SBF transmission.

**Figure 15-13. LIN Reception Manipulation Outline**




**Notes 1.** The wakeup signal is sent by the pin edge detector, UART0 is enabled, and the SBF reception mode is set.

**2.** The receive operation is performed until detection of the stop bit. Upon detection of SBF reception of 11 or more bits, normal SBF reception end is judged, and an interrupt signal is output. Upon detection of SBF reception of less than 11 bits, an SBF reception error is judged, no interrupt signal is output, and the mode returns to the SBF reception mode.

**3.** If SBF reception ends normally, an interrupt request signal is output. The timer is enabled by an SBF reception completion interrupt. Moreover, error detection for the ASIS0.PE0, ASIS0.FE0, and ASIS0.OVE0 bits is suppressed and UART communication error detection processing and data transfer of the receive shift register and RXB0 register are not performed. The receive shift register holds the initial value, FFH.

**4.** The RXD0 pin is connected to TI (capture input: refer to **15.3 (7) Selector operation control register 0 (SELCNT0)**) of the timer, the transfer rate is calculated, and the baud rate error is calculated. The value of BRG (refer to **15.6 Dedicated Baud Rate Generator n (BRGn)**) obtained by correcting the baud rate error after dropping UART0 enable is set again, causing the status to become the reception status.

**5.** Checksum field distinctions are made by software. UART0 is initialized following CSF reception, and the processing for setting the SBF reception mode again is performed by software.

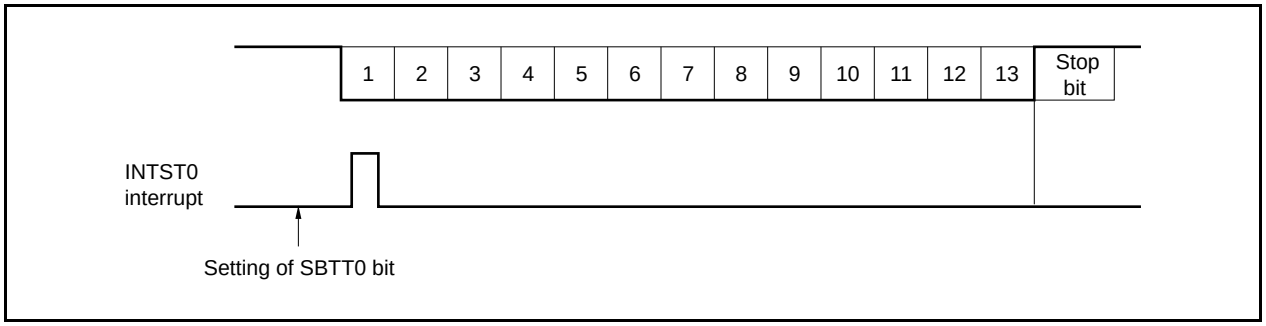
**(2) SBF transmission**

When the ASIM0.UARTE0 bit = ASIM0.TXE0 bit = 1, the transmission enabled status is entered, and SBF transmission is started by setting the SBF transmission trigger (ASICL0.SBRT0 bit) to 1.

Thereafter, a low-level width of bits 13 to 20 specified by the ASICL0.SBL02 to ASICL0.SBL00 bits is output. A transmission completion interrupt request signal (INTST0) is generated upon SBF transmission start. Following the end of SBF transmission, the ASICL0.SBTT0 bit is automatically cleared to 0. Thereafter, the UART transmission mode is restored.

Transmission is suspended until the data to be transmitted next is written to the TXB0 register, or until the SBF transmission trigger (SBTT0 bit) is set to 1.

**Figure 15-14. SBF Transmission**



**(3) SBF reception**

The reception enabled status is achieved by setting the ASIM0.UARTE0 bit to 1 and then setting the ASIM0.RXE0 bit to 1.

The SBF reception wait status is set by setting the SBF reception trigger (ASICL0.SBRT0 bit) to 1.

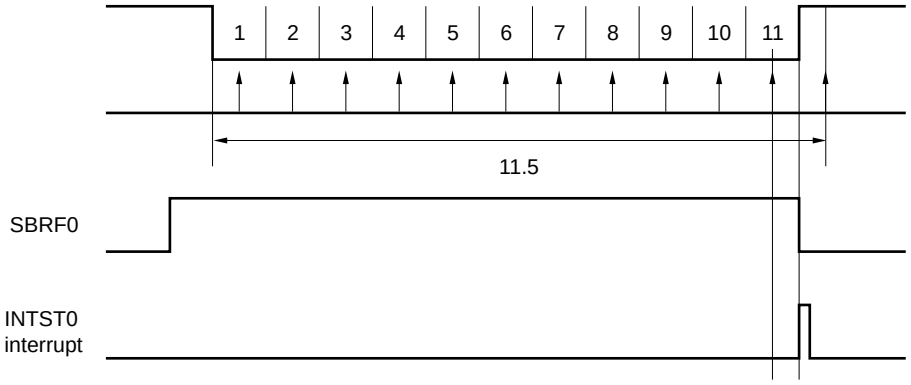
In the SBF reception wait status, similarly to the UART reception wait status, the RXD0 pin is monitored and start bit detection is performed.

Following detection of the start bit, reception is started and the internal counter counts up according to the set baud rate.

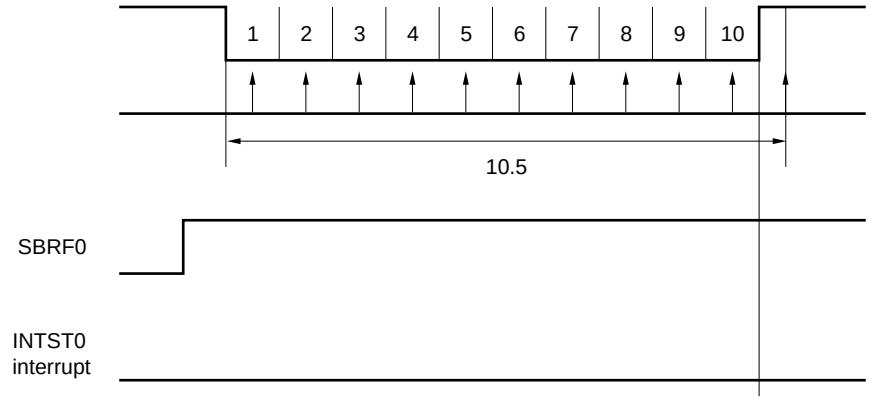
When a stop bit is received, if the SBF width is 11 or more bits, normal processing is judged and a reception completion interrupt request signal (INTSR0) is output. The ASICL0.SBRF0 bit is automatically cleared and SBF reception ends. Error detection for the ASIS0.PE0, ASIS0.FE0, and ASIS0.OVE0 bits is suppressed and UART communication error detection processing is not performed. Moreover, data transfer of the reception shift register and RXB0 register is not performed and FFH, the initial value, is held. If the SBF width is 10 or fewer bits, reception is terminated as error processing without outputting an interrupt, and the SBF reception mode is returned to. The ASICL0.SBRF0 bit is not cleared at this time.

**Figure 15-15. SBF Reception**

**(a) Normal SBF reception (detection of stop bit in more than 10.5 bits)**




**(b) SBF reception error (detection of stop bit in 10.5 or fewer bits)**



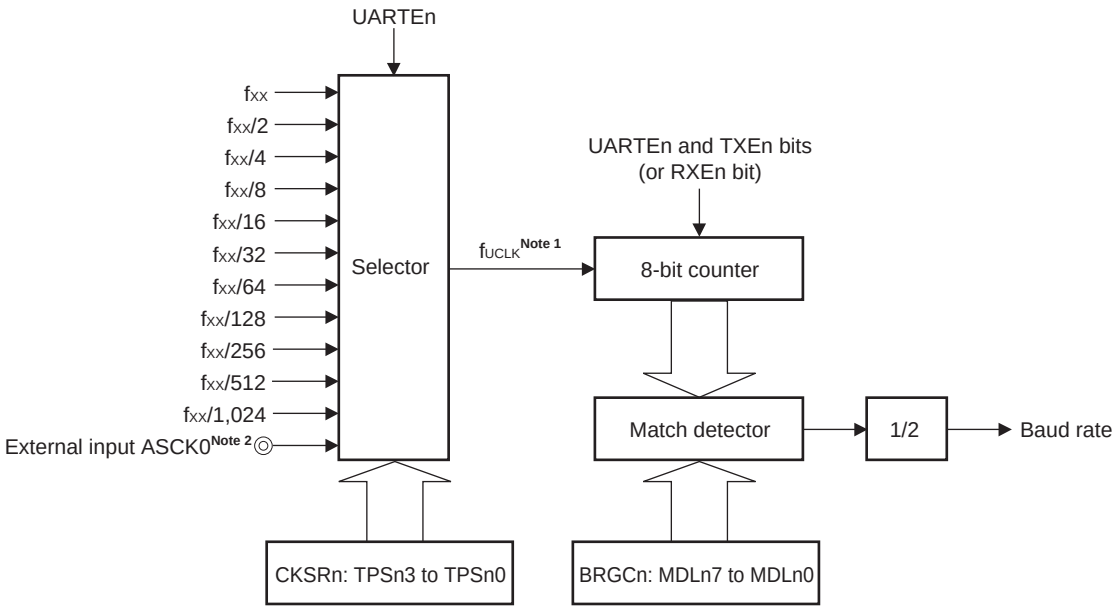
## 15.6 Dedicated Baud Rate Generator n (BRGn)

A dedicated baud rate generator, which consists of a source clock selector and an 8-bit programmable counter, generates serial clocks during transmission/reception by UARTn. The dedicated baud rate generator output can be selected as the serial clock for each channel.

Separate 8-bit counters exist for transmission and for reception.

### 15.6.1 Baud rate generator n (BRGn) configuration

**Figure 15-16. Configuration of Baud Rate Generator n (BRGn)**




**Notes 1.** Set $f_{UCLK}$ so as to satisfy the following conditions.

- $V_{DD}$ = REGC = 4.0 to 5.5 V: $f_{UCLK} \leq 12$ MHz
- $V_{DD}$ = 4.0 to 5.5 V, REGC = Capacity: $f_{UCLK} \leq 6$ MHz
- $V_{DD}$ = REGC = 2.7 to 4.0 V: $f_{UCLK} \leq 6$ MHz

**2.** ASCK0 pin input can be used only by UART0.

**Remark** $f_{XX}$: Main clock frequency

**(1) Base clock**

When the ASIMn.UARTEn bit = 1, the clock selected according to the CKSRn.TPSn3 to CKSRn.TPSn0 bits is supplied to the transmission/reception unit. This clock is called the base clock ($f_{UCLK}$). When the UARTEn bit = 0, $f_{UCLK}$ is fixed to low level.

### 15.6.2 Serial clock generation

A serial clock can be generated according to the settings of the CKSRn and BRGCn registers.

The base clock to the 8-bit counter is selected by the CKSRn.TPSn3 to CKSRn.TPSn0 bits.

The 8-bit counter divisor value can be set by the BRGCn.MDLn7 to BRGCn.MDLn0 bits.

#### (1) Clock select register n (CKSRn)

The CKSRn register is an 8-bit register for selecting the base clock using the TPSn3 to TPSn0 bits. The clock selected by the TPSn3 to TPSn0 bits becomes the base clock ($f_{UCLK}$) of the transmission/reception module.

This register can be read or written in 8-bit units.

After reset, CKSRn is cleared to 00H.

**Caution Clear the ASIMn.UARTEn bit to 0 before rewriting the TPSn3 to TPSn0 bits.**

After reset: 00H R/W Address: CKSR0 FFFFFA06H, CKSR1 FFFFFA16H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CKSRn | 0 | 0 | 0 | 0 | TPSn3 | TPSn2 | TPSn1 | TPSn0 |

(n = 0, 1)

| TPSn3 | TPSn2 | TPSn1 | TPSn0 | Base clock ($f_{UCLK}$)[Note 1] |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | $f_{XX}$ |
| 0 | 0 | 0 | 1 | $f_{XX}/2$ |
| 0 | 0 | 1 | 0 | $f_{XX}/4$ |
| 0 | 0 | 1 | 1 | $f_{XX}/8$ |
| 0 | 1 | 0 | 0 | $f_{XX}/16$ |
| 0 | 1 | 0 | 1 | $f_{XX}/32$ |
| 0 | 1 | 1 | 0 | $f_{XX}/64$ |
| 0 | 1 | 1 | 1 | $f_{XX}/128$ |
| 1 | 0 | 0 | 0 | $f_{XX}/256$ |
| 1 | 0 | 0 | 1 | $f_{XX}/512$ |
| 1 | 0 | 1 | 0 | $f_{XX}/1,024$ |
| 1 | 0 | 1 | 1 | External clock[Note 2] (ASCK0 pin) |
| Other than above | | | | Setting prohibited |

**Notes 1.** Set $f_{UCLK}$ so as to satisfy the following conditions.

- $V_{DD}$ = REGC = 4.0 to 5.5 V: $f_{UCLK} \leq 12$ MHz
- $V_{DD}$ = 4.0 to 5.5 V, REGC = Capacity: $f_{UCLK} \leq 6$ MHz
- $V_{DD}$ = REGC = 2.7 to 4.0 V: $f_{UCLK} \leq 6$ MHz

**2.** ASCK0 pin input clock can be used only by UART0.
Setting of UART1 is prohibited.

**Remark** $f_{XX}$: Main clock frequency

**(2) Baud rate generator control register n (BRGCn)**

The BRGCn register is an 8-bit register that controls the baud rate (serial transfer speed) of UARTn.

This register can be read or written in 8-bit units.

After reset, BRGCn is set to FFH.

**Caution If the MDLn7 to MDLn0 bits are to be overwritten, the ASIMn.TXEn and ASIMn.RXEn bits should be cleared to 0 first.**

After reset: FFH R/W Address: BRGC0 FFFFFA07H, BRGC1 FFFFFA17H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| BRGCn | MDLn7 | MDLn6 | MDLn5 | MDLn4 | MDLn3 | MDLn2 | MDLn1 | MDLn0 |

(n = 0, 1)

| MDLn7 | MDLn6 | MDLn5 | MDLn4 | MDLn3 | MDLn2 | MDLn1 | MDLn0 | Set value (k) | Serial clock |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | × | × | × | – | Setting prohibited |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 8 | $f_{UCLK}/8$ |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 9 | $f_{UCLK}/9$ |
| 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 10 | $f_{UCLK}/10$ |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 250 | $f_{UCLK}/250$ |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 251 | $f_{UCLK}/251$ |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 252 | $f_{UCLK}/252$ |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 253 | $f_{UCLK}/253$ |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 254 | $f_{UCLK}/254$ |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 255 | $f_{UCLK}/255$ |

**Remarks 1.** $f_{UCLK}$: Frequency [Hz] of base clock selected by CKSRn.TPSn3 to CKSRn.TPSn0 bits

**2.** k: Value set by MDLn7 to MDLn0 bits (k = 8, 9, 10, ..., 255)

**3.** The baud rate is the output clock for the 8-bit counter divided by 2.

**4.** ×: don't care

**(3) Baud rate**

The baud rate is the value obtained by the following formula.

$$\text{Baud rate [bps]} = \frac{f_{UCLK}}{2 \times k}$$

$f_{UCLK}$ = Frequency [Hz] of base clock selected by CKSRn.TPSn3 to CKSRn.TPSn0 bits
k = Value set by BRGCn.MDLn7 to BRGCn.MDLn0 bits (k = 8, 9, 10, ..., 255)

**(4) Baud rate error**

The baud rate error is obtained by the following formula.

$$\text{Error (\%)} = \left( \frac{\text{Actual baud rate (baud rate with error)}}{\text{Target baud rate (normal baud rate)}} - 1 \right) \times 100 \text{ [\%]}$$

**Cautions 1. Make sure that the baud rate error during transmission does not exceed the allowable error of the reception destination.**

**2. Make sure that the baud rate error during reception is within the allowable baud rate range during reception, which is described in 15.6.4 Allowable baud rate range during reception.**

**Example:** Base clock frequency = 10 MHz = 10,000,000 Hz
Setting of BRGCn.MDLn7 to BRGCn.MDLn0 bits = 00100001B (k = 33)
Target baud rate = 153,600 bps

Baud rate = 10,000,000/(2 × 33)
= 151,515 [bps]

Error = (151,515/153,600 − 1) × 100
= −1.357 [%]

### 15.6.3 Baud rate setting example

**Table 15-4. Baud Rate Generator Setting Data**

| Baud Rate (bps) | $f_{XX}$ = 20 MHz | | | $f_{XX}$ = 16 MHz | | | $f_{XX}$ = 10 MHz | | |
|---|---|---|---|---|---|---|---|---|---|
| | $f_{UCLK}$ | k | ERR | $f_{UCLK}$ | k | ERR | $f_{UCLK}$ | k | ERR |
| 300 | $f_{XX}$/512 | 41H (65) | 0.16 | $f_{XX}$/1024 | 1AH (26) | 0.16 | $f_{XX}$/256 | 41H (65) | 0.16 |
| 600 | $f_{XX}$/256 | 41H (65) | 0.16 | $f_{XX}$/1024 | 0DH (13) | 0.16 | $f_{XX}$/128 | 41H (65) | 0.16 |
| 1200 | $f_{XX}$/128 | 41H (65) | 0.16 | $f_{XX}$/512 | 0DH (13) | 0.16 | $f_{XX}$/64 | 41H (65) | 0.16 |
| 2400 | $f_{XX}$/64 | 41H (65) | 0.16 | $f_{XX}$/256 | 0DH (13) | 0.16 | $f_{XX}$/32 | 41H (65) | 0.16 |
| 4800 | $f_{XX}$/32 | 41H (65) | 0.16 | $f_{XX}$/128 | 0DH (13) | 0.16 | $f_{XX}$/16 | 41H (65) | 0.16 |
| 9600 | $f_{XX}$/16 | 41H (65) | 0.16 | $f_{XX}$/64 | 0DH (13) | 0.16 | $f_{XX}$/8 | 41H (65) | 0.16 |
| 10400 | $f_{XX}$/64 | 0FH (15) | 0.16 | $f_{XX}$/64 | 0CH (12) | 0.16 | $f_{XX}$/32 | 0FH (15) | 0.16 |
| 19200 | $f_{XX}$/8 | 41H (65) | 0.16 | $f_{XX}$/32 | 0DH (13) | 0.16 | $f_{XX}$/4 | 41H (65) | 0.16 |
| 24000 | $f_{XX}$/32 | 0DH (13) | 0.16 | $f_{XX}$/2 | A7H (167) | –0.20 | $f_{XX}$/16 | 0DH (13) | 0.16 |
| 31250 | $f_{XX}$/32 | 0AH (10) | 0.00 | $f_{XX}$/32 | 08H (8) | 0.00 | $f_{XX}$/16 | 0AH (10) | 0 |
| 33600 | $f_{XX}$/2 | 95H (149) | –0.13 | $f_{XX}$/2 | 77H (119) | 0.04 | $f_{XX}$ | 95H (149) | –0.13 |
| 38400 | $f_{XX}$/4 | 41H (65) | 0.16 | $f_{XX}$/16 | 0DH (13) | 0.16 | $f_{XX}$/2 | 41H (65) | 0.16 |
| 48000 | $f_{XX}$/16 | 0DH (13) | 0.16 | $f_{XX}$/2 | 53H (83) | 0.40 | $f_{XX}$/8 | 0DH (13) | 0.16 |
| 56000 | $f_{XX}$/2 | 59H (89) | 0.32 | $f_{XX}$/2 | 47H (71) | 0.60 | $f_{XX}$ | 59H (89) | 0.32 |
| 62500 | $f_{XX}$/16 | 0AH (10) | 0.00 | $f_{XX}$/16 | 08H (8) | 0.00 | $f_{XX}$/8 | 0AH (10) | 0.00 |
| 76800 | $f_{XX}$/2 | 41H (65) | 0.16 | $f_{XX}$/8 | 0DH (13) | 0.16 | $f_{XX}$ | 41H (65) | 0.16 |
| 115200 | $f_{XX}$/2 | 2BH (43) | 0.94 | $f_{XX}$/2 | 23H (35) | –0.79 | $f_{XX}$ | 2BH (43) | 0.94 |
| 153600 | $f_{XX}$/2 | 21H (33) | –1.36 | $f_{XX}$/4 | 0DH (13) | 0.16 | $f_{XX}$ | 21H (33) | –1.36 |
| 312500 | $f_{XX}$/4 | 08H (8) | 0 | $f_{XX}$/2 | 0DH (13) | –1.54 | $f_{XX}$/2 | 08H (8) | 0.00 |

**Caution The allowable frequency of the base clock ($f_{UCLK}$) is as follows.**

- **$V_{DD}$ = REGC = 4.0 to 5.5 V: $f_{UCLK} \leq$ 12 MHz**
- **$V_{DD}$ = 4.0 to 5.5 V, REGC = Capacity: $f_{UCLK} \leq$ 6 MHz**
- **$V_{DD}$ = REGC = 2.7 to 4.0 V: $f_{UCLK} \leq$ 6 MHz**

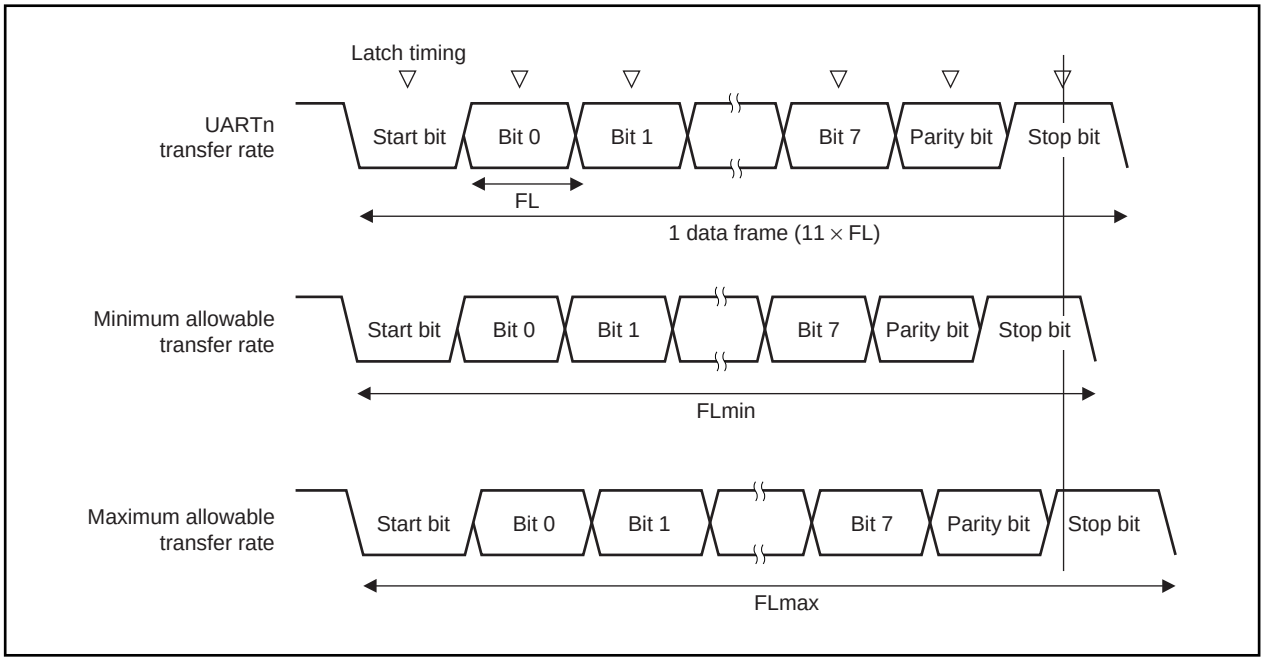
**Remark** $f_{XX}$: Main clock frequency
$f_{UCLK}$: Base clock frequency
k: Set values of BRGCn.MDLn7 to BRGCn.MDLn0 bits
ERR: Baud rate error [%]
n = 0, 1

### 15.6.4 Allowable baud rate range during reception

The degree to which a discrepancy from the transmission destination's baud rate is allowed during reception is shown below.

**Caution The equations described below should be used to set the baud rate error during reception so that it always is within the allowable error range.**

**Figure 15-17. Allowable Baud Rate Range During Reception**




As shown in Figure 15-17, after the start bit is detected, the receive data latch timing is determined according to the counter that was set by the BRGCn register. If all data up to the final data (stop bit) is in time for this latch timing, the data can be received normally.
If this is applied to 11-bit reception, the following is theoretically true.

$$FL = (Brate)^{-1}$$

Brate: UARTn baud rate
k: BRGCn register set value
FL: 1-bit data length

When the latch timing margin is 2 base clocks, the minimum allowable transfer rate (FLmin) is as follows.

$$FLmin = 11 \times FL - \frac{k-2}{2k} \times FL = \frac{21k+2}{2k} FL$$

Therefore, the transfer destination's maximum receivable baud rate (BRmax) is as follows.

$$BRmax = (FLmin/11)^{-1} = \frac{22k}{21k + 2} \text{ Brate}$$

Similarly, the maximum allowable transfer rate (FLmax) can be obtained as follows.

$$\frac{10}{11} \times FLmax = 11 \times FL - \frac{k + 2}{2 \times k} \times FL = \frac{21k - 2}{2 \times k} FL$$

$$FLmax = \frac{21k - 2}{20k} FL \times 11$$

Therefore, the transfer destination's minimum receivable baud rate (BRmin) is as follows.

$$BRmin = (FLmax/11)^{-1} = \frac{20k}{21k - 2} \text{ Brate}$$

The allowable baud rate error of UARTn and the transfer destination can be obtained as follows from the expressions described above for computing the minimum and maximum baud rate values.

**Table 15-5. Maximum and Minimum Allowable Baud Rate Error**

| Division Ratio (k) | Maximum Allowable Baud Rate Error | Minimum Allowable Baud Rate Error |
|---|---|---|
| 8 | +3.53% | –3.61% |
| 20 | +4.26% | –4.31% |
| 50 | +4.56% | –4.58% |
| 100 | +4.66% | –4.67% |
| 255 | +4.72% | –4.73% |

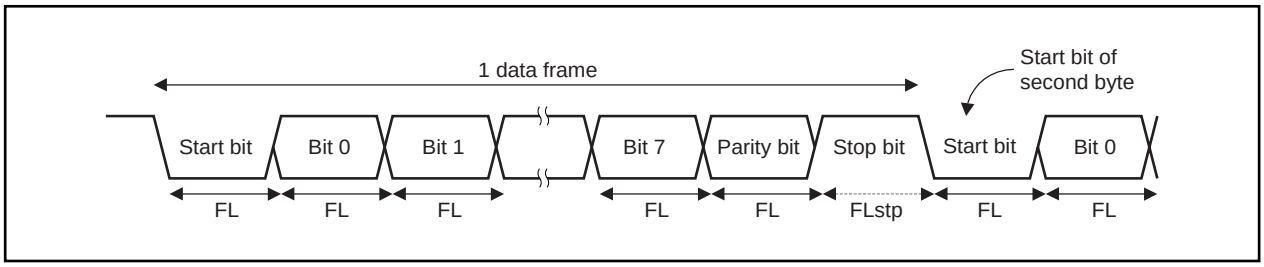
**Remarks 1.** The reception precision depends on the number of bits in one frame, the base clock frequency, and the division ratio (k). The higher the base clock frequency and the larger the division ratio (k), the higher the precision.

**2.** k: BRGCn register set value

### 15.6.5 Transfer rate during continuous transmission

During continuous transmission, the transfer rate from a stop bit to the next start bit is extended two clocks of the base clock longer than normal. However, on the reception side, the transfer result is not affected since the timing is initialized by the detection of the start bit.

**Figure 15-18. Transfer Rate During Continuous Transmission**




Representing the 1-bit data length by FL, the stop bit length by FLstp, and the base clock frequency by $f_{UCLK}$ yields the following equation.

$$FLstp = FL + 2/f_{UCLK}$$

Therefore, the transfer rate during continuous transmission is as follows (when the stop bit length = 1).

$$\text{Transfer rate} = 11 \times FL + (2/f_{UCLK})$$

## 15.7 Cautions

Cautions to be observed when using UARTn are shown below.

(1) When the supply of clocks to UARTn is stopped (for example, in IDLE or STOP mode), operation stops with each register retaining the value it had immediately before the supply of clocks was stopped. The TXDn pin output also holds and outputs the value it had immediately before the supply of clocks was stopped. However, operation is not guaranteed after the supply of clocks is restarted. Therefore, after the supply of clocks is restarted, the circuits should be initialized by clearing the ASIMn.UARTEn, ASIMn.RXEn, and ASIMn.TXEn bits to 000.

(2) UARTn has a 2-stage buffer configuration consisting of the TXBn register and the transmission shift register, and has status flags (ASIFn.TXBFn and ASIFn.TXSFn bits) that indicate the status of each buffer. If the TXBFn and TXSFn bits are read in continuous transmission, the value changes 10 → 11 → 01. For the timing to write the next data to the TXBn register, read only the TXBFn bit during continuous transmission.

# CHAPTER 16 CLOCKED SERIAL INTERFACE 0 (CSI0)

In the V850ES/KF1+, two channels of clocked serial interface 0 (CSI0) are provided.

## 16.1 Features

- Maximum transfer speed: 5 Mbps
- Master mode/slave mode selectable
- Transmission data length: 8 bits or 16 bits can be set
- MSB/LSB-first selectable for transfer data
- Eight clock signals can be selected (7 master clocks and 1 slave clock)
- 3-wire type SO0n: Serial transmit data output
  SI0n: Serial receive data input
  $\overline{\text{SCK0n}}$: Serial clock I/O
- Interrupt sources: 1 type
  - Transmission/reception completion interrupt request signal (INTCSI0n)
- Transmission/reception mode or reception-only mode selectable
- Two transmission buffer registers (SOTBFn/SOTBFLn, SOTBn/SOTBLn) and two reception buffer registers (SIRBn/SIRBLn, SIRBEn/SIRBELn) are provided on chip
- Single transfer mode/continuous transfer mode selectable

**Remark** n = 0, 1

## 16.2 Configuration

CSI0n is controlled via the CSIM0n register.

**(1) Clocked serial interface mode register 0n (CSIM0n)**

The CSIM0n register is an 8-bit register that specifies the operation of CSI0n.

**(2) Clocked serial interface clock selection register n (CSICn)**

The CSICn register is an 8-bit register that controls the CSI0n serial transfer operation.

**(3) Serial I/O shift register 0n (SIO0n)**

The SIO0n register is a 16-bit shift register that converts parallel data into serial data.
The SIO0n register is used for both transmission and reception.
Data is shifted in (reception) and shifted out (transmission) from the MSB or LSB side.
The actual transmission/reception operations are started up by accessing the buffer register.

**(4) Serial I/O shift register 0nL (SIO0nL)**

The SIO0nL register is an 8-bit shift register that converts parallel data into serial data.
The SIO0nL register is used for both transmission and reception.
Data is shifted in (reception) and shifted out (transmission) from the MSB or LSB side.
The actual transmission/reception operations are started up by accessing the buffer register.

**(5) Clocked serial interface receive buffer register n (SIRBn)**

The SIRBn register is a 16-bit buffer register that stores receive data.

**(6) Clocked serial interface receive buffer register nL (SIRBnL)**

The SIRBnL register is an 8-bit buffer register that stores receive data.

**(7) Clocked serial interface read-only receive buffer register n (SIRBEn)**

The SIRBEn register is a 16-bit buffer register that stores receive data.
The SIRBEn register is the same as the SIRBn register. It is used to read the contents of the SIRBn register.

**(8) Clocked serial interface read-only receive buffer register nL (SIRBEnL)**

The SIRBEnL register is an 8-bit buffer register that stores receive data.
The SIRBEnL register is the same as the SIRBnL register. It is used to read the contents of the SIRBnL register.

**(9) Clocked serial interface transmit buffer register n (SOTBn)**

The SOTBn register is a 16-bit buffer register that stores transmit data.

**(10) Clocked serial interface transmit buffer register nL (SOTBnL)**

The SOTBnL register is an 8-bit buffer register that stores transmit data.

**(11) Clocked serial interface initial transmit buffer register n (SOTBFn)**

The SOTBFn register is a 16-bit buffer register that stores the initial transmit data in the continuous transfer mode.

**(12) Clocked serial interface initial transmit buffer register nL (SOTBFnL)**

The SOTBFnL register is an 8-bit buffer register that stores initial transmit data in the continuous transfer mode.

**(13) Selector**

The selector selects the serial clock to be used.

**(14) Serial clock controller**

Controls the serial clock supply to the shift register. Also controls the clock output to the $\overline{\text{SCK0n}}$ pin when the internal clock is used.

**(15) Serial clock counter**

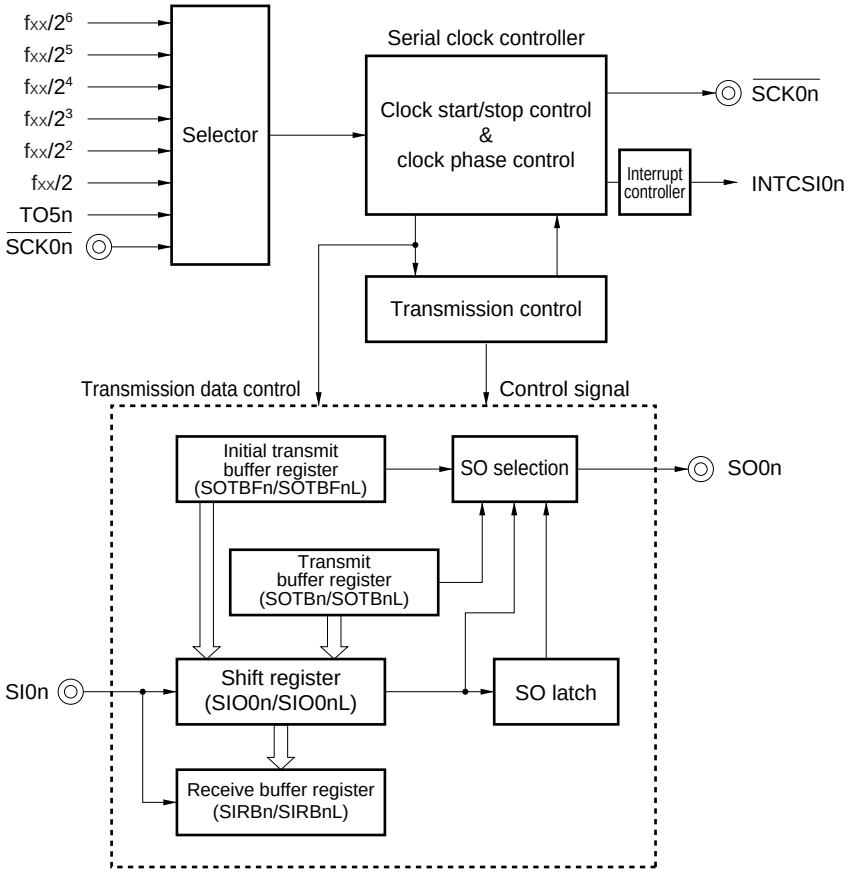
Counts the serial clock output or input during transmission/reception, and checks whether 8-bit or 16-bit data transmission/reception has been performed.

**(16) Interrupt controller**

Controls the interrupt request timing.

**Remark** n = 0, 1

**Figure 16-1. Block Diagram of Clocked Serial Interface**




**Remarks 1.** n = 0, 1

**2.** fxx: Main clock frequency

## 16.3 Registers

**(1) Clocked serial interface mode register 0n (CSIM0n)**

The CSIM0n register controls the CSI0n operation.

This register can be read or written in 8-bit or 1-bit units (however, CSOTn bit is read-only).

After reset, CSIM0n is cleared to 00H.

**Caution Overwriting the TRMDn, CCLn, DIRn, CSITn, and AUTOn bits can be done only when the CSOTn bit = 0. If these bits are overwritten when the CSOTn bit = 1, the operation cannot be guaranteed.**

After reset: 00H R/W Address: CSIM00 FFFFFD00H, CSIM01 FFFFFD10H

| | <7> | <6> | 5 | <4> | 3 | 2 | 1 | <0> |
|---|---|---|---|---|---|---|---|---|
| CSIM0n | CSI0En | TRMDn | CCLn | DIRn | CSITn | AUTOn | 0 | CSOTn |

(n = 0, 1)

| CSI0En | CSI0n operation enable/disable |
|---|---|
| 0 | Disable CSI0n operation. |
| 1 | Enable CSI0n operation. |
| The internal CSI0n circuit can be reset[Note] asynchronously by clearing the CSI0En bit to 0. For the $\overline{\text{SCK0n}}$ and SO0n pin output status when the CSI0En bit = 0, refer to **16.5 Output Pins**. | |

| TRMDn | Specification of transmission/reception mode |
|---|---|
| 0 | Receive-only mode |
| 1 | Transmission/reception mode |
| When the TRMDn bit = 0, reception is performed and the SO0n pin outputs a low level. Data reception is started by reading the SIRBn register.<br>When the TRMDn bit = 1, transmission/reception is started by writing data to the SOTBn register. | |

| CCLn | Specification of data length |
|---|---|
| 0 | 8 bits |
| 1 | 16 bits |

| DIRn | Specification of transfer direction mode (MSB/LSB) |
|---|---|
| 0 | First bit of transfer data is MSB |
| 1 | First bit of transfer data is LSB |

| CSITn | Control of delay of interrupt request signal |
|---|---|
| 0 | No delay |
| 1 | Delay mode (interrupt request signal is delayed 1/2 cycle compared to the serial clock) |
| The delay mode (CSITn bit = 1) is valid only in the master mode (CSICn.CKS0n2 to CSICn.CKS0n0 bits are not 111B). In the slave mode (CKS0n2 to CKS0n0 bits are 111B), do not set the delay mode. | |

| AUTOn | Specification of single transfer mode or continuous transfer mode |
|---|---|
| 0 | Single transfer mode |
| 1 | Continuous transfer mode |

| CSOTn | Communication status flag |
|---|---|
| 0 | Communication stopped |
| 1 | Communication in progress |
| The CSOTn bit is cleared (0) by writing 0 to the CSI0En bit. | |

**Note** The CSOTn bit and the SIRBn, SIRBnL, SIRBE, SIRBEnL, SIOn, and SIOnL registers are reset.

**Remark** n = 0, 1

**(2) Clocked serial interface clock selection register n (CSICn)**

The CSICn register is an 8-bit register that controls the CSI0n transfer operation.

This register can be read or written in 8-bit or 1-bit units.

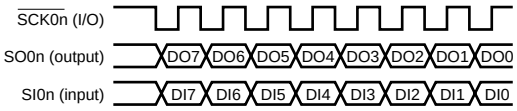
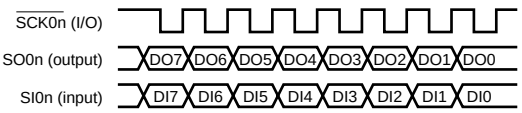
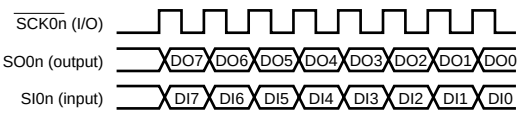
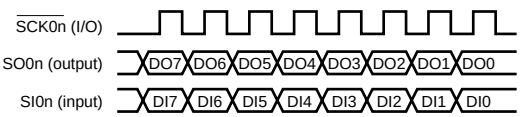
After reset, CSICn is cleared to 00H.

**Caution The CSICn register can be overwritten only when the CSIM0n.CSI0En bit = 0.**

After reset: 00H R/W Address: CSIC0 FFFFFD01H, CSIC1 FFFFFD11H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CSICn | 0 | 0 | 0 | CKPn | DAPn | CKS0n2 | CKS0n1 | CKS0n0 |

(n = 0, 1)

| CKPn | DAPn | Specification of timing of transmitting/receiving data to/from $\overline{SCK0n}$ |
|---|---|---|
| 0 | 0 | (Type 1)   |
| 0 | 1 | (Type 2)   |
| 1 | 0 | (Type 3)   |
| 1 | 1 | (Type 4)   |

| CKS0n2 | CKS0n1 | CKS0n0 | Serial clock[Note] | Mode |
|---|---|---|---|---|
| 0 | 0 | 0 | $f_{XX}/2$ | Master mode |
| 0 | 0 | 1 | $f_{XX}/2^2$ | Master mode |
| 0 | 1 | 0 | $f_{XX}/2^3$ | Master mode |
| 0 | 1 | 1 | $f_{XX}/2^4$ | Master mode |
| 1 | 0 | 0 | $f_{XX}/2^5$ | Master mode |
| 1 | 0 | 1 | $f_{XX}/2^6$ | Master mode |
| 1 | 1 | 0 | Clock generated by TO5n | Master mode |
| 1 | 1 | 1 | External clock ($\overline{SCK0n}$ pin) | Slave mode |

**Note** Set the serial clock so as to satisfy the following conditions.

- $V_{DD}$ = REGC = 4.0 to 5.5 V: Serial clock ≤ 5 MHz
- $V_{DD}$ = 4.0 to 5.5 V, REGC = Capacity: Serial clock ≤ 2.5 MHz
- $V_{DD}$ = REGC = 2.7 to 4.0 V: Serial clock ≤ 2.5 MHz

**Remark** $f_{XX}$: Main clock frequency

**(3) Clocked serial interface receive buffer registers n, nL (SIRBn, SIRBnL)**

The SIRBn register is a 16-bit buffer register that stores receive data.

When the receive-only mode is set (CSIM0n.TRMDn bit = 0), the reception operation is started by reading data from the SIRBn register.

This register is read-only in 16-bit units. When the lower 8 bits are used as the SIRBnL register, this register is read-only in 8-bit units.

In addition to reset input, this register is also cleared to 0000H by clearing (0) the CSIM0n.CSI0En bit.

**Cautions 1. Read the SIRBn register only when a 16-bit data length has been set (CSIM0n.CCLn bit = 1).**

**Read the SIRBnL register only when an 8-bit data length has been set (CCLn bit = 0).**

**2. When the single transfer mode has been set (CSIM0n.AUTOn bit = 0), perform a read operation only in the idle state (CSIM0n.CSOTn bit = 0). If the SIRBn or SIRBnL register is read during data transfer, the data cannot be guaranteed.**

**(a) SIRBn register**

After reset: 0000H R Address: SIRB0 FFFFFD02H, SIRB1 FFFFFD12H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SIRBn | SIRBn | SIRBn | SIRBn | SIRBn | SIRBn | SIRBn | SIRBn | SIRBn | SIRBn | SIRBn | SIRBn | SIRBn | SIRBn | SIRBn | SIRBn | SIRBn |
| (n = 0, 1) | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

**(b) SIRBnL register**

After reset: 00H R Address: SIRB0L FFFFFD02H, SIRB1L FFFFFD12H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| SIRBnL | SIRBn7 | SIRBn6 | SIRBn5 | SIRBn4 | SIRBn3 | SIRBn2 | SIRBn1 | SIRBn0 |
| (n = 0, 1) | | | | | | | | |

**(4) Clocked serial interface read-only receive buffer registers n, nL (SIRBEn, SIRBEnL)**

The SIRBEn register is a 16-bit buffer register that stores receive data.

The SIRBEn register is the same as the SIRBn register. Even if the SIRBEn register is read, the next operation will not start. The SIRBEn register is used to read the contents of the SIRBn register when the serial reception is not continued.

This register is read-only in 16-bit units. However, when the lower 8 bits are used as the SIRBEnL register, the register is read-only in 8-bit units.

In addition to reset input, this register is also cleared to 0000H by clearing (0) the CSIM0n.CSI0En bit.

**Cautions 1. The receive operation is not started even if data is read from the SIRBEn and SIRBEnL registers.**

**2. The SIRBEn register can be read only if a 16-bit data length has been set (CSIM0n.CCLn bit = 1).**

**The SIRBEnL register can be read only if an 8-bit data length has been set (CCLn bit = 0).**

**(a) SIRBEn register**

After reset: 0000H R Address: SIRBE0 FFFFFD06H, SIRBE1 FFFFFD16H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SIRBEn | SIRBEn | SIRBEn | SIRBEn | SIRBEn | SIRBEn | SIRBEn | SIRBEn | SIRBEn | SIRBEn | SIRBEn | SIRBEn | SIRBEn | SIRBEn | SIRBEn | SIRBEn | SIRBEn |
| (n = 0, 1) | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

**(b) SIRBEnL register**

After reset: 00H R Address: SIRBE0L FFFFFD06H, SIRBE1L FFFFFD16H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| SIRBEnL | SIRBEn7 | SIRBEn6 | SIRBEn5 | SIRBEn4 | SIRBEn3 | SIRBEn2 | SIRBEn1 | SIRBEn0 |
| (n = 0, 1) | | | | | | | | |

**(5) Clocked serial interface transmit buffer registers n, nL (SOTBn, SOTBnL)**

The SOTBn register is a 16-bit buffer register that stores transmit data.

When the transmission/reception mode is set (CSIM0n.TRMDn bit = 1), the transmission operation is started by writing data to the SOTBn register.

This register can be read or written in 16-bit units. However, when the lower 8 bits are used as the SOTBnL register, the register can be read or written in 8-bit units.

After reset, this register is cleared to 0000H.

**Cautions 1. Access the SOTBn register only when a 16-bit data length has been set (CSIM0n.CCLn bit = 1).**

**Access the SOTBnL register only when an 8-bit data length has been set (CCLn bit = 0).**

**2. When the single transfer mode is set (CSIM0n.AUTOn bit = 0), perform access only in the idle state (CSIM0n.CSOTn bit = 0). If the SOTBn and SOTBnL registers are accessed during data transfer, the data cannot be guaranteed.**

**(a) SOTBn register**

After reset: 0000H R/W Address: SOTB0 FFFFFD04H, SOTB1 FFFFFD14H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SOTBn | SOTBn | SOTBn | SOTBn | SOTBn | SOTBn | SOTBn | SOTBn | SOTBn | SOTBn | SOTBn | SOTBn | SOTBn | SOTBn | SOTBn | SOTBn | SOTBn |
| (n = 0, 1) | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

**(b) SOTBnL register**

After reset: 00H R/W Address: SOTB0L FFFFFD04H, SOTB1L FFFFFD14H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| SOTBnL | SOTBn7 | SOTBn6 | SOTBn5 | SOTBn4 | SOTBn3 | SOTBn2 | SOTBn1 | SOTBn0 |
| (n = 0, 1) | | | | | | | | |

**(6) Clocked serial interface initial transmit buffer registers n, nL (SOTBFn, SOTBFnL)**

The SOTBFn register is a 16-bit buffer register that stores initial transmission data in the continuous transfer mode.

The transmission operation is not started even if data is written to the SOTBFn register.

This register can be read or written in 16-bit units. However, when the lower 8 bits are used as the SOTBFnL register, the register can be read or written in 8-bit units.

After reset, this register is cleared to 0000H.

**Caution Access the SOTBFn register and SOTBFnL register only when a 16-bit data length has been set (CSIM0n.CCLn bit = 1), and only when an 8-bit data length has been set (CCLn bit = 0), respectively, and only in the idle state (CSIM0n.CSOTn bit = 0). If the SOTBFn and SOTBFnL registers are accessed during data transfer, the data cannot be guaranteed.**

**(a) SOTBFn register**

After reset: 0000H R/W Address: SOTBF0 FFFFFD08H, SOTBF1 FFFFFD18H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SOTBFn | SOTBFn 15 | SOTBFn 14 | SOTBFn 13 | SOTBFn 12 | SOTBFn 11 | SOTBFn 10 | SOTBFn 9 | SOTBFn 8 | SOTBFn 7 | SOTBFn 6 | SOTBFn 5 | SOTBFn 4 | SOTBFn 3 | SOTBFn 2 | SOTBFn 1 | SOTBFn 0 |
| (n = 0, 1) | | | | | | | | | | | | | | | | |

**(b) SOTBFnL register**

After reset: 00H R/W Address: SOTBF0L FFFFFD08H, SOTBF1L FFFFFD18H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| SOTBFnL | SOTBFn7 | SOTBFn6 | SOTBFn5 | SOTBFn4 | SOTBFn3 | SOTBFn2 | SOTBFn1 | SOTBFn0 |
| (n = 0, 1) | | | | | | | | |

**(7) Serial I/O shift registers n, nL (SIO0n, SIO0nL)**

The SIO0n register is a 16-bit shift register that converts parallel data into serial data.

The transfer operation is not started even if the SIO0n register is read.

This register is read-only in 16-bit units. However, when the lower 8 bits are used as the SIO0nL register, the register is read-only in 8-bit units.

In addition to reset input, this register is also cleared to 0000H by clearing (0) the CSIM0n.CSI0En bit.

**Caution Read the SIO0n register and SIO0nL register only when a 16-bit data length has been set (CSIM0n.CCLn bit = 1), and only when an 8-bit data length has been set (CCLn bit = 0), respectively, and only in the idle state (CSIM0n.CSOTn bit = 0). If the SIO0n and SIO0nL registers are read during data transfer, the data cannot be guaranteed.**

**(a) SIO0n register**

After reset: 0000H R Address: SIO00 FFFFFD0AH, SIO01 FFFFFD1AH

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SIO0n | SIOn15 | SIOn14 | SIOn13 | SIOn12 | SIOn11 | SIOn10 | SIOn9 | SIOn8 | SIOn7 | SIOn6 | SIOn5 | SIOn4 | SIOn3 | SIOn2 | SIOn1 | SIOn0 |

(n = 0, 1)

**(b) SIO0nL register**

After reset: 00H R Address: SIO00L FFFFFD0AH, SIO01L FFFFFD1AH

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| SIO0nL | SIOn7 | SIOn6 | SIOn5 | SIOn4 | SIOn3 | SIOn2 | SIOn1 | SIOn0 |

(n = 0, 1)

**Table 16-1. Use of Each Buffer Register**

| Register Name | R/W | | Single Transfer | | Continuous Trans |
|---|---|---|---|---|---|
| | | | Transmission/Reception Mode | Receive-Only Mode | Transmission/Reception Mode |
| SIRBn (SIRBnL) | Read | Function | Storing received data[Note 2] | • Reading starts reception<br>• Storing received data | Storing up to the (N – 1)th received data (other than the last)[Note 2] |
| | | Use method | When transmission and reception are complete, read the received data from this register. | • First, read dummy data and start transfer.<br>• To perform reception of the next data after reception is complete, read the received data from this register. | When reception is complete, read the received data from this register. Repeat this operation until the (N – 1)th data has been received. |
| SIRBEn (SIRBEnL) | Read | Function | – | Storing the data received last[Note 2] | – |
| | | Use method | Not used. | If reception of the next data will not be performed after reception is complete, read the received data from this register. | Not used |
| SIO0n (SIO0nL) | Read | Function | – | – | Storing the Nth (last) received data[Note 2] |
| | | Use method | Not used. | Not used | When the Nth (last) transmission/reception is complete, read the Nth (last) data. |
| SOTBn (SOTBnL) | Write | Function | • Starting transmission/reception when written<br>• Storing the data to be transmitted | – | • Starting transmission/reception when written<br>• Storing the data to be transmitted second and subsequently |
| | | Use method | • First, write a dummy data (FFH) to start transmission/reception.<br>• When transmission/reception is complete, write the data to be transmitted next. | Not used | When transmission/reception is complete, write the data to be transmitted next to this register to start the next transmission/reception. |
| SOTBFn (SOTBFnL) | Write | Function | – | – | Storing the data to be transmitted first[Note 2] |
| | | Use method | Not used | Not used | Before starting transmission/reception (writing to SOTBn), write the data to be transmitted first. |

**Notes 1.** It is assumed that the number of data to be transmitted is N.

**2.** Neither reading nor writing will start communication.

**Remark** In the 16-bit mode, the registers not enclosed in parentheses are used; in the 8-bit mode, the registers in parentheses are used

## 16.4 Operation

### 16.4.1 Transmission/reception completion interrupt request signal (INTCSI0n)

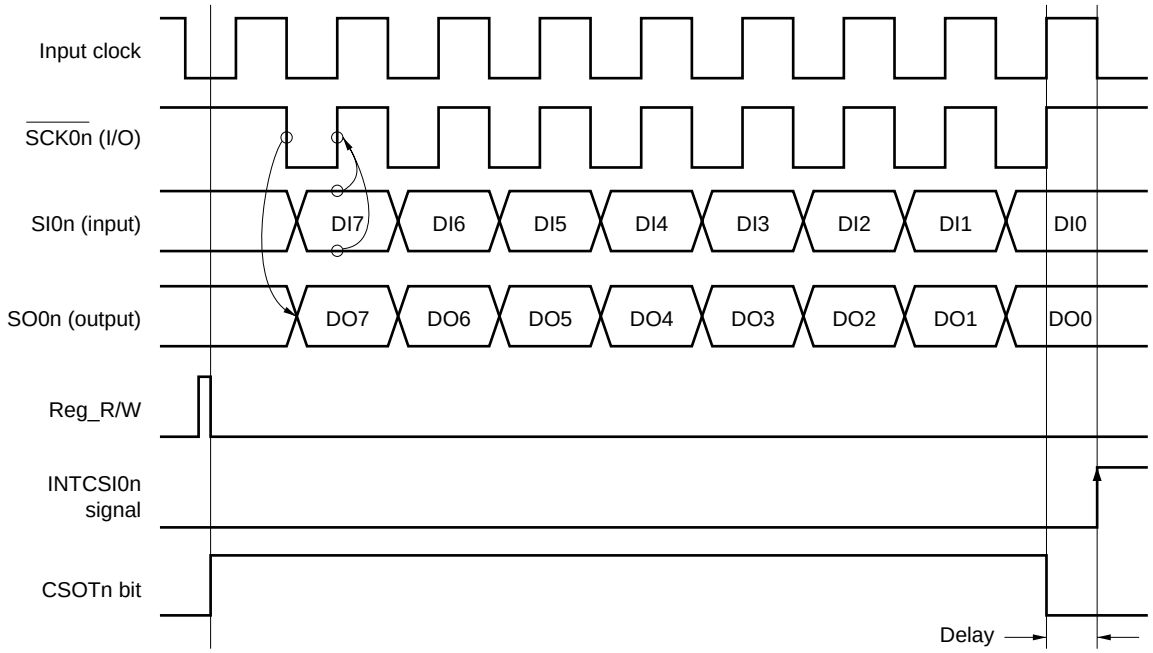
The INTCSI0n signal is set (1) upon completion of data transmission/reception.

Writing to the CSIM0n register clears (0) the INTCSI0n signal.

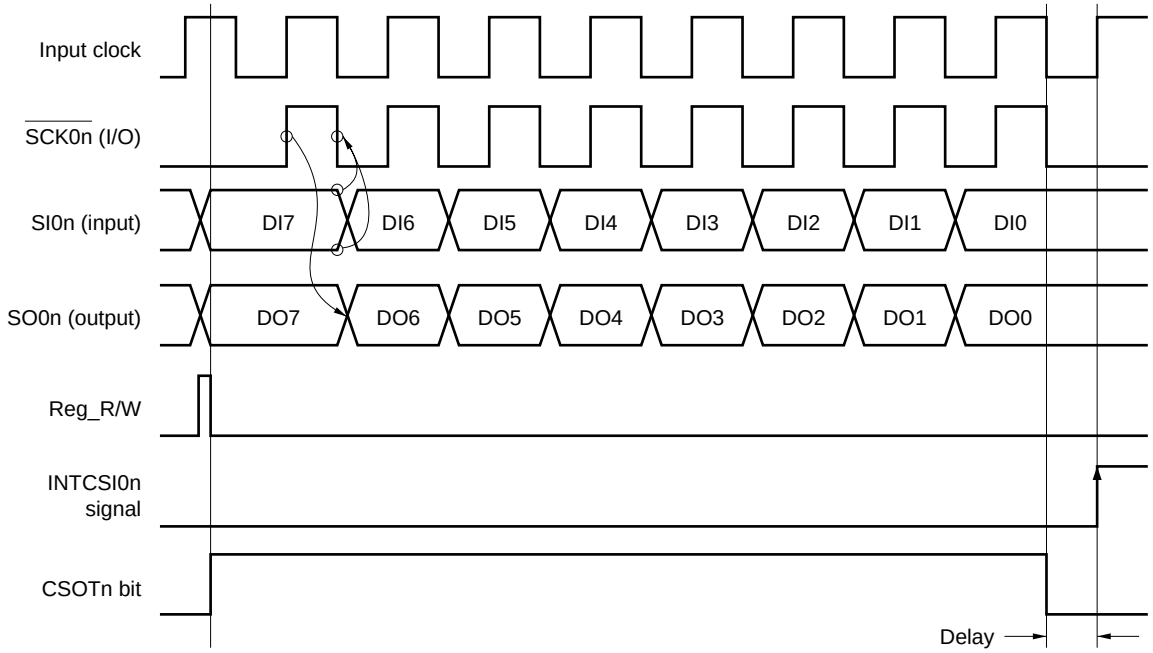
**Caution The delay mode (CSIM0n.CSITn bit = 1) is valid only in the master mode (CSICn.CKS0n2 to CSICn.CKS0n0 bits are not 111B). The delay mode cannot be set when the slave mode is set (CKS0n2 to CKS0n0 bits = 111B).**

**Figure 16-2. Timing Chart of INTCSI0n Signal Output in Delay Mode**

**(a) Transmit/receive type 1**




**(b) Transmit/receive type 4**




**Remarks 1.** Reg_R/W: Internal signal. This signal indicates that the SIRBn/SIRBnL register read or the SOTBn/SOTBnL register write was performed.

**2.** n = 0, 1

### 16.4.2 Single transfer mode

**(1) Usage**

In the receive-only mode (CSIM0n.TRMDn bit = 0), communication is started by reading the SIRBn/SIRBnL register.

In the transmission/reception mode (TRMDn bit = 1), communication is started by writing to the SOTBn/SOTBnL register.

In the slave mode, the operation must be enabled beforehand (CSIM0n.CSI0En bit = 1).

When communication is started, the value of the CSIM0n.CSOTn bit becomes 1 (transmission execution status).

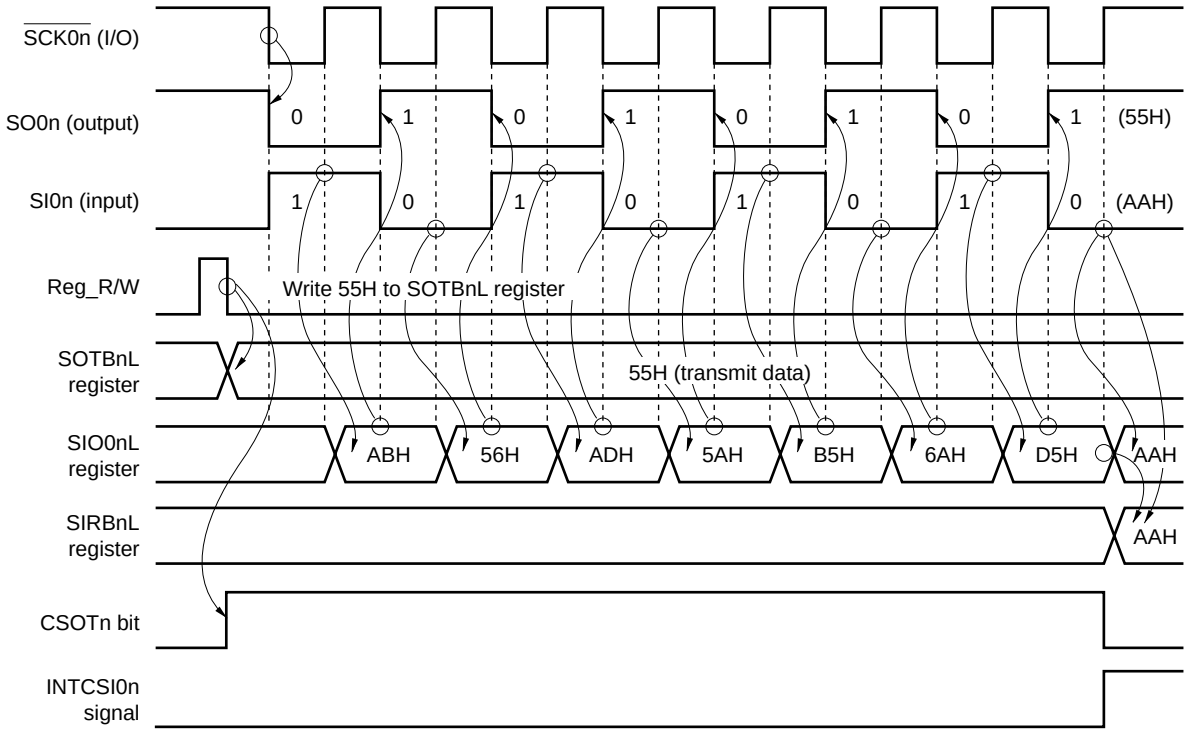
Upon communication completion, the transmission/reception completion interrupt request signal (INTCSI0n) is generated, and the CSOTn bit is cleared (0). The next data communication request is then waited for.

**Caution When the CSOTn bit = 1, do not manipulate the CSI0n register.**

**Remark** n = 0, 1

**Figure 16-3. Timing Chart in Single Transfer Mode (1/2)**

**(a) In transmission/reception mode, data length: 8 bits, transfer direction: MSB first, no interrupt delay, single transfer mode, when AAH is received and 55H is transmitted, transmit/receive type 1**



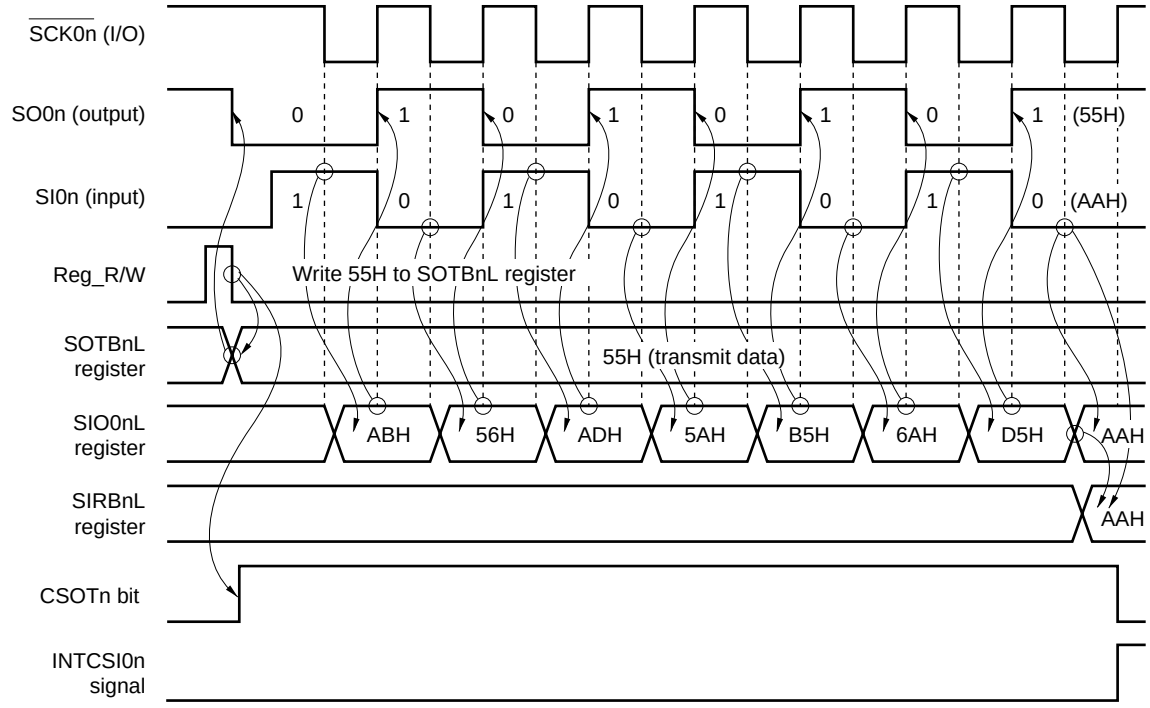

**Remarks 1.** Reg_R/W: Internal signal. This signal indicates that the SIRBn/SIRBnL register read or the SOTBn/SOTBnL register write was performed.

**2.** For the transmit/receive types, refer to **16.3 (2) Clocked serial interface clock selection register n (CSICn)**.

**3.** n = 0, 1

**Figure 16-3. Timing Chart in Single Transfer Mode (2/2)**

**(b) In transmission/reception mode, data length: 8 bits, transfer direction: MSB first, no interrupt delay, single transfer mode, when AAH is received and 55H is transmitted, transmit/receive type 2**




**Remarks 1.** Reg_R/W: Internal signal. This signal indicates that the SIRBn/SIRBnL register read or the SOTBn/SOTBnL register write was performed.

**2.** For the transmit/receive types, refer to **16.3 (2) Clocked serial interface clock selection register n (CSICn)**.

**3.** n = 0, 1

### 16.4.3 Continuous transfer mode

#### (1) Usage (receive-only: 8-bit data length)

<1> Set the continuous transfer mode (CSIM0n.AUTOn bit = 1) and the receive-only mode (CSIM0n.TRMDn bit = 0).

<2> Read the SIRBnL register (start transfer with dummy read).

<3> When the transmission/reception completion interrupt request signal (INTCSI0n) has been generated, read the SIRBnL register[Note] (reserve next transfer).

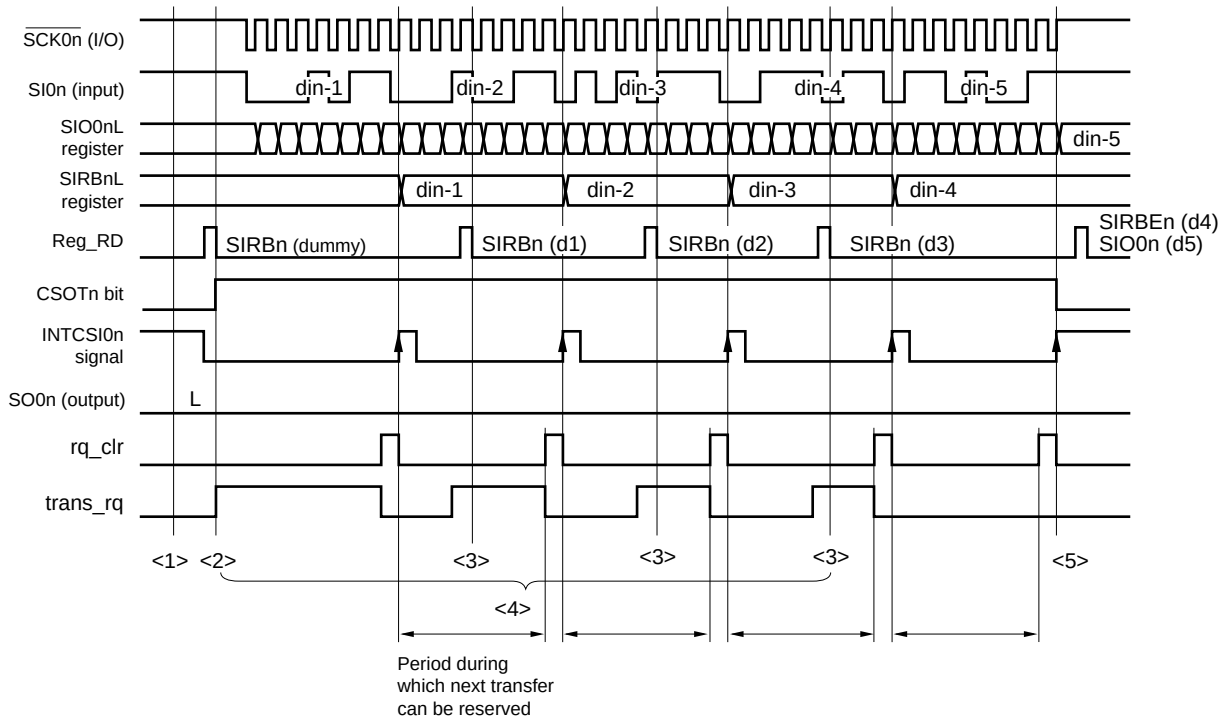
<4> Repeat step <3> (N – 2) times. (N: Number of transfer data)
Ignore the interrupt triggered by reception of the (N – 1)th data (at this time, the SIRBEnL register can be read).

<5> Following generation of the last INTCSI0n signal, read the SIRBEnL register and the SIO0nL register[Note].

**Note** When transferring N number of data, receive data is loaded by reading the SIRBnL register from the first data to the (N – 2)th data. The (N – 1)th data is loaded by reading the SIRBEnL register, and the Nth (last) data is loaded by reading the SIO0nL register (refer to **Table 16-1 Use of Each Buffer Register**).

**Figure 16-4. Continuous Transfer (Receive-Only) Timing Chart**

- Transmit/receive type 1, 8-bit data length




**Remarks 1.** Reg_RD: Internal signal. This signal indicates that the SIRBnL register has been read.
rq_clr: Internal signal. Transfer request clear signal.
trans_rq: Internal signal. Transfer request signal.

**2.** n = 0, 1

In the case of the continuous transfer mode, two transfer requests are set at the start of the first transfer. Following the INTCSI0n signal, transfer is continued if the SIRBnL register can be read within the next transfer reservation period. If the SIRBnL register cannot be read, transfer ends and the SIRBnL register does not receive the new value of the SIO0nL register.

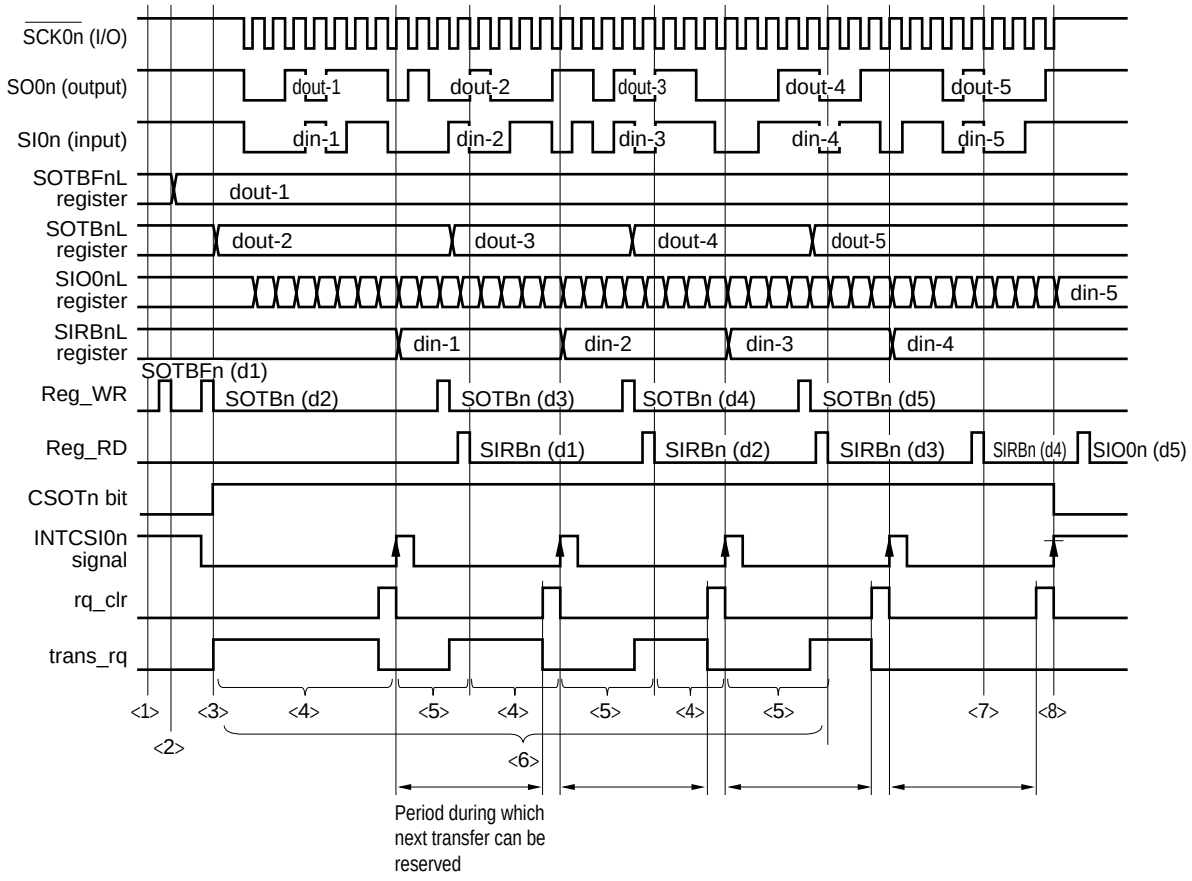
The last data can be obtained by reading the SIO0nL register following completion of the transfer.

**(2) Usage (transmission/reception: 8-bit data length)**

<1> Set the continuous transfer mode (CSIM0n.AUTOn bit = 1) and the transmission/reception mode (CSIM0n.TRMDn bit = 1).
<2> Write the first data to the SOTBFnL register.
<3> Write the 2nd data to the SOTBnL register (start transfer).
<4> When the transmission/reception completion interrupt request signal (INTCSI0n) has been generated, write the next data to the SOTBnL register (reserve next transfer). Read the SIRBnL register to load the receive data.
<5> Repeat step <4> as long as data to be sent remains.
<6> When the INTCSI0n signal is generated, read the SIRBnL register to load the (N – 1)th receive data (N: Number of transfer data).
<7> Following the last INTCSI0n signal, read the SIO0nL register to load the Nth (last) receive data.

**Figure 16-5. Continuous Transfer (Transmission/Reception) Timing Chart**

- Transmit/receive type 1, 8-bit data length




**Remarks 1.** Reg_WR: Internal signal. This signal indicates that the SOTBnL register has been written.
Reg_RD: Internal signal. This signal indicates that the SIRBnL register has been read.
rq_clr: Internal signal. Transfer request clear signal.
trans_rq: Internal signal. Transfer request signal.

**2.** n = 0, 1

In the case of the continuous transfer mode, two transfer requests are set at the start of the first transfer. Following the INTCSI0n signal, transfer is continued if the SOTBnL register can be written within the next transfer reservation period. If the SOTBnL register cannot be written, transfer ends and the SIRBnL register does not receive the new value of the SIO0nL register.

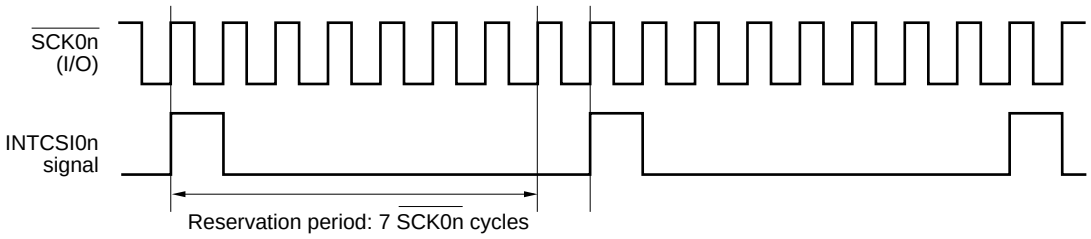
The last receive data can be obtained by reading the SIO0nL register following completion of the transfer.

**(3) Next transfer reservation period**

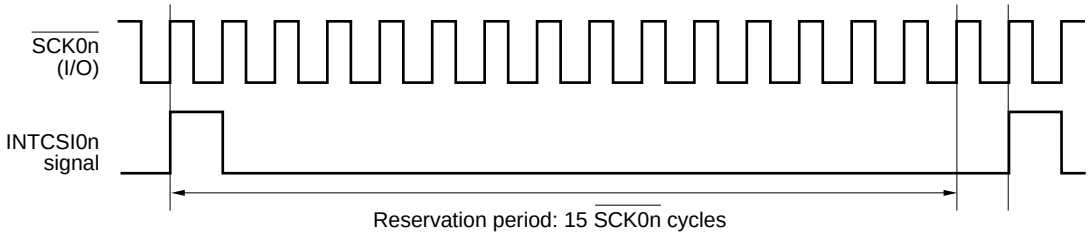
In the continuous transfer mode, the next transfer must be prepared with the period shown in Figure 16-6.

**Figure 16-6. Timing Chart of Next Transfer Reservation Period (1/2)**

**(a) When data length: 8 bits, transmit/receive type 1**



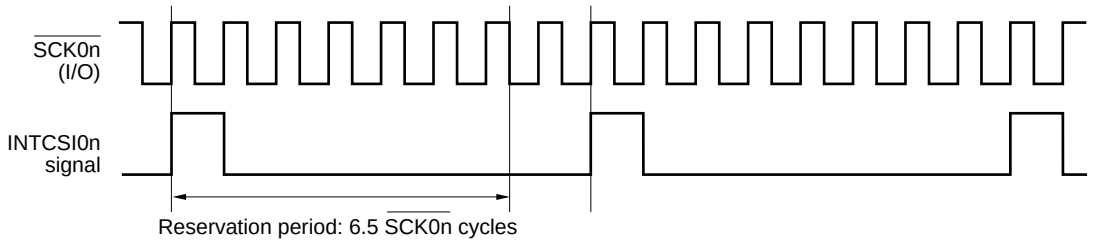

**(b) When data length: 16 bits, transmit/receive type 1**



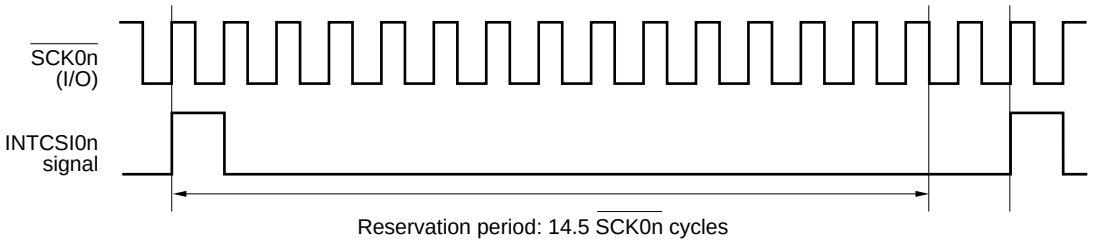

**Remark** n = 0, 1

**Figure 16-6. Timing Chart of Next Transfer Reservation Period (2/2)**

**(c) When data length: 8 bits, transmit/receive type 2**




**(d) When data length: 16 bits, transmit/receive type 2**




**Remark** n = 0, 1

**(4) Cautions**

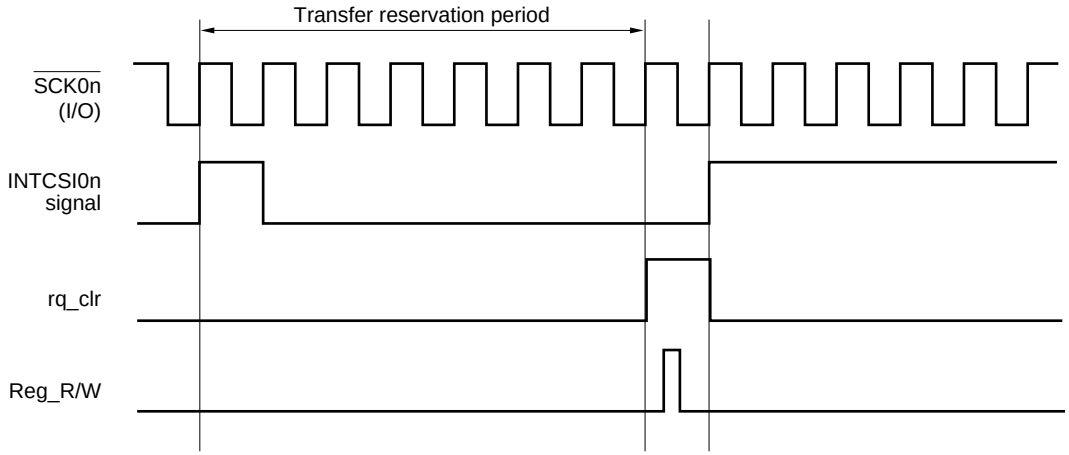
To continue continuous transfers, it is necessary to either read the SIRBn register or write to the SOTBn register during the transfer reservation period.

If access is performed to the SIRBn register or the SOTBn register when the transfer reservation period is over, the following occurs.

**(i) In case of conflict between transfer request clear and register access**

Since transfer request clear has higher priority, the next transfer request is ignored. Therefore, transfer is interrupted, and normal data transfer cannot be performed.

**Figure 16-7. Transfer Request Clear and Register Access Conflict**




**Remarks 1.** rq_clr: Internal signal. Transfer request clear signal.

Reg_R/W: Internal signal. This signal indicates that the SIRBn/SIRBnL register read or the SOTBn/SOTBnL register write was performed.

**2.** n = 0, 1

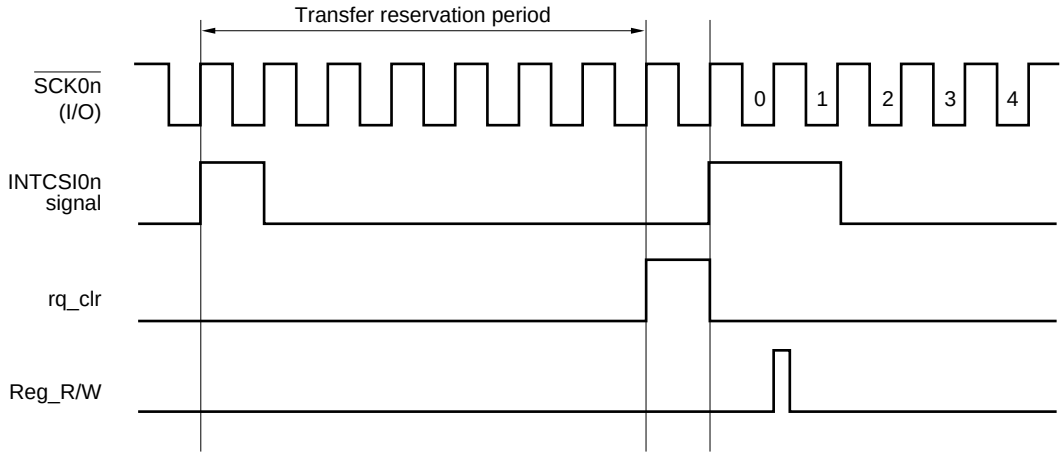
**(ii) In case of conflict between transmission/reception completion interrupt request signal (INTCSI0n) generation and register access**

Since continuous transfer has stopped once, executed as a new continuous transfer.

In the slave mode, a bit phase error transfer error results (refer to **Figure 16-8**).

In the transmission/reception mode, the value of the SOTBFn register is retransmitted, and illegal data is sent.

**Figure 16-8. Interrupt Request and Register Access Conflict**




**Remarks 1.** rq_clr: Internal signal. Transfer request clear signal.

Reg_R/W: Internal signal. This signal indicates that the SIRBn/SIRBnL register read or the SOTBn/SOTBnL register write was performed.

**2.** n = 0, 1

## 16.5 Output Pins

The following describes the output pins. For the setting of each pin, refer to **Table 4-14 Settings When Port Pins Are Used for Alternate Functions**.

**(1) $\overline{\text{SCK0n}}$ pin**

When the CSI0n operation is disabled (CSIM0n.CSI0En bit = 0), the $\overline{\text{SCK0n}}$ pin output status is as follows.

**Table 16-2. $\overline{\text{SCK0n}}$ Pin Output Status**

| CKPn | CKS0n2 | CKS0n1 | CKS0n0 | $\overline{\text{SCK0n}}$ Pin Output |
|---|---|---|---|---|
| 0 | Don't care | Don't care | Don't care | Fixed to high level |
| 1 | 1 | 1 | 1 | High impedance |
| | Other than above | | | Fixed to low level |

**Remark** n = 0, 1

**(2) SO0n pin**

When the CSI0n operation is disabled (CSI0En bit = 0), the SO0n pin output status is as follows.

**Table 16-3. SO0n Pin Output Status**

| TRMDn | DAPn | AUTOn | CCLn | DIRn | SO0n Pin Output |
|---|---|---|---|---|---|
| 0 | Don't care | Don't care | Don't care | Don't care | Fixed to low level |
| 1 | 0 | Don't care | Don't care | Don't care | SO latch value (low level) |
| | 1 | 0 | 0 | 0 | SOTBn7 bit value |
| | | | | 1 | SOTBn0 bit value |
| | | | 1 | 0 | SOTBn15 bit value |
| | | | | 1 | SOTBn0 bit value |
| | | 1 | 0 | 0 | SOTBFn7 bit value |
| | | | | 1 | SOTBFn0 bit value |
| | | | 1 | 0 | SOTBFn15 bit value |
| | | | | 1 | SOTBFn0 bit value |

**Remark** n = 0, 1

# CHAPTER 17 CLOCKED SERIAL INTERFACE A (CSIA) WITH AUTOMATIC TRANSMIT/RECEIVE FUNCTION

In the V850ES/KF1+, one channel of CSIA is provided.

## 17.1 Functions

CSIA0 has the following two modes.

- 3-wire serial I/O mode
- 3-wire serial I/O mode with automatic transmit/receive function

**(1) 3-wire serial I/O mode**

This mode is used to transfer 8-bit data using three lines: a serial clock pin ($\overline{\text{SCKA0}}$) and two serial data pins (SIA0 and SOA0).

In addition, whether 8-bit data is transferred MSB or LSB first can be specified, so this interface can be connected to any device.

**(2) 3-wire serial I/O mode with automatic transmit/receive function**

This mode is used to transfer 8-bit data using three lines: a serial clock pin ($\overline{\text{SCKA0}}$) and two serial data pins (SIA0 and SOA0).

In addition, whether 8-bit data is transferred MSB or LSB first can be specified, so this interface can be connected to any device.

Data can be transferred to/from a display driver etc. without using software since a 32-byte buffer RAM is incorporated for automatic transfer.

- Maximum transfer speed: 2 MHz (in master mode)
- Master mode/slave mode selectable
- Transfer data length: 8 bits
- MSB/LSB-first selectable for transfer data
- Automatic transmit/receive function:
  Number of transfer bytes can be specified between 1 and 32
  Transfer interval can be specified (0 to 63 clocks)
  Single transfer/repeat transfer selectable
- On-chip dedicated baud rate generator (6/8/16/32 divisions)
- 3-wire SOA0: Serial data output
  SIA0: Serial data input
  $\overline{\text{SCKA0}}$: Serial clock I/O
- Transmission/reception completion interrupt request signal: INTCSIA0
- Internal 32-byte buffer RAM

## 17.2 Configuration

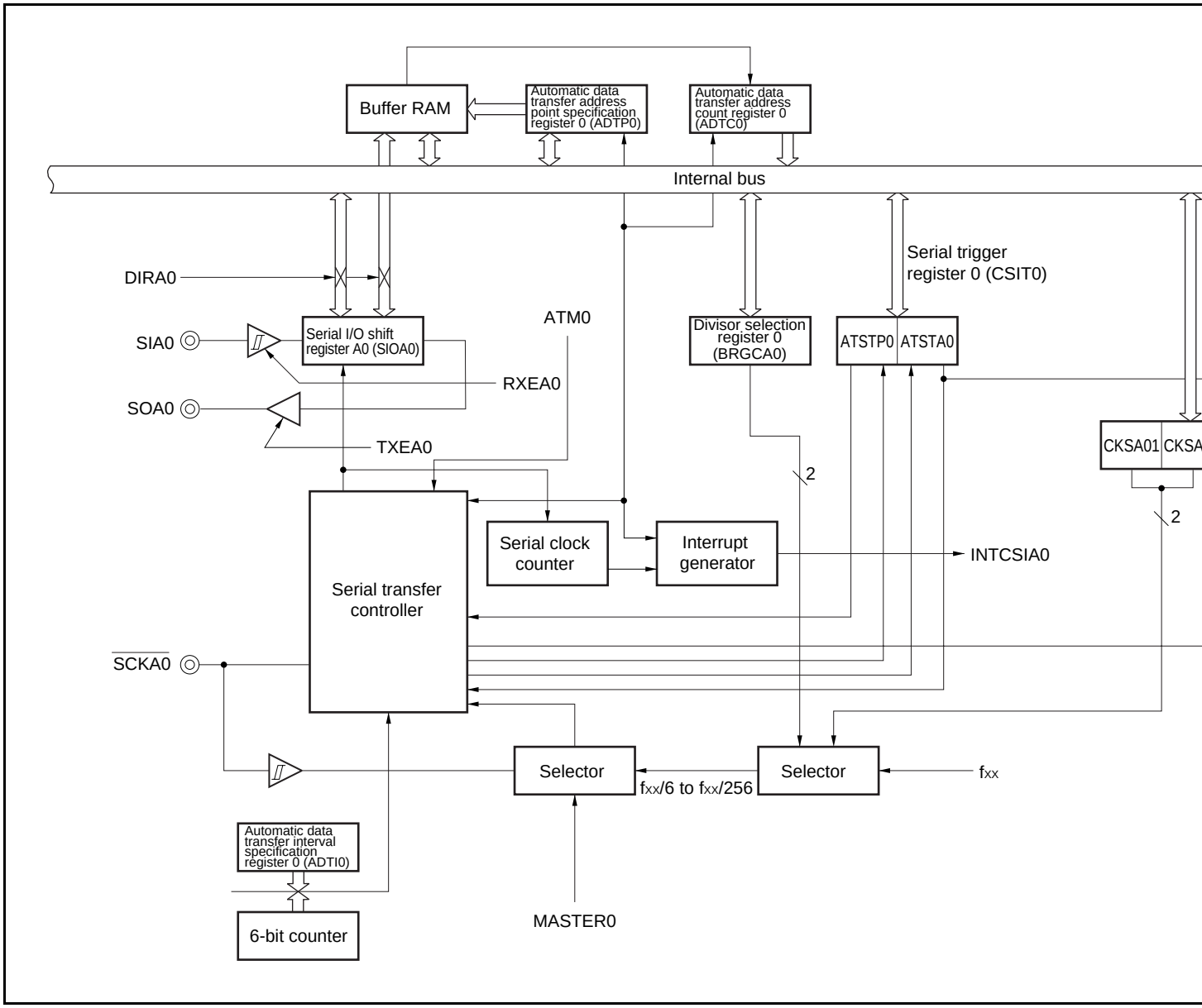
CSIA0 consists of the following hardware.

**Table 17-1. Configuration of CSIA0**

| Item | Configuration |
|---|---|
| Registers | Serial I/O shift register A0 (SIOA0)<br>Automatic data transfer address count register 0 (ADTC0)<br>CSIA0 buffer RAM (CSIA0Bm, CSIA0BmL, CSIA0BmH) (m = 0 to F) |
| Control registers | Serial operation mode specification register 0 (CSIMA0)<br>Serial status register 0 (CSIS0)<br>Serial trigger register 0 (CSIT0)<br>Divisor selection register 0 (BRGCA0)<br>Automatic data transfer address point specification register 0 (ADTP0)<br>Automatic data transfer interval specification register 0 (ADTI0) |

**Remark** For the alternate-function pin settings, refer to **Table 4-14 Settings When Port Pins Are Used for Alternate Functions**.

**Figure 17-1. Block Diagram of CSIA0**



**(1) Serial I/O shift register A0 (SIOA0)**

This is an 8-bit register used to store transmit/receive data in 1-byte transfer mode (CSIMA0.ATE0 bit = 0). Writing transmit data to the SIOA0 register starts the transfer. In addition, after a transfer completion interrupt request signal (INTCSIA0) is generated (CSIS0.TSF0 bit = 0), data can be received by reading data from the SIOA0 register.

This register can be read or written in 8-bit units. However, writing to the SIOA0 register is prohibited when the TSF0 bit = 1.

After reset, this register is cleared to 00H.

**Cautions 1. A transfer operation is started by writing to the SIOA0 register. Consequently, when transmission is disabled (CSIMA0.TXEA0 bit = 0), write dummy data to the SIOA0 register to start the transfer operation, and then perform a receive operation.**

**2. Do not write data to the SIOA0 register while the automatic transmit/receive function is operating.**

After reset: 00H R/W Address: FFFFFD46H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| SIOA0 | SIOA07 | SIOA06 | SIOA05 | SIOA04 | SIOA03 | SIOA02 | SIOA01 | SIOA00 |

**(2) Automatic data transfer address count register 0 (ADTC0)**

This is a register used to indicate buffer RAM addresses during automatic transfer. When automatic transfer is stopped, the data position when transfer stopped can be ascertained by reading ADTC0 register value.

This register is read-only in 8-bit units. However, reading from the ADTC0 register is prohibited when the CSIS0.TSF0 bit = 1.

After reset, this register is cleared to 00H.

After reset: 00H R Address: FFFFFD47H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ADTC0 | ADTC07 | ADTC06 | ADTC05 | ADTC04 | ADTC03 | ADTC02 | ADTC01 | ADTC00 |

## 17.3 Registers

Serial interface CSIA0 is controlled by the following six registers.

- Serial operation mode specification register 0 (CSIMA0)
- Serial status register 0 (CSIS0)
- Serial trigger register 0 (CSIT0)
- Divisor selection register 0 (BRGCA0)
- Automatic data transfer address point specification register 0 (ADTP0)
- Automatic data transfer interval specification register 0 (ADTI0)

**(1) Serial operation mode specification register 0 (CSIMA0)**

This is an 8-bit register used to control the serial transfer operation.

This register can be read or written in 8-bit or 1-bit units.

After reset, this register is cleared to 00H.

After reset: 00H R/W Address: FFFFFD40H

| | <7> | 6 | 5 | 4 | <3> | <2> | <1> | 0 |
|---|---|---|---|---|---|---|---|---|
| CSIMA0 | CSIAE0 | ATE0 | ATM0 | MASTER0 | TXEA0 | RXEA0 | DIRA0 | 0 |

| CSIAE0 | CSIA0 operation enable/disable control |
|---|---|
| 0 | Disable CSIA0 operation (SOA0: Low level, $\overline{\text{SCKA0}}$: High level) |
| 1 | Enable CSIA0 operation |

- When the CSIAE0 bit is cleared to 0, the CSIA0 unit is reset[Note] asynchronously.
- When the CSIAE0 bit = 0, the CSIA0 unit is reset, so to operate CSIA0, first set the CSIAE0 bit to 1.
- If the CSIAE0 bit is cleared from 1 to 0, all the registers in the CSIA0 unit are initialized. Before the CSIAE0 bit is set to 1 again, first re-set the registers of the CSIA0 unit.
- If the CSIAE0 bit is cleared from 1 to 0, the buffer RAM value is not held. Also, when the CSIAE0 bit = 0, the buffer RAM cannot be accessed.

| ATE0 | Automatic transfer operation enable/disable control |
|---|---|
| 0 | 1-byte transfer mode |
| 1 | Automatic transfer mode |

| ATM0 | Specification of automatic transfer mode |
|---|---|
| 0 | Single transfer mode (stops at address specified with ADTP0 register) |
| 1 | Repeat transfer mode (Following transfer completion, the ADTC0 register is cleared to 00H and transmission starts again.) |

| MASTER0 | Specification of CSIA0 master/slave mode |
|---|---|
| 0 | Slave mode (synchronized with $\overline{\text{SCKA0}}$ input clock) |
| 1 | Master mode (synchronized with internal clock) |

| TXEA0 | Transmission enable/disable control |
|---|---|
| 0 | Disable transmission (SOA0: Low level) |
| 1 | Enable transmission |

| RXEA0 | Reception enable/disable control |
|---|---|
| 0 | Disable reception |
| 1 | Enable reception |

| DIRA0 | Specification of transfer data direction |
|---|---|
| 0 | MSB first |
| 1 | LSB first |

**Note** The ADTC0, CSIT0, and SIOA0 registers and the CSIS0.TSF0 bit are reset.

**(2) Serial status register 0 (CSIS0)**

This is an 8-bit register used to select the serial clock and to indicate the transfer status of CSIA0.

This register can be read or written in 8-bit or 1-bit units.

After reset, this register is cleared to 00H. However, rewriting the CSIS0 register is prohibited when the TSF0 bit is 1.

After reset: 00H R/W Address: FFFFFD41H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| CSIS0 | CKSA01 | CKSA00 | 0 | 0 | 0 | 0 | 0 | TSF0 |

| CKSA01 | CKSA00 | Serial clock ($f_{SCKA}$) selection[Note] | | | |
|---|---|---|---|---|---|
| | | | 20 MHz | 16 MHz | 10 MHz |
| 0 | 0 | $f_{XX}$ | Setting prohibited | Setting prohibited | 100 ns |
| 0 | 1 | $f_{XX}/2$ | 100 ns | 125 ns | 200 ns |
| 1 | 0 | $f_{XX}/4$ | 200 ns | 250 ns | 400 ns |
| 1 | 1 | $f_{XX}/8$ | 400 ns | 500 ns | 800 ns |
| Rewriting CSIS0 is prohibited when the CSIMA0.CSIAE0 bit is 1. | | | | | |

| TSF0 | Transfer status |
|---|---|
| 0 | CSIAE0 bit = 0<br>At reset input<br>At completion of specified transfer<br>When transfer has been suspended by setting the CSIT0.ATSTP0 bit to 1 |
| 1 | From transfer start to completion of specified transfer |

**Note** Set $f_{SCKA}$ so as to satisfy the following conditions.

- $V_{DD}$ = REGC = 4.0 to 5.5 V: $f_{SCKA} \leq 12$ MHz
- $V_{DD}$ = 4.0 to 5.5 V, REGC = Capacity: $f_{SCKA} \leq 6$ MHz
- $V_{DD}$ = REGC = 2.7 to 4.0 V: $f_{SCKA} \leq 6$ MHz

**Cautions 1. The TSF0 bit is read-only.**

**2. When the TSF0 bit = 1, rewriting the CSIMA0, CSIS0, BRGCA0, ADTP0, ADTI0, and SIOA0 registers is prohibited. However, the transfer buffer RAM can be rewritten.**

**3. Be sure to clear bits 1 to 5 to 0.**

### (3) Serial trigger register 0 (CSIT0)

The CSIT0 register between the buffer RAM and shift register is an 8-bit register used to control execution/stop of automatic data transfer.

This register can be read or written in 8-bit or 1-bit units. However, manipulate only when the CSIMA0.ATE0 bit is 1 (manipulation prohibited when ATE0 bit = 0).

After reset, this register is cleared to 00H.

After reset: 00H R/W Address: FFFFFD42H

| | 7 | 6 | 5 | 4 | 3 | 2 | <1> | <0> |
|---|---|---|---|---|---|---|---|---|
| CSIT0 | 0 | 0 | 0 | 0 | 0 | 0 | ATSTP0 | ATSTA0 |

| ATSTP0 | Automatic data transfer suspension |
|---|---|
| 0 | – |
| 1 | Stop automatic data transfer |

Even when the ATSTP0 bit is set to 1, transfer does not stop until 1 byte has been transferred.
1 is held until immediately before the transmission/reception completion interrupt request signal (INTCSIA0) is generated, and ATSTP0 is automatically cleared to 0 after that.
After automatic transfer has been suspended, the data address at the point of suspension is stored in the ADTC0 register.
A function to resume automatic data transfer is not provided, so if transfer has been interrupted by setting the ATSTP0 bit to 1, set each register again, and set the ATSTA0 bit to 1 to start automatic data transfer.

| ATSTA0 | Automatic data transfer start |
|---|---|
| 0 | – |
| 1 | Start automatic data transfer |

Even when the ATSTA0 bit is set to 1, automatic data transfer does not start until 1 byte has been transferred.
1 is held until immediately before the INTCSIA0 signal is generated, and ATSTA0 is automatically cleared to 0 after that.

**(4) Divisor selection register 0 (BRGCA0)**

This is an 8-bit register used to control the serial transfer speed (divisor of CSIA clock).

This register can be read or written in 8-bit units. However, when the CSIS0.TSF0 bit is 1, rewriting the BRGCA0 register is prohibited.

After reset, this register is set to 03H.

After reset: 03H R/W Address: FFFFFD43H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| BRGCA0 | 0 | 0 | 0 | 0 | 0 | 0 | BRGC01 | BRGC00 |

| BRGC01 | BRGC00 | Selection of CSIA0 serial clock ($f_{SCKA}$ division ratio) |
|---|---|---|
| 0 | 0 | 6 ($f_{SCKA}$/6) |
| 0 | 1 | 8 ($f_{SCKA}$/8) |
| 1 | 0 | 16 ($f_{SCKA}$/16) |
| 1 | 1 | 32 ($f_{SCKA}$/32) |

**(5) Automatic data transfer address point specification register 0 (ADTP0)**

This is an 8-bit register used to specify the buffer RAM address that ends transfer during automatic data transfer (CSIMA0.ATE0 bit = 1).

This register can be read or written in 8-bit units. However, when the CSIS0.TSF0 bit is 1, rewriting the ADTP0 register is prohibited.

After reset, this register is cleared to 00H.

In the V850ES/KF1+, 00H to 1FH can be specified because 32 bytes of buffer RAM are incorporated.

**Example** When the ADTP0 register is set to 07H
8 bytes of FFFFFE00H to FFFFFE07H are transferred.

In repeat transfer mode (CSIMA0.ATM0 bit = 1), transfer is performed repeatedly up to the address value specified by the ADTP0 register.

**Example** When the ADTP0 register is set to 07H (repeat transfer mode)
Transfer is repeated as FFFFFE00H to FFFFFE07H, … .

After reset: 00H R/W Address: FFFFFD44H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ADTP0 | 0 | 0 | 0 | ADTP04 | ADTP03 | ADTP02 | ADTP01 | ADTP00 |

**Caution Be sure to clear bits 5 to 7 to 0.**

The relationship between buffer RAM address values and the ADTP0 register setting values is shown below.

**Table 17-2. Relationship Between Buffer RAM Address Values and ADTP0 Register Setting Values**

| Buffer RAM Address Value | ADTP0 Register Setting Value | Buffer RAM Address Value | ADTP0 Register Setting Value |
|---|---|---|---|
| FFFFFE00H | 00H | FFFFFE10H | 10H |
| FFFFFE01H | 01H | FFFFFE11H | 11H |
| FFFFFE02H | 02H | FFFFFE12H | 12H |
| FFFFFE03H | 03H | FFFFFE13H | 13H |
| FFFFFE04H | 04H | FFFFFE14H | 14H |
| FFFFFE05H | 05H | FFFFFE15H | 15H |
| FFFFFE06H | 06H | FFFFFE16H | 16H |
| FFFFFE07H | 07H | FFFFFE17H | 17H |
| FFFFFE08H | 08H | FFFFFE18H | 18H |
| FFFFFE09H | 09H | FFFFFE19H | 19H |
| FFFFFE0AH | 0AH | FFFFFE1AH | 1AH |
| FFFFFE0BH | 0BH | FFFFFE1BH | 1BH |
| FFFFFE0CH | 0CH | FFFFFE1CH | 1CH |
| FFFFFE0DH | 0DH | FFFFFE1DH | 1DH |
| FFFFFE0EH | 0EH | FFFFFE1EH | 1EH |
| FFFFFE0FH | 0FH | FFFFFE1FH | 1FH |

**(6) Automatic data transfer interval specification register 0 (ADTI0)**

This is an 8-bit register used to specify the interval period between 1-byte transfers during automatic data transfer (CSIMA0.ATE0 bit = 1).

Set this register when in master mode (CSIMA0.MASTER0 bit = 1) (setting is unnecessary in slave mode). Setting in 1-byte transfer mode (ATE0 bit = 0) is also valid. When the interval time specified by the ADTI0 register after the end of 1-byte transfer has elapsed, a transmission/reception completion interrupt request signal (INTCSIA0) is output. The number of clocks for the interval can be set to between 0 and 63 clocks.

This register can be read or written in 8-bit units. However, when the CSIS0.TSF0 bit is 1, rewriting the ADTI0 register is prohibited.

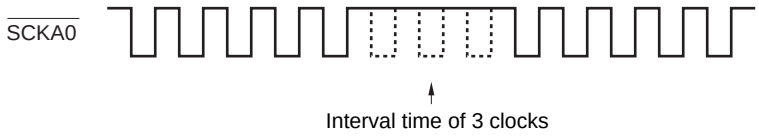
After reset, this register is cleared to 00H.

After reset: 00H R/W Address: FFFFFD45H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| ADTI0 | 0 | 0 | ADTI05 | ADTI04 | ADTI03 | ADTI02 | ADTI01 | ADTI00 |

The specified interval time is the transfer clock (specified by the BRGCA0 register) multiplied by an integer value.

**Example** When ADTI0 register = 03H




**(7) CSIA0 buffer RAM (CSIA0Bm)**

This area holds transmit/receive data (up to 32 bytes) in automatic transfer mode in 1-byte units.

This register can be read or written in 16-bit units. However, when the higher 8 bits and the lower 8 bits of the CSIA0Bm register are used as the CSIA0BmH register and CSIA0BmL register, respectively, these registers can be read or written in 8-bit units.

After automatic transfer is started, only data equal to one byte more than the number of bytes stored in the ADTP0 register is transmitted/received in sequence from the CSIA0B0L register.

**Cautions 1. To read the value of the CSIA0Bm register after data is written to the register, wait for the duration of more than six clocks of $f_{SCKA}$ (serial clock set by the CSIS0.CKSA01 and CSIS0.CKSA00 bits) or until data is written to the buffer RAM at another address.**

**2. When the main clock stops and the CPU operates on the subclock, do not access the CSIA0Bm register.**
**For details, refer to 3.4.8 (2).**

**Remark** m = 0 to F

**Table 17-3. CSIA0 Buffer RAM**

| Address | Symbol | R/W | Manipulatable Bits | | After Reset |
|---|---|---|---|---|---|
| | | | 8 | 16 | |
| FFFFFE00H | CSIA0B0 | R/W | | √ | Undefined |
| FFFFFE00H | CSIA0B0L | R/W | √ | | Undefined |
| FFFFFE01H | CSIA0B0H | R/W | √ | | Undefined |
| FFFFFE02H | CSIA0B1 | R/W | | √ | Undefined |
| FFFFFE02H | CSIA0B1L | R/W | √ | | Undefined |
| FFFFFE03H | CSIA0B1H | R/W | √ | | Undefined |
| FFFFFE04H | CSIA0B2 | R/W | | √ | Undefined |
| FFFFFE04H | CSIA0B2L | R/W | √ | | Undefined |
| FFFFFE05H | CSIA0B2H | R/W | √ | | Undefined |
| FFFFFE06H | CSIA0B3 | R/W | | √ | Undefined |
| FFFFFE06H | CSIA0B3L | R/W | √ | | Undefined |
| FFFFFE07H | CSIA0B3H | R/W | √ | | Undefined |
| FFFFFE08H | CSIA0B4 | R/W | | √ | Undefined |
| FFFFFE08H | CSIA0B4L | R/W | √ | | Undefined |
| FFFFFE09H | CSIA0B4H | R/W | √ | | Undefined |
| FFFFFE0AH | CSIA0B5 | R/W | | √ | Undefined |
| FFFFFE0AH | CSIA0B5L | R/W | √ | | Undefined |
| FFFFFE0BH | CSIA0B5H | R/W | √ | | Undefined |
| FFFFFE0CH | CSIA0B6 | R/W | | √ | Undefined |
| FFFFFE0CH | CSIA0B6L | R/W | √ | | Undefined |
| FFFFFE0DH | CSIA0B6H | R/W | √ | | Undefined |
| FFFFFE0EH | CSIA0B7 | R/W | | √ | Undefined |
| FFFFFE0EH | CSIA0B7L | R/W | √ | | Undefined |
| FFFFFE0FH | CSIA0B7H | R/W | √ | | Undefined |
| FFFFFE10H | CSIA0B8 | R/W | | √ | Undefined |
| FFFFFE10H | CSIA0B8L | R/W | √ | | Undefined |
| FFFFFE11H | CSIA0B8H | R/W | √ | | Undefined |
| FFFFFE12H | CSIA0B9 | R/W | | √ | Undefined |
| FFFFFE12H | CSIA0B9L | R/W | √ | | Undefined |
| FFFFFE13H | CSIA0B9H | R/W | √ | | Undefined |
| FFFFFE14H | CSIA0BA | R/W | | √ | Undefined |
| FFFFFE14H | CSIA0BAL | R/W | √ | | Undefined |
| FFFFFE15H | CSIA0BAH | R/W | √ | | Undefined |
| FFFFFE16H | CSIA0BB | R/W | | √ | Undefined |
| FFFFFE16H | CSIA0BBL | R/W | √ | | Undefined |
| FFFFFE17H | CSIA0BBH | R/W | √ | | Undefined |
| FFFFFE18H | CSIA0BC | R/W | | √ | Undefined |
| FFFFFE18H | CSIA0BCL | R/W | √ | | Undefined |
| FFFFFE19H | CSIA0BCH | R/W | √ | | Undefined |
| FFFFFE1AH | CSIA0BD | R/W | | √ | Undefined |
| FFFFFE1AH | CSIA0BDL | R/W | √ | | Undefined |
| FFFFFE1BH | CSIA0BDH | R/W | √ | | Undefined |
| FFFFFE1CH | CSIA0BE | R/W | | √ | Undefined |
| FFFFFE1CH | CSIA0BEL | R/W | √ | | Undefined |
| FFFFFE1DH | CSIA0BEH | R/W | √ | | Undefined |
| FFFFFE1EH | CSIA0BF | R/W | | √ | Undefined |
| FFFFFE1EH | CSIA0BFL | R/W | √ | | Undefined |
| FFFFFE1FH | CSIA0BFH | R/W | √ | | Undefined |

## 17.4 Operation

CSIA0 can be used in the following two modes.

- 3-wire serial I/O mode
- 3-wire serial I/O mode with automatic transmit/receive function

### 17.4.1 3-wire serial I/O mode

The one-byte data transmission/reception is executed in the mode in which the CSIMA0.ATE0 bit is cleared to 0.

In this mode, communication is executed by using three lines: serial clock ($\overline{\text{SCKA0}}$), serial data output (SOA0), and serial data input (SIA0) pins.

The 3-wire serial I/O mode is controlled by the following three registers.

- Serial operation mode specification register 0 (CSIMA0)
- Serial status register 0 (CSIS0)
- Divisor selection register 0 (BRGCA0)

**Remark** For the alternate-function pin settings, refer to **Table 4-14 Settings When Port Pins Are Used for Alternate Functions**.

### (1) 1-byte transmission/reception communication operation

#### (a) 1-byte transmission/reception

When the CSIMA0.CSIAE0 bit and the CSIMA0.ATE0 bit = 1, 0, respectively, if transfer data is written to the SIOA0 register, the data is output via the SOA0 pin in synchronization with the $\overline{\text{SCKA0}}$ pin falling edge, and then input via the SIA0 pin in synchronization with the falling edge of the $\overline{\text{SCKA0}}$ pin, and stored in the SIOA0 register in synchronization with the rising edge 1 clock later.

Data transmission and data reception can be performed simultaneously.

If only reception is to be performed, transfer can only be started by writing a dummy value to the SIOA0 register.

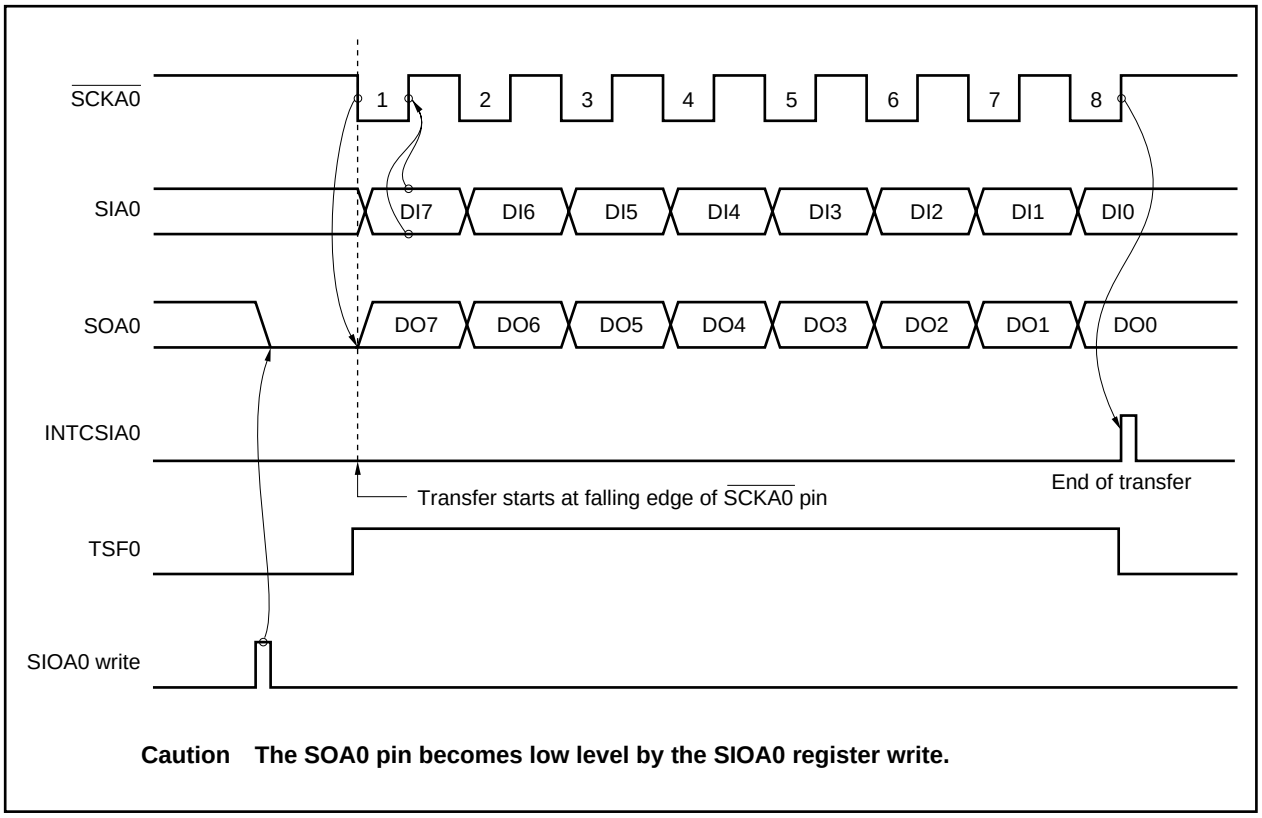
When transfer of 1 byte is complete, a transmission/reception completion interrupt request signal (INTCSIA0) is generated.

In 1-byte transmission/reception, the setting of the CSIMA0.ATM0 bit is invalid.

Be sure to read data after confirming that the CSIS0.TSF0 bit = 0.

**Caution Determine the setting procedure of alternate-function pins considering the relationship with the communication partner.**

**Figure 17-2. 3-Wire Serial I/O Mode Timing**



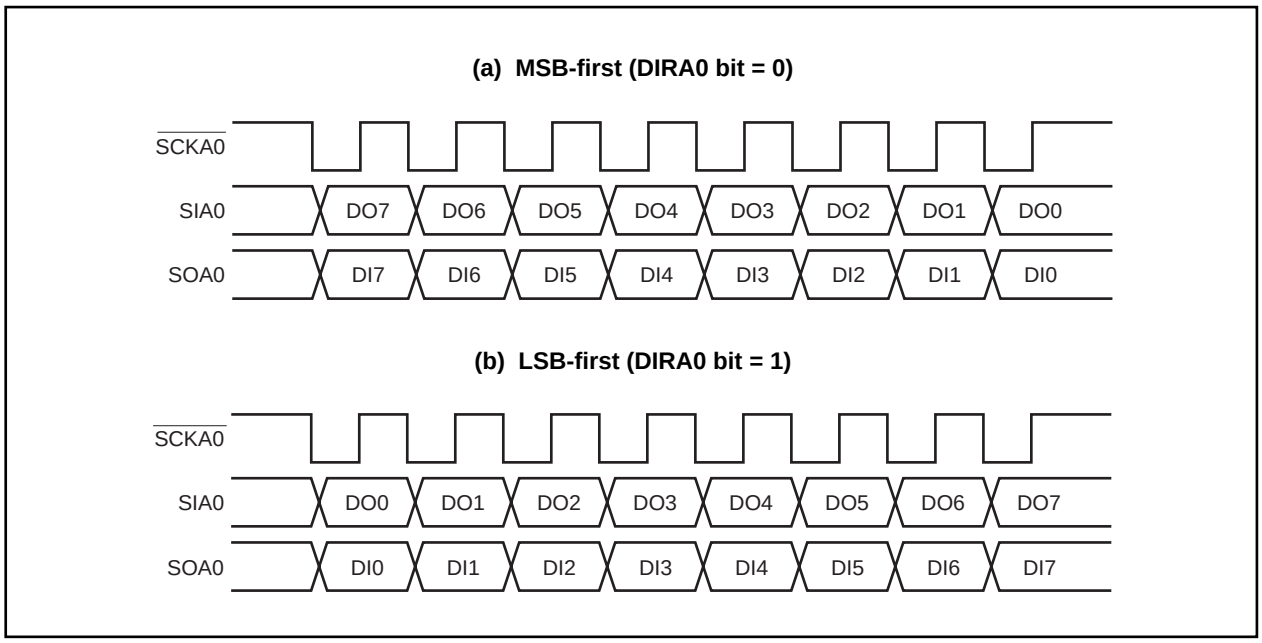

**Caution The SOA0 pin becomes low level by the SIOA0 register write.**

**(b) Data format**

In the data format, data is changed in synchronization with the $\overline{\text{SCKA0}}$ pin falling edge as shown in Figure 17-3.

The data length is fixed to 8 bits and the data transfer direction can be switched by the specification of the CSIMA0.DIRA0 bit.

**Figure 17-3. Format of Transmit/Receive Data**



**(c) Switching MSB/LSB as start bit**

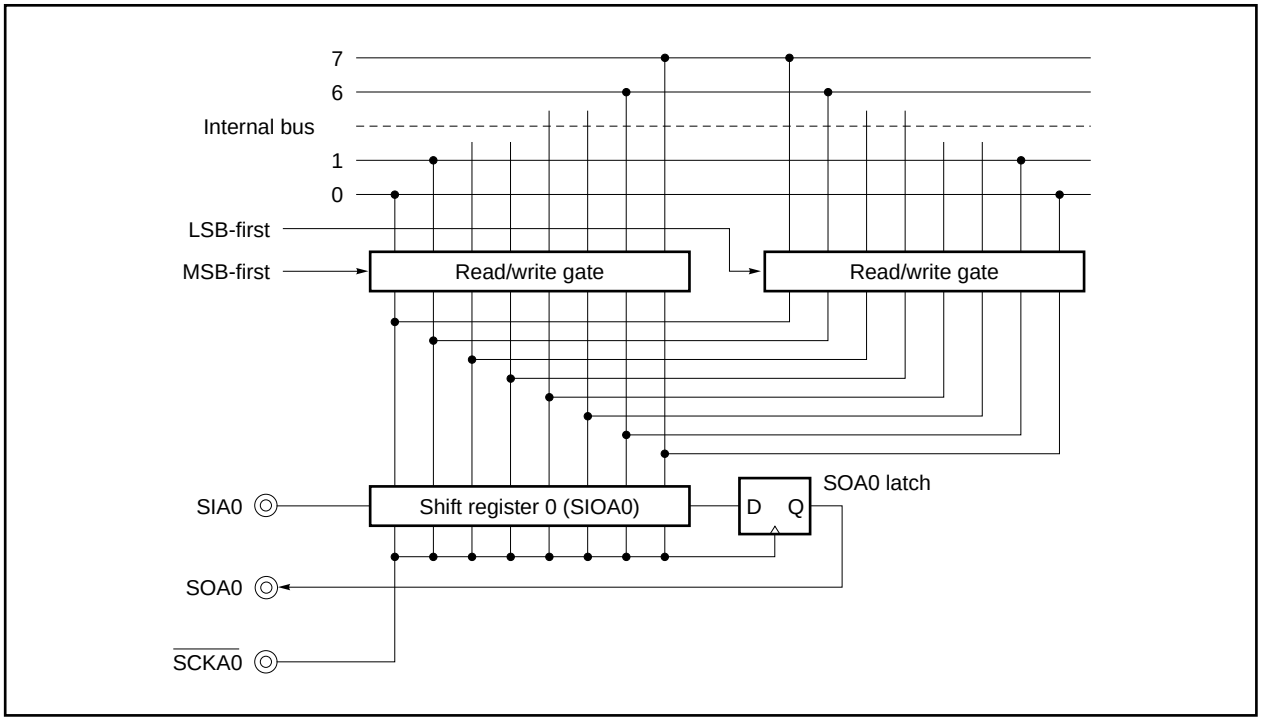
Figure 17-4 shows the configuration of the SIOA0 register and the internal bus. As shown in the figure, MSB/LSB can be read or written in reverse form.

Switching MSB/LSB as the start bit can be specified using the CSIMA0.DIRA0 bit.

Start bit switching is realized by switching the bit order for data written to the SIOA0 register. The SIOA0 register shift order remains unchanged.

Thus, switching between MSB-first and LSB-first must be performed before writing data to the SIOA0 register.

**Figure 17-4. Transfer Bit Order Switching Circuit**




**(d) Transfer start**

Serial transfer is started by setting transfer data to the SIOA0 register when the following two conditions are satisfied.

- CSIA0 operation control bit (CSIMA0.CSIAE0) = 1
- Other than during serial communication

**Caution If the CSIAE0 bit is set to 1 after data is written to the SIOA0 register, communication does not start.**

Upon termination of 8-bit communication, serial communication automatically stops and the transmission/reception completion interrupt request signal (INTCSIA0) is generated.

### 17.4.2 3-wire serial I/O mode with automatic transmit/receive function

Up to 32 bytes of data can be transmitted/received without using software in the mode in which the CSIMA0.ATE0 bit is set to 1. After communication is started, only data of the set number of bytes stored in RAM in advance can be transmitted, and only data of the set number of bytes can be received and stored in RAM.

The 3-wire serial I/O mode with automatic transmit/receive function is controlled by the following registers.

- Serial operation mode specification register 0 (CSIMA0)
- Serial status register 0 (CSIS0)
- Serial trigger register 0 (CSIT0)
- Divisor selection register 0 (BRGCA0)
- Automatic data transfer address point specification register 0 (ADTP0)
- Automatic data transfer interval specification register 0 (ADTI0)

**Remark** For the alternate-function pin settings, refer to **Table 4-14 Settings When Port Pins Are Used for Alternate Functions**.

#### (1) Automatic transmit/receive data setting

##### (a) Transmit data setting

<1> Write transmit data from the least significant address FFFFFE00H of buffer RAM (up to FFFFFE1FH at maximum). The transmit data should be in the order from lower address to higher address.
<2> Set the ADTP0 register to the value obtained by subtracting 1 from the number of transmit data bytes.

##### (b) Automatic transmission/reception mode setting

<1> Set the CSIMA0.CSIAE0 bit and the CSIMA0.ATE0 bit to 11.
<2> Set the CSIMA0.RXEA0 bit and the CSIMA0.TXEA0 bit to 11.
<3> Set a data transfer interval in the ADTI0 register.
<4> Set the CSIT0.ATSTA0 bit to 1.

The following operations are automatically carried out when (a) and (b) are carried out.

- After the buffer RAM data indicated by the ADTC0 register (initial value: 00H) is transferred to the SIOA0 register, transmission is carried out (start of automatic transmission/reception).
- The received data is written to the buffer RAM address indicated by the ADTC0 register.
- The ADTC0 register is incremented and the next data transmission/reception is carried out. Data transmission/reception continues until the ADTC0 register incremental output matches the set value of the ADTP0 register (end of automatic transmission/reception). However, if the CSIMA0.ATM0 bit is set to 1 (continuous transfer mode), the ADTC0 register is cleared after a match between the ADTP0 and ADTC0 registers, and then repeated transmission/reception is started.
- When automatic transmission/reception is terminated, the CSIS0.TSF0 bit is cleared to 0.

**Caution Determine the setting procedure of alternate-function pins considering the relationship with the communication partner.**

### (2) Automatic transmission/reception communication operation

#### (a) Automatic transmission/reception mode

Automatic transmission/reception can be performed using buffer RAM.

The data stored in the buffer RAM is output from the SOA0 pin via the SIOA0 register in synchronization with the $\overline{\text{SCKA0}}$ pin falling edge by performing (a) and (b) in **(1) Automatic transmit/receive data setting**.

The data is then input from the SIA0 pin via the SIOA0 register in synchronization with the falling edge of the $\overline{\text{SCKA0}}$ pin and the receive data is stored in the buffer RAM in synchronization with the rising edge 1 clock later.

Data transfer ends if the CSIS0.TSF0 bit is cleared to 0 when any of the following conditions is met.

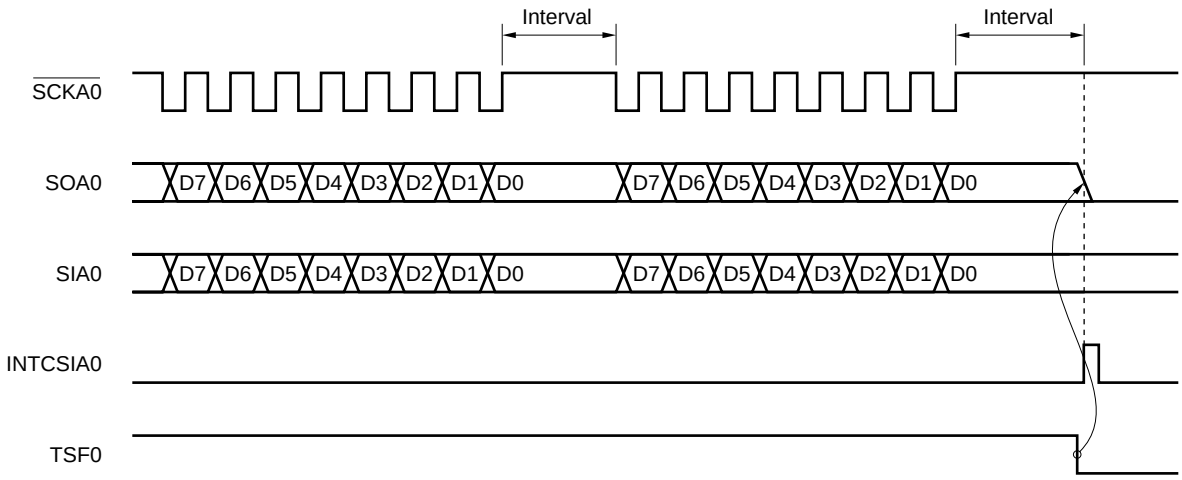
- Reset by clearing the CSIMA0.CSIAE0 bit to 0
- Transfer of 1 byte is complete by setting the CSIT0.ATSTP0 bit to 1
- Transfer of the range specified by the ADTP0 register is complete

At this time, a transmission/reception completion interrupt request signal (INTCSIA0) is generated except when the CSIAE0 bit = 0.

If a transfer is terminated in the middle, transfer starting from the remaining data is not possible. Read the ADTC0 register to confirm how much of the data has already been transferred, set the transfer data again, and perform (a) and (b) in **(1) Automatic transmit/receive data setting**.

Figure 17-5 shows the operation timing in automatic transmission/reception mode and Figure 17-6 shows the operation flowchart. Figure 17-7 shows the operation of the buffer RAM when 6 bytes of data are transmitted/received.

**Figure 17-5. Automatic Transmission/Reception Mode Operation Timings**

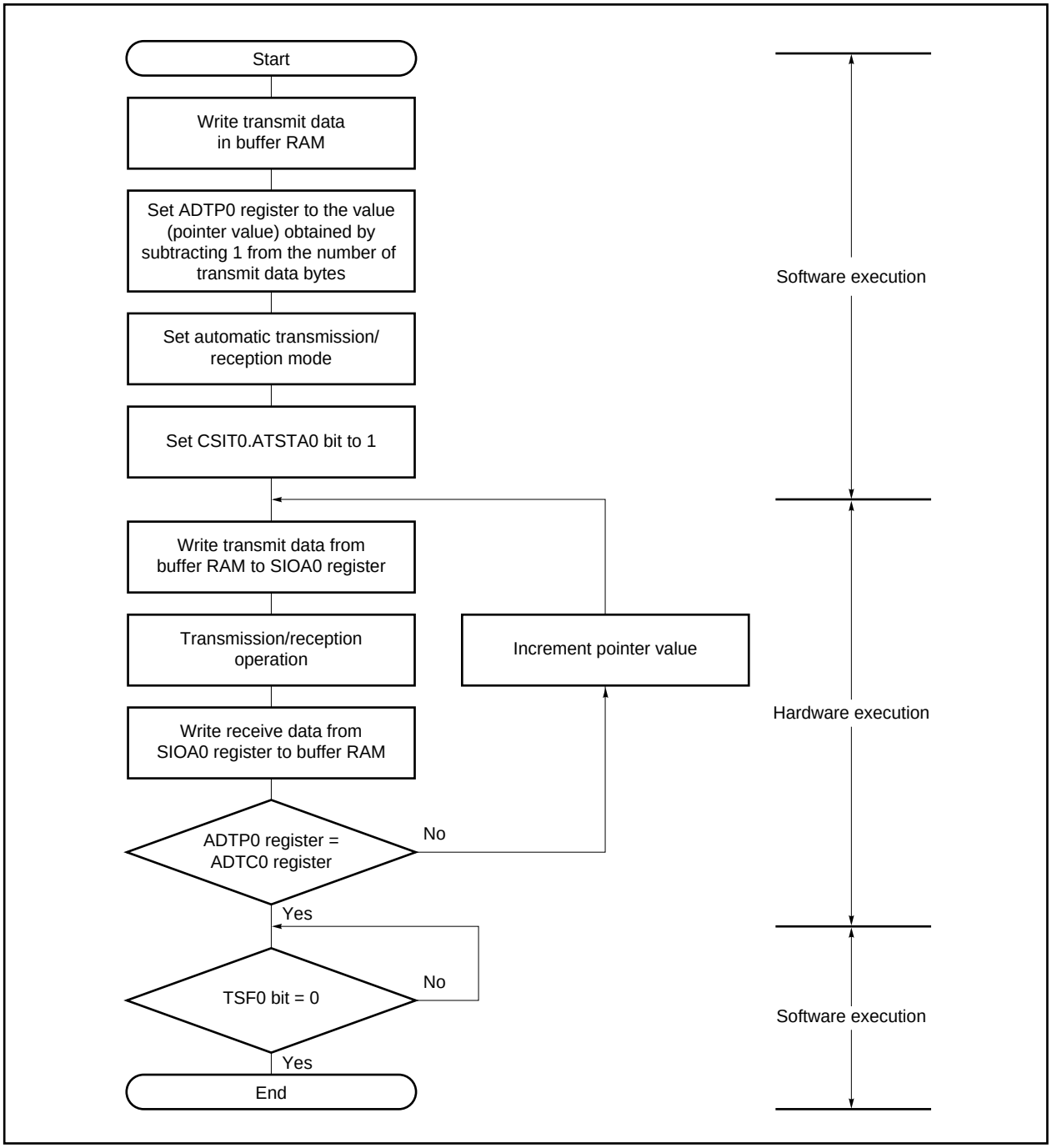



**Cautions 1. Because, in the automatic transmission/reception mode, the automatic transmit/receive function reads/writes data from/to the buffer RAM after 1-byte transmission/reception, an interval is inserted until the next transmission/reception. As the buffer RAM read/write is performed at the same time as CPU processing, the interval is dependent upon the value of the ADTI0 register.**

**2. When the TSF0 bit is cleared, the SOA0 pin becomes low level.**

**3. If CPU access to the buffer RAM conflicts with CSIA0 read/write during the interval time, the interval time becomes longer.**

**Figure 17-6. Automatic Transmission/Reception Mode Flowchart**



In 6-byte transmission/reception (CSIMA0.ATM0 bit = 0, CSIMA0.RXEA0 bit = 1, CSIMA0.TXEA0 bit = 1) in automatic transmission/reception mode, buffer RAM operates as follows.

**(i) When transmission/reception operation is started (refer to Figure 17-7 (a).)**

When the CSIT0.ATSTA0 bit is set to 1, transmit data 1 (T1) is transferred from the buffer RAM to the SIOA0 register. When transmission of the first byte is completed, receive data 1 (R1) is transferred from the SIOA0 register to the buffer RAM, and the ADTC0 register is incremented. Then transmit data 2 (T2) is transferred from the buffer RAM to the SIOA0 register.

**(ii) 4th byte transmission/reception point (refer to Figure 17-7 (b).)**

Transmission/reception of the third byte is completed, and transmit data 4 (T4) is transferred from the buffer RAM to the SIOA0 register. When transmission of the fourth byte is completed, the receive data 4 (R4) is transferred from the SIOA0 register to the buffer RAM, and the value of the ADTC0 register is incremented.

**(iii) Completion of transmission/reception (refer to Figure 17-7 (c).)**

When transmission of the sixth byte is completed, receive data 6 (R6) is transferred from the SIOA0 register to the buffer RAM, and the transmission/reception completion interrupt request signal (INTCSIA0) is generated.

**Figure 17-7. Buffer RAM Operation in 6-Byte Transmission/Reception (in Automatic Transmission/Reception Mode) (1/2)**

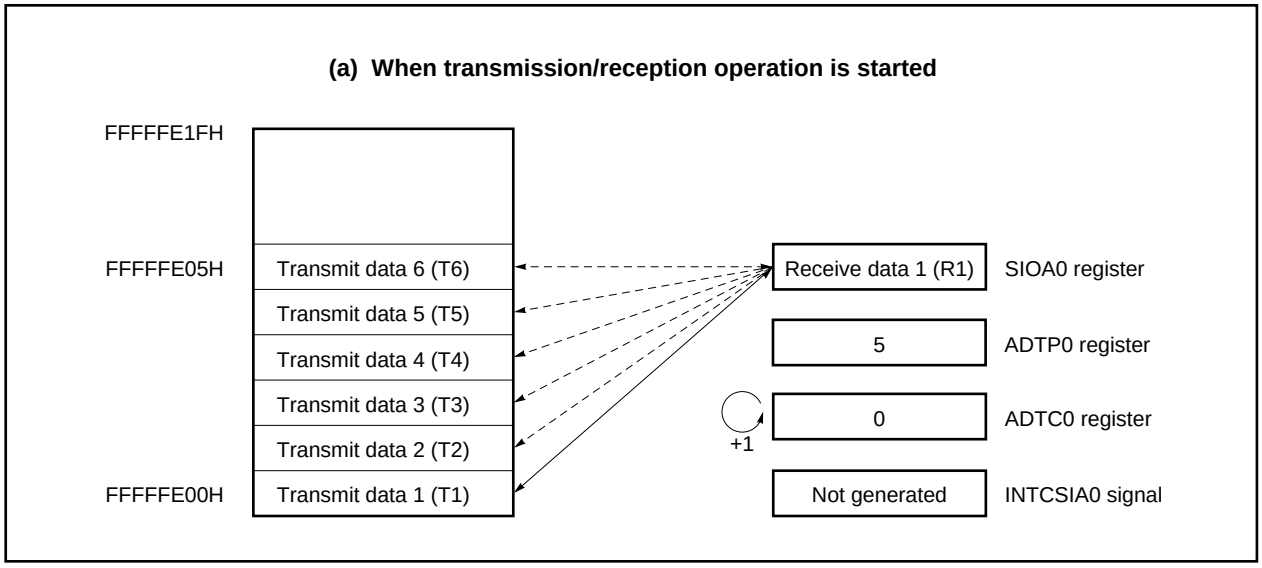
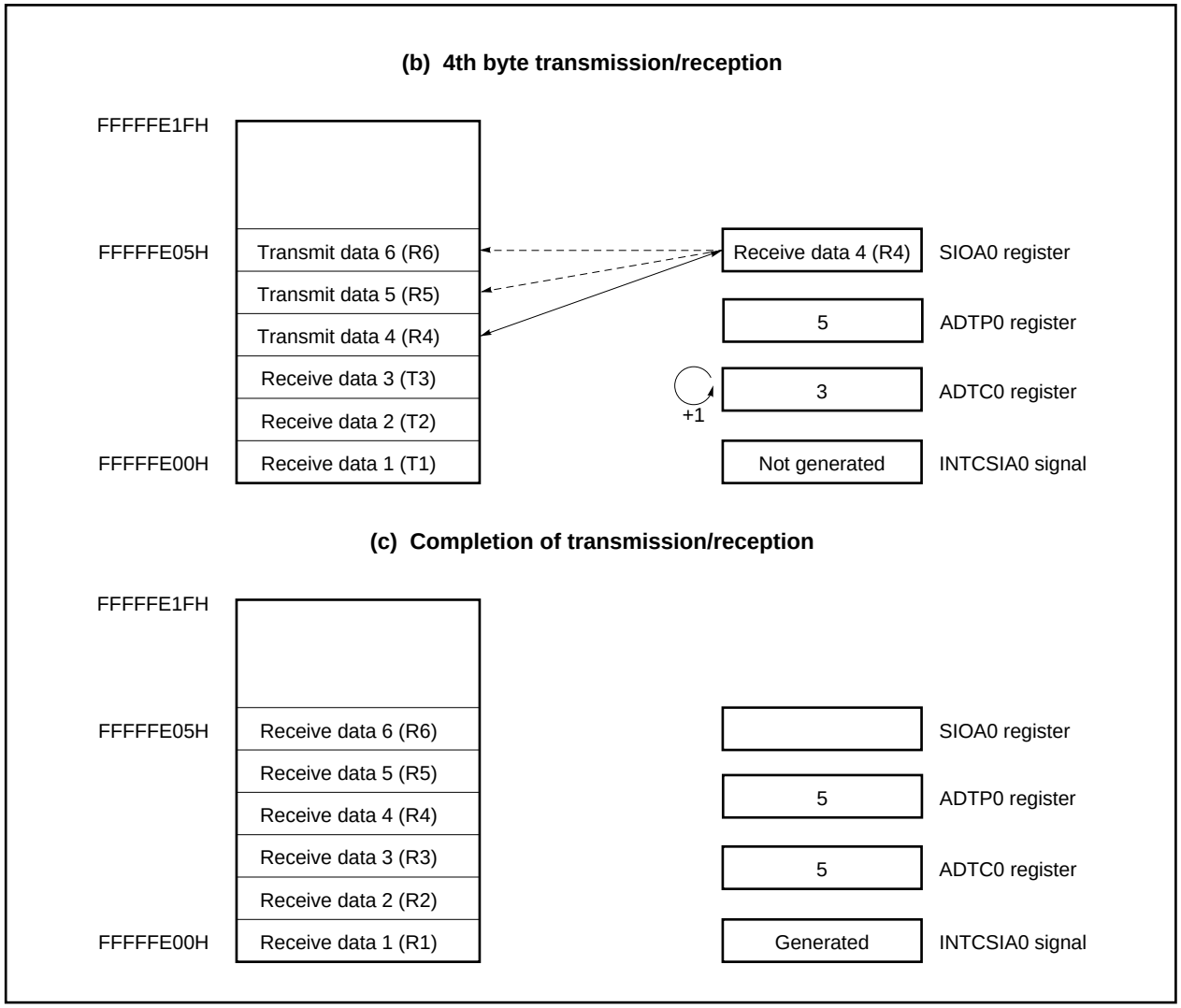


**Figure 17-7. Buffer RAM Operation in 6-Byte Transmission/Reception (in Automatic Transmission/Reception Mode) (2/2)**



**(b) Automatic transmission mode**

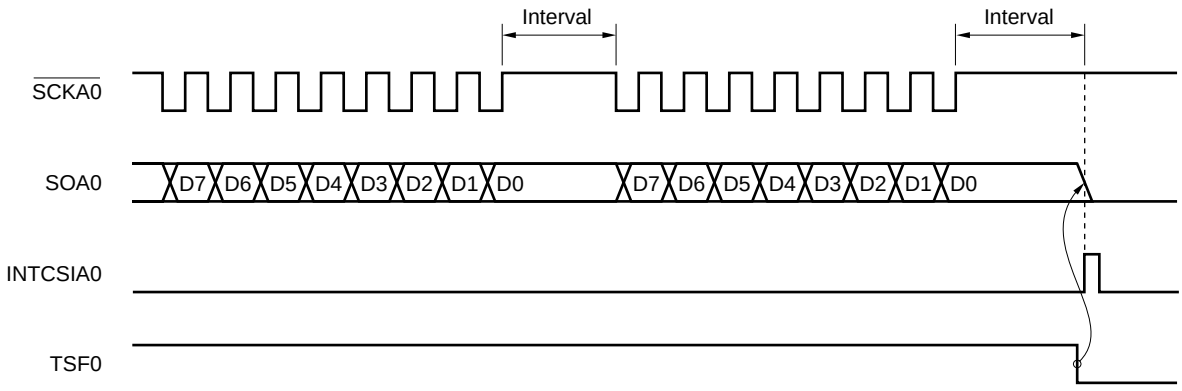
In this mode, the specified number of 8-bit unit data are transmitted.

Serial transfer is started when the CSIT0.ATSTA0 bit is set to 1 while the CSIMA0.CSIAE0, CSIMA0.ATE0, and CSIMA0.TXEA0 bits are set to 1.

When the final byte has been transmitted, an interrupt request signal (INTCSIA0) is generated.

Figure 17-8 shows the automatic transmission mode operation timing, and Figure 17-9 shows the operation flowchart. Figure 17-10 shows the operation of the buffer RAM when 6 bytes of data are transmitted.

**Figure 17-8. Automatic Transmission Mode Operation Timing**

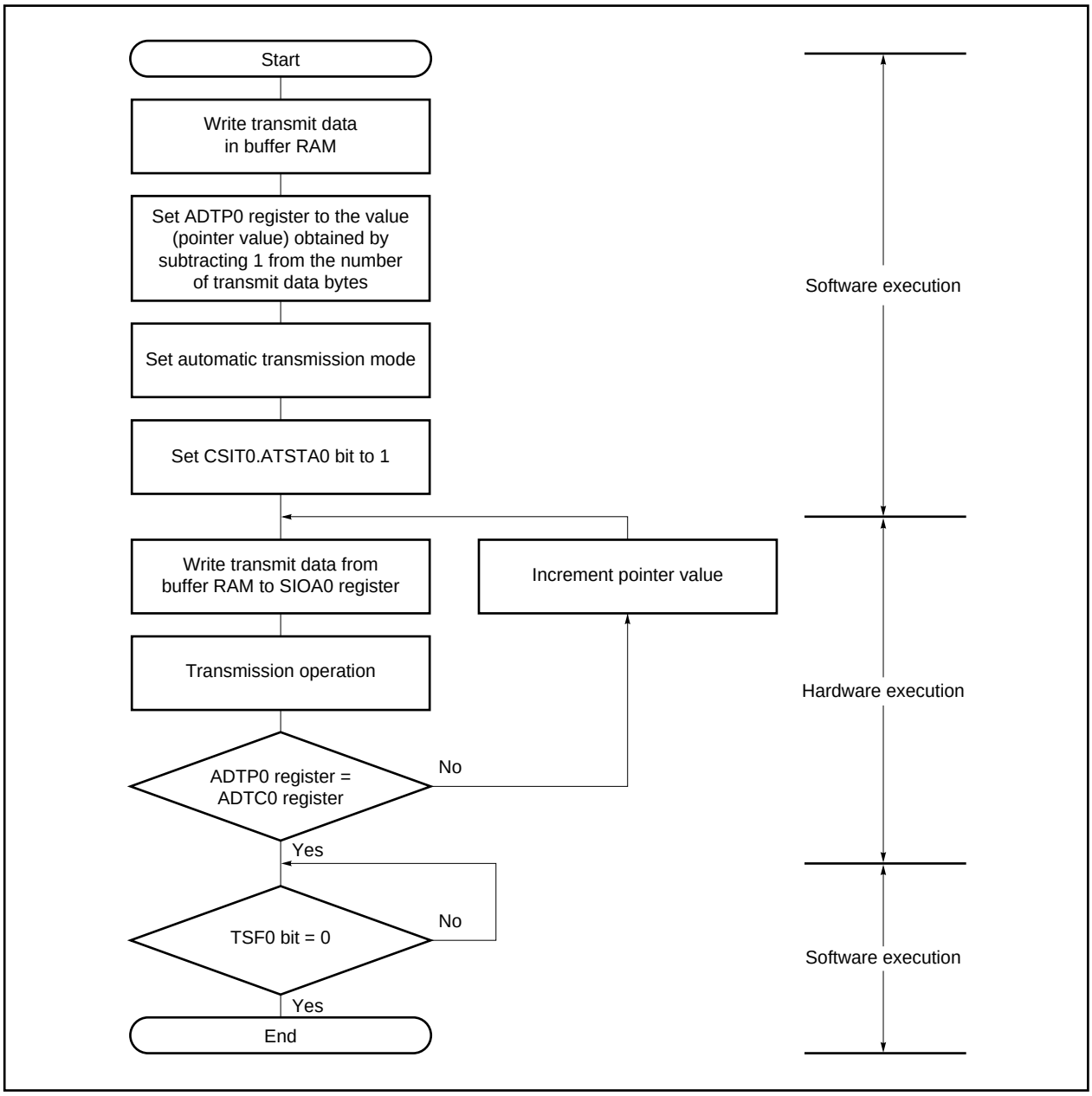



**Cautions 1. Because, in the automatic transmission mode, the automatic transmit/receive function reads data from the buffer RAM after 1-byte transmission, an interval is inserted until the next transmission. As the buffer RAM read is performed at the same time as CPU processing, the interval is dependent upon the value of the ADTI0 register.**

**2. When the TSF0 bit is cleared, the SOA0 pin becomes low level.**

**3. If CPU access to the buffer RAM conflicts with CSIA0 read/write during the interval time, the interval time becomes longer.**

**Figure 17-9. Automatic Transmission Mode Flowchart**



In 6-byte transmission (CSIMA0.ATM0 bit = 0, CSIMA0.RXEA0 bit = 0, CSIMA0.TXEA0 bit = 1, CSIMA0.ATE0 bit = 1) in automatic transmission mode, buffer RAM operates as follows.

**(i) When transmission is started (refer to Figure 17-10 (a).)**

When the CSIT0.ATSTA0 bit is set to 1, transmit data 1 (T1) is transferred from the buffer RAM to the SIOA0 register. When transmission of the first byte is completed, the ADTC0 register is incremented. Then transmit data 2 (T2) is transferred from the buffer RAM to the SIOA0 register.

**(ii) 4th byte transmission point (refer to Figure 17-10 (b).)**

Transmission of the third byte is completed, and transmit data 4 (T4) is transferred from the buffer RAM to the SIOA0 register. When transmission of the fourth byte is completed, the value of the ADTC0 register is incremented.

**(iii) Completion of transmission (refer to Figure 17-10 (c).)**

When transmission of the sixth byte is completed, the interrupt request signal (INTCSIA0) is generated, and the TFS0 flag is cleared to 0.

**Figure 17-10. Buffer RAM Operation in 6-Byte Transmission (in Automatic Transmission Mode) (1/2)**

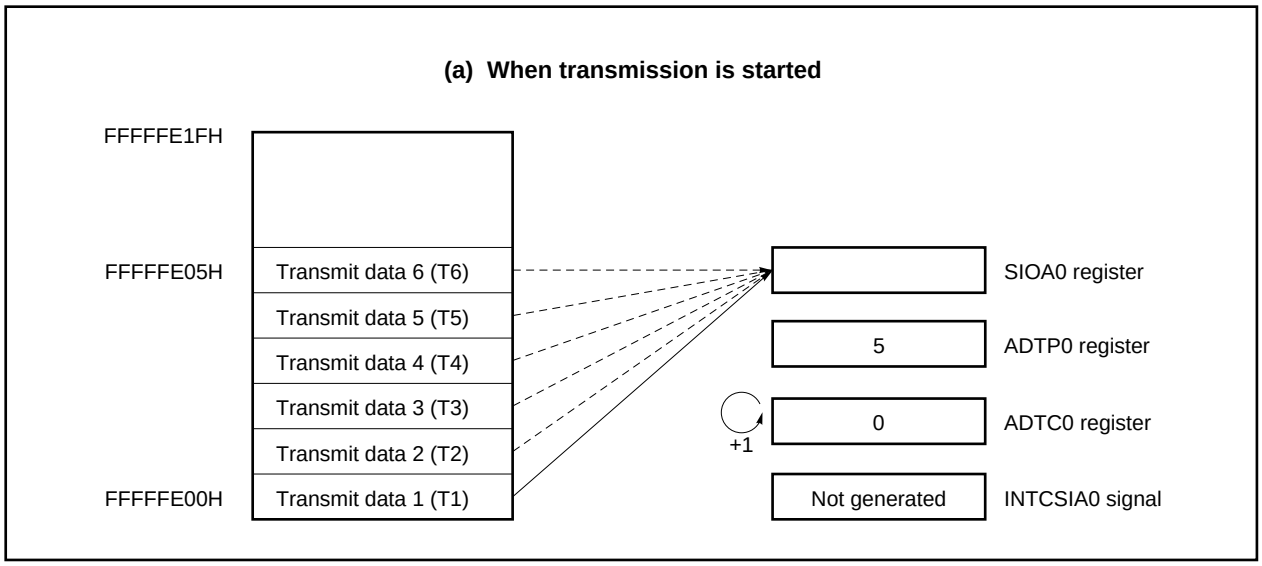
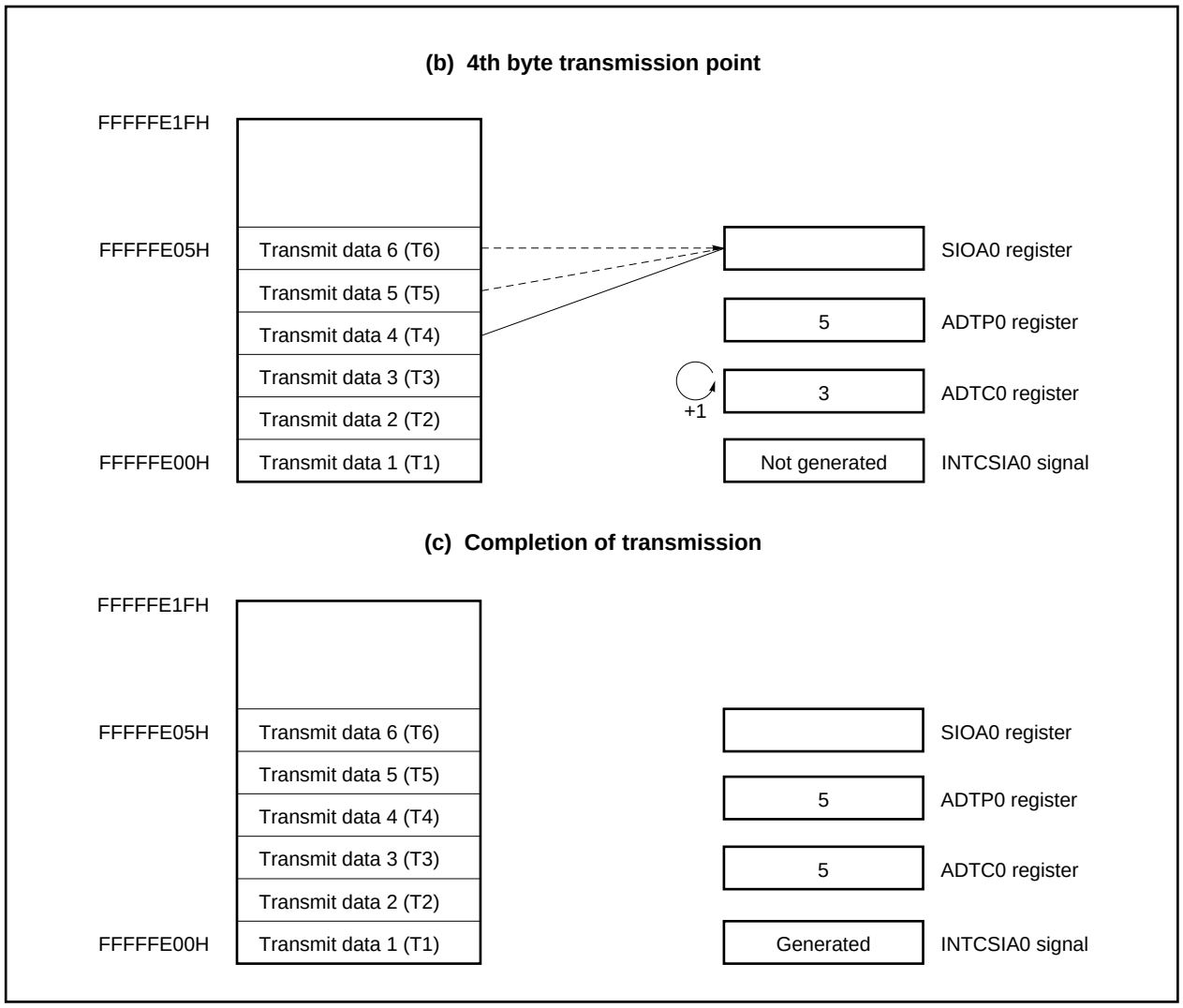


**Figure 17-10. Buffer RAM Operation in 6-Byte Transmission (in Automatic Transmission Mode) (2/2)**



**(c) Repeat transmission mode**

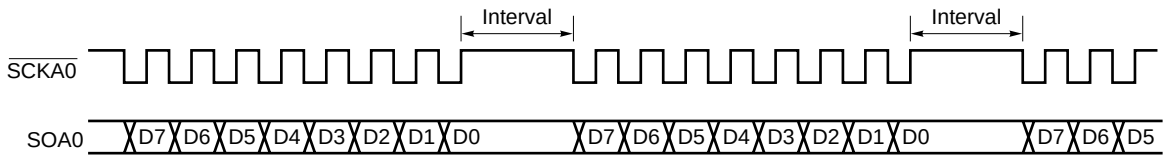
In this mode, data stored in the buffer RAM is transmitted repeatedly.

Serial transfer is started when the CSIT0.ATSTA0 bit is set to 1 while the CSIMA0.CSIAE0, CSIMA0.ATE0, CSIMA0.ATM0, and CSIMA0.TXEA0 bits are set to 1.

Unlike the basic transmission mode, after the specified number of bytes has been transmitted, the transmission/reception completion interrupt request signal (INTCSIA0) is not generated, the ADTC0 register is reset to 0, and the buffer RAM contents are transmitted again.

The repeat transmission mode operation timing is shown in Figure 17-11, and the operation flowchart in Figure 17-12. Figure 17-13 shows the operation of the buffer RAM when 6 bytes of data are transmitted in the repeat transmission mode.

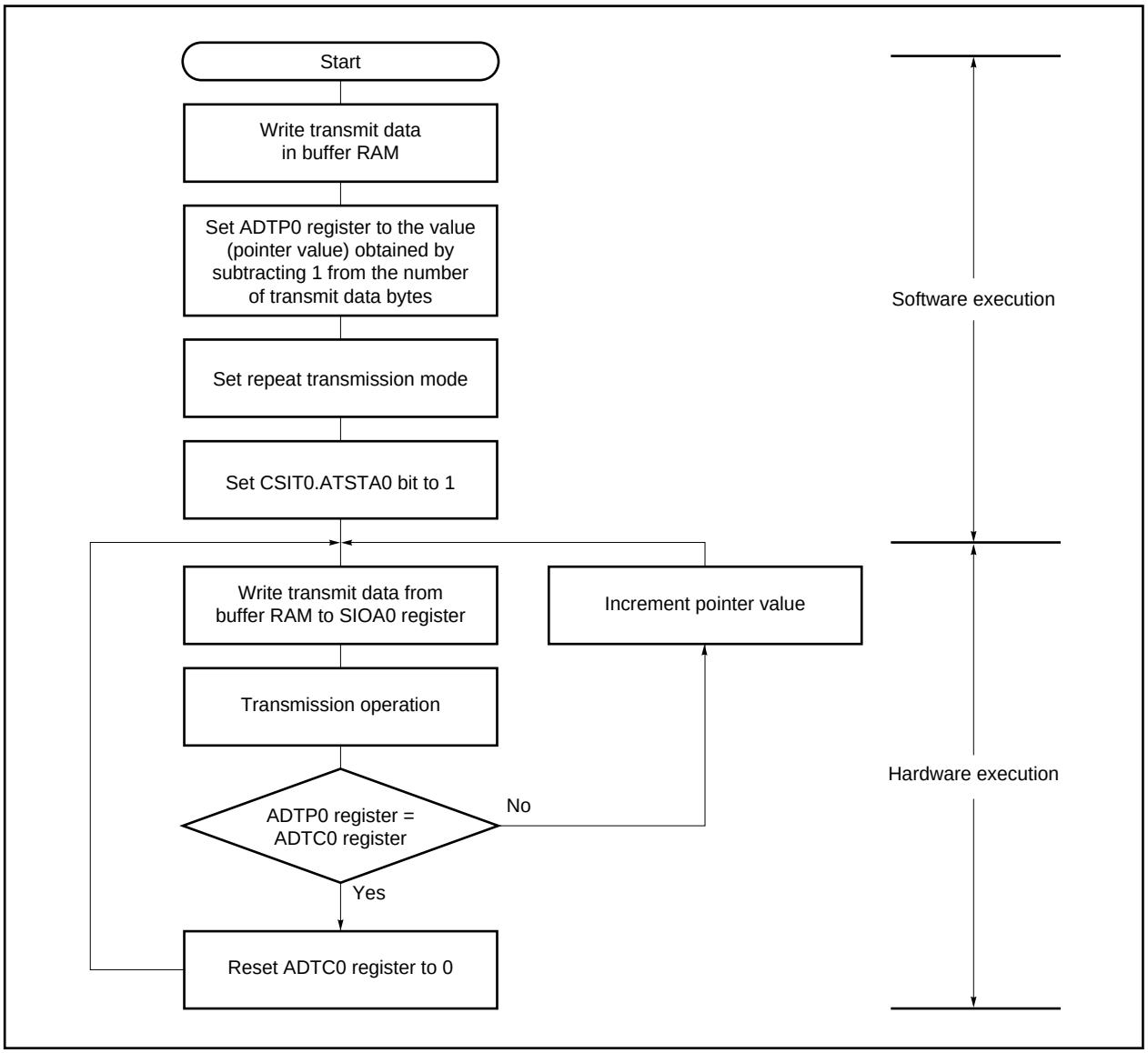
**Figure 17-11. Repeat Transmission Mode Operation Timing**




**Cautions 1. Because, in the repeat transmission mode, a read is performed on the buffer RAM after the transmission of one byte, the interval is included in the period up to the next transmission. As the buffer RAM read is performed at the same time as CPU processing, the interval is dependent upon the value of the ADTI0 register.**

**2. If CPU access to the buffer RAM conflicts with CSIA0 read/write during the interval time, the interval time becomes longer.**

**Figure 17-12. Repeat Transmission Mode Flowchart**



In 6-byte transmission (CSIMA0.ATM0 bit = 1, CSIMA0.RXEA0 bit = 0, CSIMA0.TXEA0 bit = 1, CSIMA0.ATE0 bit = 1) in repeat transmission mode, buffer RAM operates as follows.

**(i) When transmission is started (refer to Figure 17-13 (a).)**

When the CSIT0.ATSTA0 bit is set to 1, transmit data 1 (T1) is transferred from the buffer RAM to the SIOA0 register. When transmission of the first byte is completed, the value of the ADTC0 register is incremented. Then transmit data 2 (T2) is transferred from the buffer RAM to the SIOA0 register.

**(ii) Upon completion of transmission of 6 bytes (refer to Figure 17-13 (b).)**

When transmission of the sixth byte is completed, the interrupt request signal (INTCSIA0) is not generated.

The ADTC0 register is reset to 0.

**(iii) 7th byte transmission point (refer to Figure 17-13 (c).)**

Transmit data 1 (T1) is transferred from the buffer RAM to the SIOA0 register again. When transmission of the first byte is completed, the value of the ADTC0 register is incremented. Then transmit data 2 (T2) is transferred from the buffer RAM to the SIOA0 register.

**Figure 17-13. Buffer RAM Operation in 6-Byte Transmission (in Repeat Transmission Mode) (1/2)**

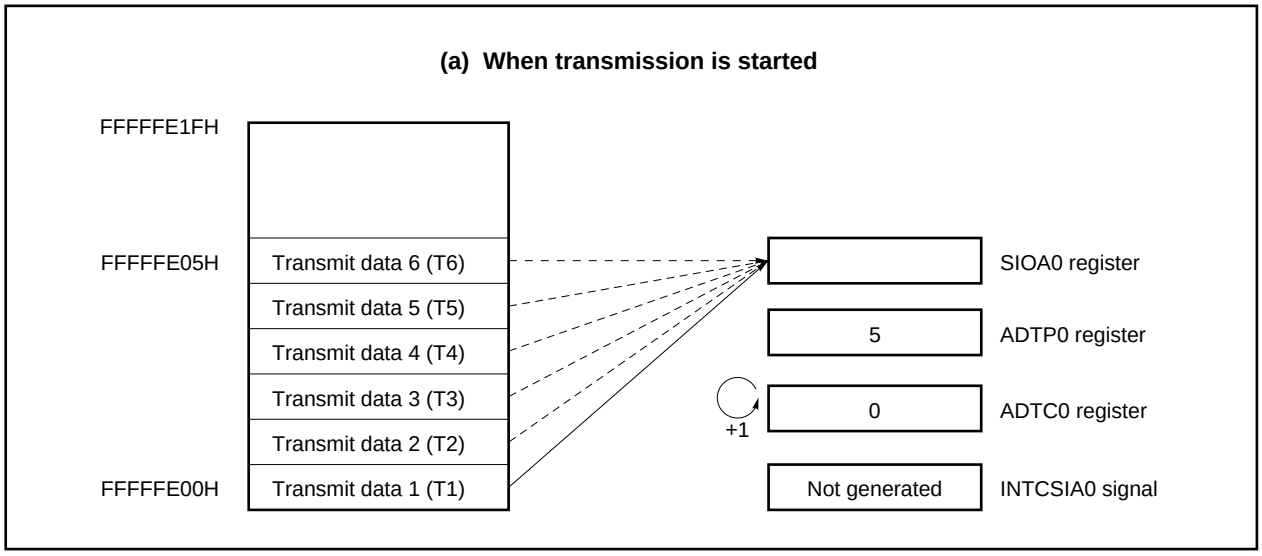
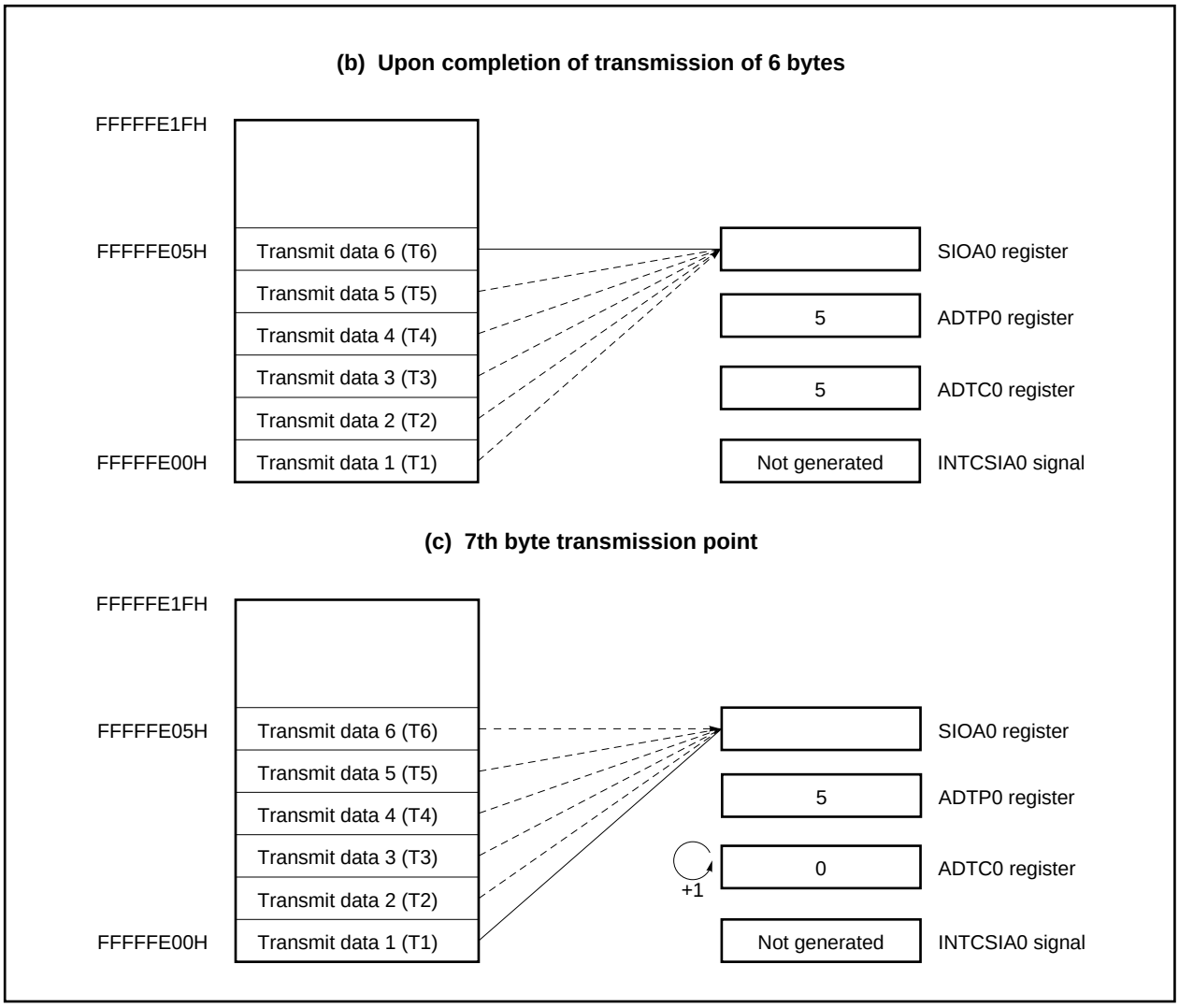


**Figure 17-13. Buffer RAM Operation in 6-Byte Transmission (in Repeat Transmission Mode) (2/2)**

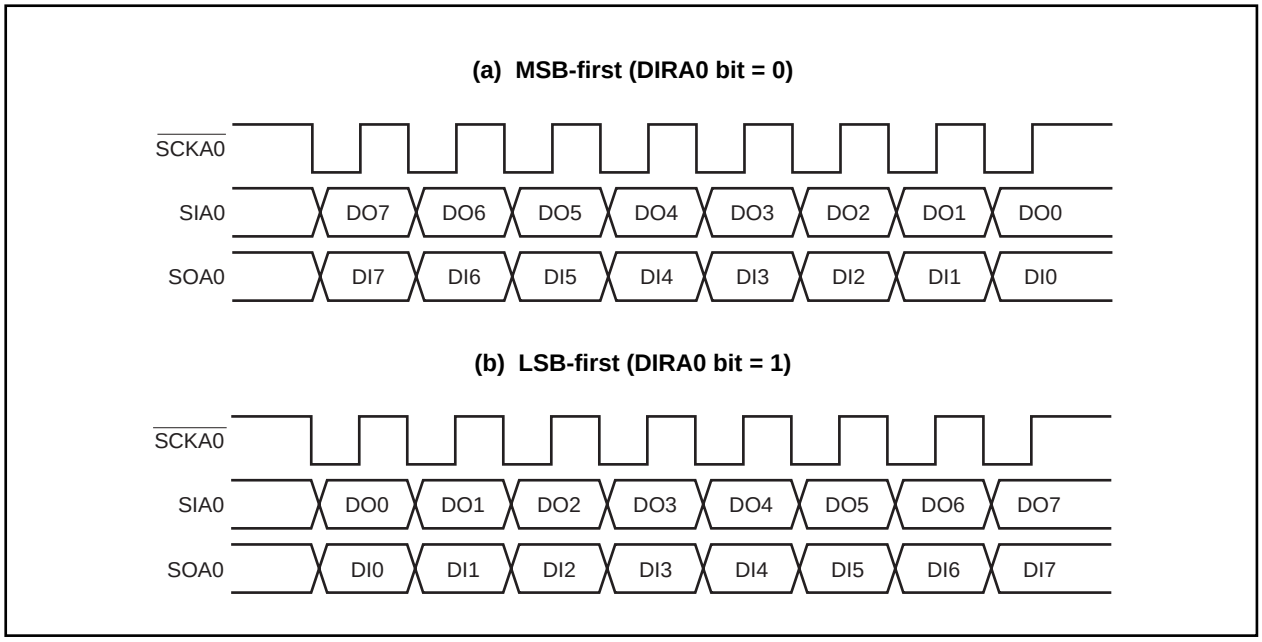


**(d) Data format**

In the data format, data is changed in synchronization with the $\overline{\text{SCKA0}}$ pin falling edge as shown in Figure 17-14.

The data length is fixed to 8 bits and the data transfer direction can be switched by the specification of the CSIMA0.DIRA0 bit.

**Figure 17-14. Format of CSIA0 Transmit/Receive Data**



**(e) Automatic transmission/reception suspension and restart**

Automatic transmission/reception can be temporarily suspended by setting the CSIT0.ATSTP0 bit to 1. During 8-bit data transfer, the transmission/reception is not suspended. It is suspended upon completion of 8-bit data transfer.

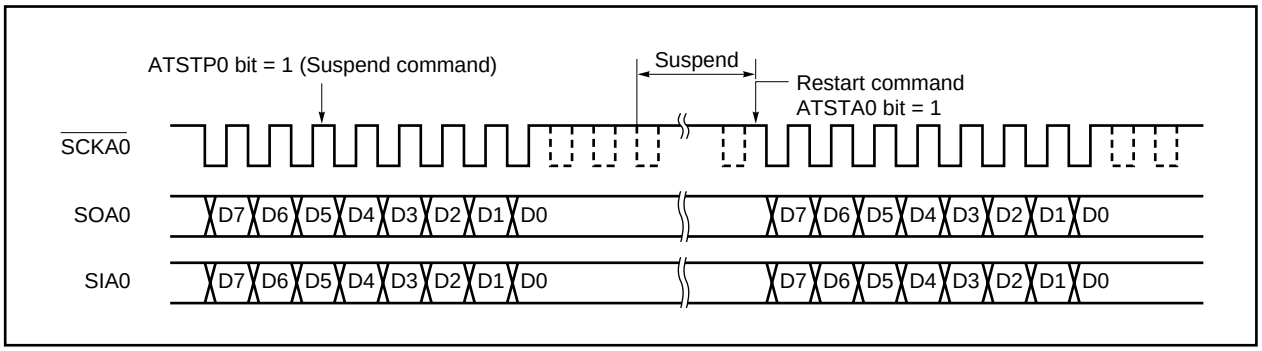
When suspended, the CSIS0.TSF0 bit is cleared to 0 after transfer of the 8th bit.

To restart automatic transmission/reception, set the CSIT0.ATSTA0 bit to 1. The remaining data can be transmitted in this way.

**Cautions 1. If the IDLE instruction is executed during automatic transmission/reception, transfer is suspended and the IDLE mode is set if during 8-bit data transfer. When the IDLE mode is cleared, automatic transmission/reception is restarted from the suspended point.**

**2. When suspending automatic transmission/reception, do not change the operating mode to 3-wire serial I/O mode while the TSF0 bit = 1.**

**Figure 17-15. Automatic Transmission/Reception Suspension and Restart**



# CHAPTER 18 I²C BUS

To use the I²C bus function, set the P38/SDA0 and P39/SCL0 pins to N-ch open drain output as the alternate function.

In the V850ES/KF1+, one channel of I²C bus is provided.
The products with an on-chip I²C bus are shown below.

μPD703308Y, 70F3306Y, 70F3308Y

## 18.1 Features

The I²C0 has the following two modes.

- Operation stop mode
- I²C (Inter IC) bus mode (multimaster supported)

**(1) Operation stop mode**

This mode is used when serial transfers are not performed. It can therefore be used to reduce power consumption.

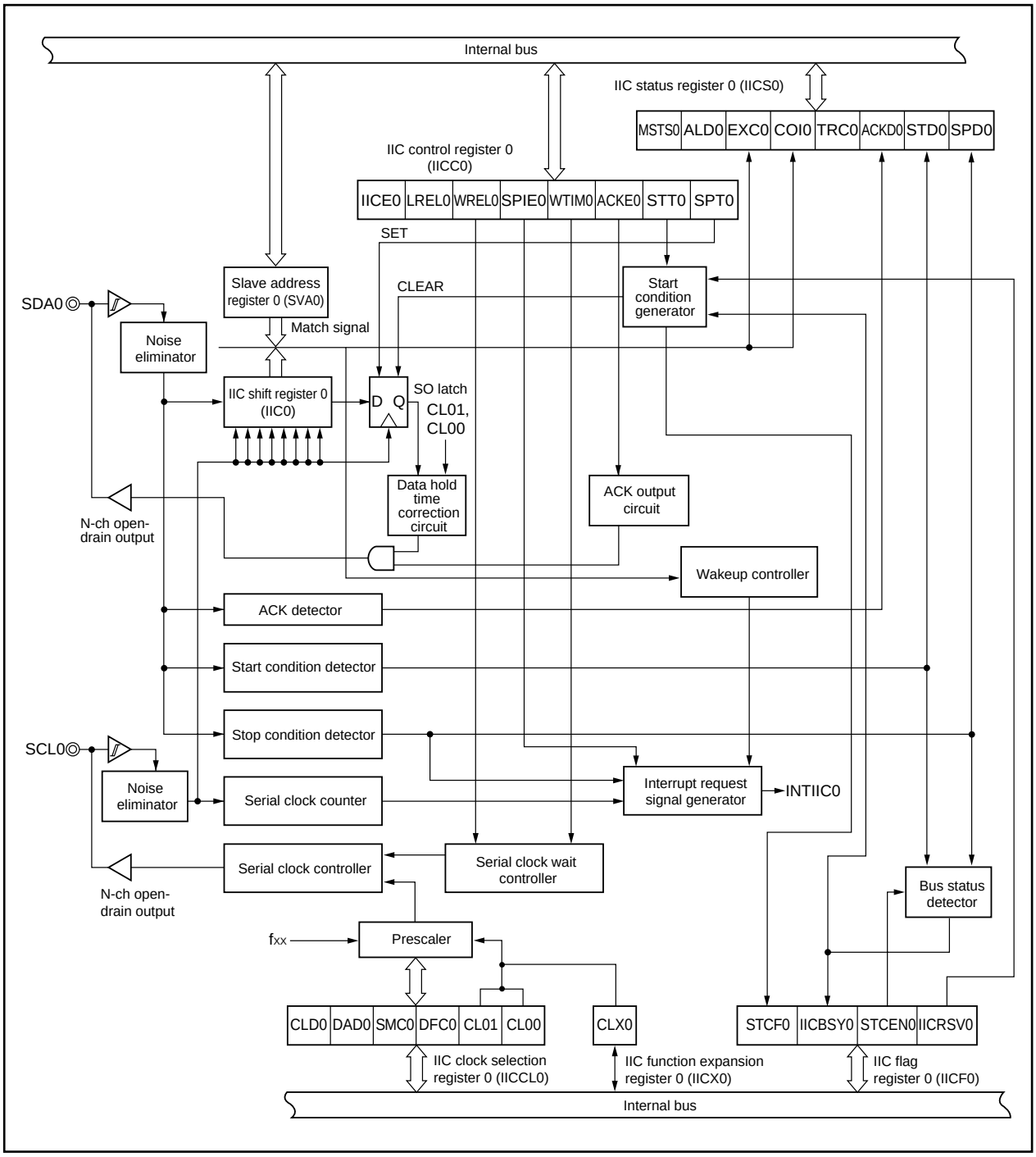
**(2) I²C bus mode (multimaster supported)**

This mode is used for 8-bit data transfers with several devices via two lines: a serial clock (SCL0) line and a serial data bus (SDA0) line.

This mode complies with the I²C bus format and the master device can output “start condition”, “data”, and “stop condition” data to the slave device, via the serial data bus. The slave device automatically detects these received data by hardware. This function can simplify the part of application program that controls the I²C bus.

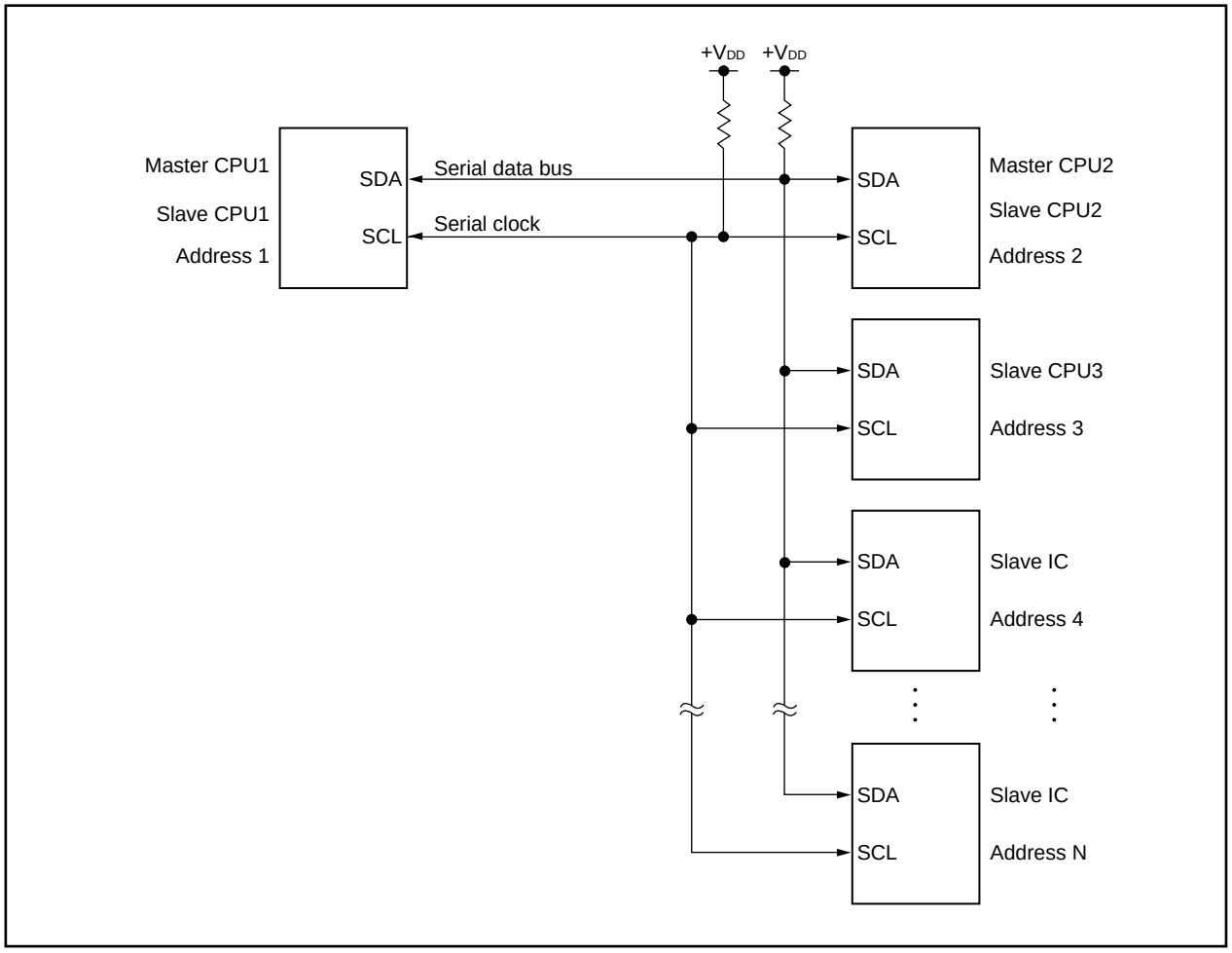
Since the SCL0 and SDA0 pins are used for N-ch open drain outputs, I²C0 requires pull-up resistors for the serial clock line and the serial data bus line.

**Figure 18-1. Block Diagram of I²C0**



A serial bus configuration example is shown below.

**Figure 18-2. Serial Bus Configuration Example Using I²C Bus**



## 18.2 Configuration

I²C0 includes the following hardware.

**Table 18-1. Configuration of I²C0**

| Item | Configuration |
|---|---|
| Registers | IIC shift register 0 (IIC0)<br>Slave address register 0 (SVA0) |
| Control registers | IIC control register 0 (IICC0)<br>IIC status register 0 (IICS0)<br>IIC flag register 0 (IICCF0)<br>IIC clock selection register 0 (IICCL0)<br>IIC function expansion register 0 (IICX0) |

**(1) IIC shift register 0 (IIC0)**

The IIC0 register is used to convert 8-bit serial data to 8-bit parallel data and to convert 8-bit parallel data to 8-bit serial data. The IIC0 register can be used for both transmission and reception.

Write and read operations to the IIC0 register are used to control the actual transmit and receive operations.

The IIC0 register can be read or written in 8-bit units.

After reset, IIC0 is cleared to 00H.

**(2) Slave address register 0 (SVA0)**

The SVA0 register sets local addresses when in slave mode.

The SVA0 register can be read or written in 8-bit units.

After reset, SVA0 is cleared to 00H.

**(3) SO latch**

The SO latch is used to retain the SDA0 pin's output level.

**(4) Wakeup controller**

This circuit generates an interrupt request signal (INTIIC0) when the address received by this register matches the address value set to the SVA0 register or when an extension code is received.

**(5) Prescaler**

This selects the sampling clock to be used.

**(6) Serial clock counter**

This counter counts the serial clocks that are output and the serial clocks that are input during transmit/receive operations and is used to verify that 8-bit data was sent or received.

**(7) Interrupt request signal generator**

This circuit controls the generation of interrupt request signals (INTIIC0).

An I²C interrupt is generated by the following two triggers.

- Falling edge of the eighth or ninth clock of the serial clock (set by IICC0.WTIM0 bit)
- Interrupt request generated when a stop condition is detected (set by IICC0.SPIE0 bit)

**(8) Serial clock controller**

In master mode, this circuit generates the clock output via the SCL0 pin from a sampling clock.

**(9) Serial clock wait controller**

This circuit controls the wait timing.

**(10) ACK output circuit, stop condition detector, start condition detector, and ACK detector**

These circuits are used to output and detect various control signals.

**(11) Data hold time correction circuit**

This circuit generates the hold time for data corresponding to the falling edge of the serial clock.

**(12) Start condition generator**

This circuit generates a start condition when the IICC0.STT0 bit is set.

However, in the communication reservation disabled status (IICF0.IICRSV0 bit = 1), when the bus is not released (IICF0.IICBSY0 bit = 1), start condition requests are ignored and the IICF0.STCF0 bit is set to 1.

**(13) Bus status detector**

This circuit detects whether or not the bus is released by detecting start conditions and stop conditions.

However, as the bus status cannot be detected immediately following operation, the initial status is set by the IICF0.STCEN0 bit.

## 18.3 Registers

I²C0 is controlled by the following registers.

- IIC control register 0 (IICC0)
- IIC status register 0 (IICS0)
- IIC flag register 0 (IICF0)
- IIC clock selection register 0 (IICCL0)
- IIC function expansion register 0 (IICX0)

The following registers are also used.

- IIC shift register 0 (IIC0)
- Slave address register 0 (SVA0)

**Remark** For the alternate-function pin settings, refer to **Table 4-14 Settings When Port Pins Are Used for Alternate Functions**.

**(1) IIC control register 0 (IICC0)**

The IICC0 register is used to enable/stop I²C0 operations, set wait timing, and set other I²C operations.

The IICC0 register can be read or written in 8-bit or 1-bit units.

After reset, IICC0 is cleared to 00H.

(1/4)

After reset: 00H R/W Address: FFFFFD82H

| | <7> | <6> | <5> | <4> | <3> | <2> | <1> | <0> |
|---|---|---|---|---|---|---|---|---|
| IICC0 | IICE0 | LREL0 | WREL0 | SPIE0 | WTIM0 | ACKE0 | STT0 | SPT0 |

| IICE0 | I²C0 operation enable/disable specification |
|---|---|
| 0 | Stop operation. Reset the IICS0 register[Note 1]. Stop internal operation. |
| 1 | Enable operation. |

| Condition for clearing (IICE0 bit = 0) | Condition for setting (IICE0 bit = 1) |
|---|---|
| • Cleared by instruction<br>• Reset | • Set by instruction |

| LREL0 | Exit from communications |
|---|---|
| 0 | Normal operation |
| 1 | This exits from the current communications and sets standby mode. This setting is automatically cleared to 0 after being executed.<br>Its uses include cases in which a locally irrelevant extension code has been received.<br>The SCL0 and SDA0 lines are set to high impedance.<br>The STT0, SPT0, IICS0.MSTS0, IICS0.EXC0, IICS0.COI0, IICS0.TRC0, IICS0.ACKD0, and IICS0.STD0 bits are cleared to 0. |
| The standby mode following exit from communications remains in effect until the following communications entry conditions are met.<br>• After a stop condition is detected, restart is in master mode.<br>• An address match or extension code reception occurs after the start condition. | |

| Condition for clearing (LREL0 bit = 0)[Note 2] | Condition for setting (LREL0 bit = 1) |
|---|---|
| • Automatically cleared after execution<br>• Reset | • Set by instruction |

| WREL0 | Wait cancellation control |
|---|---|
| 0 | Do not cancel wait |
| 1 | Cancel wait. This setting is automatically cleared to 0 after wait is canceled. |

| Condition for clearing (WREL0 bit = 0)[Note 2] | Condition for setting (WREL0 bit = 1) |
|---|---|
| • Automatically cleared after execution<br>• Reset | • Set by instruction |

**Notes 1.** The IICS0 register, and the IICF0.STCF0, IICF0.IICBSY0, IICCL0.CLD0, and IICCL0.DAD0 bits are reset.

**2.** This flag's signal is invalid when the IICE0 bit = 0.

(2/4)

| SPIE0 | Enable/disable generation of interrupt request when stop condition is detected |
|---|---|
| 0 | Disable |
| 1 | Enable |

| Condition for clearing (SPIE0 bit = 0)[Note] | Condition for setting (SPIE0 bit = 1) |
|---|---|
| • Cleared by instruction<br>• Reset | • Set by instruction |

| WTIM0 | Control of wait and interrupt request generation |
|---|---|
| 0 | Interrupt request is generated at the eighth clock's falling edge.<br>Master mode: After output of eight clocks, clock output is set to low level and wait is set.<br>Slave mode: After input of eight clocks, the clock is set to low level and wait is set for master device. |
| 1 | Interrupt request is generated at the ninth clock's falling edge.<br>Master mode: After output of nine clocks, clock output is set to low level and wait is set.<br>Slave mode: After input of nine clocks, the clock is set to low level and wait is set for master device. |

An interrupt is generated at the falling edge of the ninth clock during address transfer independently of the setting of this bit. The setting of this bit is valid when the address transfer is completed. When in master mode, a wait is inserted at the falling edge of the ninth clock during address transfers. For a slave device that has received a local address, a wait is inserted at the falling edge of the ninth clock after an acknowledge signal ($\overline{ACK}$) is issued. However, when the slave device has received an extension code, a wait is inserted at the falling edge of the eighth clock.

| Condition for clearing (WTIM0 bit = 0)[Note] | Condition for setting (WTIM0 bit = 1) |
|---|---|
| • Cleared by instruction<br>• Reset | • Set by instruction |

| ACKE0 | Acknowledgment control |
|---|---|
| 0 | Disable acknowledgment. |
| 1 | Enable acknowledgment. During the ninth clock period, the SDA0 line is set to low level. However, $\overline{ACK}$ is invalid during address transfers and other than in expansion mode. |

| Condition for clearing (ACKE0 bit = 0)[Note] | Condition for setting (ACKE0 bit = 1) |
|---|---|
| • Cleared by instruction<br>• Reset | • Set by instruction |

**Note** This flag's signal is invalid when the IICE0 bit = 0.

(3/4)

| STT0 | Start condition trigger |
|---|---|
| 0 | Do not generate a start condition. |
| 1 | When bus is released (in STOP mode):<br>Generate a start condition (for starting as master). The SDA0 line is changed from high level to low level and then the start condition is generated. Next, after the rated amount of time has elapsed, the SCL0 line is changed to low level.<br>When a third party is communicating:<br>• When communication reservation function is enabled (IICF0.IICRSV0 bit = 0)<br>Functions as the start condition reservation flag. When set to 1, automatically generates a start condition after the bus is released.<br>• When communication reservation function is disabled (IICRSV0 bit = 1)<br>The IICF0.STCF0 bit is set to 1. No start condition is generated.<br>In the wait state (when master device):<br>Generates a restart condition after releasing the wait. |

Cautions concerning set timing

For master reception: Cannot be set to 1 during transfer. Can be set to 1 only when the ACKE0 bit has been cleared to 0 and slave has been notified of final reception.

For master transmission: A start condition cannot be generated normally during the ACK0 period. Set to 1 during the wait period.

• Cannot be set to 1 at the same time as the SPT0 bit.

| Condition for clearing (STT0 bit = 0)[Note] | Condition for setting (STT0 bit = 1) |
|---|---|
| • Cleared by loss in arbitration<br>• Cleared after start condition is generated by master device<br>• When the LREL0 bit = 1 (exit from communications)<br>• When the IICE0 bit = 0 (operation stop)<br>• Reset | • Set by instruction |

**Note** This flag's signal is invalid when the IICE0 bit = 0.

**Remark** The STT0 bit is 0 if it is read after data setting.

(4/4)

| SPT0 | Stop condition trigger |
|---|---|
| 0 | Stop condition is not generated. |
| 1 | Stop condition is generated (termination of master device's transfer).<br>After the SDA0 line goes to low level, either set the SCL0 line to high level or wait until the SCL0 pin goes to high level. Next, after the rated amount of time has elapsed, the SDA0 line is changed from low level to high level and a stop condition is generated. |

Cautions concerning set timing

For master reception: Cannot be set to 1 during transfer. Can be set to 1 only when the ACKE0 bit has been cleared to 0 and during the wait period after slave has been notified of final reception.

For master transmission: A stop condition cannot be generated normally during the $\overline{\text{ACK}}$ signal period. Set to 1 during the wait period.

- Cannot be set to 1 at the same time as the STT0 bit.
  The SPT0 bit can be set to 1 only when in master mode[Note 1].
  When the WTIM0 bit has been cleared to 0, if the SPT0 bit is set to 1 during the wait period that follows output of eight clocks, note that a stop condition will be generated during the high-level period of the ninth clock.
  When a ninth clock must be output, the WTIM0 bit should be set from 0 to 1 during the wait period following output of eight clocks, and the SPT0 bit should be set to 1 during the wait period that follows output of the ninth clock.

| Condition for clearing (SPT0 bit = 0)[Note 2] | Condition for setting (SPT0 bit = 1) |
|---|---|
| • Cleared by loss in arbitration<br>• Automatically cleared after stop condition is detected<br>• When the LREL0 bit = 1 (exit from communications)<br>• When the IICE0 bit = 0 (operation stop)<br>• Reset | • Set by instruction |

**Notes 1.** Set the SPT0 bit to 1 only in master mode. However, the SPT0 bit must be set to 1 and a stop condition generated before the first stop condition is detected following the switch to operation enable status. For details, refer to **18.14 Cautions**.

**2.** This flag's signal is invalid when the IICE0 bit = 0.

**Caution When the IICS0.TRC0 bit is set to 1, the WREL0 bit is set to 1 during the ninth clock and wait is canceled, after which the TRC0 bit is cleared to 0 and the SDA0 line is set to high impedance.**

**Remark** The SPT0 bit is 0 if it is read after data setting.

**(2) IIC status register 0 (IICS0)**

The IICS0 register indicates the status of the I²C0 bus.

The IICS0 register is read-only, in 8-bit or 1-bit units.

After reset, IICS0 is cleared to 00H.

**Caution When the main clock is stopped and the CPU is operating on the subclock, do not access the IICS0 register using an access method that causes a wait.**
**For details, refer to 3.4.8 (2).**

(1/3)

After reset: 00H R Address: FFFFFD86H

| | <7> | <6> | <5> | <4> | <3> | <2> | <1> | <0> |
|---|---|---|---|---|---|---|---|---|
| IICS0 | MSTS0 | ALD0 | EXC0 | COI0 | TRC0 | ACKD0 | STD0 | SPD0 |

| MSTS0 | Master device status |
|---|---|
| 0 | Slave device status or communication standby status |
| 1 | Master device communication status |

| Condition for clearing (MSTS0 bit = 0) | Condition for setting (MSTS0 bit = 1) |
|---|---|
| • When a stop condition is detected<br>• When the ALD0 bit = 1 (arbitration loss)<br>• Cleared by the IICC0.LREL0 bit = 1 (exit from communications)<br>• When the IICC0.IICE0 bit changes from 1 to 0 (operation stop)<br>• Reset | • When a start condition is generated |

| ALD0 | Detection of arbitration loss |
|---|---|
| 0 | This status means either that there was no arbitration or that the arbitration result was a “win”. |
| 1 | This status indicates the arbitration result was a “loss”. The MSTS0 bit is cleared to 0. |

| Condition for clearing (ALD0 bit = 0) | Condition for setting (ALD0 bit = 1) |
|---|---|
| • Automatically cleared after the IICS0 register is read[Note]<br>• When the IICE0 bit changes from 1 to 0 (operation stop)<br>• Reset | • When the arbitration result is a “loss”. |

**Note** This register is also cleared when a bit manipulation instruction is executed for bits other than the IICS0 register.

(2/3)

| EXC0 | Detection of extension code reception |
|---|---|
| 0 | Extension code was not received. |
| 1 | Extension code was received. |

| Condition for clearing (EXC0 bit = 0) | Condition for setting (EXC0 bit = 1) |
|---|---|
| • When a start condition is detected<br>• When a stop condition is detected<br>• Cleared by the LREL0 bit = 1 (exit from communications)<br>• When the IICE0 bit changes from 1 to 0 (operation stop)<br>• Reset | • When the higher four bits of the received address data is either “0000” or “1111” (set at the rising edge of the eighth clock). |

| COI0 | Detection of matching addresses |
|---|---|
| 0 | Addresses do not match. |
| 1 | Addresses match. |

| Condition for clearing (COI0 bit = 0) | Condition for setting (COI0 bit = 1) |
|---|---|
| • When a start condition is detected<br>• When a stop condition is detected<br>• Cleared by the LREL0 bit = 1 (exit from communications)<br>• When the IICE0 bit changes from 1 to 0 (operation stop)<br>• Reset | • When the received address matches the local address (SVA0 register) (set at the rising edge of the eighth clock). |

| TRC0 | Detection of transmit/receive status |
|---|---|
| 0 | Receive status (other than transmit status). The SDA0 line is set for high impedance. |
| 1 | Transmit status. The value in the SO latch is enabled for output to the SDA0 line (valid starting at the rising edge of the first byte’s ninth clock). |

| Condition for clearing (TRC0 bit = 0) | Condition for setting (TRC0 bit = 1) |
|---|---|
| • When a stop condition is detected<br>• Cleared by the LREL0 bit = 1 (exit from communications)<br>• When the IICE0 bit changes from 1 to 0 (operation stop)<br>• Cleared by the IICC0.WREL0 bit = 1[Note] (wait release)<br>• When the ALD0 bit changes from 0 to 1 (arbitration loss)<br>• Reset<br>Master<br>• When “1” is output to the first byte’s LSB (transfer direction specification bit)<br>Slave<br>• When a start condition is detected<br>When not used for communication | Master<br>• When a start condition is generated<br>Slave<br>• When “1” is input in the first byte’s LSB (transfer direction specification bit) |

**Note** The TRC0 bit is cleared to 0 and the SDA0 line becomes high impedance when the WREL0 bit is set to 1 and wait state is released at the ninth clock with the TRC0 bit = 1.

(3/3)

| ACKD0 | Detection of acknowledge signal ($\overline{ACK}$) |
|---|---|
| 0 | $\overline{ACK}$ signal was not detected. |
| 1 | $\overline{ACK}$ signal was detected. |

| Condition for clearing (ACKD0 bit = 0) | Condition for setting (ACKD0 bit = 1) |
|---|---|
| • When a stop condition is detected<br>• At the rising edge of the next byte's first clock<br>• Cleared by the LREL0 bit = 1 (exit from communications)<br>• When the IICE0 bit changes from 1 to 0 (operation stop)<br>• Reset | • After the SDA0 pin is set to low level at the rising edge of the SCL0 pin's ninth clock |

| STD0 | Detection of start condition |
|---|---|
| 0 | Start condition was not detected. |
| 1 | Start condition was detected. This indicates that the address transfer period is in effect. |

| Condition for clearing (STD0 bit = 0) | Condition for setting (STD0 bit = 1) |
|---|---|
| • When a stop condition is detected<br>• At the rising edge of the next byte's first clock following address transfer<br>• Cleared by the LREL0 bit = 1 (exit from communications)<br>• When the IICE0 bit changes from 1 to 0 (operation stop)<br>• Reset | • When a start condition is detected |

| SPD0 | Detection of stop condition |
|---|---|
| 0 | Stop condition was not detected. |
| 1 | Stop condition was detected. The master device's communication is terminated and the bus is released. |

| Condition for clearing (SPD0 bit = 0) | Condition for setting (SPD0 bit = 1) |
|---|---|
| • At the rising edge of the address transfer byte's first clock following setting of this bit and detection of a start condition<br>• When the IICE0 bit changes from 1 to 0 (operation stop)<br>• Reset | • When a stop condition is detected |

**(3) IIC flag register 0 (IICF0)**

IICF0 is a register that sets the operation mode of I²C0 and indicates the status of the I²C bus.

This register can be read or written in 8-bit or 1-bit units. However, the STCF0 and IICBSY0 bits are read-only.

The IICRSV0 bit can be used to enable/disable the communication reservation function (refer to **18.13 Communication Reservation**).

The STCEN0 bit can be used to set the initial value of the IICBSY0 bit (refer to **18.14 Cautions**).

The IICRSV0 and STCEN0 bits can be written only when the operation of I²C0 is disabled (IICC0.IICE0 bit = 0).

When operation is enabled, the IICF0 register can be read.

After reset, IICF0 is cleared to 00H.

After reset: 00H R/W[Note] Address: FFFFFD8AH

| | <7> | <6> | 5 | 4 | 3 | 2 | <1> | <0> |
|---|---|---|---|---|---|---|---|---|
| IICF0 | STCF0 | IICBSY0 | 0 | 0 | 0 | 0 | STCEN0 | IICRSV0 |

| STCF0 | STT0 clear flag |
|---|---|
| 0 | Generate start condition |
| 1 | Start condition generation unsuccessful: clear STT0 flag |

| Condition for clearing (STCF0 bit = 0) | Condition for setting (STCF0 bit = 1) |
|---|---|
| • Cleared by the STT0 bit = 1<br>• Reset | • Generating start condition unsuccessful and the STT0 bit cleared to 0 when communication reservation is disabled (IICRSV0 bit = 1). |

| IICBSY0 | I²C0 bus status flag |
|---|---|
| 0 | Bus release status |
| 1 | Bus communication status |

| Condition for clearing (IICBSY0 bit = 0) | Condition for setting (IICBSY0 bit = 1) |
|---|---|
| • Detection of stop condition<br>• Reset | • Detection of start condition<br>• Setting of the IICE0 bit when the STCEN0 bit = 0 |

| STCEN0 | Initial start enable trigger |
|---|---|
| 0 | After operation is enabled (IICE0 bit = 1), enable generation of a start condition upon detection of a stop condition. |
| 1 | After operation is enabled (IICE0 bit = 1), enable generation of a start condition without detecting a stop condition. |

| Condition for clearing (STCE0 bit = 0) | Condition for setting (STCE0 bit = 1) |
|---|---|
| • Detection of start condition<br>• Reset | • Setting by instruction |

| IICRSV0 | Communication reservation function disable bit |
|---|---|
| 0 | Enable communication reservation |
| 1 | Disable communication reservation |

| Condition for clearing (IICRSV0 bit = 0) | Condition for setting (IICRSV0 bit = 1) |
|---|---|
| • Cleared by instruction<br>• Reset | • Setting by instruction |

**Note** Bits 6 and 7 are read-only bits.

**Cautions 1. Write to the STCEN0 bit only when the operation is stopped (IICE0 bit = 0).**

**2. As the bus release status (IICBSY0 bit = 0) is recognized regardless of the actual bus status when the STCEN0 bit = 1, when generating the first start condition (STT0 bit = 1), it is necessary to verify that no third party communications are in progress in order to prevent such communications from being destroyed.**

**3. Write to the IICRSV0 bit only when the operation is stopped (IICE0 bit = 0).**

**(4) IIC clock selection register 0 (IICCL0)**

The IICCL0 register is used to set the transfer clock for I²C0.

The IICCL0 register can be read or written in 8-bit or 1-bit units. However, the CLD0 and DAD0 bits are read-only. The SMC0, CL01, and CL00 bits are set in combination with the IICX0.CLX0 bit (refer to **18.3 (6) I²C0 transfer clock setting method**).

After reset, IICCL0 is cleared to 00H.

After reset: 00H R/W[Note] Address: FFFFFD84H

| | 7 | 6 | <5> | <4> | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| IICCL0 | 0 | 0 | CLD0 | DAD0 | SMC0 | DFC0 | CL01 | CL00 |

| CLD0 | Detection of SCL0 pin level (valid only when IICC0.IICE0 bit = 1) |
|---|---|
| 0 | The SCL0 pin was detected at low level. |
| 1 | The SCL0 pin was detected at high level. |

| Condition for clearing (CLD0 bit = 0) | Condition for setting (CLD0 bit = 1) |
|---|---|
| • When the SCL0 pin is at low level<br>• When the IICE0 bit = 0 (operation stop)<br>• Reset | • When the SCL0 pin is at high level |

| DAD0 | Detection of SDA0 pin level (valid only when IICE0 bit = 1) |
|---|---|
| 0 | The SDA0 pin was detected at low level. |
| 1 | The SDA0 pin was detected at high level. |

| Condition for clearing (DAD0 bit = 0) | Condition for setting (DAD0 bit = 1) |
|---|---|
| • When the SDA0 pin is at low level<br>• When the IICE0 bit = 0 (operation stop)<br>• Reset | • When the SDA0 pin is at high level |

| SMC0 | Operation mode switching |
|---|---|
| 0 | Operates in standard mode. |
| 1 | Operates in high-speed mode. |

| DFC0 | Digital filter operation control |
|---|---|
| 0 | Digital filter off. |
| 1 | Digital filter on. |
| Digital filter can be used only in high-speed mode.<br>In high-speed mode, the transfer clock does not vary regardless of DFC0 bit set/clear.<br>The digital filter is used for noise elimination in high-speed mode. | |

**Note** Bits 4 and 5 are read-only bits.

**(5) IIC function expansion register 0 (IICX0)**

This register sets the function expansion of I²C0 (valid only in high-speed mode).

This register can be read or written in 8-bit or 1-bit units. The CLX0 bit is set in combination with the IICCL0.SMC0, IICCL0.CL01, and IICCL0.CL00 bits (refer to **18.3 (6) I²C0 transfer clock setting method**).

After reset, IICX0 is cleared to 00H.

After reset: 00H R/W Address: FFFFFD85H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | <0> |
|---|---|---|---|---|---|---|---|---|
| IICX0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CLX0 |

**(6) I²C0 transfer clock setting method**

The I²C0 transfer clock frequency ($f_{SCL}$) is calculated using the following expression.

$$f_{SCL} = 1/(m \times T + t_R + t_F)$$

m = 12, 24, 48, 54, 86, 88, 172, 198 (refer to **Table 18-2 Selection Clock Setting**)

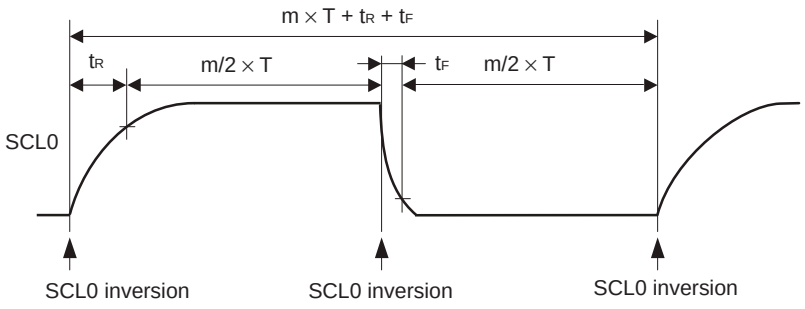
T: $1/f_{XX}$

$t_R$: SCL0 rise time

$t_F$: SCL0 fall time

For example, the I²C0 transfer clock frequency ($f_{SCL}$) when $f_{XX}$ = 16 MHz, m = 172, $t_R$ = 200 ns, and $t_F$ = 50 ns is calculated using following expression.

$$f_{SCL} = 1/(172 \times 62.5 \text{ ns} + 200 \text{ ns} + 50 \text{ ns}) \cong 90.9 \text{ kHz}$$




The selection clock is set using a combination of the IICCL0.SMC0, IICCL0.CL01, and IICCL0.CL00 bits and the IICX0.CLX0 bit.

**Table 18-2. Selection Clock Setting**

| IICX0 | IICCL0 | | | Selection Clock | Transfer Clock ($f_{XX}$/m) | Settable Internal System Clock Frequency ($f_{XX}$) Range | Operation Mode |
|---|---|---|---|---|---|---|---|
| Bit 0 | Bit 3 | Bit 1 | Bit 0 | | | | |
| CLX0 | SMC0 | CL01 | CL00 | | | | |
| 0 | 0 | 0 | 0 | $f_{XX}$/2 | $f_{XX}$/88 | 4.0 MHz to 8.38 MHz | Normal mode (SMC0 bit = 0) |
| 0 | 0 | 0 | 1 | $f_{XX}$/2 | $f_{XX}$/172 | 8.38 MHz to 16.76 MHz | |
| 0 | 0 | 1 | 0 | $f_{XX}$ | $f_{XX}$/86 | 4.19 MHz to 8.38 MHz | |
| 0 | 0 | 1 | 1 | $f_{XX}$/3 | $f_{XX}$/198 | 16.0 MHz to 19.8 MHz | |
| 0 | 1 | 0 | x | $f_{XX}$/2 | $f_{XX}$/48 | 8 MHz to 16.76 MHz | High-speed mode (SMC0 bit = 1) |
| 0 | 1 | 1 | 0 | $f_{XX}$ | $f_{XX}$/24 | 4 MHz to 8.38 MHz | |
| 0 | 1 | 1 | 1 | $f_{XX}$/3 | $f_{XX}$/54 | 16 MHz to 20 MHz | |
| 1 | 0 | x | x | Setting prohibited | | | |
| 1 | 1 | 0 | x | $f_{XX}$/2 | $f_{XX}$/24 | 8.00 MHz to 8.38 MHz | High-speed mode (SMC0 bit = 1) |
| 1 | 1 | 1 | 0 | $f_{XX}$ | $f_{XX}$/12 | 4.00 MHz to 4.19 MHz | |
| 1 | 1 | 1 | 1 | Setting prohibited | | | |

**Remark** x: don't care

**(7) IIC shift register 0 (IIC0)**

The IIC0 register is used for serial transmission/reception (shift operations) that is synchronized with the serial clock.

The IIC0 register can be read or written in 8-bit units, but data should not be written to the IIC0 register during a data transfer.

When the IIC0 register is written during wait, the wait is cancelled and data transfer is started.

After reset, IIC0 is cleared to 00H.

After reset: 00H R/W Address: FFFFFD80H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| IIC0 | | | | | | | | |

**(8) Slave address register 0 (SVA0)**

The SVA0 register holds the I²C bus's slave addresses.

The SVA0 register can be read or written in 8-bit units, but bit 0 should be fixed as 0.

After reset, SVA0 is cleared to 00H.

After reset: 00H R/W Address: FFFFFD83H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| SVA0 | | | | | | | | 0 |

## 18.4 Functions

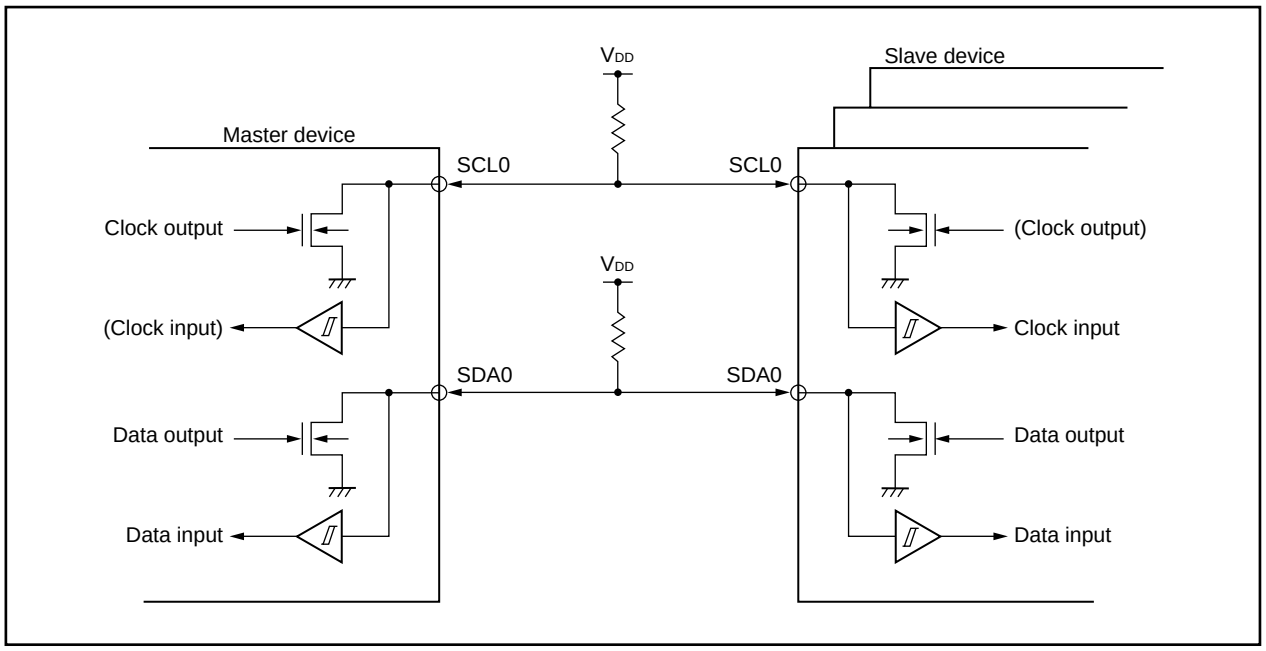
### 18.4.1 Pin configuration

The serial clock pin (SCL0) and serial data bus pin (SDA0) are configured as follows.

SCL0 ..... This pin is used for serial clock input and output.
This pin is an N-ch open-drain output for both master and slave devices. Input is Schmitt input.
SDA0 ..... This pin is used for serial data input and output.
This pin is an N-ch open-drain output for both master and slave devices. Input is Schmitt input.

Since outputs from the serial clock line and the serial data bus line are N-ch open-drain outputs, an external pull-up resistor is required.

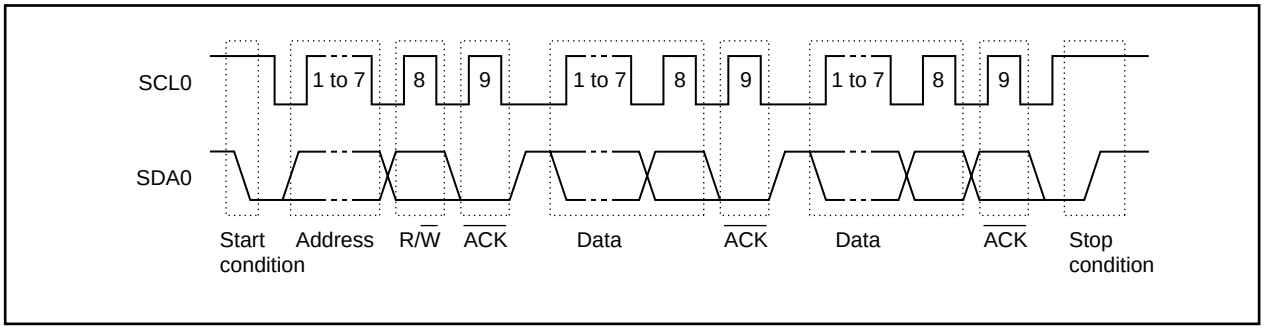
**Figure 18-3. Pin Configuration Diagram**



## 18.5 I²C Bus Definitions and Control Methods

The following section describes the I²C bus's serial data communication format and the signals used by the I²C bus. The transfer timing for the "start condition", "data", and "stop condition" output via the I²C bus's serial data bus is shown below.

**Figure 18-4. I²C Bus's Serial Data Transfer Timing**




The master device outputs the start condition, slave address, and stop condition.

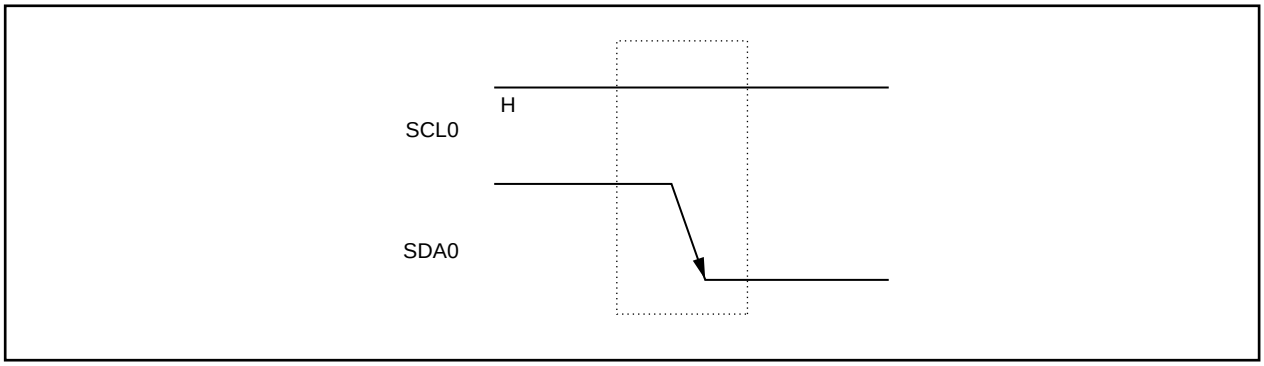
The acknowledge signal ($\overline{ACK}$) can be output by either the master or slave device (normally, it is output by the device that receives 8-bit data).

The serial clock (SCL0) is continuously output by the master device. However, in the slave device, the SCL0 pin's low-level period can be extended and a wait can be inserted.

### 18.5.1 Start condition

A start condition is met when the SCL0 pin is at high level and the SDA0 pin changes from high level to low level. The start conditions for the SCL0 pin and SDA0 pin are signals that the master device outputs to the slave device when starting a serial transfer. Start conditions can be detected when the device is used as a slave.

**Figure 18-5. Start Conditions**




A start condition is output when the IICC0.STT0 bit is set to 1 after a stop condition has been detected (IICS0.SPD0 bit = 1). When a start condition is detected, the IICS0.STD0 bit is set to 1.

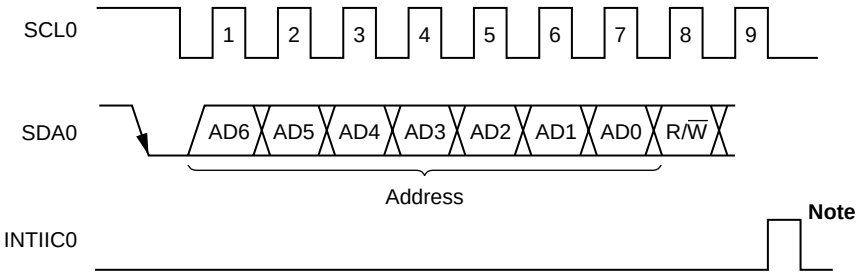
### 18.5.2 Addresses

The 7 bits of data that follow the start condition are defined as an address.

An address is a 7-bit data segment that is output in order to select one of the slave devices that are connected to the master device via bus lines. Therefore, each slave device connected via the bus lines must have a unique address.

The slave devices include hardware that detects the start condition and checks whether or not the 7-bit address data matches the data values stored in the SVA0 register. If the address data matches the SVA0 register values, the slave device is selected and communicates with the master device until the master device transmits a start condition or stop condition.

**Figure 18-6. Address**




**Note** The interrupt request signal (INTIIC0) is generated if a local address or extension code is received during slave device operation.

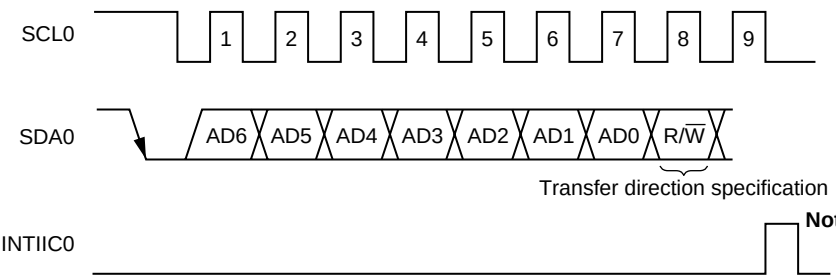
The slave address and the eighth bit, which specifies the transfer direction as described in **18.5.3 Transfer direction specification** below, are together written to the IIC0 register and are then output. Received addresses are written to the IIC0 register.

The slave address is assigned to the higher 7 bits of the IIC0 register.

### 18.5.3 Transfer direction specification

In addition to the 7-bit address data, the master device sends 1 bit that specifies the transfer direction. When this transfer direction specification bit has a value of 0, it indicates that the master device is transmitting data to a slave device. When the transfer direction specification bit has a value of 1, it indicates that the master device is receiving data from a slave device.

**Figure 18-7. Transfer Direction Specification**




**Note** The interrupt request signal (INTIIC0) is generated if a local address or extension code is received during slave device operation.

### 18.5.4 Acknowledge signal ($\overline{\text{ACK}}$)

The acknowledge signal ($\overline{\text{ACK}}$) is used by the transmitting and receiving devices to confirm serial data reception.

The receiving device returns one $\overline{\text{ACK}}$ signal for each 8 bits of data it receives. The transmitting device normally receives an $\overline{\text{ACK}}$ signal after transmitting 8 bits of data. However, when the master device is the receiving device, it does not output an $\overline{\text{ACK}}$ signal after receiving the final data to be transmitted. The transmitting device detects whether or not an $\overline{\text{ACK}}$ signal is returned after it transmits 8 bits of data. When an $\overline{\text{ACK}}$ signal is returned, the reception is judged as normal and processing continues. If the slave device does not return an $\overline{\text{ACK}}$ signal, the master device outputs either a stop condition or a restart condition and then stops the current transmission. Failure to return an $\overline{\text{ACK}}$ signal may be caused by the following two factors.

<1> Reception was not performed normally.

<2> The final data was received.

When the receiving device sets the SDA0 line to low level during the ninth clock, the $\overline{\text{ACK}}$ signal becomes active (normal receive response).

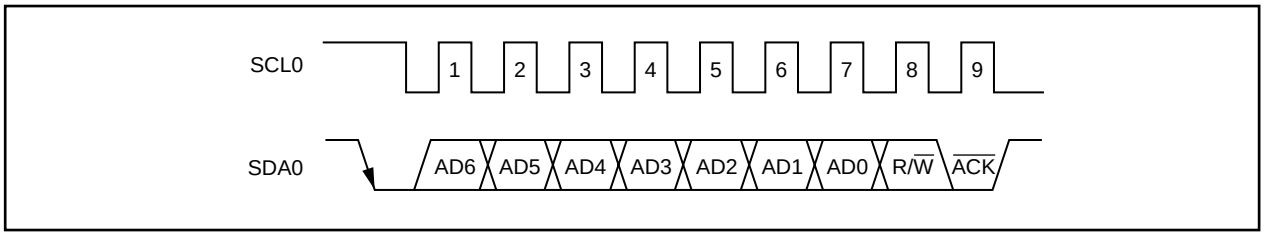
When the IICC0.ACKE0 bit is set to 1, automatic $\overline{\text{ACK}}$ signal generation is enabled.

Transmission of the eighth bit following the 7 address data bits causes the IICS0.TRC0 bit to be set. When this TRC0 bit's value is 0, it indicates receive mode. Therefore, the ACKE0 bit should be set to 1.

When the slave device is receiving (when TRC0 bit = 0), if the slave device does not need to receive any more data after receiving several bytes, clearing the ACKE0 bit to 0 will prevent the master device from starting transmission of the subsequent data.

Similarly, when the master device is receiving (when TRC0 bit = 0) and the subsequent data is not needed and when either a restart condition or a stop condition should therefore be output, clearing the ACKE0 bit to 0 will prevent the $\overline{\text{ACK}}$ signal from being returned. This prevents the MSB data from being output via the SDA0 line (i.e., stops transmission) during transmission from the slave device.

**Figure 18-8. Acknowledge Signal ($\overline{\text{ACK}}$)**




When the local address is received, an $\overline{\text{ACK}}$ signal is automatically output in synchronization with the falling edge of the SCL0 pin's eighth clock regardless of the ACKE0 bit value. No $\overline{\text{ACK}}$ signal is output if the received address is not a local address.

The $\overline{\text{ACK}}$ signal output method during data reception is based on the wait timing setting, as described below.

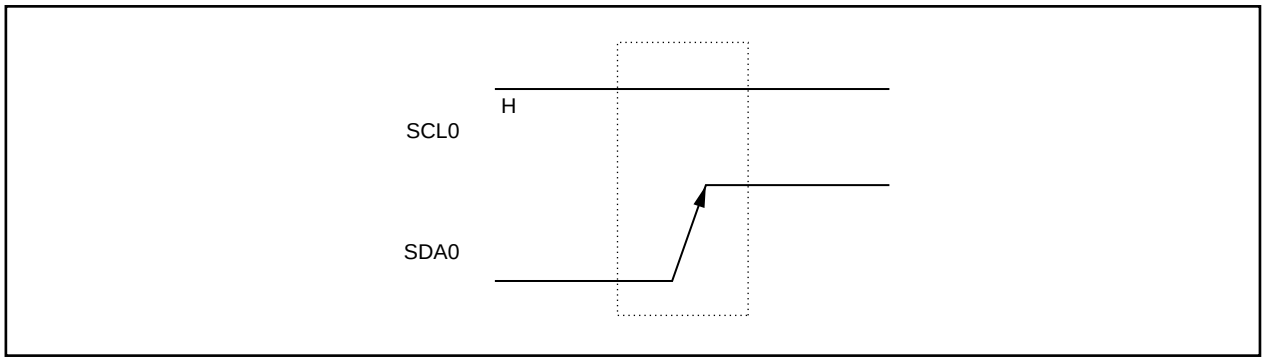
- When 8-clock wait is selected: (IICC0.WTIM0 bit = 0) — $\overline{\text{ACK}}$ signal is output at the falling edge of the SCL0 pin's eighth clock if the ACKE0 bit is set to 1 before wait cancellation.
- When 9-clock wait is selected: (WTIM0 bit = 1) — $\overline{\text{ACK}}$ signal is automatically output at the falling edge of the SCL0 pin's eighth clock if the ACKE0 bit has already been set to 1.

### 18.5.5 Stop condition

When the SCL0 pin is at high level, changing the SDA0 pin from low level to high level generates a stop condition.

A stop condition is a signal that the master device outputs to the slave device when serial transfer has been completed. Stop conditions can be detected when the device is used as a slave.

**Figure 18-9. Stop Condition**




A stop condition is generated when the IICC0.SPT0 bit is set to 1. When the stop condition is detected, the IICS0.SPD0 bit is set to 1 and the interrupt request signal (INTIIC0) is generated when the IICC0.SPIE0 bit is set to 1.

### 18.5.6 Wait signal ($\overline{\text{WAIT}}$)

The wait signal ($\overline{\text{WAIT}}$) is used to notify the communication partner that a device (master or slave) is preparing to transmit or receive data (i.e., is in a wait state).

Setting the SCL0 pin to low level notifies the communication partner of the wait status. When wait status has been canceled for both the master and slave devices, the next data transfer can begin.

**Figure 18-10. Wait Signal (1/2)**

**(a) When master device has a nine-clock wait and slave device has an eight-clock wait**

**(master: transmission, slave: reception, and IICC0.ACKE0 bit = 1)**

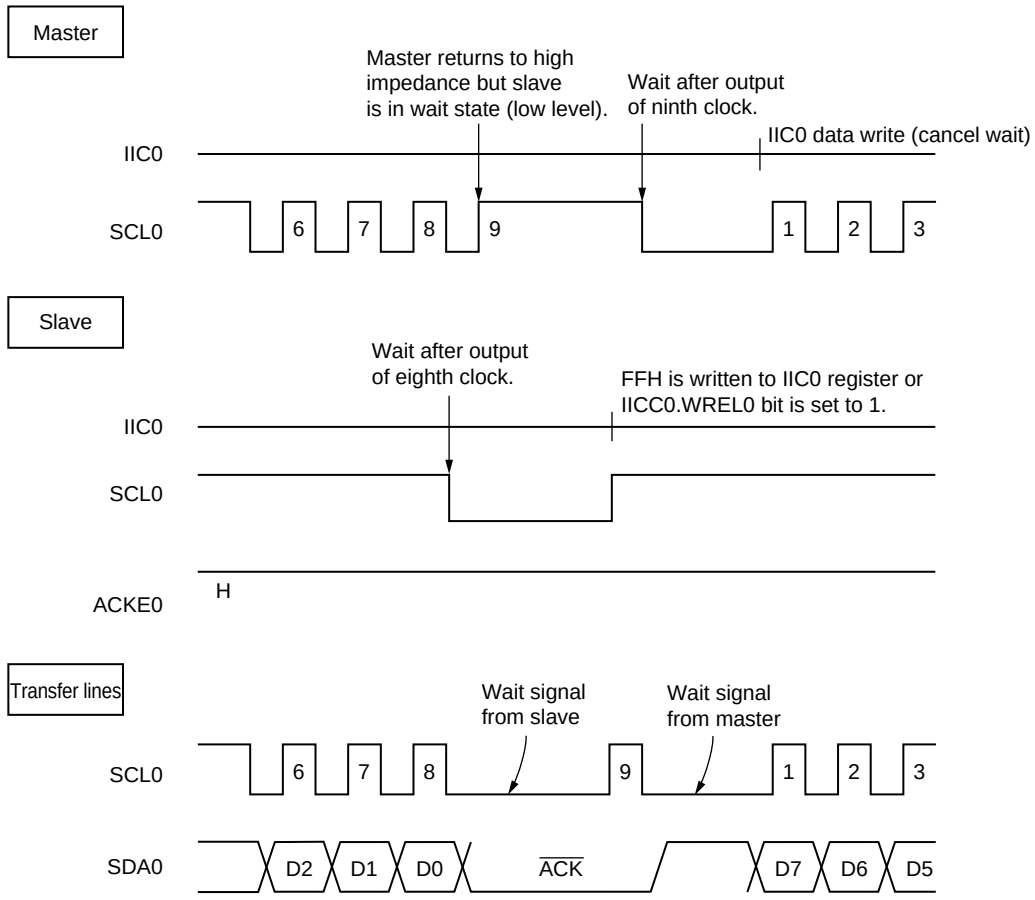
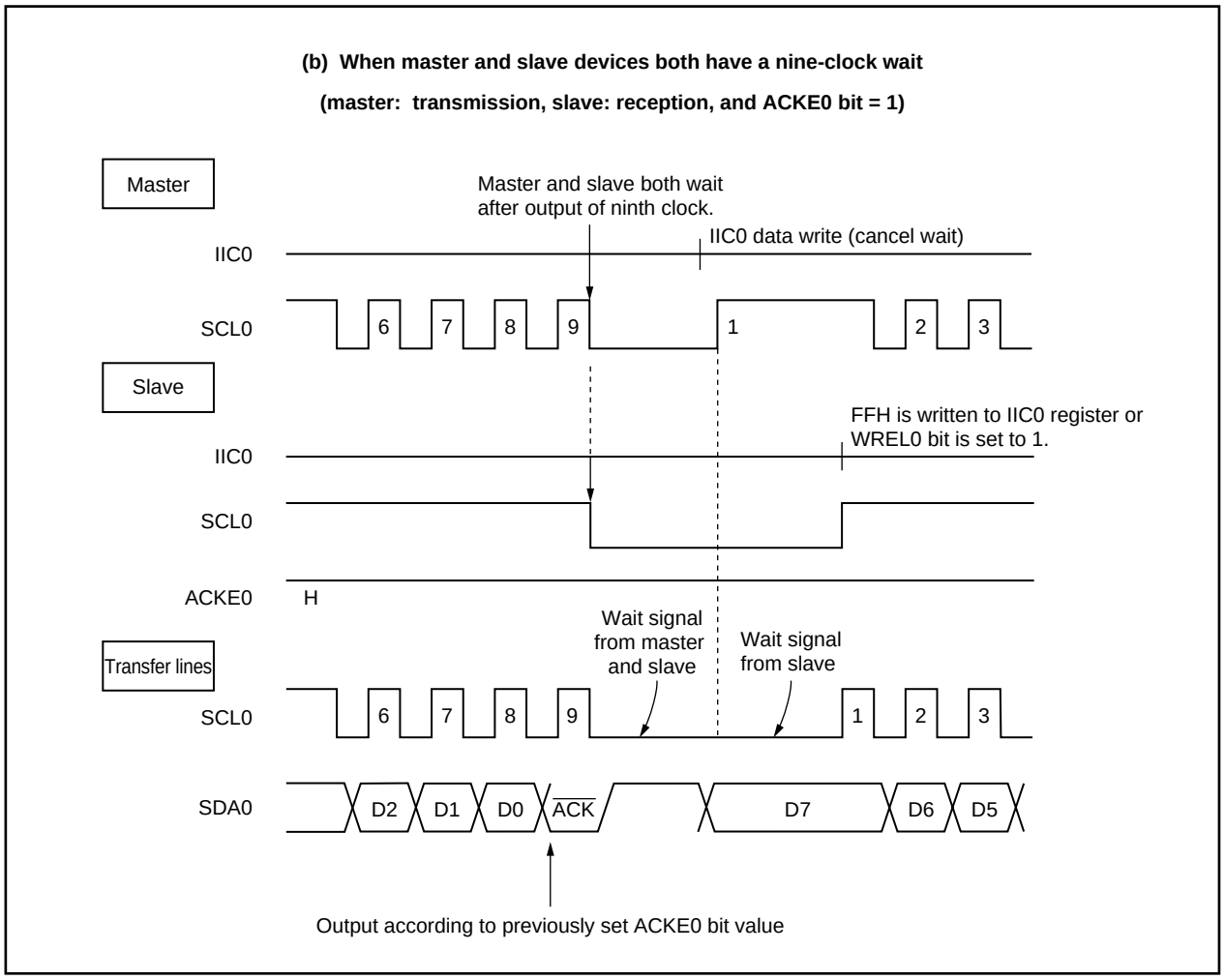


**Figure 18-10. Wait Signal (2/2)**




A wait may be automatically generated depending on the setting for the IICC0.WTIM0 bit.

Normally, when the WREL0 bit is set to 1 or when FFH is written to the IIC0 register, the wait status is canceled and the transmitting side writes data to the IIC0 register to cancel the wait status.

The master device can also cancel the wait status via either of the following methods.

- By setting the IICC0.STT0 bit to 1
- By setting the IICC0.SPT0 bit to 1

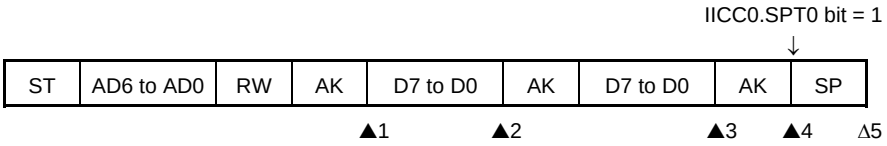
## 18.6 I²C Interrupt Request Signals (INTIIC0)

The following shows the value of the IICS0 register at the INTIIC0 interrupt request signal generation timing and at the INTIIC0 signal timing.

### 18.6.1 Master device operation

#### (1) Start ~ Address ~ Data ~ Data ~ Stop (normal transmission/reception)

**<1> When IICC0.WTIM0 bit = 0**




▲1: IICS0 register = 10XXX110B

▲2: IICS0 register = 10XXX000B

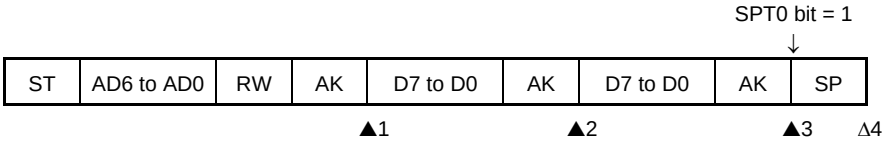
▲3: IICS0 register = 10XXX000B (WTIM0 bit = 1)

▲4: IICS0 register = 10XXXX00B

Δ 5: IICS0 register = 00000001B

**Remark** ▲: Always generated
Δ: Generated only when IICC0.SPIE0 bit = 1
X: don't care

**<2> When WTIM0 bit = 1**




▲1: IICS0 register = 10XXX110B

▲2: IICS0 register = 10XXX100B

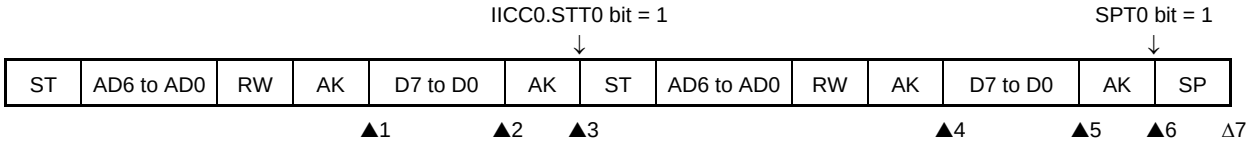
▲3: IICS0 register = 10XXXX00B

Δ 4: IICS0 register = 00000001B

**Remark** ▲: Always generated
Δ: Generated only when SPIE0 bit = 1
X: don't care

**(2) Start ~ Address ~ Data ~ Start ~ Address ~ Data ~ Stop (restart)**

**<1> When WTIM0 bit = 0**




▲1: IICS0 register = 10XXX110B

▲2: IICS0 register = 10XXX000B (WTIM0 bit = 1)

▲3: IICS0 register = 10XXXX00B (WTIM0 bit = 0)

▲4: IICS0 register = 10XXX110B (WTIM0 bit = 0)

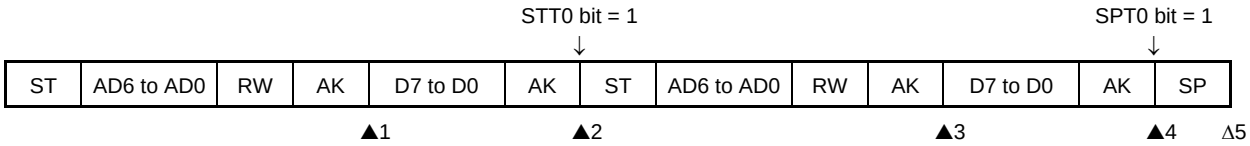
▲5: IICS0 register = 10XXX000B (WTIM0 bit = 1)

▲6: IICS0 register = 10XXXX00B

Δ 7: IICS0 register = 00000001B

**Remark** ▲: Always generated
Δ: Generated only when SPIE0 bit = 1
X: don't care

**<2> When WTIM0 bit = 1**




▲1: IICS0 register = 10XXX110B

▲2: IICS0 register = 10XXXX00B

▲3: IICS0 register = 10XXX110B

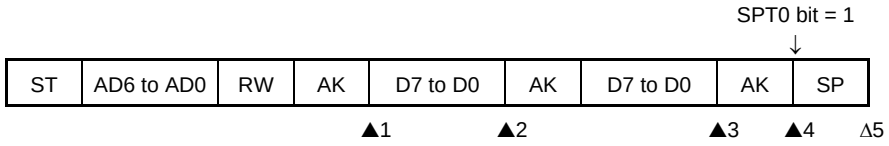
▲4: IICS0 register = 10XXXX00B

Δ 5: IICS0 register = 00000001B

**Remark** ▲: Always generated
Δ: Generated only when SPIE0 bit = 1
X: don't care

### (3) Start ~ Code ~ Data ~ Data ~ Stop (extension code transmission)

**<1> When WTIM0 bit = 0**




▲1: IICS0 register = 1010X110B

▲2: IICS0 register = 1010X000B

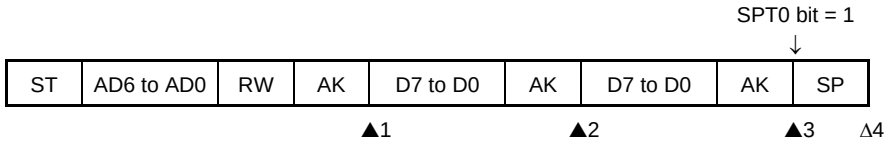
▲3: IICS0 register = 1010X000B (WTIM0 bit = 1)

▲4: IICS0 register = 1010XX00B

Δ 5: IICS0 register = 00000001B

**Remark** ▲: Always generated
Δ: Generated only when SPIE0 bit = 1
X: don't care

**<2> When WTIM0 bit = 1**




▲1: IICS0 register = 1010X110B

▲2: IICS0 register = 1010X100B

▲3: IICS0 register = 1010XX00B

Δ 4: IICS0 register = 00000001B

**Remark** ▲: Always generated
Δ: Generated only when SPIE0 bit = 1
X: don't care

### 18.6.2 Slave device operation (when receiving slave address data (match with address))

#### (1) Start ~ Address ~ Data ~ Data ~ Stop

**<1> When IICC0.WTIM0 bit = 0**

| ST | AD6 to AD0 | RW | AK | D7 to D0 | AK | D7 to D0 | AK | SP |
|---|---|---|---|---|---|---|---|---|

▲1 ▲2 ▲3 Δ4

▲1: IICS0 register = 0001X110B

▲2: IICS0 register = 0001X000B

▲3: IICS0 register = 0001X000B

Δ 4: IICS0 register = 00000001B

**Remark** ▲: Always generated
Δ: Generated only when IICC0.SPIE0 bit = 1
X: don't care

**<2> When WTIM0 bit = 1**

| ST | AD6 to AD0 | RW | AK | D7 to D0 | AK | D7 to D0 | AK | SP |
|---|---|---|---|---|---|---|---|---|

▲1 ▲2 ▲3 Δ4

▲1: IICS0 register = 0001X110B

▲2: IICS0 register = 0001X100B

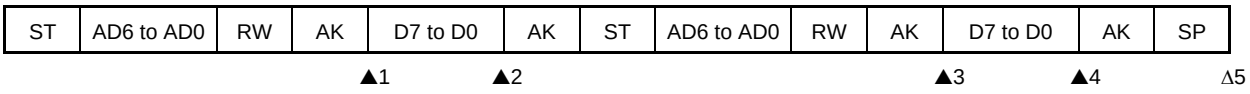
▲3: IICS0 register = 0001XX00B

Δ 4: IICS0 register = 00000001B

**Remark** ▲: Always generated
Δ: Generated only when SPIE0 bit = 1
X: don't care

**(2) Start ~ Address ~ Data ~ Start ~ Address ~ Data ~ Stop**

**<1> When WTIM0 bit = 0 (after restart, match with address)**




▲1: IICS0 register = 0001X110B

▲2: IICS0 register = 0001X000B

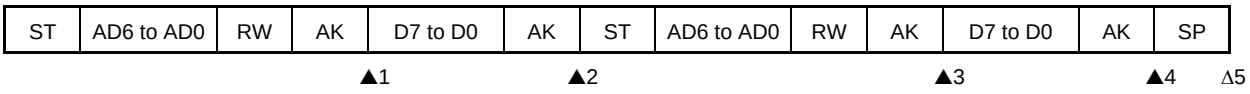
▲3: IICS0 register = 0001X110B

▲4: IICS0 register = 0001X000B

∆ 5: IICS0 register = 00000001B

**Remark** ▲: Always generated
∆: Generated only when SPIE0 bit = 1
X: don't care

**<2> When WTIM0 bit = 1 (after restart, match with address)**




▲1: IICS0 register = 0001X110B

▲2: IICS0 register = 0001XX00B

▲3: IICS0 register = 0001X110B

▲4: IICS0 register = 0001XX00B

∆ 5: IICS0 register = 00000001B

**Remark** ▲: Always generated
∆: Generated only when SPIE0 bit = 1
X: don't care

**(3) Start ~ Address ~ Data ~ Start ~ Code ~ Data ~ Stop**

**<1> When WTIM0 bit = 0 (after restart, extension code reception)**

| ST | AD6 to AD0 | RW | AK | D7 to D0 | AK | ST | AD6 to AD0 | RW | AK | D7 to D0 | AK | SP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | ▲1 | ▲2 | | | | ▲3 | | ▲4 | Δ5 |

▲1: IICS0 register = 0001X110B

▲2: IICS0 register = 0001X000B

▲3: IICS0 register = 0010X010B

▲4: IICS0 register = 0010X000B

Δ 5: IICS0 register = 00000001B

**Remark** ▲: Always generated
Δ: Generated only when SPIE0 bit = 1
X: don't care

**<2> When WTIM0 bit = 1 (after restart, extension code reception)**

| ST | AD6 to AD0 | RW | AK | D7 to D0 | AK | ST | AD6 to AD0 | RW | AK | D7 to D0 | AK | SP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | ▲1 | | ▲2 | | | ▲3 | ▲4 | ▲5 | Δ6 |

▲1: IICS0 register = 0001X110B

▲2: IICS0 register = 0001XX00B

▲3: IICS0 register = 0010X010B

▲4: IICS0 register = 0010X110B

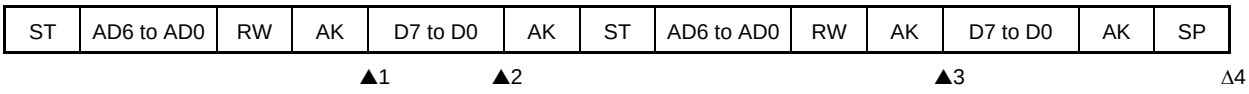
▲5: IICS0 register = 0010XX00B

Δ 6: IICS0 register = 00000001B

**Remark** ▲: Always generated
Δ: Generated only when SPIE0 bit = 1
X: don't care

**(4) Start ~ Address ~ Data ~ Start ~ Address ~ Data ~ Stop**

**<1> When WTIM0 bit = 0 (after restart, mismatch with address (= not extension code))**




▲1: IICS0 register = 0001X110B

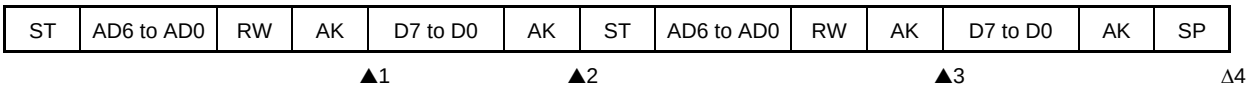
▲2: IICS0 register = 0001X000B

▲3: IICS0 register = 00000X10B

Δ 4: IICS0 register = 00000001B

**Remark** ▲: Always generated
Δ: Generated only when SPIE0 bit = 1
X: don't care

**<2> When WTIM0 bit = 1 (after restart, mismatch with address (= not extension code))**




▲1: IICS0 register = 0001X110B

▲2: IICS0 register = 0001XX00B

▲3: IICS0 register = 00000X10B

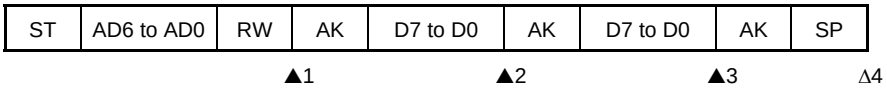
Δ 4: IICS0 register = 00000001B

**Remark** ▲: Always generated
Δ: Generated only when SPIE0 bit = 1
X: don't care

### 18.6.3 Slave device operation (when receiving extension code)

#### (1) Start ~ Code ~ Data ~ Data ~ Stop

**<1> When IICC0.WTIM0 bit = 0**




▲1: IICS0 register = 0010X010B

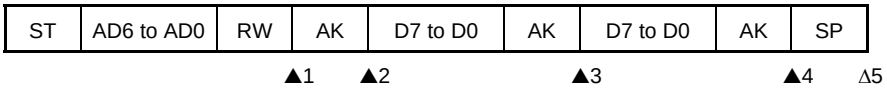
▲2: IICS0 register = 0010X000B

▲3: IICS0 register = 0010X000B

Δ 4: IICS0 register = 00000001B

**Remark** ▲: Always generated
Δ: Generated only when IICC0.SPIE0 bit = 1
X: don't care

**<2> When WTIM0 bit = 1**




▲1: IICS0 register = 0010X010B

▲2: IICS0 register = 0010X110B

▲3: IICS0 register = 0010X100B

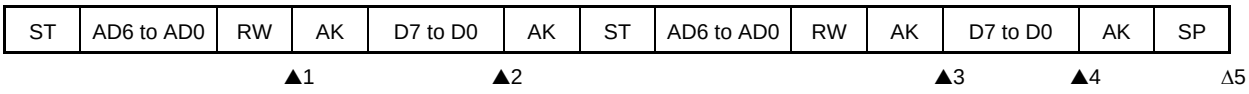
▲4: IICS0 register = 0010XX00B

Δ 5: IICS0 register = 00000001B

**Remark** ▲: Always generated
Δ: Generated only when SPIE0 bit = 1
X: don't care

**(2) Start ~ Code ~ Data ~ Start ~ Address ~ Data ~ Stop**

**<1> When WTIM0 bit = 0 (after restart, match with address)**




▲1: IICS0 register = 0010X010B

▲2: IICS0 register = 0010X000B

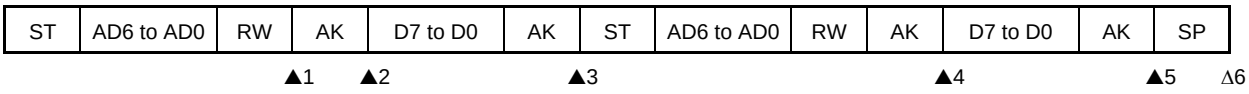
▲3: IICS0 register = 0001X110B

▲4: IICS0 register = 0001X000B

∆ 5: IICS0 register = 00000001B

**Remark** ▲: Always generated
∆: Generated only when SPIE0 bit = 1
X: don't care

**<2> When WTIM0 bit = 1 (after restart, match with address)**




▲1: IICS0 register = 0010X010B

▲2: IICS0 register = 0010X110B

▲3: IICS0 register = 0010XX00B

▲4: IICS0 register = 0001X110B

▲5: IICS0 register = 0001XX00B

∆ 6: IICS0 register = 00000001B

**Remark** ▲: Always generated
∆: Generated only when SPIE0 bit = 1
X: don't care

**(3) Start ~ Code ~ Data ~ Start ~ Code ~ Data ~ Stop**

**<1> When WTIM0 bit = 0 (after restart, extension code reception)**

| ST | AD6 to AD0 | RW | AK | D7 to D0 | AK | ST | AD6 to AD0 | RW | AK | D7 to D0 | AK | SP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | ▲1 | | ▲2 | | | | ▲3 | | ▲4 | Δ5 |

▲1: IICS0 register = 0010X010B

▲2: IICS0 register = 0010X000B

▲3: IICS0 register = 0010X010B

▲4: IICS0 register = 0010X000B

Δ 5: IICS0 register = 00000001B

**Remark** ▲: Always generated

Δ: Generated only when SPIE0 bit = 1

X: don't care

**<2> When WTIM0 bit = 1 (after restart, extension code reception)**

| ST | AD6 to AD0 | RW | AK | D7 to D0 | AK | ST | AD6 to AD0 | RW | AK | D7 to D0 | AK | SP |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | ▲1 | ▲2 | | ▲3 | | | ▲4 | ▲5 | | ▲6 Δ7 |

▲1: IICS0 register = 0010X010B

▲2: IICS0 register = 0010X110B

▲3: IICS0 register = 0010XX00B

▲4: IICS0 register = 0010X010B

▲5: IICS0 register = 0010X110B

▲6: IICS0 register = 0010XX00B

Δ 7: IICS0 register = 00000001B

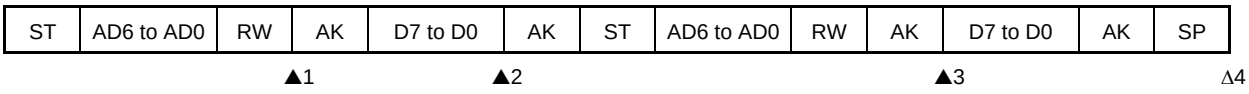
**Remark** ▲: Always generated

Δ: Generated only when SPIE0 bit = 1

X: don't care

**(4) Start ~ Code ~ Data ~ Start ~ Address ~ Data ~ Stop**

**<1> When WTIM0 bit = 0 (after restart, mismatch with address (= not extension code))**




▲1: IICS0 register = 0010X010B

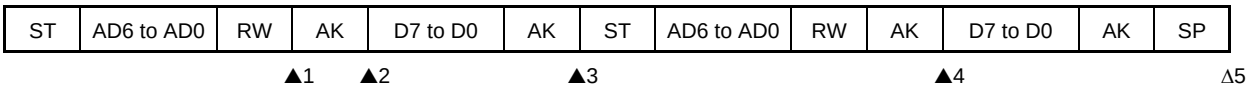
▲2: IICS0 register = 0010X000B

▲3: IICS0 register = 00000X10B

Δ 4: IICS0 register = 00000001B

**Remark** ▲: Always generated
Δ: Generated only when SPIE0 bit = 1
X: don't care

**<2> When WTIM0 bit = 1 (after restart, mismatch with address (= not extension code))**




▲1: IICS0 register = 0010X010B

▲2: IICS0 register = 0010X110B

▲3: IICS0 register = 0010XX00B

▲4: IICS0 register = 00000X10B

Δ 5: IICS0 register = 00000001B

**Remark** ▲: Always generated
Δ: Generated only when SPIE0 bit = 1
X: don't care

### 18.6.4 Operation without communication

#### (1) Start ~ Code ~ Data ~ Data ~ Stop

| ST | AD6 to AD0 | RW | AK | D7 to D0 | AK | D7 to D0 | AK | SP |
|---|---|---|---|---|---|---|---|---|

Δ1

Δ 1: IICS0 register = 00000001B

**Remark** Δ: Generated only when IICC0.SPIE0 bit = 1

### 18.6.5 Arbitration loss operation (operation as slave after arbitration loss)

#### (1) When arbitration loss occurs during transmission of slave address data

**<1> When IICC0.WTIM0 bit = 0**

| ST | AD6 to AD0 | RW | AK | D7 to D0 | AK | D7 to D0 | AK | SP |
|---|---|---|---|---|---|---|---|---|

▲1 ▲2 ▲3 Δ4

▲1: IICS0 register = 0101X110B (Example: when IICS0.ALD0 bit is read during interrupt servicing)

▲2: IICS0 register = 0001X000B

▲3: IICS0 register = 0001X000B

Δ 4: IICS0 register = 00000001B

**Remark** ▲: Always generated
Δ: Generated only when IICC0.SPIE0 bit = 1
X: don't care

**<2> When WTIM0 bit = 1**

| ST | AD6 to AD0 | RW | AK | D7 to D0 | AK | D7 to D0 | AK | SP |
|---|---|---|---|---|---|---|---|---|

▲1 ▲2 ▲3 Δ4

▲1: IICS0 register = 0101X110B (Example: when ALD0 bit is read during interrupt servicing)

▲2: IICS0 register = 0001X100B

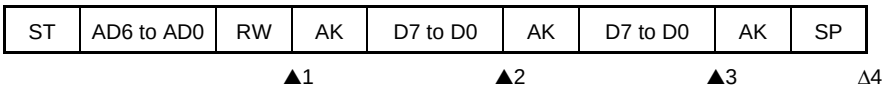
▲3: IICS0 register = 0001XX00B

Δ 4: IICS0 register = 00000001B

**Remark** ▲: Always generated
Δ: Generated only when SPIE0 bit = 1
X: don't care

**(2) When arbitration loss occurs during transmission of extension code**

**<1> When WTIM0 bit = 0**




▲1: IICS0 register = 0110X010B (Example: when ALD0 bit is read during interrupt servicing)

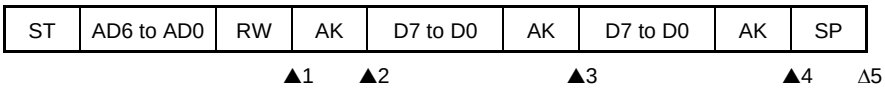
▲2: IICS0 register = 0010X000B

▲3: IICS0 register = 0010X000B

Δ 4: IICS0 register = 00000001B

**Remark** ▲: Always generated
Δ: Generated only when SPIE0 bit = 1
X: don't care

**<2> When WTIM0 bit = 1**




▲1: IICS0 register = 0110X010B (Example: when ALD0 bit is read during interrupt servicing)

▲2: IICS0 register = 0010X110B

▲3: IICS0 register = 0010X100B

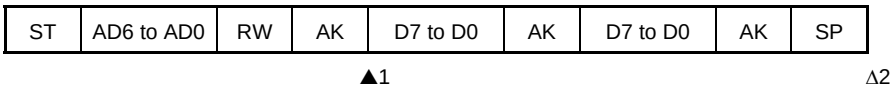
▲4: IICS0 register = 0010XX00B

Δ 5: IICS0 register = 00000001B

**Remark** ▲: Always generated
Δ: Generated only when SPIE0 bit = 1
X: don't care

### 18.6.6 Operation when arbitration loss occurs (no communication after arbitration loss)

#### (1) When arbitration loss occurs during transmission of slave address data

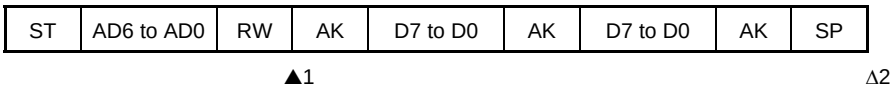



▲1: IICS0 register = 01000110B (Example: when IICS0.ALD0 bit is read during interrupt servicing)

Δ 2: IICS0 register = 00000001B

**Remark** ▲: Always generated
Δ: Generated only when IICC0.SPIE0 bit = 1

#### (2) When arbitration loss occurs during transmission of extension code




▲1: IICS0 register = 0110X010B (Example: when ALD0 bit is read during interrupt servicing)

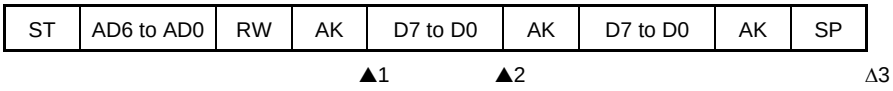
IICC0.LREL0 bit is set to 1 by software

Δ 2: IICS0 register = 00000001B

**Remark** ▲: Always generated
Δ: Generated only when SPIE0 bit = 1
X: don't care

(**3) When arbitration loss occurs during data transfer**

**<1> When IICC0.WTIM0 bit = 0**




▲1: IICS0 register = 10001110B

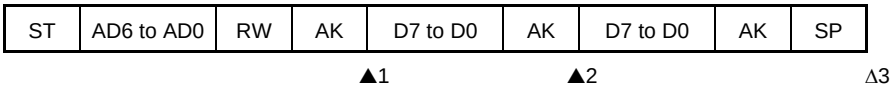
▲2: IICS0 register = 01000000B (Example: when ALD0 bit is read during interrupt servicing)

Δ 3: IICS0 register = 00000001B

**Remark** ▲: Always generated

Δ: Generated only when SPIE0 bit = 1

**<2> When WTIM0 bit = 1**




▲1: IICS0 register = 10001110B

▲2: IICS0 register = 01000100B (Example: when ALD0 bit is read during interrupt servicing)

Δ 3: IICS0 register = 00000001B

**Remark** ▲: Always generated

Δ: Generated only when SPIE0 bit = 1

### (4) When loss occurs due to restart condition during data transfer

#### <1> Not extension code (Example: mismatches with address)

| ST | AD6 to AD0 | RW | AK | D7 to Dn | ST | AD6 to AD0 | RW | AK | D7 to D0 | AK | SP |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | ▲1 | | | | | ▲2 | | Δ3 |

▲1: IICS0 register = 1000X110B

▲2: IICS0 register = 01000110B (Example: when ALD0 bit is read during interrupt servicing)

Δ 3: IICS0 register = 00000001B

**Remark** ▲: Always generated
Δ: Generated only when SPIE0 bit = 1
X: don't care
Dn = D6 to D0

#### <2> Extension code

| ST | AD6 to AD0 | RW | AK | D7 to Dn | ST | AD6 to AD0 | RW | AK | D7 to D0 | AK | SP |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | ▲1 | | | | | ▲2 | | Δ3 |

▲1: IICS0 register = 1000X110B

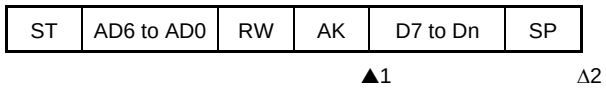
▲2: IICS0 register = 0110X010B (Example: when ALD0 bit is read during interrupt servicing)

LREL0 bit is set to 1 by software

Δ 3: IICS0 register = 00000001B

**Remark** ▲: Always generated
Δ: Generated only when SPIE0 bit = 1
X: don't care
Dn = D6 to D0

**(5) When loss occurs due to stop condition during data transfer**



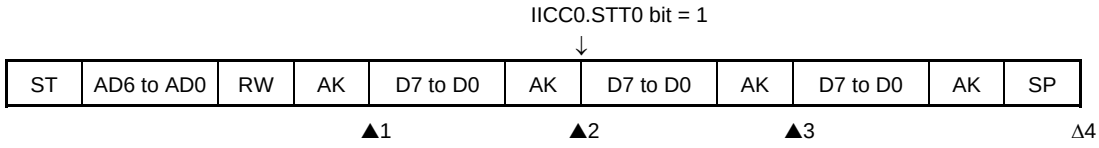

▲1: IICS0 register = 1000X110B

Δ 2: IICS0 register = 01000001B

**Remark** ▲: Always generated
Δ: Generated only when SPIE0 bit = 1
X: don't care
Dn = D6 to D0

**(6) When arbitration loss occurs due to low-level data when attempting to generate a restart condition**

**When WTIM0 bit = 1**




▲1: IICS0 register = 1000X110B

▲2: IICS0 register = 1000XX00B

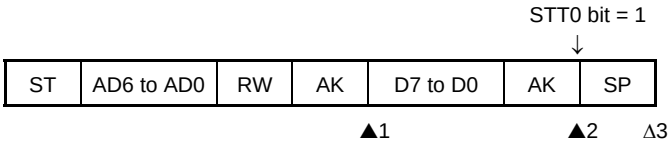
▲3: IICS0 register = 01000100B (Example: when ALD0 bit is read during interrupt servicing)

Δ 4: IICS0 register = 00000001B

**Remark** ▲: Always generated
Δ: Generated only when SPIE0 bit = 1
X: don't care

### (7) When arbitration loss occurs due to a stop condition when attempting to generate a restart condition

**When WTIM0 bit = 1**




▲1: IICS0 register = 1000X110B

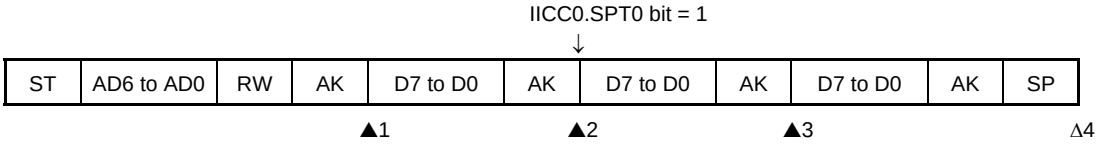
▲2: IICS0 register = 1000XX00B

Δ 3: IICS0 register = 01000001B

**Remark** ▲: Always generated
Δ: Generated only when SPIE0 bit = 1
X: don't care

### (8) When arbitration loss occurs due to low-level data when attempting to generate a stop condition

**When WTIM0 bit = 1**




▲1: IICS0 register = 1000X110B

▲2: IICS0 register = 1000XX00B

▲3: IICS0 register = 01000000B (Example: when ALD0 bit is read during interrupt servicing)

Δ 4: IICS0 register = 00000001B

**Remark** ▲: Always generated
Δ: Generated only when SPIE0 bit = 1
X: don't care

## 18.7 Interrupt Request Signal (INTIIC0) Generation Timing and Wait Control

The setting of the IICC0.WTIM0 bit determines the timing by which the INTIIC0 signal is generated and the corresponding wait control, as shown below.

**Table 18-3. INTIIC0 Signal Generation Timing and Wait Control**

| WTIM0 Bit | During Slave Device Operation | | | During Master Device 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 slave device's INTIIC0 signal and wait period occurs at the falling edge of the ninth clock only when there is a match with the address set to the SVA0 register.
At this point, an $\overline{ACK}$ signal is output regardless of the value set to the IICC0.ACKE0 bit. For a slave device that has received an extension code, the INTIIC0 signal occurs at the falling edge of the eighth clock.
When the address does not match after restart, the INTIIC0 signal is generated at the falling edge of the ninth clock, but no wait occurs.

**2.** If the received address does not match the contents of the SVA0 register and extension codes have not been received, neither the INTIIC0 signal nor a wait occurs.

**Remark** The numbers in the table indicate the number of the serial clock's clock signals. Interrupt requests and wait control are both synchronized with the falling edge of these clock signals.

**(1) During address transmission/reception**

- Slave device operation: Interrupt and wait timing are determined depending on the conditions in Notes 1 and 2 above regardless of the WTIM0 bit.
- Master device operation: Interrupt and wait timing occur at the falling edge of the ninth clock regardless of the WTIM0 bit.

**(2) During data reception**

- Master/slave device operation: Interrupt and wait timing are determined according to the WTIM0 bit.

**(3) During data transmission**

- Master/slave device operation: Interrupt and wait timing are determined according to the WTIM0 bit.

**(4) Wait cancellation method**

The four wait cancellation methods are as follows.

- By setting the IICC0.WREL0 bit to 1
- By writing to the IIC0 register
- By start condition setting (IICC0.STT0 bit = 1)[Note]
- By stop condition setting (IICC0.SPT0 bit = 1)[Note]

**Note** Master only

When an 8-clock wait has been selected (WTIM0 bit = 0), the output level of the $\overline{ACK}$ signal must be determined prior to wait cancellation.

**(5) Stop condition detection**

The INTIIC0 signal is generated when a stop condition is detected.

## 18.8 Address Match Detection Method

When in I²C bus mode, the master device can select a particular slave device by transmitting the corresponding slave address.

Address match detection is performed automatically by hardware. An INTIIC0 interrupt request signal occurs when a local address has been set to the SVA0 register and when the address set to the SVA0 register matches the slave address sent by the master device, or when an extension code has been received.

## 18.9 Error Detection

In I²C bus mode, the status of the serial data bus (SDA0) during data transmission is captured by the IIC0 register of the transmitting device, so the IIC0 register data prior to transmission can be compared with the transmitted IIC0 register data to enable detection of transmission errors. A transmission error is judged as having occurred when the compared data values do not match.

## 18.10 Extension Code

(1) When the higher 4 bits of the receive address are either 0000 or 1111, the extension code flag (EXC0) is set for extension code reception and an interrupt request signal (INTIIC0) is issued at the falling edge of the eighth clock. The local address stored in the SVA0 register is not affected.

(2) If 11110xx0 is set to the SVA0 register by a 10-bit address transfer and 11110xx0 is transferred from the master device, the results are as follows. Note that the INTIIC0 signal occurs at the falling edge of the eighth clock.

- Higher 4 bits of data match: IICS0.EXC0 bit = 1
- 7 bits of data match: IICS0.COI0 bit = 1

(3) Since the processing after the INTIIC0 signal occurs differs according to the data that follows the extension code, such processing is performed by software.
For example, when operation as a slave is not desired after the extension code is received, set the IICC0.LREL0 bit to 1 and the CPU will enter the next communication wait state.

**Table 18-4. Extension Code Bit Definitions**

| Slave Address | R/W Bit | Description |
|---|---|---|
| 0000 000 | 0 | General call address |
| 0000 000 | 1 | Start byte |
| 0000 001 | X | CBUS address |
| 0000 010 | X | Address that is reserved for different bus format |
| 1111 0xx | X | 10-bit slave address specification |

## 18.11 Arbitration

When several master devices simultaneously output a start condition (when the IICC0.STT0 bit is set to 1 before the IICS0.STD0 bit is set to 1), communication among the master devices is performed as the number of clocks is adjusted until the data differs. This kind of operation is called arbitration.

When one of the master devices loses in arbitration, an arbitration loss flag (IICS0.ALD0 bit) is set (1) via the timing by which the arbitration loss occurred, and the SCL0 and SDA0 lines are both set for high impedance, which releases the bus.

The arbitration loss is detected based on the timing of the next interrupt request signal (INTIIC0) (the eighth or ninth clock, when a stop condition is detected, etc.) and the ALD0 bit = 1 setting that has been made by software.

For details of interrupt request timing, refer to **18.6 I²C Interrupt Request Signals (INTIIC0)**.

**Figure 18-11. Arbitration Timing Example**

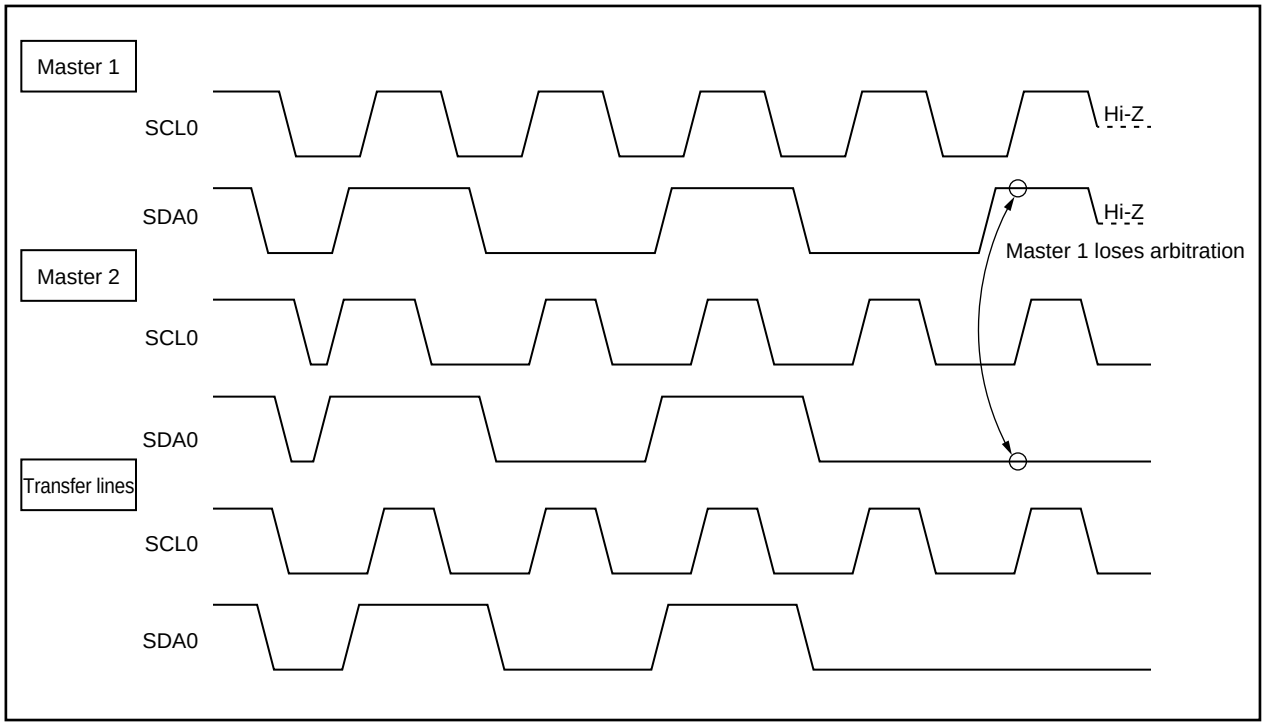


**Table 18-5. Status During Arbitration and Interrupt Request Generation Timing**

| Status During Arbitration | Interrupt Request Generation Timing |
|---|---|
| During address transmission | At falling edge of eighth or ninth clock following byte transfer[Note 1] |
| Read/write data after address transmission | |
| During extension code transmission | |
| Read/write data after extension code transmission | |
| During data transmission | |
| During $\overline{ACK}$ signal transfer period after data reception | |
| When restart condition is detected during data transfer | |
| When stop condition is detected during data transfer | When stop condition is output (when IICC0.SPIE0 bit = 1)[Note 2] |
| When the SDA0 pin is at low level while attempting to output a restart condition | At falling edge of eighth or ninth clock following byte transfer[Note 1] |
| When stop condition is detected while attempting to output a restart condition | When stop condition is output (when SPIE0 bit = 1)[Note 2] |
| When the SDA0 pin is at low level while attempting to output a stop condition | At falling edge of eighth or ninth clock following byte transfer[Note 1] |
| When the SCL0 pin is at low level while attempting to output a restart condition | |

**Notes 1.** When the IICC0.WTIM0 bit = 1, an INTIIC0 signal occurs at the falling edge of the ninth clock. When the WTIM0 bit = 0 and the extension code's slave address is received, an INTIIC0 signal occurs at the falling edge of the eighth clock.

**2.** When there is a possibility that arbitration will occur, set the SPIE0 bit = 1 for master device operation.

## 18.12 Wakeup Function

The I²C bus slave function is a function that generates an interrupt request signal (INTIIC0) when a local address or extension code has been received. This function makes processing more efficient by preventing the unnecessary INTIIC0 signal from occurring when addresses do not match.

When a start condition is detected, wakeup standby mode is set. This wakeup standby mode is in effect while addresses are transmitted due to the possibility that an arbitration loss may change the master device (which has output a start condition) to a slave device.

However, when a stop condition is detected, the IICC0.SPIE0 bit is set regardless of the wakeup function, and this determines whether the INTIIC0 signal is enabled or disabled.

## 18.13 Communication Reservation

### 18.13.1 When communication reservation function is enabled (IICF0.IICRSV0 bit = 0)

To start master device communications when not currently using a bus, a communication reservation can be made to enable transmission of a start condition when the bus is released. There are two modes under which the bus is not used.

- When arbitration results in neither master nor slave operation
- When an extension code is received and slave operation is disabled ($\overline{ACK}$ signal is not returned and the bus was released when the IICC0.LREL0 bit was set to 1).

If the IICC0.STT0 bit is set (1) while the bus is not used, a start condition is automatically generated and wait status is set after the bus is released (after a stop condition is detected).

When the bus release is detected (when a stop condition is detected), writing to the IIC0 register causes the master's address transfer to start. At this point, the IICC0.SPIE0 bit should be set (1).

When the STT0 bit has been set (1), the operation mode (as start condition or as communication reservation) is determined according to the bus status.

If the bus has been released.....a start condition is generated
If the bus has not been released (standby mode).....communication reservation

To detect which operation mode has been determined for the STT0 bit, set the STT0 bit (1), wait for the wait period, then check the IICS0.MSTS0 bit.

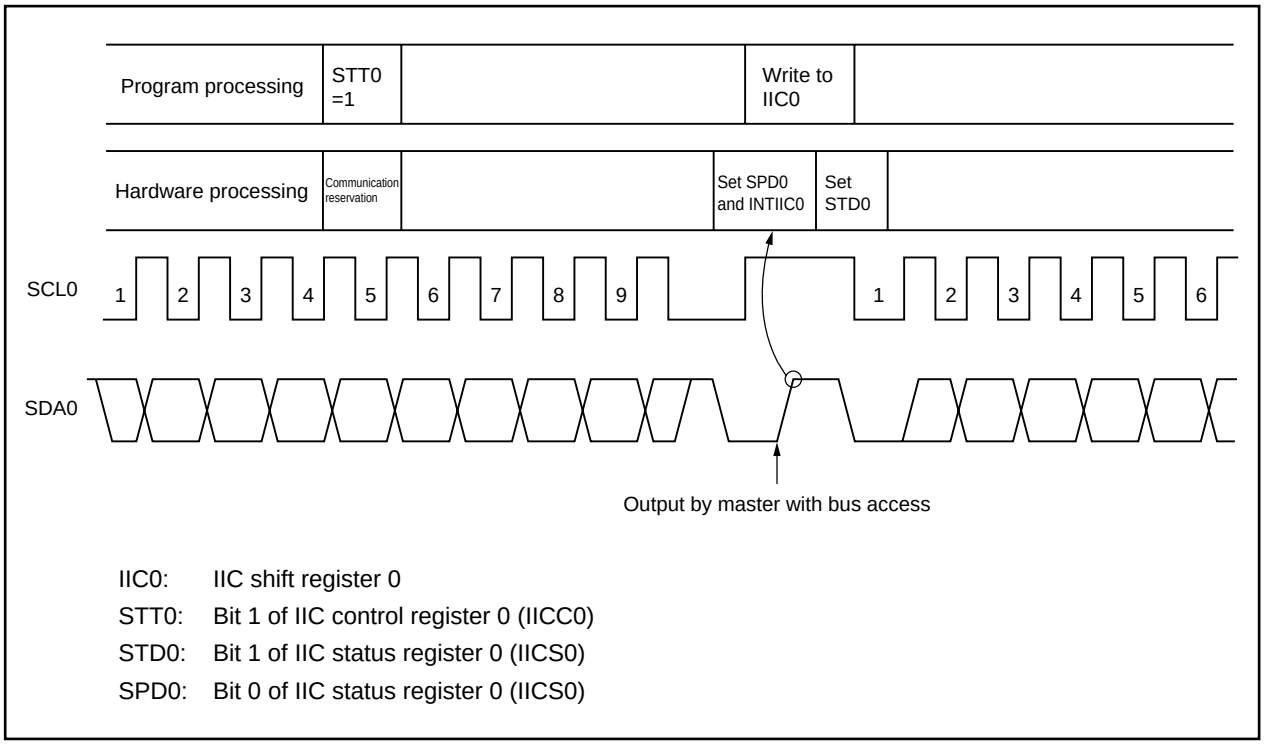
Wait periods, which should be set via software, are listed in Table 18-6. These wait periods can be set via the settings for the IICCL0.SMC0, IICCL0.CL01, and IICCL0.CL00 bits.

**Table 18-6. Wait Periods**

| SMC0 | CL01 | CL00 | Wait Period |
|---|---|---|---|
| 0 | 0 | 0 | 26 clocks |
| 0 | 0 | 1 | 46 clocks |
| 0 | 1 | 0 | 92 clocks |
| 0 | 1 | 1 | 37 clocks |
| 1 | 0 | 0 | 16 clocks |
| 1 | 0 | 1 | |
| 1 | 1 | 0 | 32 clocks |
| 1 | 1 | 1 | 13 clocks |

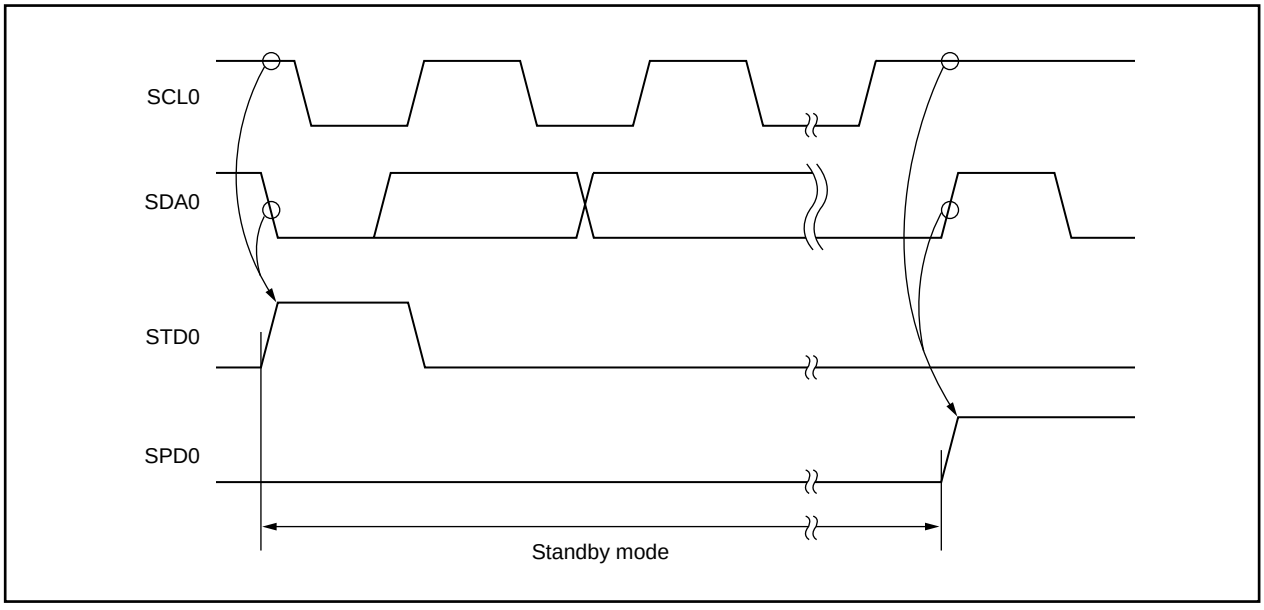
The communication reservation timing is shown below.

**Figure 18-12. Communication Reservation Timing**



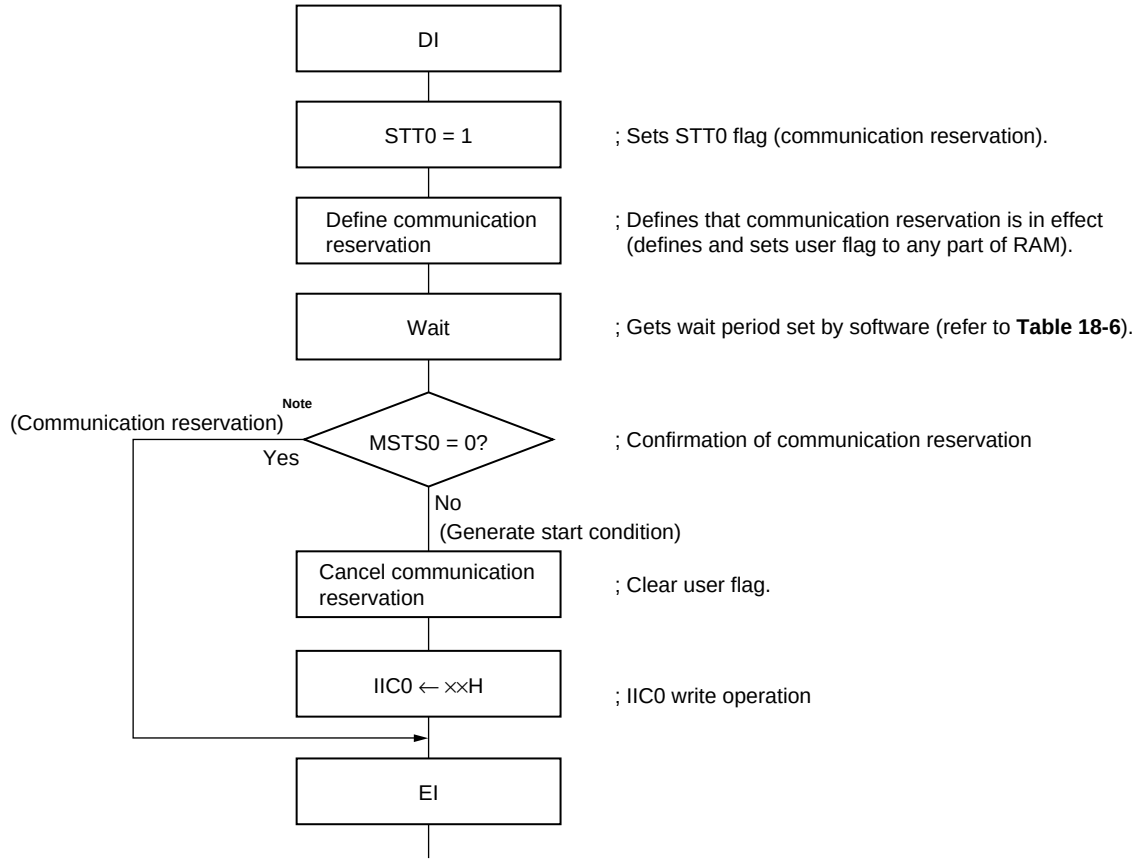

Communication reservations are accepted via the following timing. After the IICS0.STD0 bit is set to 1, a communication reservation can be made by setting the IICC0.STT0 bit to 1 before a stop condition is detected.

**Figure 18-13. Timing for Accepting Communication Reservations**



The communication reservation flowchart is illustrated below.

**Figure 18-14. Communication Reservation Flowchart**




**Note** The communication reservation operation executes a write to the IIC0 register when a stop condition interrupt request occurs.

### 18.13.2 When communication reservation function is disabled (IICF0.IICRSV0 bit = 1)

When the IICC0.STT0 bit is set when the bus is not used in a communication during bus communication, this request is rejected and a start condition is not generated. The following two statuses are included in the status where bus is not used.

- When arbitration results in neither master nor slave operation
- When an extension code is received and slave operation is disabled ($\overline{ACK}$ signal is not returned and the bus was released when the IICC0.LREL0 bit was set to 1)

To confirm whether the start condition was generated or request was rejected, check the IICF0.STCF0 flag. The time shown in Table 18-7 is required until the STCF0 flag is set after setting the STT0 bit = 1. Therefore, secure the time by software.

**Table 18-7. Wait Periods**

| CL01 | CL00 | Wait Period |
|---|---|---|
| 0 | 0 | 6 clocks |
| 0 | 1 | 6 clocks |
| 1 | 0 | 3 clocks |
| 1 | 1 | 9 clocks |

## 18.14 Cautions

**(1) When IICF0.STCEN0 bit = 0**

Immediately after I²C0 operation is enabled, the bus communication status (IICF0.IICBSY0 bit = 1) is recognized regardless of the actual bus status. To execute master communication in the status where a stop condition has not been detected, generate a stop condition and then release the bus before starting the master communication.

Use the following sequence for generating a stop condition.

<1> Set the IICCL0 register.
<2> Set the IICC0.IICE0 bit.
<3> Set the IICC0.SPT0 bit.

**(2) When IICF0.STCEN0 bit = 1**

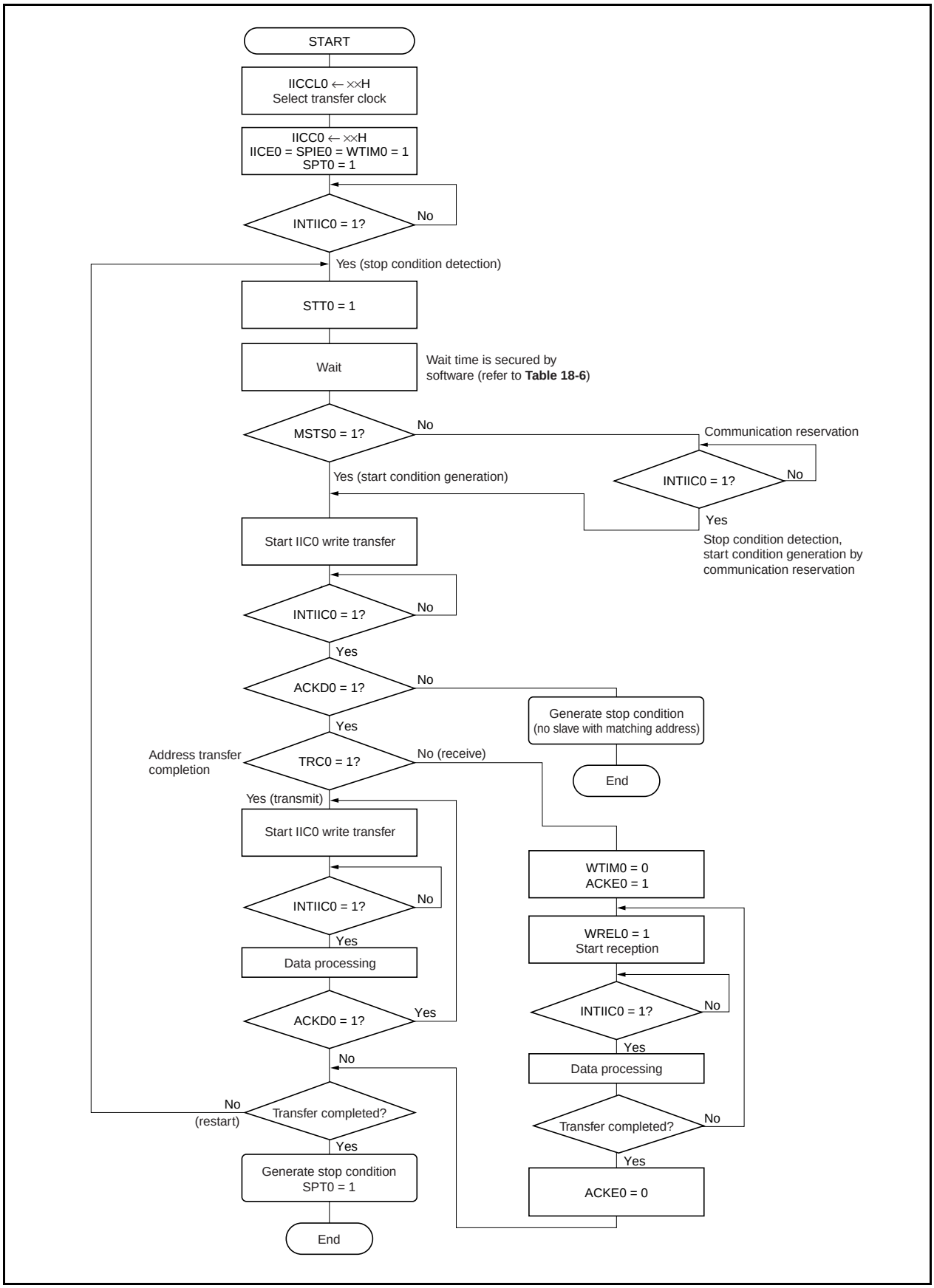
Immediately after I²C0 operation is enabled, the bus released status (IICBSY0 bit = 0) is recognized regardless of the actual bus status. To issue the first start condition (IICC0.STT0 bit = 1), it is necessary to confirm that the bus has been released, so as to not disturb other communications.

## 18.15 Communication Operations

### 18.15.1 Master operation 1

The following shows the flowchart for master communication when the communication reservation function is enabled (IICF0.IICRSV0 bit = 0) and the master operation is started after a stop condition is detected (IICF0.STCEN0 bit = 0).

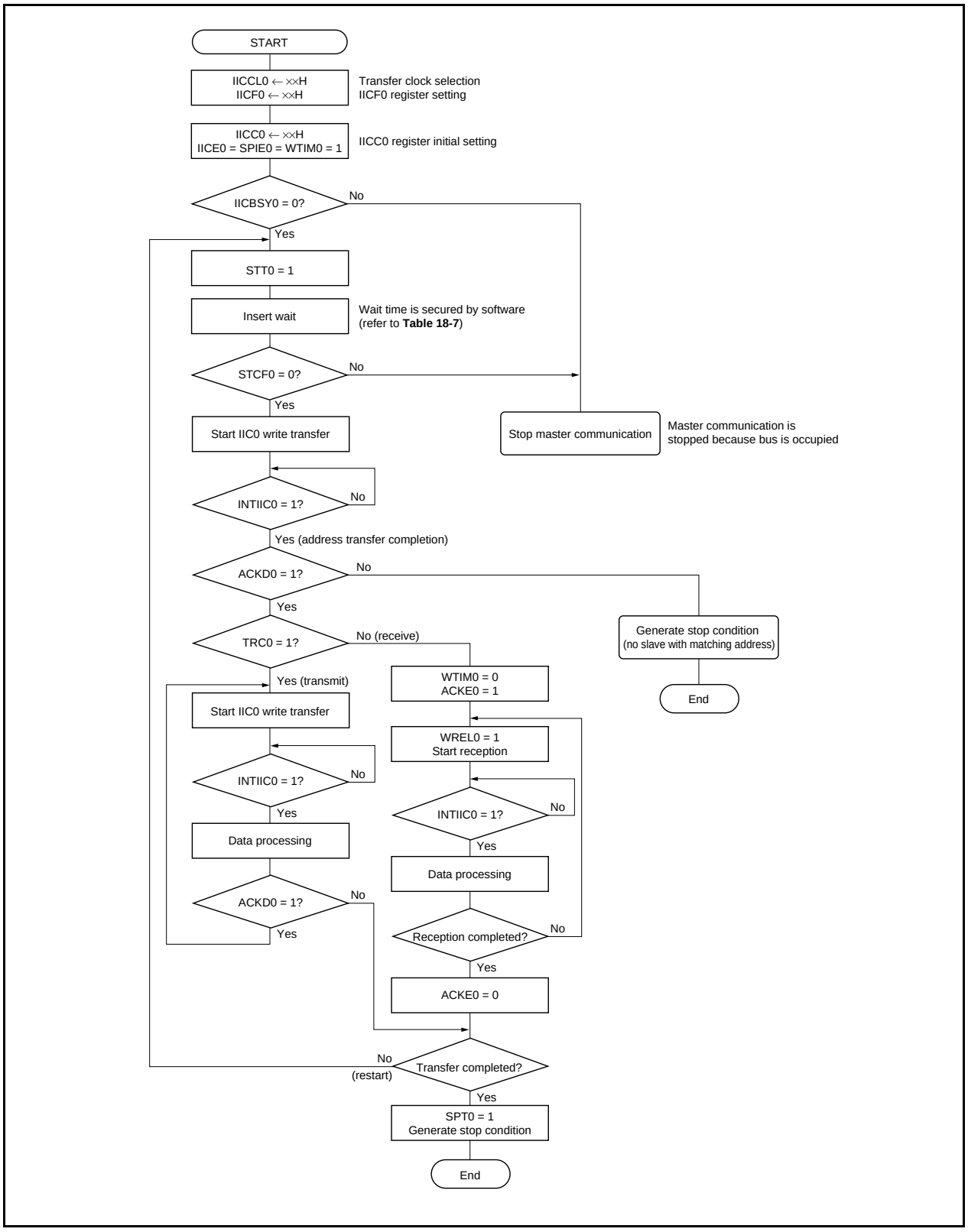
**Figure 18-15. Master Operation Flowchart (1)**



### 18.15.2 Master operation 2

The following shows the flowchart for master communication when the communication reservation function is disabled (IICRSV0 bit = 1) and the master operation is started without detecting a stop condition (STCEN0 bit = 1).

**Figure 18-16. Master Operation Flowchart (2)**



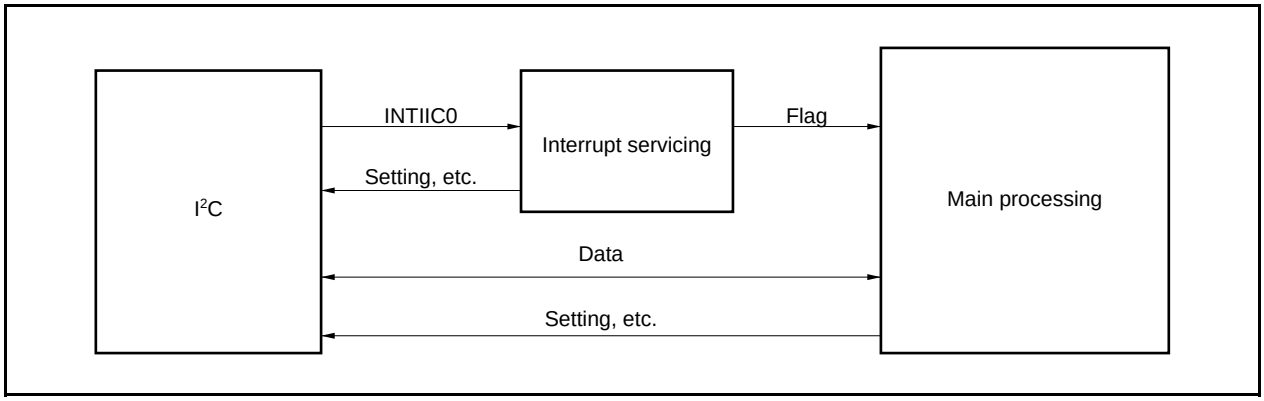
### 18.15.3 Slave operation

The following shows the processing procedure of the slave operation.

Basically, the operation of the slave device is event-driven. Therefore, processing by an INTIIC0 interrupt (processing requiring a significant change of the operation status, such as stop condition detection during communication) is necessary.

The following description assumes that data communication does not support extension codes. Also, it is assumed that the INTIIC0 interrupt servicing performs only status change processing and that the actual data communication is performed during the main processing.

**Figure 18-17. Software Outline During Slave Operation**




Therefore, the following three flags are prepared so that the data transfer processing can be performed by transmitting these flags to the main processing instead of the INTIIC0 signal.

**(1) Communication mode flag**

This flag indicates the following communication statuses.

Clear mode: Data communication not in progress

Communication mode: Data communication in progress (valid address detection stop condition detection, $\overline{\text{ACK}}$ signal from master not detected, address mismatch)

**(2) Ready flag**

This flag indicates that data communication is enabled. This is the same status as an INTIIC0 interrupt during normal data transfer. This flag is set in the interrupt servicing block and cleared in the main processing block. The ready flag for the first data for transmission is not set in the interrupt servicing block, so the first data is transmitted without clearance processing (the address match is regarded as a request for the next data).

**(3) Communication direction flag**

This flag indicates the direction of communication and is the same as the value of the IICS0.TRC0 bit.

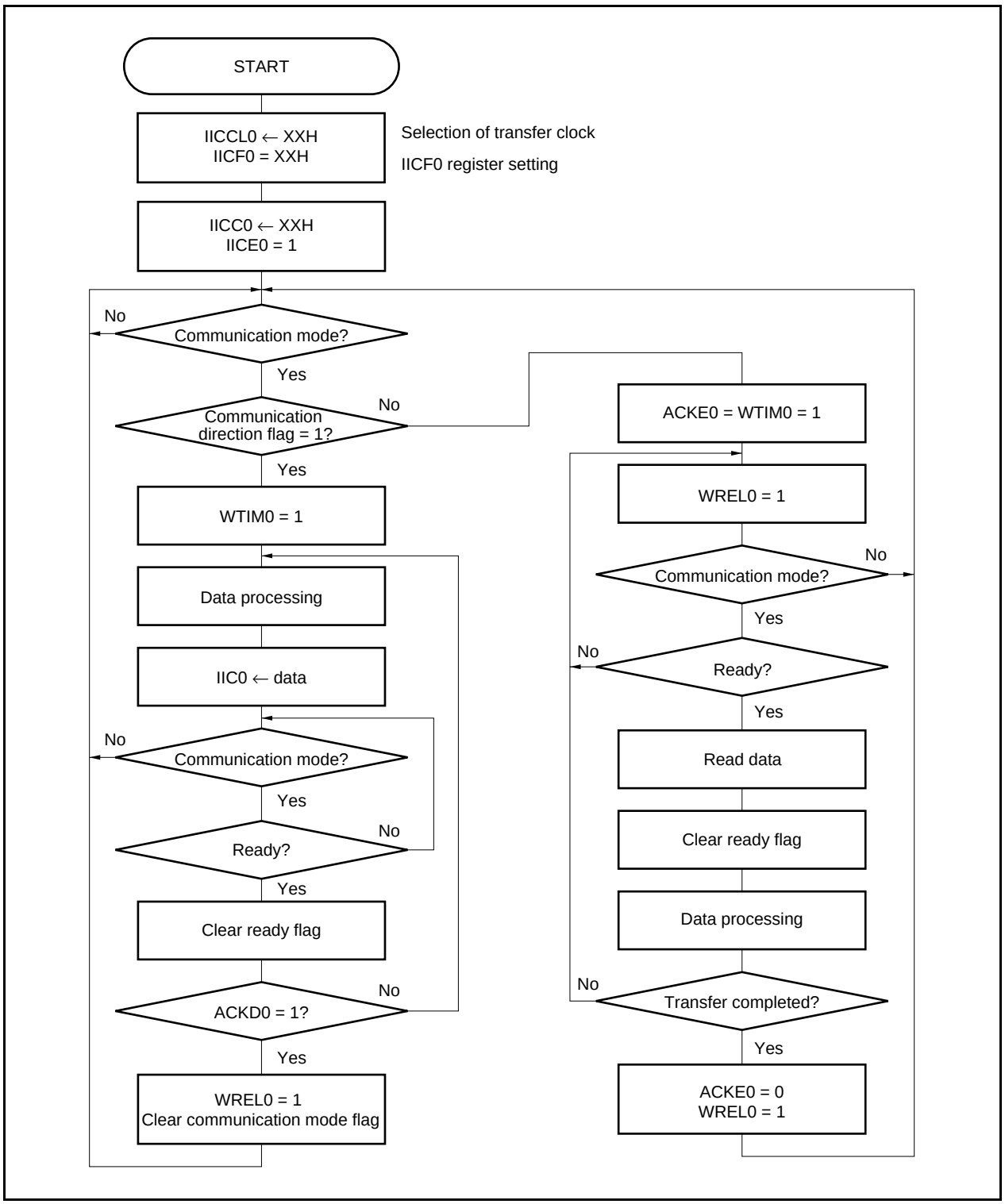
The following shows the operation of the main processing block during slave operation.

Start I²C0 and wait for the communication enabled status. When communication is enabled, perform transfer using the communication mode flag and ready flag (the processing of the stop condition and start condition is performed by interrupts, conditions are confirmed by flags).

For transmission, repeat the transmission operation until the master device stops returning $\overline{\text{ACK}}$ signal. When the master device stops returning $\overline{\text{ACK}}$ signal, transfer is complete.

For reception, receive the required number of data and do not return $\overline{ACK}$ signal for the next data immediately after transfer is complete. After that, the master device generates the stop condition or restart condition. This causes exit from communications.

**Figure 18-18. Slave Operation Flowchart (1)**



The following shows an example of the processing of the slave device by an INTIIC0 interrupt (it is assumed that no extension codes are used here). During an INTIIC0 interrupt, the status is confirmed and the following steps are executed.

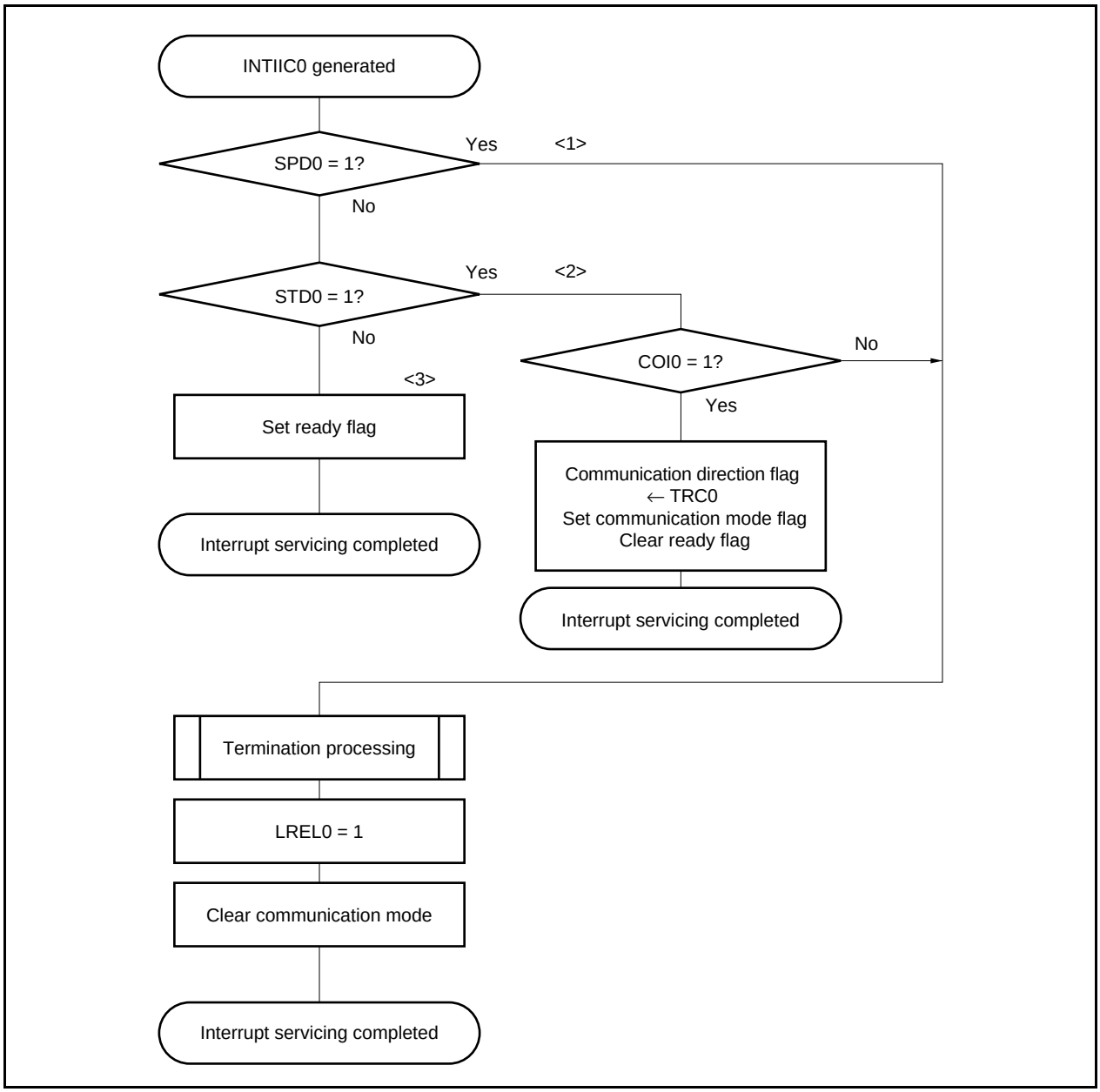
<1> When a stop condition is detected, communication is terminated.

<2> When a start condition is detected, the address is confirmed. If the address does not match, communication is terminated. If the address matches, the communication mode is set and wait is released, and operation returns from the interrupt (the ready flag is cleared).

<3> For data transmission/reception, when the ready flag is set, operation returns from the interrupt while the I²C0 bus remains in the wait status.

**Remark** <1> to <3> in the above correspond to <1> to <3> in **Figure 18-19 Slave Operation Flowchart (2)**.

**Figure 18-19. Slave Operation Flowchart (2)**



## 18.16 Timing of Data Communication

When using I²C bus mode, the master device outputs an address via the serial bus to select one of several slave devices as its communication partner.

After outputting the slave address, the master device transmits the IICS0.TRC0 bit that specifies the data transfer direction and then starts serial communication with the slave device.

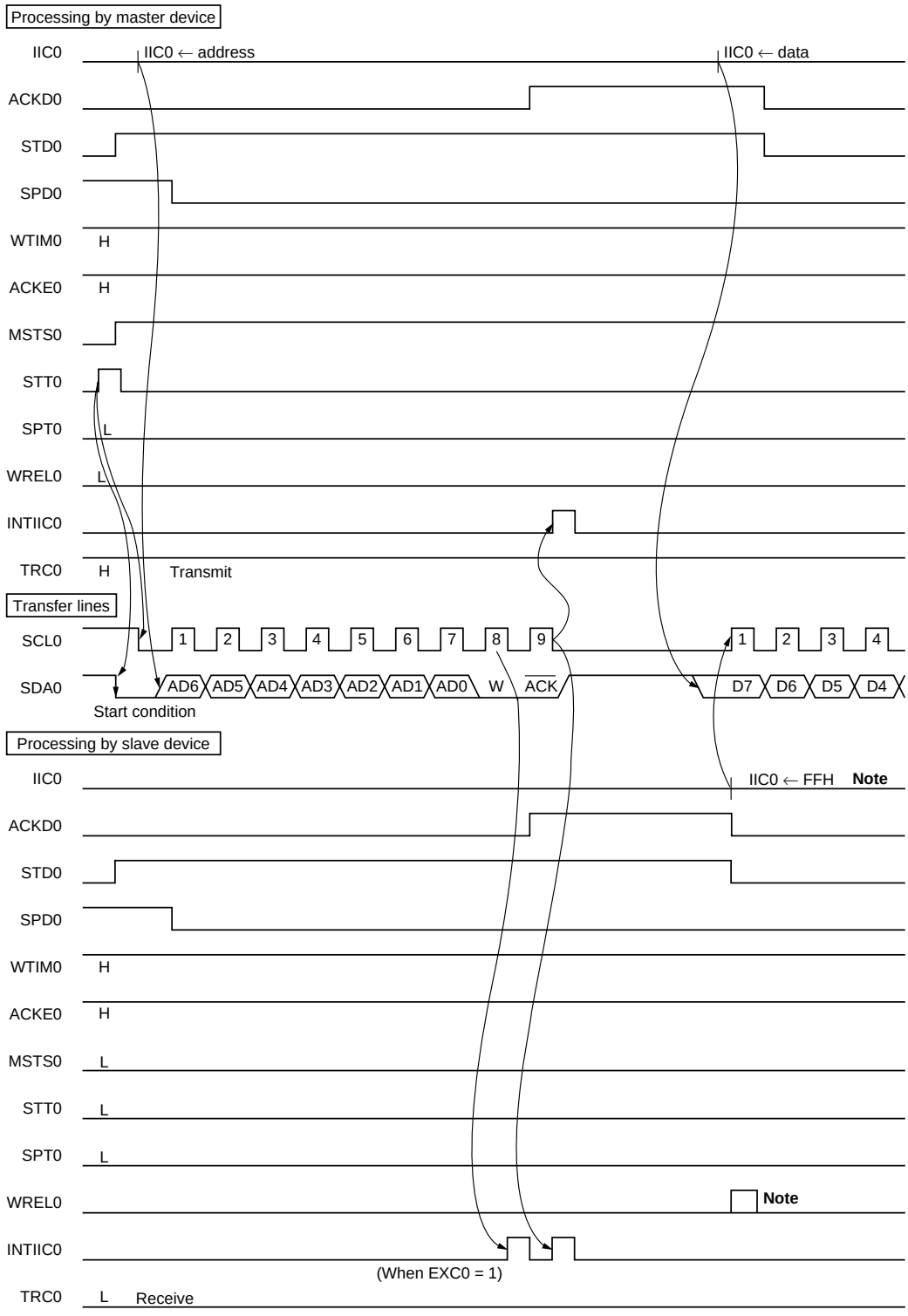
The IIC0 register's shift operation is synchronized with the falling edge of the serial clock (SCL0 pin). The transmit data is transferred to the SO latch and is output (MSB first) via the SDA0 pin.

Data input via the SDA0 pin is captured by the IIC0 register at the rising edge of the SCL0 pin.

The data communication timing is shown below.

**Figure 18-20. Example of Master to Slave Communication**
**(When 9-Clock Wait Is Selected for Both Master and Slave) (1/3)**

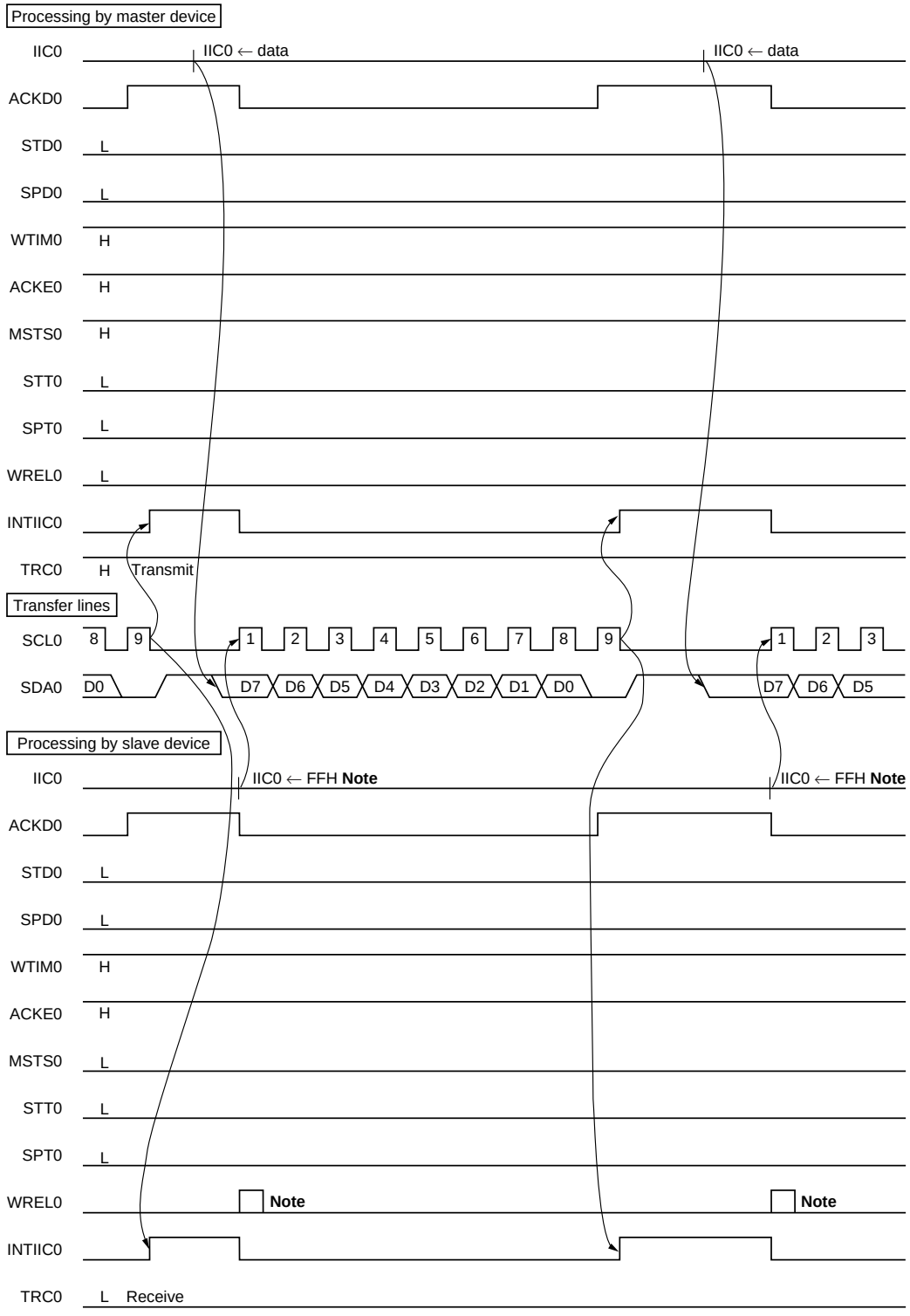
**(a) Start condition ~ address**




**Note** To cancel slave wait, write FFH to IIC0 or set WREL0.

**Figure 18-20. Example of Master to Slave Communication (When 9-Clock Wait Is Selected for Both Master and Slave) (2/3)**

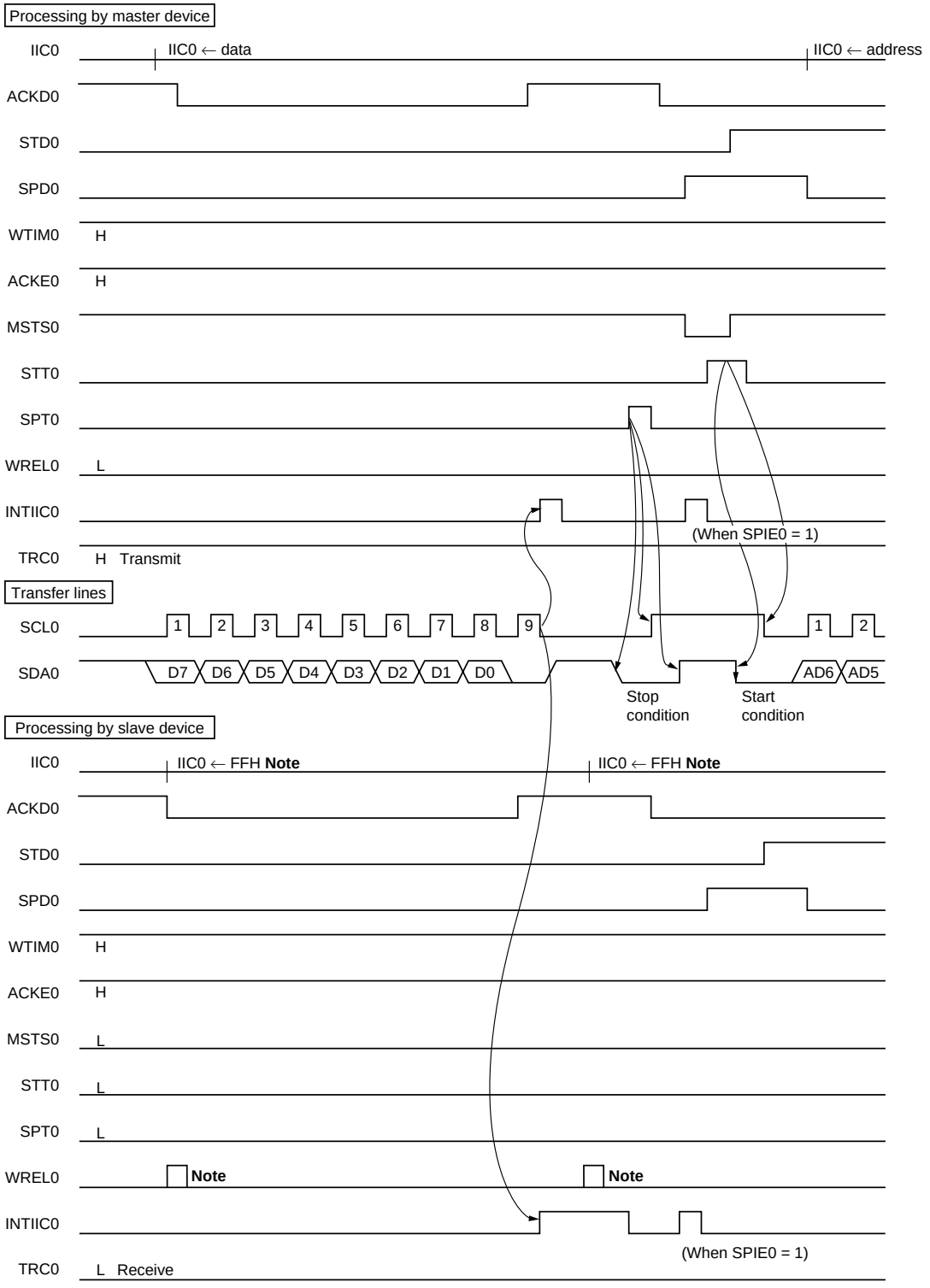
**(b) Data**




**Note** To cancel slave wait, write FFH to IIC0 or set WREL0.

**Figure 18-20. Example of Master to Slave Communication (When 9-Clock Wait Is Selected for Both Master and Slave) (3/3)**

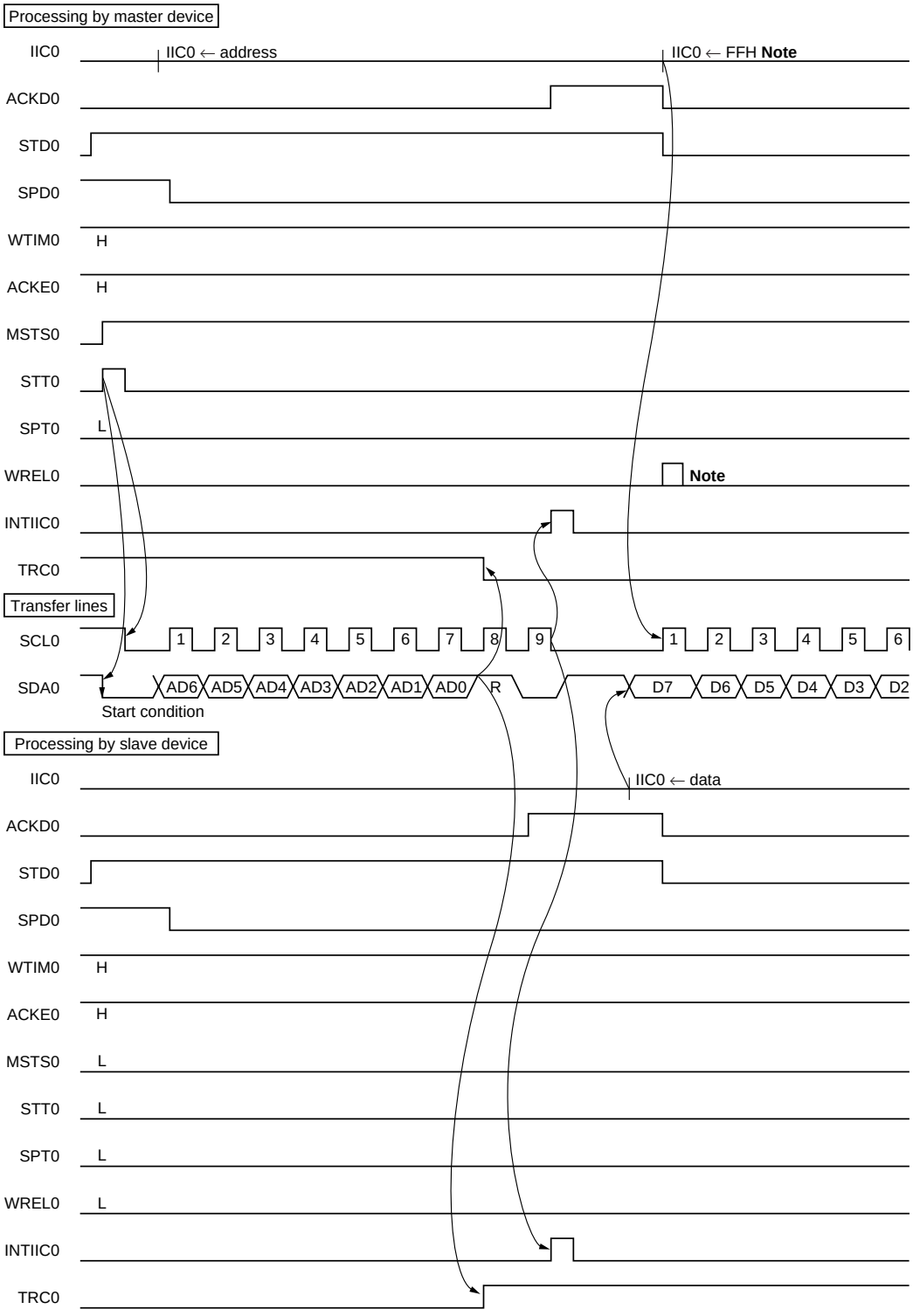
**(c) Stop condition**




**Note** To cancel slave wait, write FFH to IIC0 or set WREL0.

**Figure 18-21. Example of Slave to Master Communication (When 9-Clock Wait Is Selected for Both Master and Slave) (1/3)**

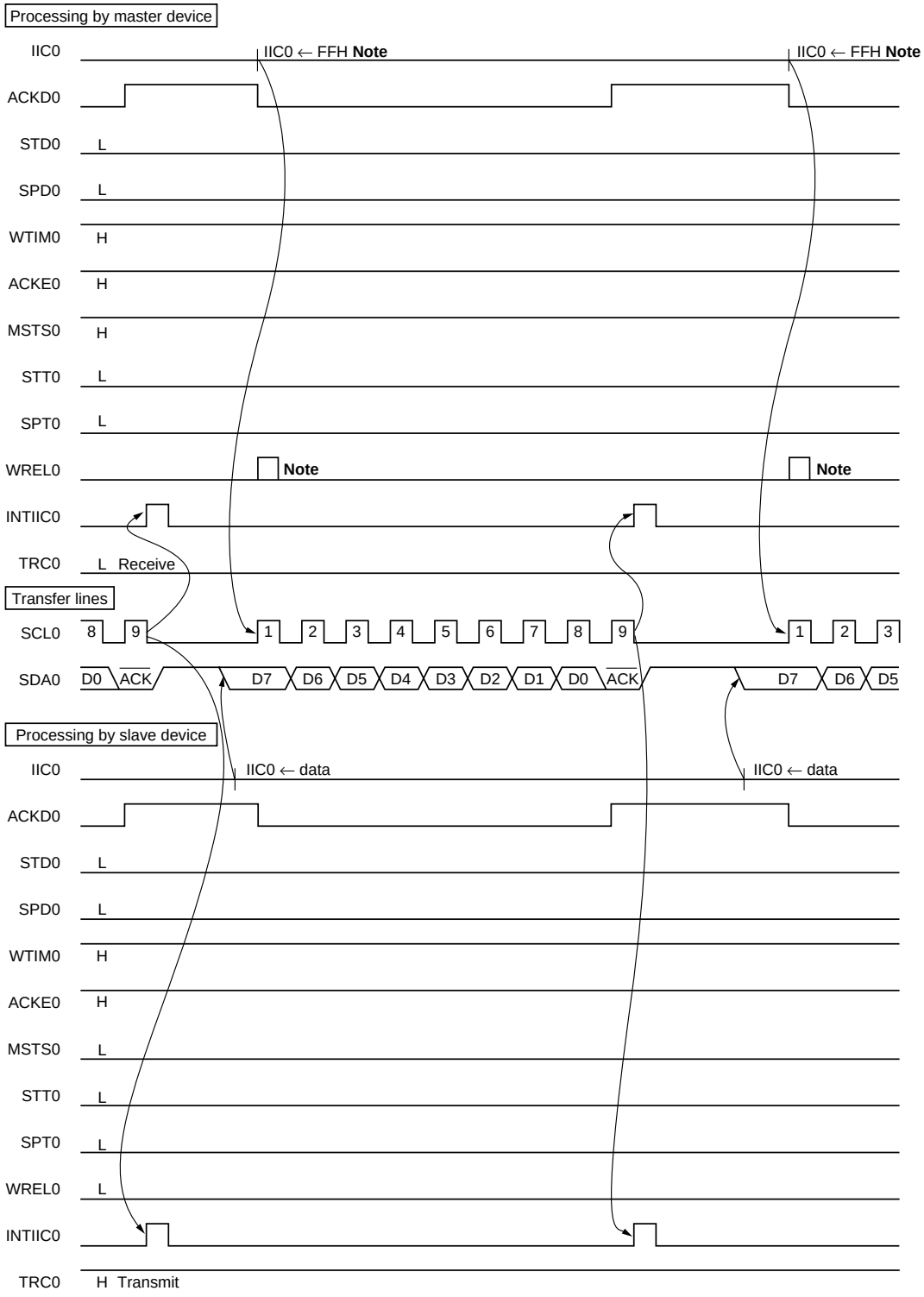
**(a) Start condition ~ address**




**Note** To cancel master wait, write FFH to IIC0 or set WREL0.

**Figure 18-21. Example of Slave to Master Communication**
**(When 9-Clock Wait Is Selected for Both Master and Slave) (2/3)**

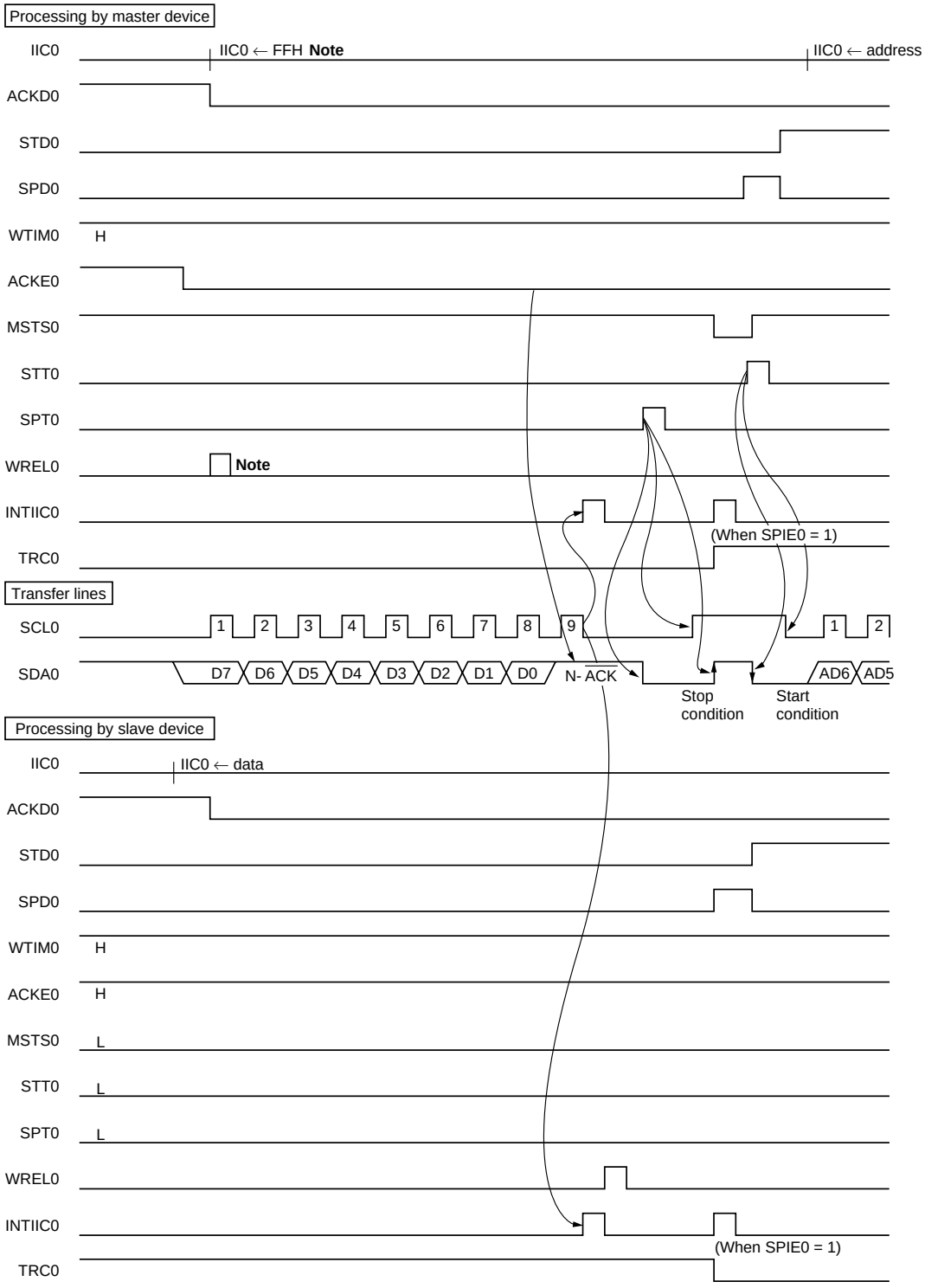
**(b) Data**




**Note** To cancel master wait, write FFH to IIC0 or set WREL0.

**Figure 18-21. Example of Slave to Master Communication (When 9-Clock Wait Is Selected for Both Master and Slave) (3/3)**

**(c) Stop condition**




**Note** To cancel master wait, write FFH to IIC0 or set WREL0.

# CHAPTER 19 INTERRUPT/EXCEPTION PROCESSING FUNCTION

## 19.1 Overview

The V850ES/KF1+ is provided with a dedicated interrupt controller (INTC) for interrupt servicing and realize an interrupt function that can service interrupt requests from a total of 38 or 39 sources.

An interrupt is an event that occurs independently of program execution, and an exception is an event whose occurrence is dependent on program execution.

The V850ES/KF1+ can process interrupt requests from the on-chip peripheral hardware and external sources. Moreover, exception processing can be started by the TRAP instruction (software exception) or by generation of an exception event (fetching of an illegal opcode) (exception trap).

### 19.1.1 Features

| Interrupt Source | | | | V850ES/KF1+ |
|---|---|---|---|---|
| Interrupt function | Non-maskable interrupt | External | | 1 channel (NMI pin) |
| | | Internal | | 2 channels (WDT1, WDT2) |
| | Maskable interrupt | External | | 8 channels (all edge detection interrupts) |
| | | Internal | WDT1 | 1 channel |
| | | | TMP | 3 channels |
| | | | TM0 | 4 channels |
| | | | TMH | 2 channels |
| | | | TM5 | 2 channels |
| | | | WT | 2 channels |
| | | | BRG | 1 channel |
| | | | UART | 6 channels |
| | | | CSI0 | 2 channels |
| | | | CSIA | 1 channel |
| | | | IIC | 1 channel[Note] |
| | | | KR | 1 channel |
| | | | AD | 1 channel |
| | | | LVI | 1 channel |
| | | | Total | 28 channels |
| Exception function | Software exception | | | 16 channels (TRAP00H to TRAP0FH) |
| | | | | 16 channels (TRAP10H to TRAP1FH) |
| | Exception trap | | | 2 channels (ILGOP/DBG0) |

**Note** Only in the μPD703308Y, 70F3306Y, 70F3308Y

Table 19-1 lists the interrupt/exception sources.

**Table 19-1. Interrupt Source List (1/2)**

| Type | Classification | Default Priority | Name | Trigger | Interrupt Source | Exception Code | Handler Address | Restored PC | Interrupt Control Register |
|---|---|---|---|---|---|---|---|---|---|
| Reset | Interrupt | – | RESET | $\overline{\text{RESET}}$ pin input | Pin | 0000H | 00000000H | Undefined | – |
| | | | | Internal reset input from WDT1, WDT2 | WDT1 WDT2 | | | | |
| Non-maskable | Interrupt | – | NMI | NMI pin valid edge input | Pin | 0010H | 00000010H | nextPC | – |
| | | – | INTWDT1 | WDT1 overflow (when non-maskable interrupt selected) | WDT1 | 0020H | 00000020H | **Note 1** | – |
| | | – | INTWDT2 | WDT2 overflow (when non-maskable interrupt selected) | WDT2 | 0030H | 00000020H | **Note 1** | – |
| Software exception | Exception | – | TRAP0n[Note 2] | TRAP instruction | – | 004nH[Note 2] | 00000040H | nextPC | – |
| | | – | TRAP1n[Note 2] | TRAP instruction | – | 005nH[Note 2] | 00000050H | nextPC | – |
| Exception trap | Exception | – | ILGOP/ DBG0 | Illegal opcode/DBTRAP instruction | – | 0060H | 00000060H | nextPC | – |
| Maskable | Interrupt | 0 | INTWDTM1 | WDT1 overflow (when interval timer selected) | WDT1 | 0080H | 00000080H | nextPC | WDT1IC |
| | | 1 | INTP0 | INTP0 pin valid edge input | Pin | 0090H | 00000090H | nextPC | PIC0 |
| | | 2 | INTP1 | INTP1 pin valid edge input | Pin | 00A0H | 000000A0H | nextPC | PIC1 |
| | | 3 | INTP2 | INTP2 pin valid edge input | Pin | 00B0H | 000000B0H | nextPC | PIC2 |
| | | 4 | INTP3 | INTP3 pin valid edge input | Pin | 00C0H | 000000C0H | nextPC | PIC3 |
| | | 5 | INTP4 | INTP4 pin valid edge input | Pin | 00D0H | 000000D0H | nextPC | PIC4 |
| | | 6 | INTP5 | INTP5 pin valid edge input | Pin | 00E0H | 000000E0H | nextPC | PIC5 |
| | | 7 | INTP6 | INTP6 pin valid edge input | Pin | 00F0H | 000000F0H | nextPC | PIC6 |
| | | 8 | INTTM000 | TM00 and CR000 match | TM00 | 0100H | 00000100H | nextPC | TM0IC00 |
| | | 9 | INTTM001 | TM00 and CR001 match | TM00 | 0110H | 00000110H | nextPC | TM0IC01 |
| | | 10 | INTTM010 | TM01 and CR010 match | TM01 | 0120H | 00000120H | nextPC | TM0IC10 |
| | | 11 | INTTM011 | TM01 and CR011 match | TM01 | 0130H | 00000130H | nextPC | TM0IC11 |
| | | 12 | INTTM50 | TM50 and CR50 match | TM50 | 0140H | 00000140H | nextPC | TM5IC0 |
| | | 13 | INTTM51 | TM51 and CR51 match | TM51 | 0150H | 00000150H | nextPC | TM5IC1 |
| | | 14 | INTCSI00 | CSI00 transfer completion | CSI00 | 0160H | 00000160H | nextPC | CSI0IC0 |
| | | 15 | INTCSI01 | CSI01 transfer completion | CSI01 | 0170H | 00000170H | nextPC | CSI0IC1 |
| | | 16 | INTSRE0 | UART0 reception error occurrence | UART0 | 0180H | 00000180H | nextPC | SREIC0 |
| | | 17 | INTSR0 | UART0 reception completion | UART0 | 0190H | 00000190H | nextPC | SRIC0 |
| | | 18 | INTST0 | UART0 transmission completion | UART0 | 01A0H | 000001AH | nextPC | STIC0 |

**Notes 1.** For restoration in the case of INTWDT1 and INTWDT2, refer to **19.10 Cautions**.

**2.** n = 0 to FH

**Table 19-1. Interrupt Source List (2/2)**

| Type | Classification | Default Priority | Name | Trigger | Interrupt Source | Exception Code | Handler Address | Restored PC | Interrupt Control Register |
|---|---|---|---|---|---|---|---|---|---|
| Maskable | Interrupt | 19 | INTSRE1 | UART1 reception error occurrence | UART1 | 01B0H | 000001B0H | nextPC | SREIC1 |
| | | 20 | INTSR1 | UART1 reception completion | UART1 | 01C0H | 000001C0H | nextPC | SRIC1 |
| | | 21 | INTST1 | UART1 transmission completion | UART1 | 01D0H | 000001D0H | nextPC | STIC1 |
| | | 22 | INTTMH0 | TMH0 and CMP00/CMP01 match | TMH0 | 01E0H | 000001E0H | nextPC | TMHIC0 |
| | | 23 | INTTMH1 | TMH1 and CMP10/CMP11 match | TMH1 | 01F0H | 000001F0H | nextPC | TMHIC1 |
| | | 24 | INTCSIA0 | CSIA0 transfer completion | CSIA0 | 0200H | 00000200H | nextPC | CSIAIC0 |
| | | 25 | INTIIC0[Note] | $I^2C0$ transfer completion | $I^2C0$ | 0210H | 00000210H | nextPC | IICIC0 |
| | | 26 | INTAD | A/D conversion completion | A/D | 0220H | 00000220H | nextPC | ADIC |
| | | 27 | INTKR | Key return interrupt | KR | 0230H | 00000230H | nextPC | KRIC |
| | | 28 | INTWTI | Watch timer interval | WT | 0240H | 00000240H | nextPC | WTIIC |
| | | 29 | INTWT | Watch timer reference time | WT | 0250H | 00000250H | nextPC | WTIC |
| | | 30 | INTBRG | 8-bit counter of prescaler 3 and PRSCM match | Prescaler 3 | 0260H | 00000260H | nextPC | BRGIC |
| | | 45 | INTLVI | Low-voltage detection | LVI | 0380H | 00000380H | nextPC | LVIIC |
| | | 46 | INTP7 | INTP7 pin valid edge input | Pin | 0390H | 00000390H | nextPC | PIC7 |
| | | 47 | INTTP0OV | TMP0 overflow | TMP | 03A0H | 000003A0H | nextPC | TPOVIC |
| | | 48 | INTTP0CC0 | TMP0 capture 0/ compare 0 match | TMP | 03B0H | 000003B0H | nextPC | TPCCIC0 |
| | | 49 | INTTP0CC1 | TMP0 capture 1/ compare 1 match | TMP | 03C0H | 000003C0H | nextPC | TPCCIC1 |

**Note** Only in the $\mu$PD703308Y, 70F3306Y, 70F3308Y

**Remarks 1.** Default priority: The priority order when two or more maskable interrupt requests with the same priority level are generated at the same time. The highest priority is 0.
The priority of non-maskable interrupt request is as follows.
INTWDT2 > INTWDT1 > NMI

Restored PC: The value of the program counter (PC) saved to EIPC, FEPC, or DBPC when interrupt/exception processing is started. The restored PC when a non-maskable or maskable interrupt is acknowledged while either of the following instructions is being executed does not become nextPC (when an interrupt is acknowledged during the execution of an instruction, the execution of that instruction is stopped and is resumed following completion of interrupt servicing).

- Load instructions (SLD.B, SLD.BU, SLD.H, SLD.HU, SLD.W)
- Divide instructions (DIV, DIVH, DIVU, DIVHU)
- PREPARE, DISPOSE instructions (only when an interrupt occurs before stack pointer update)

nextPC: The PC value at which processing is started following interrupt/exception processing.

**2.** The execution address of the illegal opcode when an illegal opcode exception occurs is calculated with (Restored PC – 4).

## 19.2 Non-Maskable Interrupts

Non-maskable interrupt request signals are acknowledged unconditionally, even when interrupts are disabled (DI state). Non-maskable interrupts (NMI) are not subject to priority control and take precedence over all other interrupt request signals.

The following three types of non-maskable interrupt request signals are available in the V850ES/KF1+.

- NMI pin input (NMI)
- Non-maskable interrupt request signal (INTWDT1) due to overflow of watchdog timer 1
- Non-maskable interrupt request signal (INTWDT2) due to overflow of watchdog timer 2

There are four choices for the valid edge of an NMI pin, namely: rising edge, falling edge, both edges, and no edge detection.

The non-maskable interrupt request signal (INTWDT1) due to overflow of watchdog timer 1 functions by setting the WDTM1.WDTM14 and WDTM1.WDTM13 bits to 10.

The non-maskable interrupt request signal (INTWDT2) due to overflow of watchdog timer 2 functions by setting the WDTM2.WDM21 and WDTM2.WDM20 bits to 01.

When two or more non-maskable interrupts occur simultaneously, they are processed in a sequence determined by the following priority order (the interrupt request signals with low priority level are ignored).

INTWDT2 > INTWDT1 > NMI

If during NMI processing, an NMI, INTWDT1, or INTWDT2 request signal newly occurs, processing is performed as follows.

**(1) If an NMI request signal newly occurs during NMI processing**

The new NMI request signal is held pending regardless of the value of the PSW.NP bit. The NMI request signal held pending is acknowledged upon completion of processing of the NMI currently being executed (following RETI instruction execution).

**(2) If an INTWDT1 request signal newly occurs during NMI processing**

If the NP bit remains set (to 1) during NMI processing, the new INTWDT1 request signal is held pending. The INTWDT1 request signal held pending is acknowledged upon completion of processing of the NMI currently being executed (following RETI instruction execution).

If the NP bit is cleared (to 0) during NMI processing, a newly generated INTWDT1 request signal is executed (NMI processing is interrupted).

**(3) If an INTWDT2 request signal newly occurs during NMI processing**

A newly generated INTWDT2 request signal is executed regardless of the value of the NP bit (NMI processing is interrupted).

**Caution For non-maskable interrupt servicing from non-maskable interrupt request signals (INTWDT1, INTWDT2), refer to 19.10 Cautions.**

**Figure 19-1. Acknowledging Non-Maskable Interrupt Request Signals (1/2)**

**(a) If two or more NMI request signals are simultaneously generated**

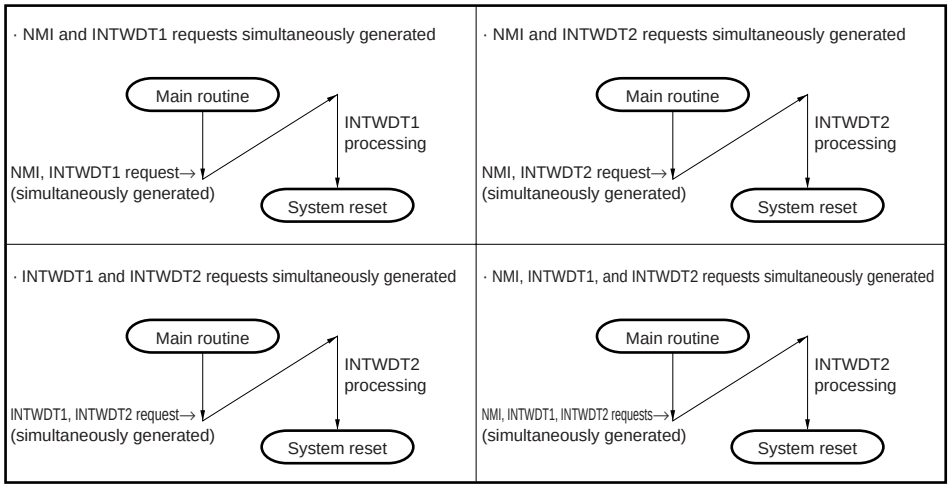
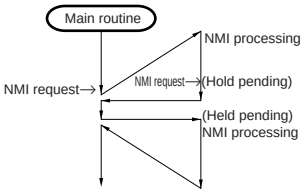
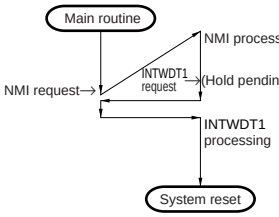
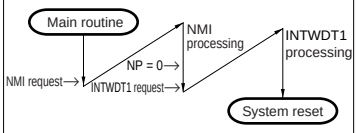
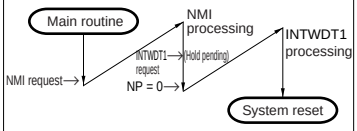
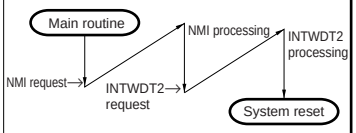
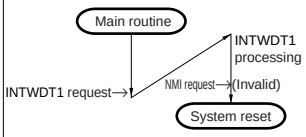
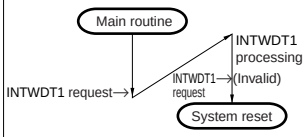
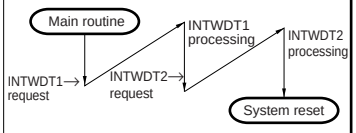
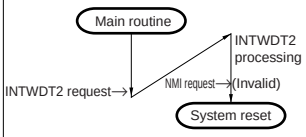
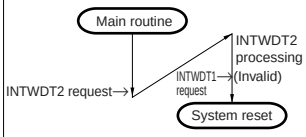
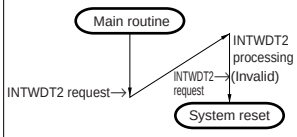


**Figure 19-1. Acknowledging Non-Maskable Interrupt Request Signals (2/2)**

**(b) If a new non-maskable interrupt request signal is generated during a non-maskable interrupt servicing**

| Non-maskable interrupt currently being serviced | Non-maskable interrupt request newly generated during non-maskable interrupt servicing | | |
|---|---|---|---|
| | NMI | INTWDT1 | INTWDT2 |
| NMI | · Generation of NMI request during NMI processing<br>  | · Generation of INTWDT1 request during NMI processing (NP = 1 state prior to INTWDT1 request is maintained)<br>  <br>· Generation of INTWDT1 request during NMI processing (Set NP = 0 before INTWDT1 request)<br>  <br>· Generation of INTWDT1 request during NMI processing (Set NP = 0 after INTWDT1 request)<br>  | · Generation of INTWDT2 request during NMI processing<br>  |
| INTWDT1 | · Generation of NMI request during INTWDT1 processing<br>  | · Generation of INTWDT1 request during INTWDT1 processing<br>  | · Generation of INTWDT2 request during INTWDT1 processing<br>  |
| INTWDT2 | · Generation of NMI request during INTWDT2 processing<br>  | · Generation of INTWDT1 request during INTWDT2 processing<br>  | · Generation of INTWDT2 request during INTWDT2 processing<br>  |

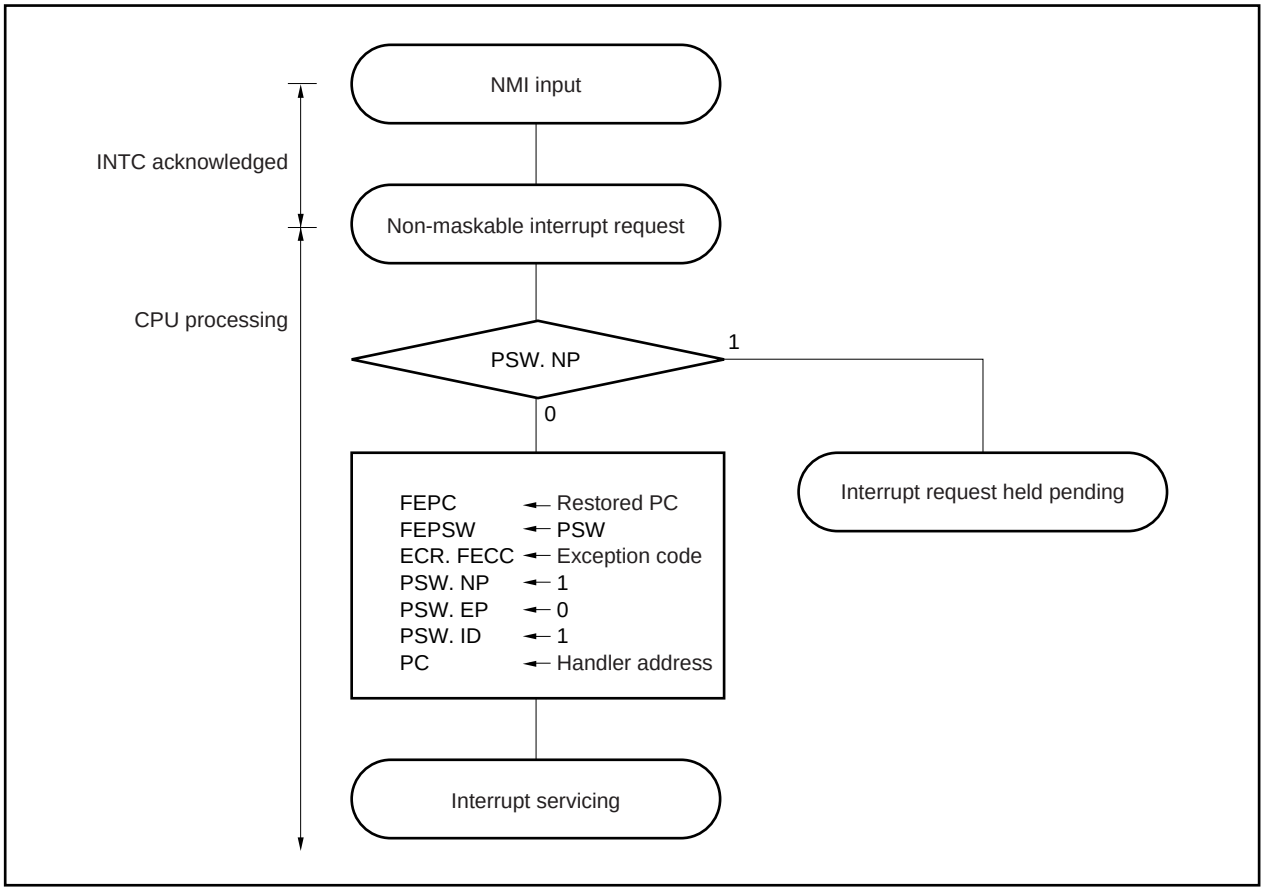
### 19.2.1 Operation

Upon generation of a non-maskable interrupt request signal, the CPU performs the following processing and transfers control to a handler routine.

<1> Saves the restored PC to FEPC.
<2> Saves the current PSW to FEPSW.
<3> Writes the exception code (0010H, 0020H, 0030H) to the higher halfword (FECC) of ECR.
<4> Sets the PSW.NP and PSW.ID bits to 1 and clears the PSW.EP bit to 0.
<5> Loads the handler address (00000010H, 00000020H, 00000030H) of the non-maskable interrupt to the PC and transfers control.

Figure 19-2 shows the servicing flow for non-maskable interrupts.

**Figure 19-2. Non-Maskable Interrupt Servicing**



### 19.2.2 Restore

Execution is restored from non-maskable interrupt servicing by the RETI instruction.

**(1) In case of NMI**

Restore from NMI processing is done with the RETI instruction.

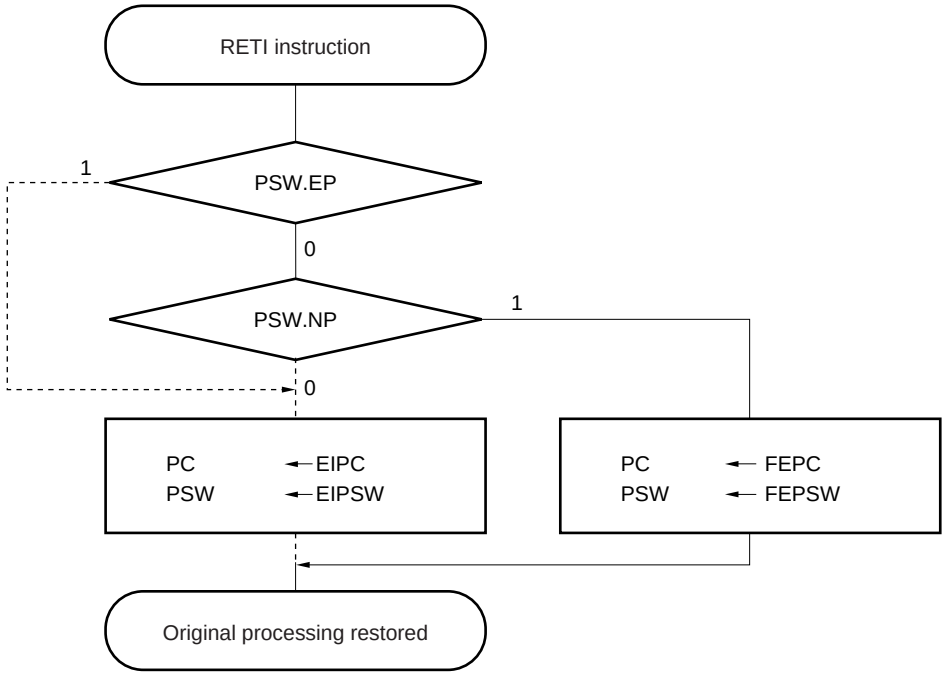
When the RETI instruction is executed, the CPU performs the following processing and transfers control to the address of the restored PC.

(i) Loads the values of the restored PC and PSW from FEPC and FEPSW, respectively, because the PSW.EP bit and the PSW.NP bit are 0 and 1, respectively.

(ii) Transfers control back to the loaded address of the restored PC and PSW.

Figure 19-3 shows the processing flow of the RETI instruction.

**Figure 19-3. RETI Instruction Processing**




**Caution When the EP bit and the NP bit are changed by the LDSR instruction during non-maskable interrupt servicing, in order to restore the PC and PSW correctly during restoring by the RETI instruction, it is necessary to clear the EP bit back to 0 and set the NP bit back to 1 using the LDSR instruction immediately before the RETI instruction.**

**Remark** The solid line shows the CPU processing flow.

**(2) In case of INTWDT1 and INTWDT2 signals**

For non-maskable interrupt servicing by the non-maskable interrupt request signals (INTWDT1, INTWDT2), refer to **19.10 Cautions**.

### 19.2.3 NP flag

The NP flag is a status flag that indicates that non-maskable interrupt servicing is in progress.

This flag is set when a non-maskable interrupt request has been acknowledged, and masks all non-maskable requests to prevent multiple interrupts.

After reset: 00000020H

| | 31 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| PSW | 0 | | NP | EP | ID | SAT | CY | OV | S | Z |

| NP | NMI servicing status |
|---|---|
| 0 | No non-maskable interrupt servicing |
| 1 | Non-maskable interrupt serving in progress |

## 19.3 Maskable Interrupts

Maskable interrupt request signals can be masked by interrupt control registers. The V850ES/KF1+ has 36 maskable interrupt sources (refer to **19.1.1 Features**).

If two or more maskable interrupt request signals are generated at the same time, they are acknowledged according to the default priority. In addition to the default priority, eight levels of interrupt priorities can be specified by using the interrupt control registers, allowing programmable priority control.

When an interrupt request signal has been acknowledged, the interrupt disabled (DI) status is set and the acknowledgment of other maskable interrupt request signals is disabled.

When the EI instruction is executed in an interrupt servicing routine, the interrupt enabled (EI) status is set, which enables acknowledgment of interrupt request signals having a priority higher than that of the interrupt request signal currently in progress. Note that only interrupt request signals with a higher priority have this capability; interrupt request signals with the same priority level cannot be nested.

To use multiple interrupts, it is necessary to save EIPC and EIPSW to memory or a register before executing the EI instruction, and restore EIPC and EIPSW to the original values by executing the DI instruction before the RETI instruction.

When the WDTM1.WDTM14 bit is cleared to 0, the watchdog timer 1 overflow interrupt functions as a maskable interrupt (INTWDTM1).

### 19.3.1 Operation

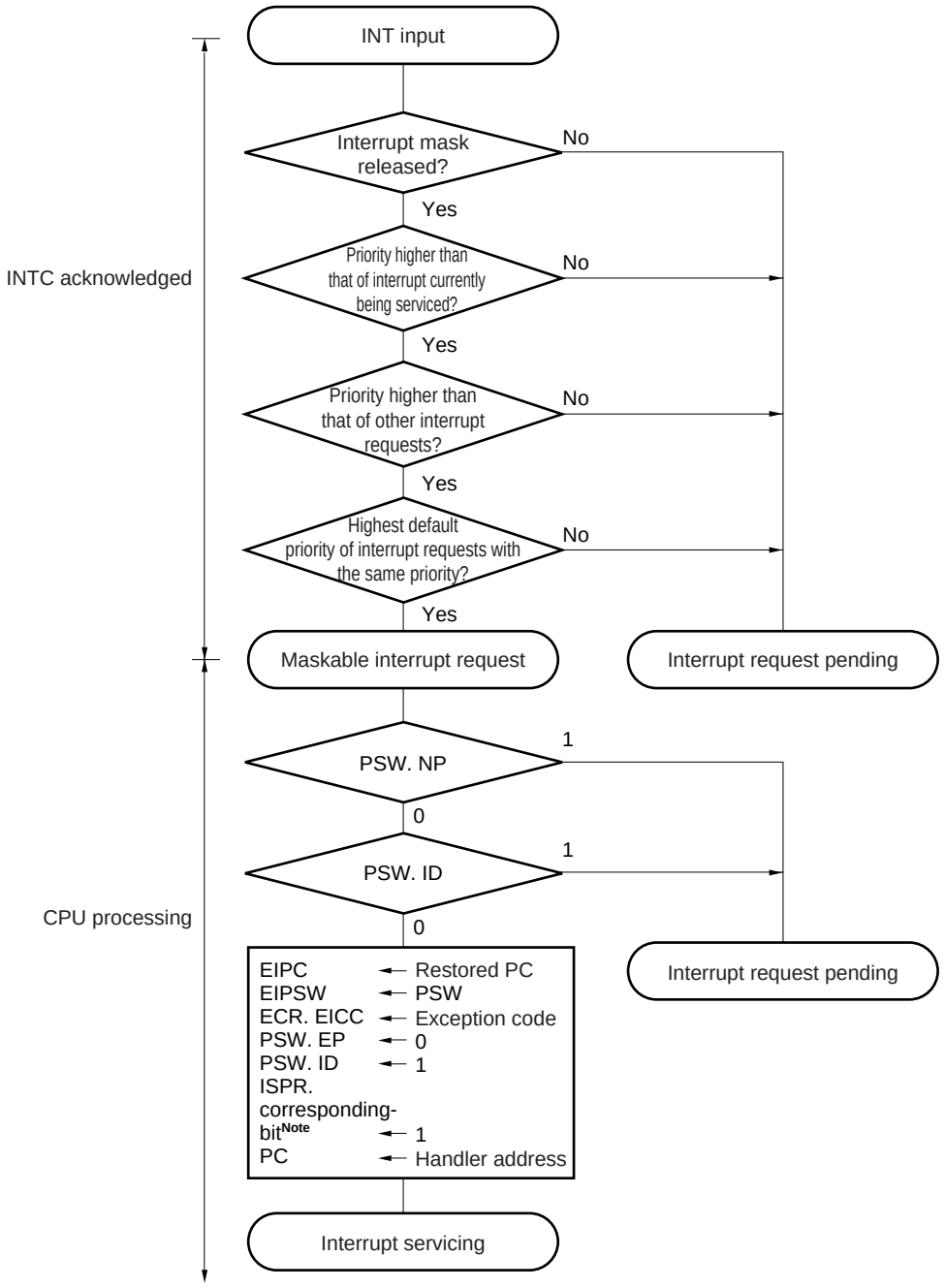
If a maskable interrupt request signal is generated, the CPU performs the following processing and transfers control to a handler routine.

<1> Saves the restored PC to EIPC.
<2> Saves the current PSW to EIPSW.
<3> Writes an exception code to the lower halfword of ECR (EICC).
<4> Sets the PSW.ID bit to 1 and clears the PSW.EP bit to 0.
<5> Loads the corresponding handler address to the PC and transfers control.

The maskable interrupt request signal masked by INTC and the maskable interrupt request signal that occurs while another interrupt is being serviced (when PSW.NP bit = 1 or ID bit = 1) are held pending internally. When the interrupts are unmasked, or when the NP bit = 0 and the ID bit = 0 by using the RETI and LDSR instructions, a new maskable interrupt servicing is started in accordance with the priority of the pending maskable interrupt request signal.

Figure 19-4 shows the servicing flow for maskable interrupts.

**Figure 19-4. Maskable Interrupt Servicing**




**Note** For the ISPR register, refer to **19.3.6 In-service priority register (ISPR)**.

### 19.3.2 Restore

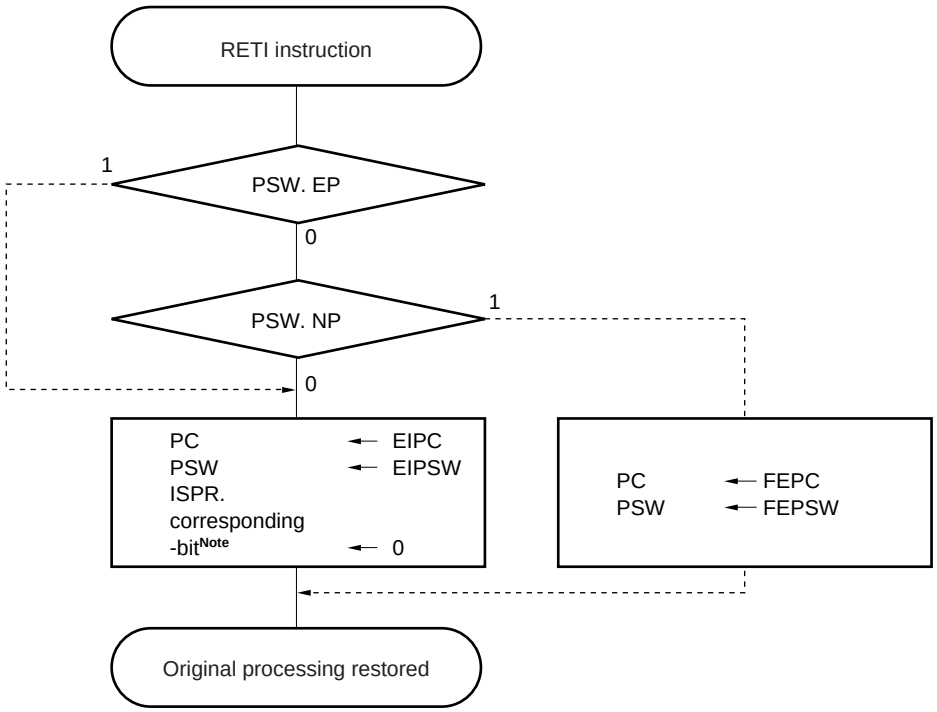
Execution is restored from maskable interrupt servicing by the RETI instruction.

When the RETI instruction is executed, the CPU performs the following processing and transfers control to the address of the restored PC.

(1) Loads the values of the restored PC and PSW from EIPC and EIPSW because the PSW.EP bit and the PSW.NP bit are both 0.
(2) Transfers control to the loaded address of the restored PC and PSW.

Figure 19-5 shows the processing flow of the RETI instruction.

**Figure 19-5. RETI Instruction Processing**




**Note** For the ISPR register, refer to **19.3.6 In-service priority register (ISPR)**.

**Caution When the EP bit and the NP bit are changed by the LDSR instruction during maskable interrupt servicing, in order to restore the PC and PSW correctly during restoring by the RETI instruction, it is necessary to clear the EP bit back to 0 and the NP bit back to 0 using the LDSR instruction immediately before the RETI instruction.**

**Remark** The solid line shows the CPU processing flow.

### 19.3.3 Priorities of maskable interrupts

INTC provides a multiple interrupt servicing in which an interrupt can be acknowledged while another interrupt is being serviced. Multiple interrupts can be controlled by priority levels.

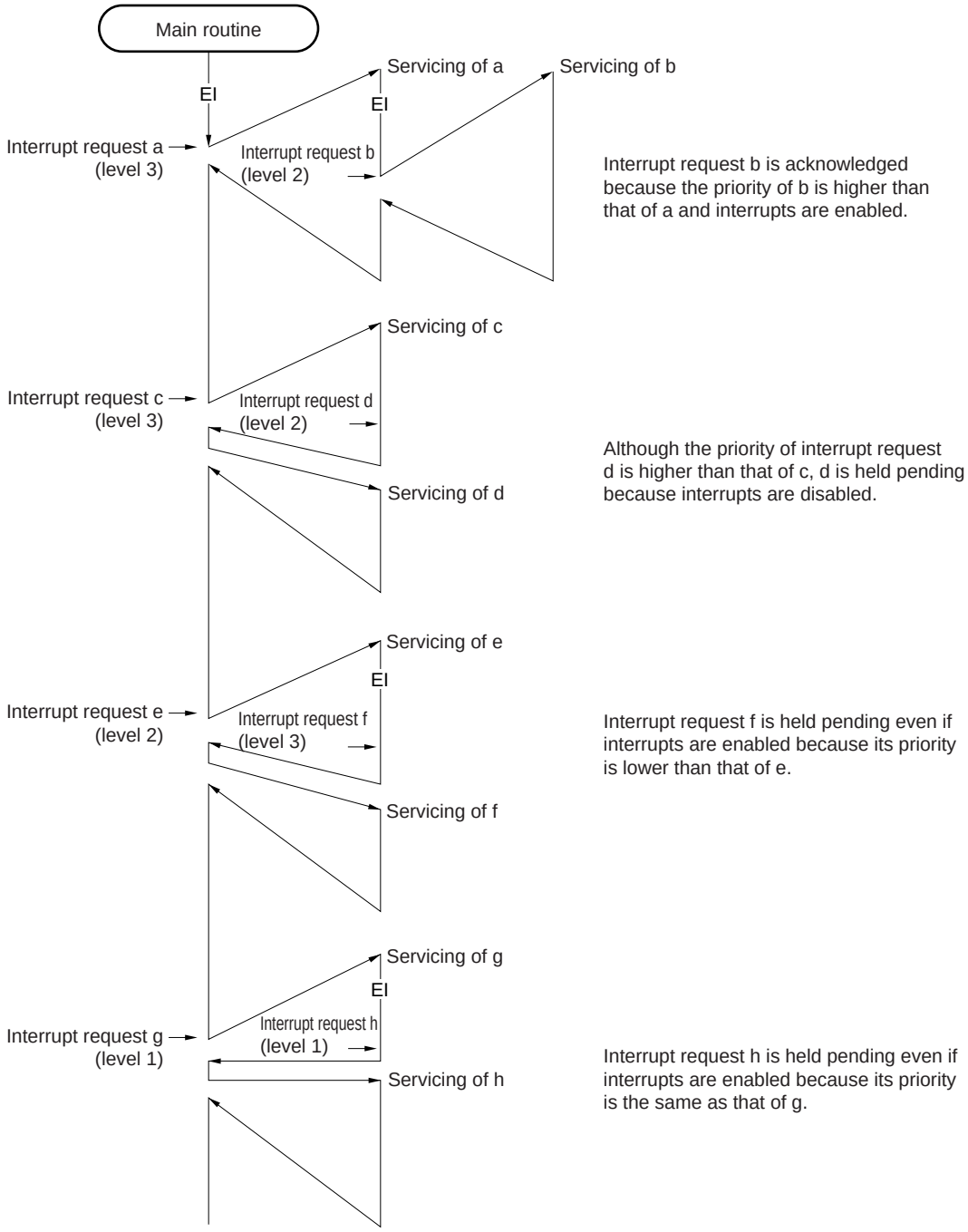
There are two types of priority level control: control based on the default priority levels, and control based on the programmable priority levels specified by the interrupt priority level specification bit (xxICn.xxPRn bit). When two or more interrupts having the same priority level specified by xxPRn are generated at the same time, interrupts are serviced in order depending on the priority level allocated to each interrupt request (default priority level) beforehand. For more information, refer to **Table 19-1 Interrupt Source List**. Programmable priority control divides interrupt requests into eight levels by setting the priority level specification flag.

Note that when an interrupt request signal is acknowledged, the PSW.ID flag is automatically set (1). Therefore, when multiple interrupts are to be used, clear (0) the ID flag beforehand (for example, by placing the EI instruction into the interrupt service program) to enable interrupts.

**Remark** xx: Identifying name of each peripheral unit (refer to **Table 19-2 Interrupt Control Registers (xxICn)**)

n: Peripheral unit number (refer to **Table 19-2 Interrupt Control Registers (xxICn)**)

**Figure 19-6. Example of Interrupt Nesting (1/2)**

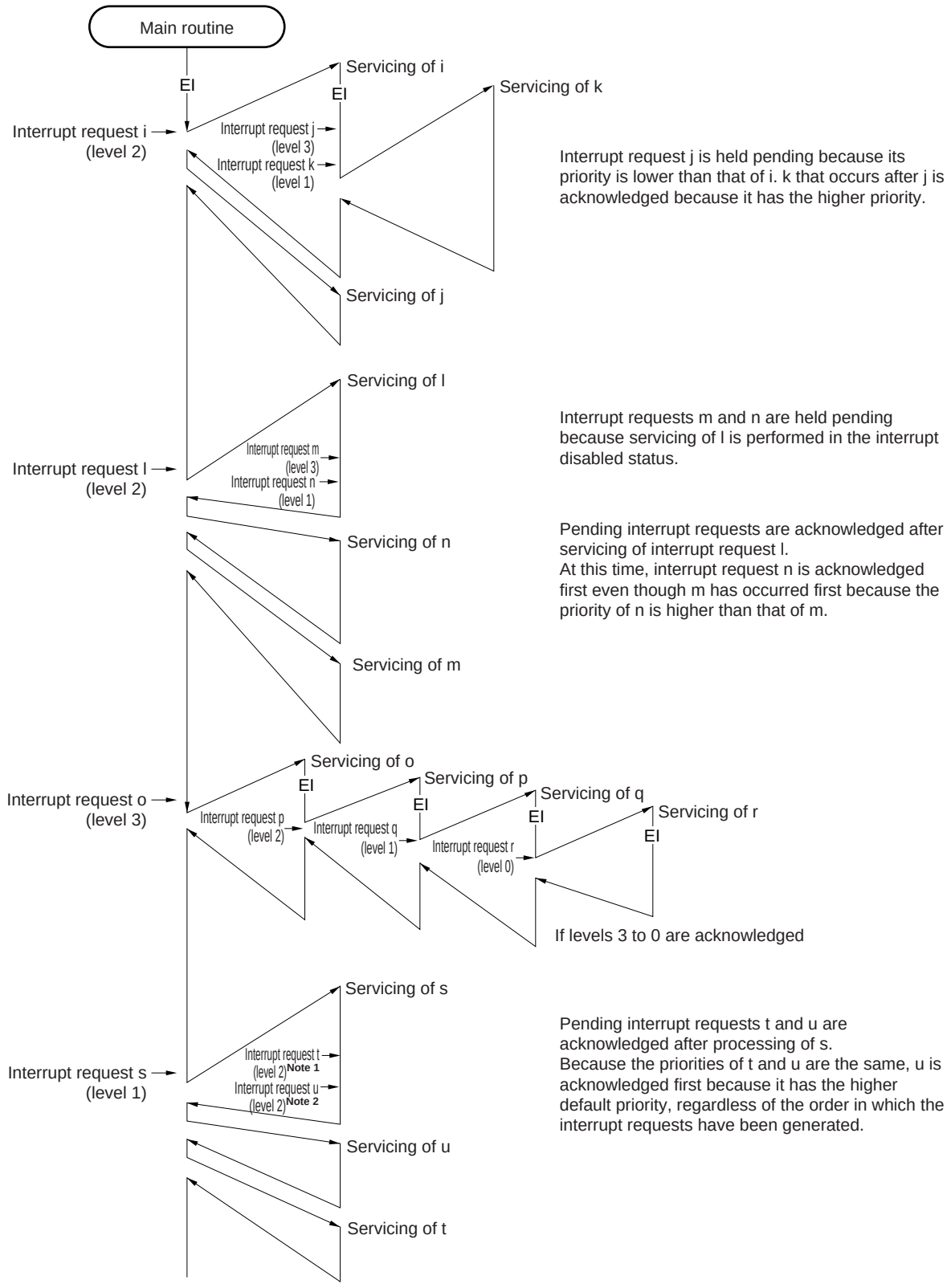



**Caution The values of EIPC and EIPSW must be saved before executing multiple interrupts.**

**Remarks 1.** a to u in the figure are the names of interrupt request signals shown for the sake of explanation.

**2.** The default priority in the figure indicates the relative priority between two interrupt request signals.

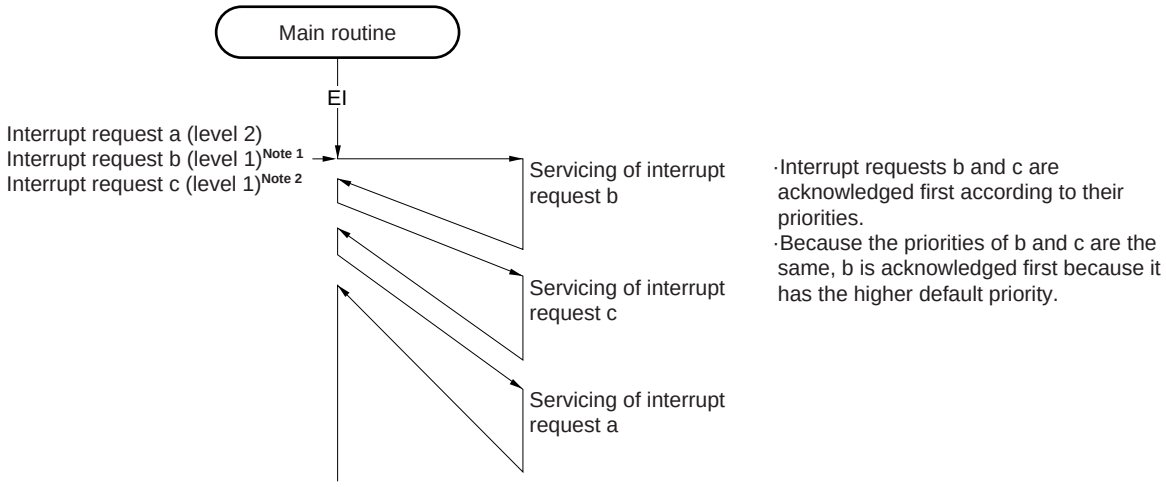
**Figure 19-6. Example of Interrupt Nesting (2/2)**




**Notes 1.** Lower default priority

**2.** Higher default priority

**Figure 19-7. Example of Servicing Simultaneously Generated Interrupt Request Signals**




**Notes 1.** Higher default priority

**2.** Lower default priority

### 19.3.4 Interrupt control register (xxICn)

An interrupt control register is assigned to each maskable interrupt and sets the control conditions for each maskable interrupt request.

The interrupt control registers can be read or written in 8-bit or 1-bit units.

After reset, xxICn is set to 47H.

**Caution Be sure to read the xxICn.xxIFn bit while interrupts are disabled (DI). If the xxIFn bit is read while interrupts are enabled (EI), an incorrect value may be read if there is a conflict between acknowledgment of the interrupt and reading of the bit.**

After reset: 47H R/W Address: FFFFF110H to FFFFF168H

| | <7> | <6> | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| xxICn | xxIFn | xxMKn | 0 | 0 | 0 | xxPRn2 | xxPRn1 | xxPRn0 |

| xxIFn | Interrupt request flag[Note] |
|---|---|
| 0 | Interrupt request not generated |
| 1 | Interrupt request generated |

| xxMKn | Interrupt mask flag |
|---|---|
| 0 | Enables interrupt servicing |
| 1 | Disables interrupt servicing (pending) |

| xxPRn2 | xxPRn1 | xxPRn0 | Interrupt priority specification bit |
|---|---|---|---|
| 0 | 0 | 0 | Specifies level 0 (highest) |
| 0 | 0 | 1 | Specifies level 1 |
| 0 | 1 | 0 | Specifies level 2 |
| 0 | 1 | 1 | Specifies level 3 |
| 1 | 0 | 0 | Specifies level 4 |
| 1 | 0 | 1 | Specifies level 5 |
| 1 | 1 | 0 | Specifies level 6 |
| 1 | 1 | 1 | Specifies level 7 (lowest) |

**Note** Automatically reset by hardware when interrupt request is acknowledged.

**Remark** xx: Identifying name of each peripheral unit (refer to **Table 19-2 Interrupt Control Registers (xxICn)**)

n: Peripheral unit number (refer to **Table 19-2 Interrupt Control Registers (xxICn)**)

Following tables list the addresses and bits of the interrupt control registers.

**Table 19-2. Interrupt Control Registers (xxICn)**

| Address | Register | Bits | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | <7> | <6> | 5 | 4 | 3 | 2 | 1 | 0 |
| FFFFF110H | WDT1IC | WDT1IF | WDT1MK | 0 | 0 | 0 | WDT1PR2 | WDT1PR1 | WDT1PR0 |
| FFFFF112H | PIC0 | PIF0 | PMK0 | 0 | 0 | 0 | PPR02 | PPR01 | PPR00 |
| FFFFF114H | PIC1 | PIF1 | PMK1 | 0 | 0 | 0 | PPR12 | PPR11 | PPR10 |
| FFFFF116H | PIC2 | PIF2 | PMK2 | 0 | 0 | 0 | PPR22 | PPR21 | PPR20 |
| FFFFF118H | PIC3 | PIF3 | PMK3 | 0 | 0 | 0 | PPR32 | PPR31 | PPR30 |
| FFFFF11AH | PIC4 | PIF4 | PMK4 | 0 | 0 | 0 | PPR42 | PPR41 | PPR40 |
| FFFFF11CH | PIC5 | PIF5 | PMK5 | 0 | 0 | 0 | PPR52 | PPR51 | PPR50 |
| FFFFF11EH | PIC6 | PIF6 | PMK6 | 0 | 0 | 0 | PPR62 | PPR61 | PPR60 |
| FFFFF120H | TM0IC00 | TM0IF00 | TM0MK00 | 0 | 0 | 0 | TM0PR002 | TM0PR001 | TM0PR000 |
| FFFFF122H | TM0IC01 | TM0IF01 | TM0MK01 | 0 | 0 | 0 | TM0PR012 | TM0PR011 | TM0PR010 |
| FFFFF124H | TM0IC10 | TM0IF10 | TM0MK10 | 0 | 0 | 0 | TM0PR102 | TM0PR101 | TM0PR100 |
| FFFFF126H | TM0IC11 | TM0IF11 | TM0MK11 | 0 | 0 | 0 | TM0PR112 | TM0PR111 | TM0PR110 |
| FFFFF128H | TM5IC0 | TM5IF0 | TM5MK0 | 0 | 0 | 0 | TM5PR02 | TM5PR01 | TM5PR00 |
| FFFFF12AH | TM5IC1 | TM5IF1 | TM5MK1 | 0 | 0 | 0 | TM5PR12 | TM5PR11 | TM5PR10 |
| FFFFF12CH | CSI0IC0 | CSI0IF0 | CSI0MK0 | 0 | 0 | 0 | CSI0PR02 | CSI0PR01 | CSI0PR00 |
| FFFFF12EH | CSI0IC1 | CSI0IF1 | CSI0MK1 | 0 | 0 | 0 | CSI0PR12 | CSI0PR11 | CSI0PR10 |
| FFFFF130H | SREIC0 | SREIF0 | SREMK0 | 0 | 0 | 0 | SREPR02 | SREPR01 | SREPR00 |
| FFFFF132H | SRIC0 | SRIF0 | SRMK0 | 0 | 0 | 0 | SRPR02 | SRPR01 | SRPR00 |
| FFFFF134H | STIC0 | STIF0 | STMK0 | 0 | 0 | 0 | STPR02 | STPR01 | STPR00 |
| FFFFF136H | SREIC1 | SREIF1 | SREMK1 | 0 | 0 | 0 | SREPR12 | SREPR11 | SREPR10 |
| FFFFF138H | SRIC1 | SRIF1 | SRMK1 | 0 | 0 | 0 | SRPR12 | SRPR11 | SRPR10 |
| FFFFF13AH | STIC1 | STIF1 | STMK1 | 0 | 0 | 0 | STPR12 | STPR11 | STPR10 |
| FFFFF13CH | TMHIC0 | TMHIF0 | TMHMK0 | 0 | 0 | 0 | TMHPR02 | TMHPR01 | TMHPR00 |
| FFFFF13EH | TMHIC1 | TMHIF1 | TMHMK1 | 0 | 0 | 0 | TMHPR12 | TMHPR11 | TMHPR10 |
| FFFFF140H | CSIAIC0 | CSIAIF0 | CSIAMK0 | 0 | 0 | 0 | CSIAPR02 | CSIAPR01 | CSIAPR00 |
| FFFFF142H | IICIC0[Note] | IICIF0 | IICMK0 | 0 | 0 | 0 | IICPR02 | IICPR01 | IICPR00 |
| FFFFF144H | ADIC | ADIF | ADMK | 0 | 0 | 0 | ADPR2 | ADPR1 | ADPR0 |
| FFFFF146H | KRIC | KRIF | KRMK | 0 | 0 | 0 | KRPR2 | KRPR1 | KRPR0 |
| FFFFF148H | WTIIC | WTIIF | WTIMK | 0 | 0 | 0 | WTIPR2 | WTIPR1 | WTIPR0 |
| FFFFF14AH | WTIC | WTIF | WTMK | 0 | 0 | 0 | WTPR2 | WTPR1 | WTPR0 |
| FFFFF14CH | BRGIC | BRGIF | BRGMK | 0 | 0 | 0 | BRGPR2 | BRGPR1 | BRGPR0 |
| FFFFF170H | LVIIC | LVIIF | LVIMK | 0 | 0 | 0 | LVIPR2 | LVIPR1 | LVIPR0 |
| FFFFF172H | PIC7 | PIF7 | PMK7 | 0 | 0 | 0 | PPR72 | PPR71 | PPR70 |
| FFFFF174H | TP0OVIC | TP0OVIF | TP0OVMK | 0 | 0 | 0 | TP0OVPR2 | TP0OVPR1 | TP0OVPR0 |
| FFFFF176H | TP0CCIC0 | TP0CCIF0 | TP0CCMK0 | 0 | 0 | 0 | TP0CCPR02 | TP0CCPR01 | TP0CCPR00 |
| FFFFF178H | TP0CCIC1 | TP0CCIF1 | TP0CCMK1 | 0 | 0 | 0 | TP0CCPR12 | TP0CCPR11 | TP0CCPR10 |

**Note** Only in the μPD703308Y, 70F3306Y, 70F3308Y

### 19.3.5 Interrupt mask registers 0, 1, 3 (IMR0, IMR1, IMR3)

These registers set the interrupt mask status for maskable interrupts. The xxMKn bit of the IMR0, IMR1, and IMR3 registers and the xxMKn bit of the xxICn register are respectively linked.

The IMRm register can be read or written in 16-bit units (m = 0, 1, 3).

When the higher 8 bits of the IMRk register are used as the IMRkH register and the lower 8 bits of the IMRk register as the IMRkL register, they can be read or written in 8-bit or 1-bit units (k = 0, 1).

**Caution In the device file, the xxMKn bit of the xxICn register is defined as a reserved word. Therefore, if bit manipulation is performed using the name xxMKn, the xxICn register, not the IMRm register, is rewritten (as a result, the IMRm register is also rewritten).**

After reset: FFFFH R/W Address: IMR0 FFFFF100H, IMR0L FFFFF100H, IMR0H FFFFF101H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| IMR0 (IMR0H[Note]) | CSI0MK1 | CSI0MK0 | TM5MK1 | TM5MK0 | TM0MK11 | TM0MK10 | TM0MK01 | TM0MK00 |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| (IMR0L) | PMK6 | PMK5 | PMK4 | PMK3 | PMK2 | PMK1 | PMK0 | WDT1MK |

After reset: FFFFH R/W Address: IMR1 FFFFF102H, IMR1L FFFFF102H, IMR1H FFFFF103H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| IMR1 (IMR1H[Note]) | 1 | BRGMK | WTMK | WTIMK | KRMK | ADMK | IICMK0 | CSIAMK0 |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| (IMR1L) | TMHMK1 | TMHMK0 | STMK1 | SRMK1 | SREMK1 | STMK0 | SRMK0 | SREMK0 |

After reset: FFFFH R/W Address: IMR3 FFFFF106H, IMR3L FFFFF106H

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| IMR3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| (IMR3L) | 1 | 1 | 1 | TP0CCMK1 | TP0CCMK2 | TP0OVFMK | PMK7 | LVIMK |

| xxMKn | Interrupt mask flag setting |
|---|---|
| 0 | Enables interrupt servicing |
| 1 | Disables interrupt servicing |

**Note** When reading from or writing to bits 8 to 15 of the IMR0 and IMR1 registers in 8-bit or 1-bit units, specify these bits as bits 0 to 7 of the IMR0H and IMR1H registers.

**Caution Set bit 15 of the IMR1 register and bits 15 to 5 of the IMR3 register to 1. The operation is not guaranteed if their value is changed.**

**Remark** xx: Identifying name of each peripheral unit (refer to **Table 19-2 Interrupt Control Registers (xxICn)**)

n: Peripheral unit number (refer to **Table 19-2 Interrupt Control Registers (xxICn)**)

### 19.3.6 In-service priority register (ISPR)

This register holds the priority level of the maskable interrupt currently being acknowledged. When the interrupt request signal is acknowledged, the bit of this register corresponding to the priority level of that interrupt request signal is set (1) and remains set while the interrupt is being serviced.

When the RETI instruction is executed, the bit among those that are set (1) in the ISPR register that corresponds to the interrupt request signal having the highest priority is automatically cleared (0) by hardware. However, it is not cleared (0) when execution is returned from non-maskable interrupt servicing or exception processing.

This register is read-only in 8-bit or 1-bit units.

After reset, ISPR is cleared to 00H.

**Caution If an interrupt is acknowledged while the ISPR register is being read in the interrupt enabled (EI) status, the value of the ISPR register after the bits of the register have been set to 1 by acknowledging the interrupt may be read. To accurately read the value of the ISPR register before an interrupt is acknowledged, read the register while interrupts are disabled (DI status).**

After reset: 00H R Address: FFFFF1FAH

| | <7> | <6> | <5> | <4> | <3> | <2> | <1> | <0> |
|---|---|---|---|---|---|---|---|---|
| ISPR | ISPR7 | ISPR6 | ISPR5 | ISPR4 | ISPR3 | ISPR2 | ISPR1 | ISPR0 |

| ISPRn | Priority of interrupt currently being acknowledged |
|---|---|
| 0 | Interrupt request with priority n is not acknowledged |
| 1 | Interrupt request with priority n is being acknowledged |

**Remark** n = 0 to 7 (priority level)

### 19.3.7 ID flag

The interrupt disable flag (ID) is allocated to the PSW and controls the maskable interrupt's operating state, and stores control information regarding enabling/disabling reception of interrupt request signals.

After reset, this flag is set to 00000020H.

After reset: 00000020H

| | 31 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| PSW | 0 | | NP | EP | ID | SAT | CY | OV | S | Z |

| ID | Maskable interrupt servicing specification[Note] |
|---|---|
| 0 | Maskable interrupt request signal acknowledgment enabled |
| 1 | Maskable interrupt request signal acknowledgment disabled |

**Note** Interrupt disable flag (ID) function

ID is set (1) by the DI instruction and cleared (0) by the EI instruction. Its value is also modified by the RETI instruction or LDSR instruction when referencing the PSW.

Non-maskable interrupt request signals and exceptions are acknowledged regardless of this flag. When a maskable interrupt request signal is acknowledged, the ID flag is automatically set (1) by hardware.

An interrupt request signal generated during the acknowledgment disabled period (ID flag = 1) can be acknowledged when the xxICn.xxIFn bit is set (1), and the ID flag is cleared (0).

### 19.3.8 Watchdog timer mode register 1 (WDTM1)

This register is a special register that can be written to only in a special sequence. To generate a maskable interrupt (INTWDT1), clear the WDTM14 bit to 0.

This register can be read or written in 8-bit or 1-bit units (for details, refer to **CHAPTER 12 WATCHDOG TIMER FUNCTIONS**).

After reset: 00H R/W Address: FFFFF6C2H

| | <7> | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| WDTM1 | RUN1 | 0 | 0 | WDTM14 | WDTM13 | 0 | 0 | 0 |

| RUN1 | Watchdog timer operation mode selection[Note 1] |
|---|---|
| 0 | Stop count operation |
| 1 | Clear counter and start count operation |

| WDTM14 | WDTM13 | Watchdog timer operation mode selection[Note 2] |
|---|---|---|
| 0 | 0 | Interval timer mode<br>(Generate maskable interrupt INTWDTM1 when overflow occurs) |
| 0 | 1 | |
| 1 | 0 | Watchdog timer mode 1[Note 3]<br>(Generate non-maskable interrupt INTWDT1 when overflow occurs) |
| 1 | 1 | Watchdog timer mode 2<br>(Start WDTRES2 reset operation when overflow occurs) |

**Notes 1.** Once the RUN1 bit has been set (1), it cannot be cleared (0) by software. Therefore, once counting starts, it cannot be stopped except by reset.

**2.** Once the WDTM14 and WDTM13 bits have been set (1), they cannot be cleared (0) by software. Reset is the only way to clear these bits.

**3.** For non-maskable interrupt servicing due to a non-maskable interrupt request signal (INTWDT1), refer to **19.10 Cautions**.

## 19.4 External Interrupt Request Input Pins (NMI, INTP0 to INTP7)

### 19.4.1 Noise elimination

**(1) Noise elimination for NMI pin**

The NMI pin includes a noise eliminator that operates using analog delay. Therefore, a signal input to the NMI pin is not detected as an edge unless it maintains its input level for a certain period. The edge is detected only after a certain period has elapsed.

The NMI pin is used for releasing the STOP mode. In the STOP mode, noise elimination using the system clock is not performed because the internal system clock is stopped.

**(2) Noise elimination for INTP0 to INTP2 and INTP4 to INTP7 pins**

The INTP0 to INTP2 and INTP4 to INTP7 pins include a noise eliminator that operates using analog delay. Therefore, a signal input to each pin is not detected as an edge unless it maintains its input level for a certain period. The edge is detected only after a certain period has elapsed.

**(3) Noise elimination for INTP3 pin**

The INTP3 pin has a digital/analog noise eliminator that can be selected by the NFC.NFEN bit.

The number of times the digital noise eliminator samples signals can be selected by the NFC.NFSTS bit from three or two. The sampling clock can be selected by the NFC.NFC2 to NFC.NFC0 bits from $f_{XX}/64$, $f_{XX}/128$, $f_{XX}/256$, $f_{XX}/512$, $f_{XX}/1024$, and $f_{XT}$. If the sampling clock is set to $f_{XX}/64$, $f_{XX}/128$, $f_{XX}/256$, $f_{XX}/512$, or $f_{XX}/1024$, the sampling clock stops in the IDLE/STOP mode. It cannot therefore be used to release the standby mode. To release the standby mode, select $f_{XT}$ as the sampling clock or select the analog noise eliminator.

**(a) Digital noise elimination control register (NFC)**

The NFC register controls elimination of noise on the INTP3 pin. If $f_{XT}$ is used as the noise elimination clock, the external interrupt function of the INTP3 pin can be used even in the IDLE/STOP mode.

This register can be read or written in 8-bit or 1-bit units.

After reset, NFC is cleared to 00H.

After reset: 00H R/W Address: FFFFF318H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| NFC | NFEN | NFSTS | 0 | 0 | 0 | NFC2 | NFC1 | NFC0 |

| NFEN | Setting of INTP3 pin noise elimination |
|---|---|
| 0 | Analog noise elimination |
| 1 | Digital noise elimination |

| NFSTS | Setting of number of samplings of digital noise elimination |
|---|---|
| 0 | Number of samplings = 3 times |
| 1 | Number of samplings = 2 times |

| NFC2 | NFC1 | NFC0 | Selection of sampling clock |
|---|---|---|---|
| 0 | 0 | 0 | $f_{XX}/64$ |
| 0 | 0 | 1 | $f_{XX}/128$ |
| 0 | 1 | 0 | $f_{XX}/256$ |
| 0 | 1 | 1 | $f_{XX}/512$ |
| 1 | 0 | 0 | $f_{XX}/1024$ |
| 1 | 0 | 1 | $f_{XT}$ |
| Other than above | | | Setting prohibited |

**Remark** $f_{XX}$: Main clock frequency

$f_{XT}$: Subclock frequency

**<Noise elimination width>**

The digital noise elimination width ($t_{WIT3}$) is as follows, where T is the sampling clock period and M is the number of samplings.

- $t_{WIT3} < (M - 1)T$: Accurately eliminated as noise
- $(M - 1)T \leq t_{WIT3} < MT$: May be eliminated as noise or detected as valid edge
- $t_{WIT3} \geq MT$: Accurately detected as valid edge

To detect the valid edge input to the INTP3 pin accurately, therefore, a pulse wider than MT must be input.

| NFSTS | NFC2 | NFC1 | NFC0 | Sampling Clock | Minimum Elimination Noise Width | | |
|---|---|---|---|---|---|---|---|
| | | | | | $f_{XX}$ = 20 MHz | $f_{XX}$ = 10 MHz | $f_{XX}$ = 8 MHz |
| 0 | 0 | 0 | 0 | $f_{XX}$/64 | 6.4 $\mu$s | 12.8 $\mu$s | 16 $\mu$s |
| 0 | 0 | 0 | 1 | $f_{XX}$/128 | 12.8 $\mu$s | 25.6 $\mu$s | 32 $\mu$s |
| 0 | 0 | 1 | 0 | $f_{XX}$/256 | 25.6 $\mu$s | 51.2 $\mu$s | 64 $\mu$s |
| 0 | 0 | 1 | 1 | $f_{XX}$/512 | 51.2 $\mu$s | 102.4 $\mu$s | 128 $\mu$s |
| 0 | 1 | 0 | 0 | $f_{XX}$/1024 | 102.4 $\mu$s | 204.8 $\mu$s | 256 $\mu$s |
| 0 | 1 | 0 | 1 | $f_{XT}$ (32.768 kHz) | 61.04 $\mu$s | | |
| 1 | 0 | 0 | 0 | $f_{XX}$/64 | 3.2 $\mu$s | 6.4 $\mu$s | 8 $\mu$s |
| 1 | 0 | 0 | 1 | $f_{XX}$/128 | 6.4 $\mu$s | 12.8 $\mu$s | 16 $\mu$s |
| 1 | 0 | 1 | 0 | $f_{XX}$/256 | 12.8 $\mu$s | 25.6 $\mu$s | 32 $\mu$s |
| 1 | 0 | 1 | 1 | $f_{XX}$/512 | 25.6 $\mu$s | 51.2 $\mu$s | 64 $\mu$s |
| 1 | 1 | 0 | 0 | $f_{XX}$/1024 | 51.2 $\mu$s | 102.4 $\mu$s | 128 $\mu$s |
| 1 | 1 | 0 | 1 | $f_{XT}$ (32.768 kHz) | 30.52 $\mu$s | | |
| Other than above | | | | Setting prohibited | | | |

### 19.4.2 Edge detection

The valid edges of the NMI and INTP0 to INTP7 pins can be selected from the following four types for each pin.

- Rising edge
- Falling edge
- Both edges
- No edge detection

After reset, the edge detection for the NMI pin is set to “no edge detection”. Therefore, interrupt requests cannot be acknowledged (the NMI pin functions as a normal port) unless a valid edge is specified by the INTR0 and INTF0 registers.

When using the P02 pin as an output port, set the NMI pin valid edge to “no edge detection”.

**(1) External interrupt rising and falling edge specification registers 0 (INTR0, INTF0)**

These are 8-bit registers that specify detection of the rising and falling edges of the NMI and INTP0 to INTP3 pins.

These registers can be read or written in 8-bit or 1-bit units.

After reset, these registers are cleared to 00H.

**Caution When switching to the port function from the external interrupt function (alternate function), edge detection may be performed. Therefore, set the port mode after setting the INTF0n and INTR0n bits = 00.**

After reset: 00H R/W Address: INTR0 FFFFFC20H, INTF0 FFFFFC00H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| INTR0 | 0 | INTR06 | INTR05 | INTR04 | INTR03 | INTR02 | 0 | 0 |
| | | INTP3 | INTP2 | INTP1 | INTP0 | NMI | | |
| INTF0 | 0 | INTF06 | INTF05 | INTF04 | INTF03 | INTF02 | 0 | 0 |
| | | INTP3 | INTP2 | INTP1 | INTP0 | NMI | | |

**Remark** For specification of the valid edge, refer to **Table 19-3**.

**Table 19-3. NMI and INTP0 to INTP3 Pins Valid Edge Specification**

| INTF0n | INTR0n | Valid edge specification (n = 2 to 6) |
|---|---|---|
| 0 | 0 | No edge detection |
| 0 | 1 | Rising edge |
| 1 | 0 | Falling edge |
| 1 | 1 | Both edges |

**Remark** n = 2: Control of NMI pin

n = 3 to 6: Control of INTP0 to INTP3 pins

**(2) External interrupt rising and falling edge specification registers 3 (INTR3, INTF3)**

These are 8-bit registers that specify detection of the rising and falling edges of the INTP7 pin.

These registers can be read or written in 8-bit or 1-bit units.

After reset, these registers are cleared to 00H.

**Caution When switching to the port function from the external interrupt function (alternate function), edge detection may be performed. Therefore, set the port mode after setting the INTF31 and INTR31 bits = 00.**

After reset: 00H R/W Address: INTR3 FFFFFC26H, INTF3 FFFFFC06H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| INTR3 | 0 | 0 | 0 | 0 | 0 | 0 | INTR31 | 0 |
| | | | | | | | INTP7 | |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTF3 | 0 | 0 | 0 | 0 | 0 | 0 | INTF31 | 0 |
| | | | | | | | INTP7 | |

**Remark** For specification of the valid edge, refer to **Table 19-4**.

**Table 19-4. INTP7 Pin Valid Edge Specification**

| INTF31 | INTR31 | Valid edge specification |
|---|---|---|
| 0 | 0 | No edge detection |
| 0 | 1 | Rising edge |
| 1 | 0 | Falling edge |
| 1 | 1 | Both edges |

**(3) External interrupt rising and falling edge specification registers 9H (INTR9H, INTF9H)**

These are 8-bit registers that specify detection of the rising edge of the INTP4 to INTP6 pins.

These registers can be read or written in 8-bit or 1-bit units.

After reset, these registers are cleared to 00H.

**Caution When switching to the port function from the external interrupt function (alternate function), edge detection may be performed. Therefore, set the port mode after setting the INTF9n and INTR9n bits = 00.**

After reset: 00H R/W Address: INTR9H FFFFFC33H, INTF9H FFFFFC13H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| INTR9H | INTR915 | INTR914 | INTR913 | 0 | 0 | 0 | 0 | 0 |
| | INTP6 | INTP5 | INTP4 | | | | | |
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTF9H | INTF915 | INTF914 | INTF913 | 0 | 0 | 0 | 0 | 0 |
| | INTP6 | INTP5 | INTP4 | | | | | |

**Remark** For specification of the valid edge, refer to **Table 19-5**.

**Table 19-5. INTP4 to INTP6 Pins Valid Edge Specification**

| INTF9n | INTR9n | Valid edge specification (n = 13 to 15) |
|---|---|---|
| 0 | 0 | No edge detection |
| 0 | 1 | Rising edge |
| 1 | 0 | Falling edge |
| 1 | 1 | Both edges |

**Remark** n = 13 to 15: Control of INTP4 to INTP6 pins

## 19.5 Software Exceptions

A software exception is generated when the CPU executes the TRAP instruction. Software exceptions can always be acknowledged.

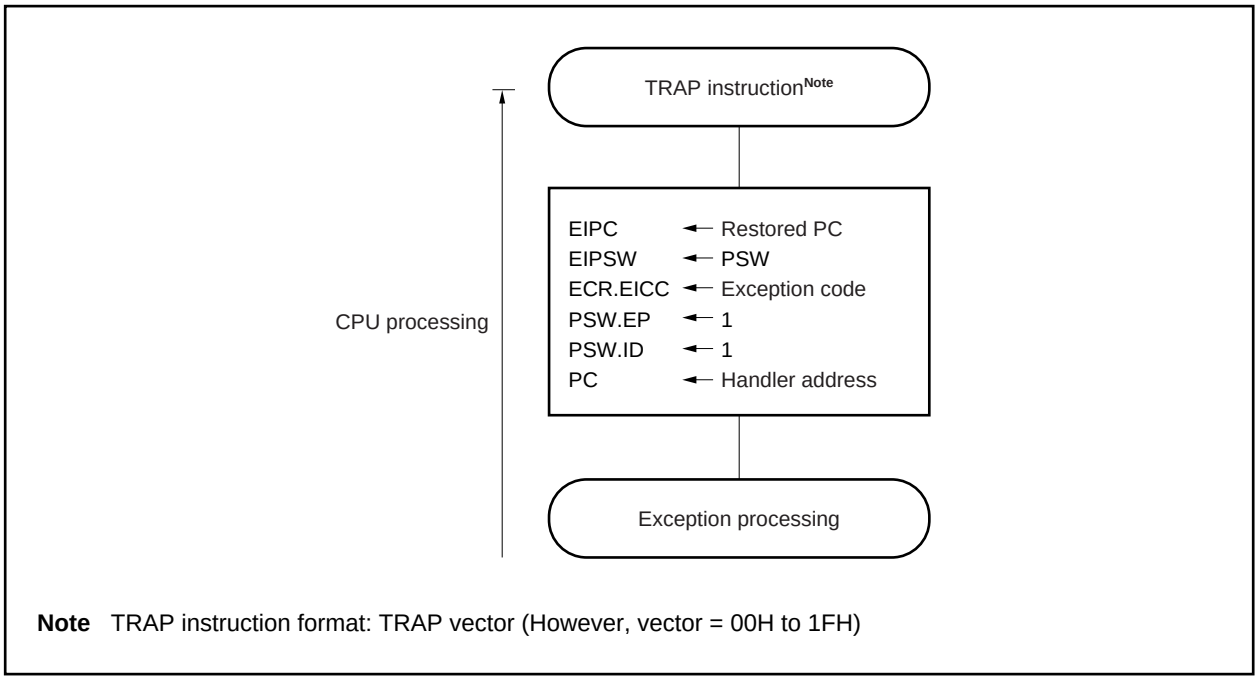
### 19.5.1 Operation

If a software exception occurs, the CPU performs the following processing and transfers control to a handler routine.

<1> Saves the restored PC to EIPC.
<2> Saves the current PSW to EIPSW.
<3> Writes an exception code to the lower 16 bits (EICC) of ECR (interrupt source).
<4> Sets the PSW.EP and PSW.ID bits to 1.
<5> Loads the handler address (00000040H or 00000050H) for the software exception routine to the PC and transfers control.

Figure 19-8 shows the software exception processing flow.

**Figure 19-8. Software Exception Processing**




**Note** TRAP instruction format: TRAP vector (However, vector = 00H to 1FH)

The handler address is determined by the operand (vector) of the TRAP instruction. If the vector is 00H to 1FH, the handler address is 00000040H, and if the vector is 10H to 1FH, the handler address is 00000050H.

### 19.5.2 Restore

Execution is restored from software exception processing by the RETI instruction.

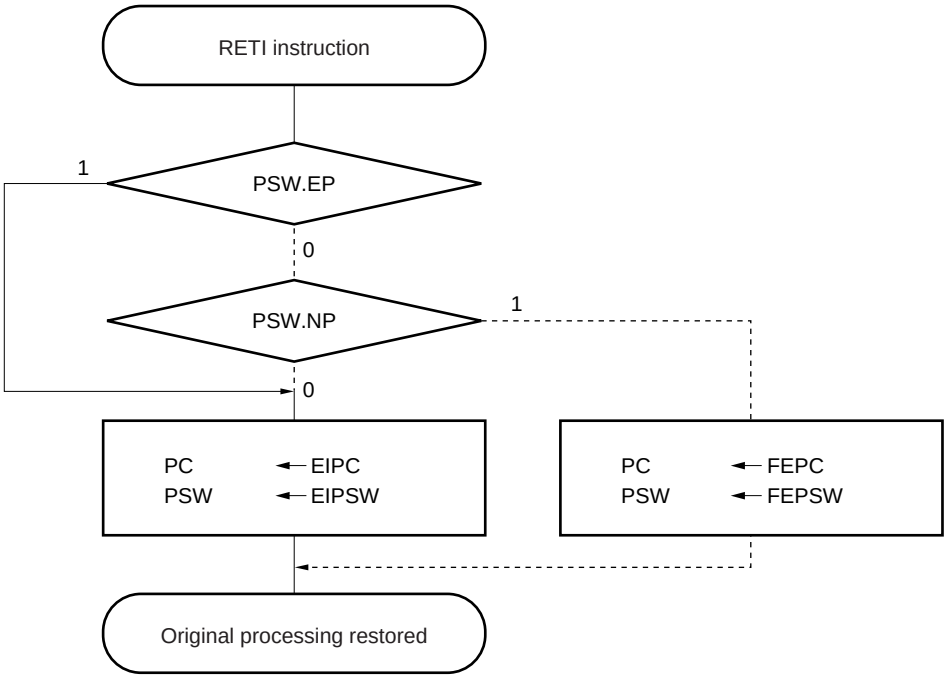
When the RETI instruction is executed, the CPU performs the following processing and transfers control to the address of the restored PC.

<1> Loads the restored PC and PSW from EIPC and EIPSW because the PSW.EP bit is 1.

<2> Transfers control to the address of the restored PC and PSW.

Figure 19-9 shows the processing flow of the RETI instruction.

**Figure 19-9. RETI Instruction Processing**




**Caution When the EP bit and the NP bit are changed by the LDSR instruction during software exception processing, in order to restore the PC and PSW correctly during restoring by the RETI instruction, it is necessary to set the EP bit back to 1 using the LDSR instruction immediately before the RETI instruction.**

**Remark** The solid line shows the CPU processing flow.

### 19.5.3 EP flag

The EP flag is a status flag that indicates that exception processing is in progress. It is set when an exception occurs.

After reset: 00000020H

| | 31 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| PSW | 0 | | NP | EP | ID | SAT | CY | OV | S | Z |

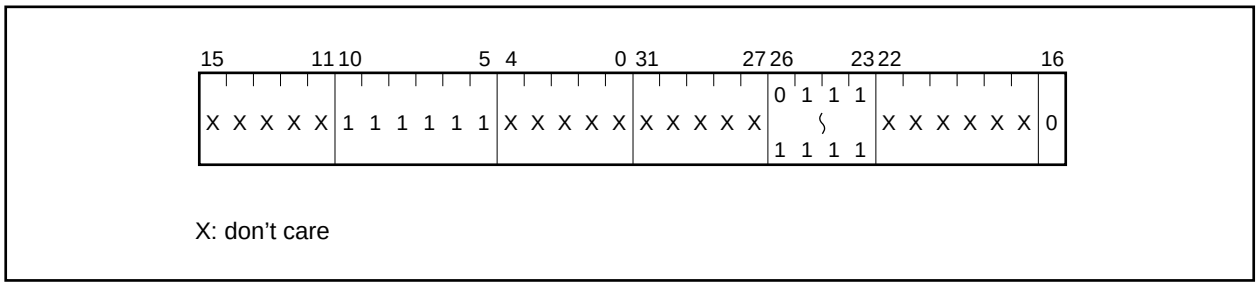
| EP | Exception processing status |
|---|---|
| 0 | Exception processing not in progress |
| 1 | Exception processing in progress |

## 19.6 Exception Trap

The exception trap is an interrupt that is requested when the illegal execution of an instruction takes place. In the V850ES/KF1+, an illegal opcode trap (ILGOP) is considered as an exception trap.

### 19.6.1 Illegal opcode

An illegal opcode is defined as an instruction with instruction opcode (bits 10 to 5) = 111111B, sub-opcode (bits 26 to 23) = 0111B to 1111B, and sub-opcode (bit 16) = 0B. When such an instruction is executed, an exception trap is generated.




X: don't care

**Caution It is recommended not to use illegal opcode because instructions may newly be assigned in the future.**

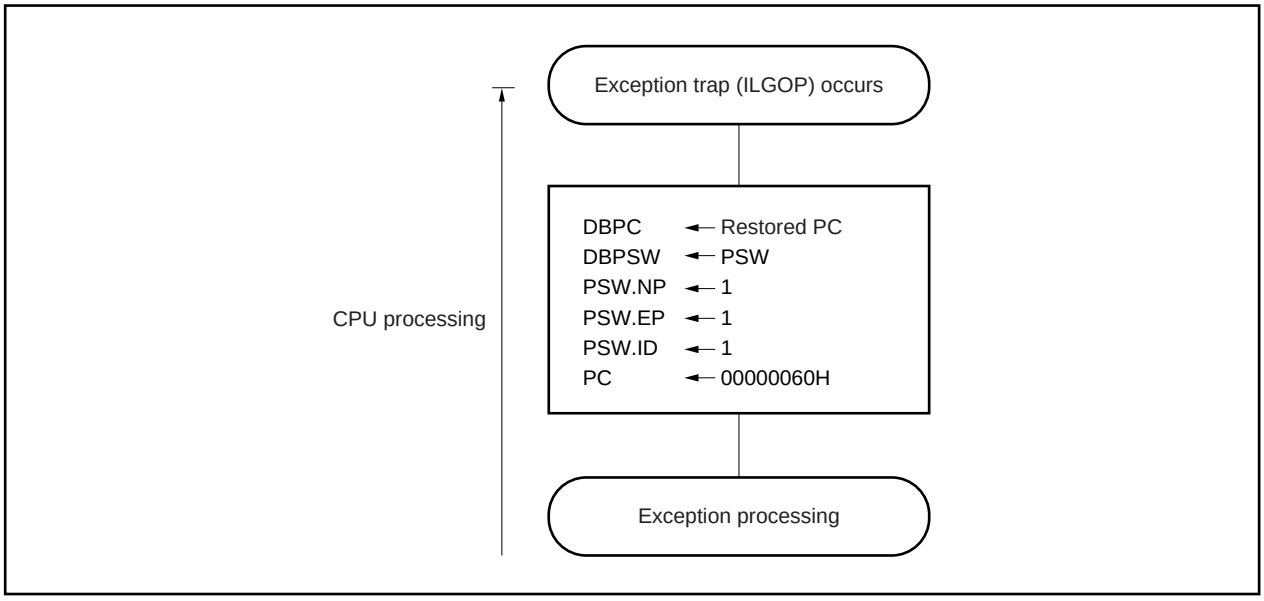
**(1) Operation**

Upon generation of an exception trap, the CPU performs the following processing and transfers control to a handler routine.

<1> Saves the restored PC to DBPC.
<2> Saves the current PSW to DBPSW.
<3> Sets the PSW.NP, PSW.EP, and PSW.ID bits.
<4> Loads the handler address (00000060H) for the exception trap routine to the PC and transfers control.

Figure 19-10 shows the exception trap processing flow.

**Figure 19-10. Exception Trap Processing**



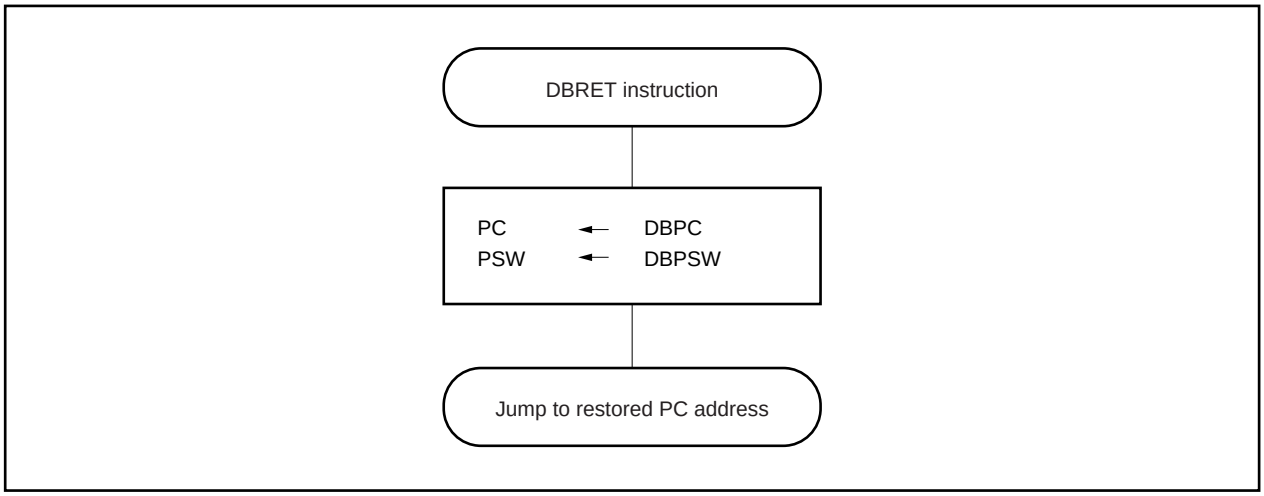

**(2) Restore**

Execution is restored from exception trap processing by the DBRET instruction. When the DBRET instruction is executed, the CPU performs the following processing and transfers control to the address of the restored PC.

<1> Loads the restored PC and PSW from DBPC and DBPSW.
<2> Transfers control to the loaded address of the restored PC and PSW.

Figure 19-11 shows the processing flow for restore from exception trap processing.

**Figure 19-11. Processing Flow for Restore from Exception Trap**



### 19.6.2 Debug trap

A debug trap is an exception that occurs upon execution of the DBTRAP instruction and that can be acknowledged at all times.

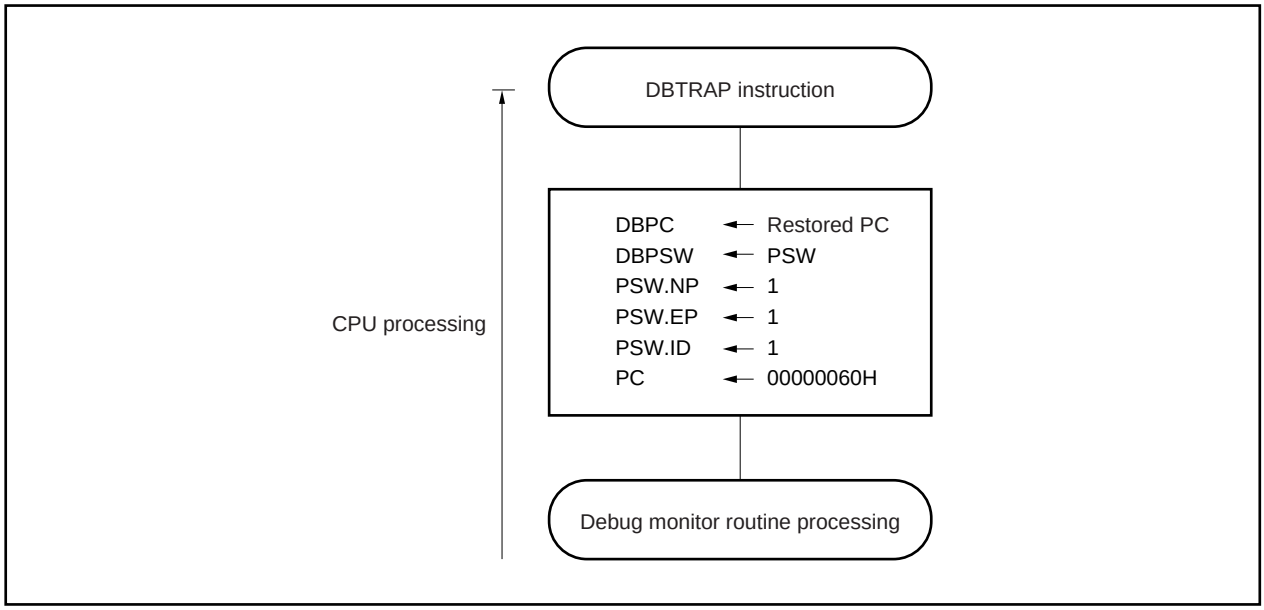
When a debug trap occurs, the CPU performs the following processing.

#### (1) Operation

<1> Saves the restored PC to DBPC.
<2> Saves the current PSW to DBPSW.
<3> Sets the PSW.NP, PSW.EP, and PSW.ID bits to 1.
<4> Sets the handler address (00000060H) for the debug trap routine to the PC and transfers control.

Figure 19-12 shows the debug trap processing flow.

**Figure 19-12. Debug Trap Processing**



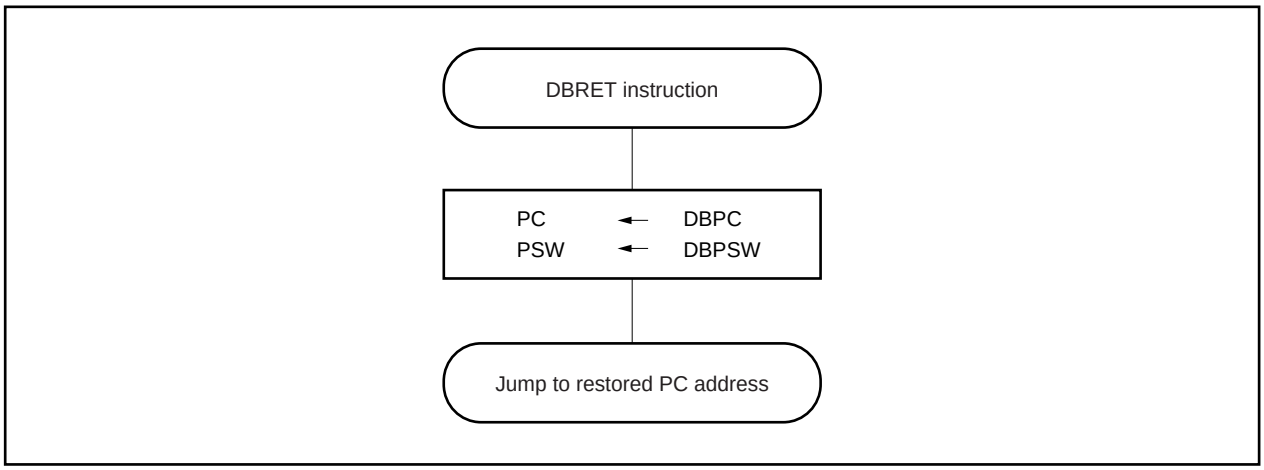
**(2) Restore**

Execution is restored from debug trap processing by the DBRET instruction. When the DBRET instruction is executed, the CPU performs the following processing and transfers control to the address of the restored PC.

<1> Loads the restored PC and PSW from DBPC and DBPSW.
<2> Transfers control to the loaded address of the restored PC and PSW.

Figure 19-13 shows the processing flow for restore from debug trap processing.

**Figure 19-13. Processing Flow for Restore from Debug Trap**



## 19.7 Multiple Interrupt Servicing Control

Multiple interrupt servicing control is a function that stops an interrupt service routine currently in progress if a higher priority interrupt request signal is generated, and processes the acknowledgment operation of the higher priority interrupt request signal.

If an interrupt request signal with a lower or equal priority is generated and a service routine is currently in progress, the later interrupt request signal will be held pending.

Multiple interrupt servicing control is performed when interrupts are enabled (PSW.ID bit = 0). Even in an interrupt servicing routine, multiple interrupt control must be performed while interrupts are enabled (ID bit = 0).

If a maskable interrupt or software exception is generated in a maskable interrupt or software exception service program, EIPC and EIPSW must be saved.

The following example illustrates the procedure.

### (1) To acknowledge maskable interrupt request signals in service program

Service program for maskable interrupt or exception

…
…
- EIPC saved to memory or register
- EIPSW saved to memory or register
- EI instruction (enables interrupt acknowledgment)

…
…
…
…

←Acknowledges maskable interrupt

- DI instruction (disables interrupt acknowledgment)
- Saved value restored to EIPSW
- Saved value restored to EIPC
- RETI instruction

**(2) To generate exception in service program**

Service program for maskable interrupt or exception

…

…

- EIPC saved to memory or register
- EIPSW saved to memory or register

…

- TRAP instruction ←Acknowledges exceptions such as TRAP instruction.

…

- Saved value restored to EIPSW
- Saved value restored to EIPC
- RETI instruction

Priorities 0 to 7 (0 is the highest) can be set for each maskable interrupt request in multiple interrupt servicing control by software. To set a priority level, write values to the xxICn.xxPRn0 to xxICn.xxPRn2 bits corresponding to each maskable interrupt request. After reset, interrupt requests are masked by the xxICn.xxMKn bit, and the priority is set to level 7 by the xxPRn0 to xxPRn2 bits.

Priorities of maskable interrupts are as follows.

(High) Level 0 > Level 1 > Level 2 > Level 3 > Level 4 > Level 5 > Level 6 > Level 7 (Low)

Interrupt servicing that has been suspended as a result of multiple interrupt servicing control is resumed after the interrupt servicing of the higher priority has been completed and the RETI instruction has been executed.
A pending interrupt request signal is acknowledged after the current interrupt servicing has been completed and the RETI instruction has been executed.

**Caution In a non-maskable interrupt servicing routine (in the time until the RETI instruction is executed), maskable interrupts are not acknowledged and held pending.**

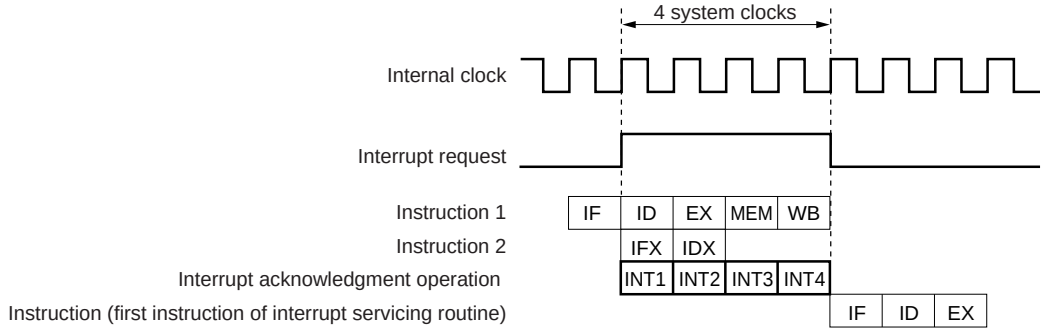
## 19.8 Interrupt Response Time

Except in the following cases, the CPU interrupt response time is a minimum of 4 clocks. If inputting consecutive interrupt request signals, at least 4 clocks must be placed between each interrupt request signal.

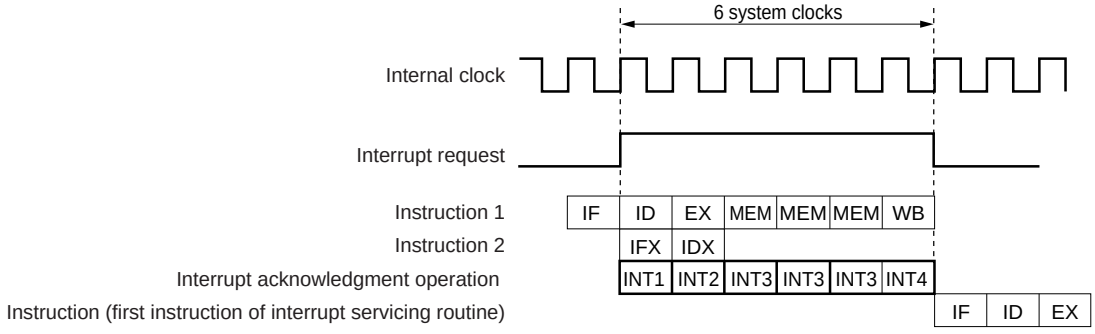
- IDLE/STOP mode
- External bus access
- Consecutive interrupt request non-sample instruction (refer to **19.9 Periods in Which Interrupts Are Not Acknowledged by CPU**)
- Access to interrupt control register
- Access to peripheral I/O register

**Figure 19-14. Pipeline Operation During Interrupt Request Signal Acknowledgment (Outline)**

**(1) Minimum interrupt response time**




**(2) Maximum interrupt response time**




**Remark** INT1 to INT4: Interrupt acknowledgment processing
IFX: Invalid instruction fetch
IDX: Invalid instruction decode

| Interrupt response time (internal system clock) | | | Condition |
|---|---|---|---|
| | Internal interrupt | External interrupt | |
| Min. | 4 | 4 + analog delay | The following cases are excluded.<br>• IDLE/STOP mode<br>• External bus access<br>• Consecutive interrupt request non-sample instruction<br>• Access to interrupt control register<br>• Access to peripheral I/O register |
| Max. | 6 | 6 + analog delay | |

## 19.9 Periods in Which Interrupts Are Not Acknowledged by CPU

Interrupts are acknowledged by the CPU while an instruction is being executed. However, no interrupt is acknowledged between an interrupt request non-sample instruction and the next instruction (interrupts are held pending).

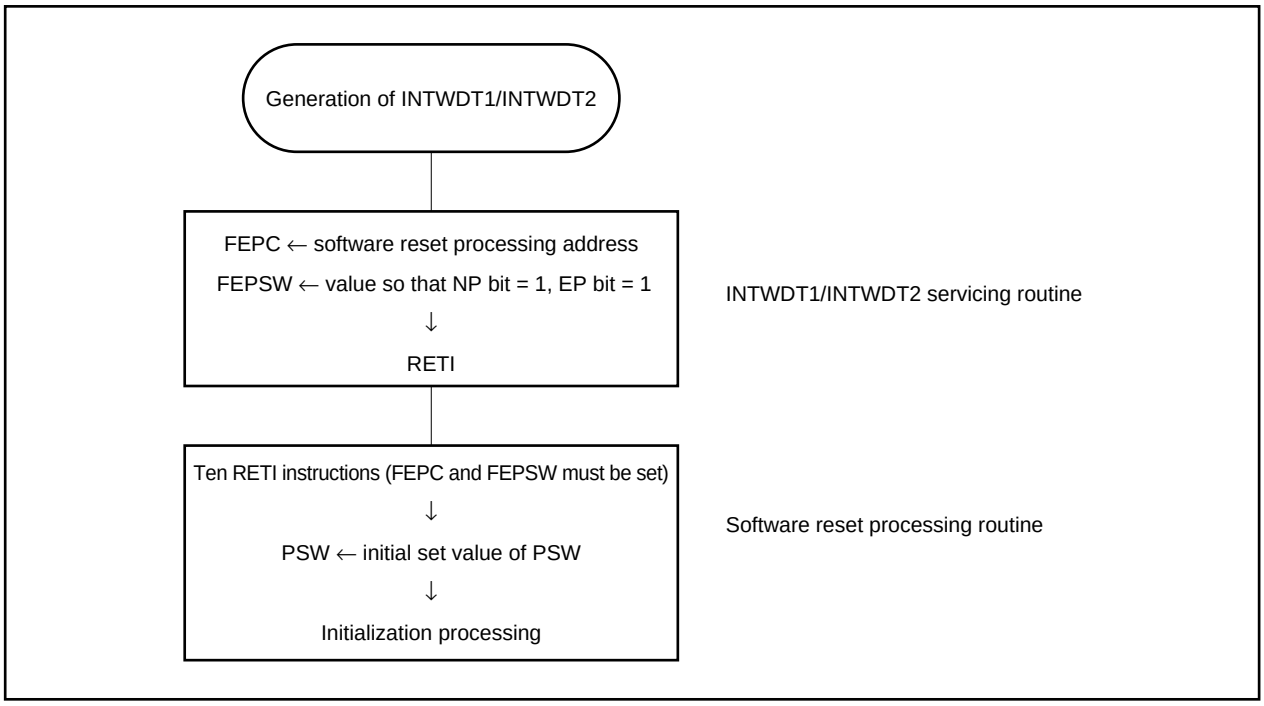
The following instructions are interrupt request non-sample instructions.

- EI instruction
- DI instruction
- LDSR reg2, 0x5 instructions (vs. PSW)
- Store instruction for the PRCMD register
- Store instruction and SET1, NOT1, and CLR1 instructions for the following registers
  - Interrupt-related registers:
    Interrupt control register (xxlCn), interrupt mask registers 0, 1, 3 (IMR0, IMR1, IMR3)
  - Power save control register (PSC)

## 19.10 Cautions

Design the system so that restoring by the RETI instruction is as follows after a non-maskable interrupt triggered by a non-maskable interrupt request signal (INTWDT1/INTWDT2) is serviced.

**Figure 19-15. Restoring by RETI Instruction**



# CHAPTER 20 KEY INTERRUPT FUNCTION

## 20.1 Function

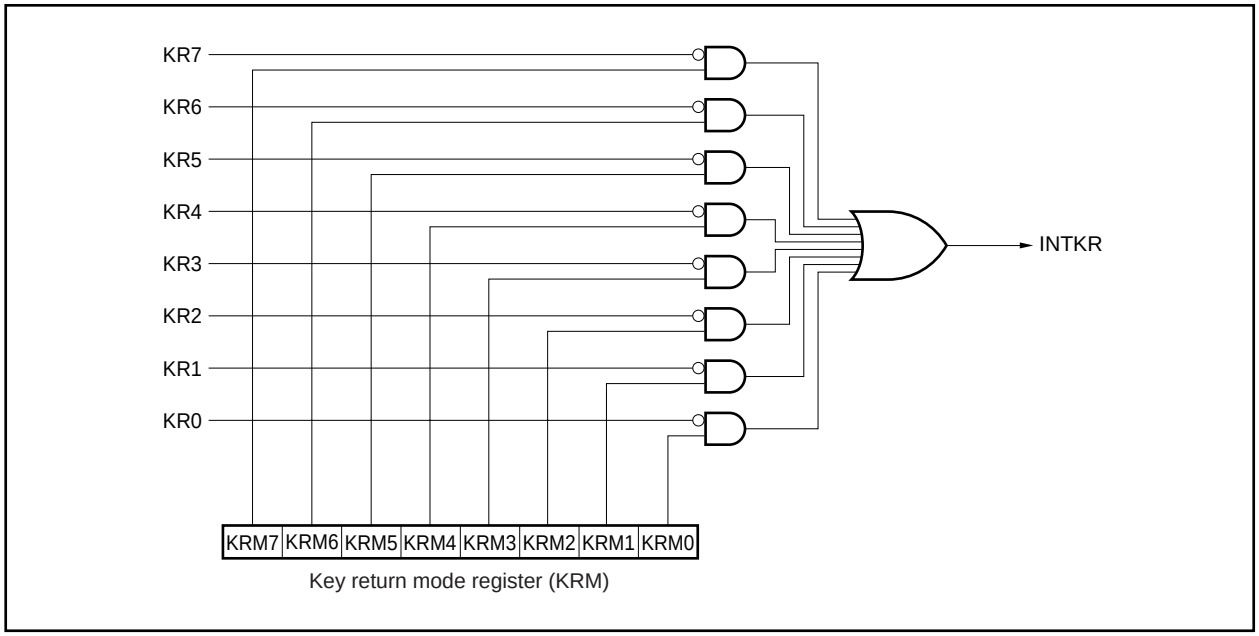
A key interrupt request signal (INTKR) can be generated by inputting a falling edge to the eight key input pins (KR0 to KR7) by setting the KRM register.

**Caution If any of the KR0 to KR7 pins is at low level, the INTKR signal is not generated even if a falling edge is input to another pin.**

**Table 20-1. Assignment of Key Return Detection Pins**

| Flag | Pin Description |
|---|---|
| KRM0 | Controls KR0 signal in 1-bit units |
| KRM1 | Controls KR1 signal in 1-bit units |
| KRM2 | Controls KR2 signal in 1-bit units |
| KRM3 | Controls KR3 signal in 1-bit units |
| KRM4 | Controls KR4 signal in 1-bit units |
| KRM5 | Controls KR5 signal in 1-bit units |
| KRM6 | Controls KR6 signal in 1-bit units |
| KRM7 | Controls KR7 signal in 1-bit units |

**Figure 20-1. Key Return Block Diagram**



## 20.2 Register

**(1) Key return mode register (KRM)**

The KRM register controls the KRM0 to KRM7 bits using the KR0 to KR7 signals.

This register can be read or written in 8-bit or 1-bit units.

After reset, KRM is cleared to 00H.

After reset: 00H R/W Address: FFFFF300H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| KRM | KRM7 | KRM6 | KRM5 | KRM4 | KRM3 | KRM2 | KRM1 | KRM0 |

| KRMn | Key return mode control |
|---|---|
| 0 | Does not detect key return signal |
| 1 | Detects key return signal |

**Caution If the KRM register is changed, an interrupt request signal (INTKR) may be generated. To prevent this, change the KRM register after disabling interrupts (DI), and then enable interrupts (EI) after clearing the interrupt request flag (KRIC.KRIF bit) to 0.**

**Remark** For the alternate-function pin settings, refer to **Table 4-14 Settings When Port Pins Are Used for Alternate Functions**.

# CHAPTER 21 STANDBY FUNCTION

## 21.1 Overview

The power consumption of the system can be effectively reduced by using the standby modes in combination and selecting the appropriate mode for the application.

The available standby modes are listed in Table 21-1.

**Table 21-1. Standby Modes**

| Mode | Functional Outline |
|---|---|
| HALT mode | Mode to stop only the operating clock of the CPU |
| IDLE mode | Mode to stop all the operations of the internal circuits except the oscillator[Note 1] |
| STOP mode | Mode to stop all the operations of the internal circuits except the subclock oscillator[Note 2] |
| Subclock operation mode | Mode to use the subclock as the internal system clock |
| Sub-IDLE mode | Mode to stop all the operations of the internal circuits, except the oscillator, in the subclock operation mode |
| Ring clock operation mode[Note 3] | Mode in which the internal system clock ($f_{CLK}$) operates on the ring clock by using the clock monitor function |
| Ring HALT mode[Note 3] | Mode in which only the operating clock of the CPU ($f_{CPU}$) is stopped in the ring clock operation mode |

**Notes 1.** The PLL does not stop. To realize low power consumption, stop the PLL and then shift to the IDLE mode.

**2.** Change to the clock-through mode, stop the PLL, then shift to the STOP mode. For details, refer to **CHAPTER 6 CLOCK GENERATION FUNCTION**.

**3.** For details of the ring clock operation mode and ring HALT mode, refer to **CHAPTER 23 CLOCK MONITOR**.

**Figure 21-1. Status Transition (1/2)**

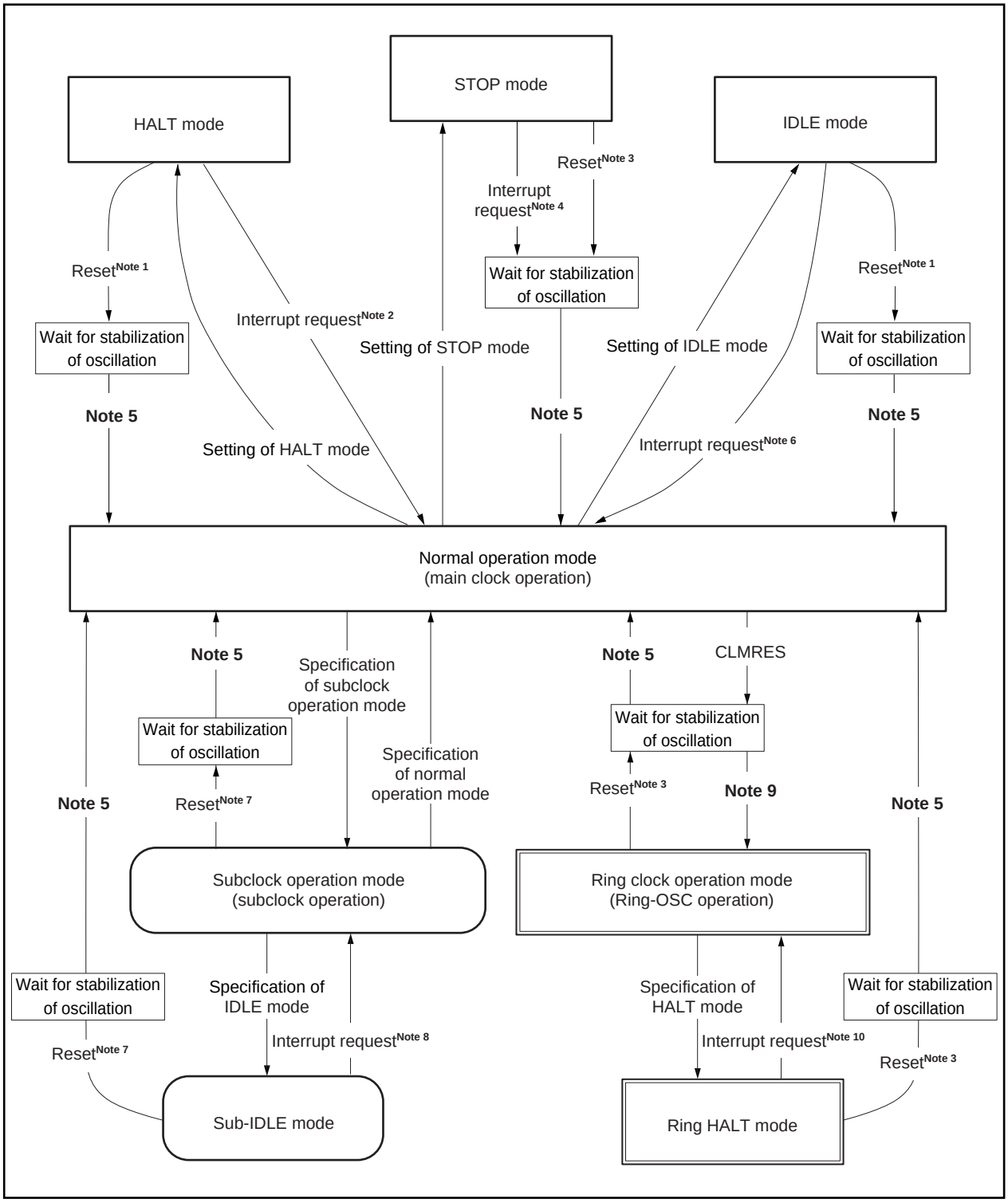


**Figure 21-1. Status Transition (2/2)**

**Notes 1.** $\overline{\text{RESET}}$ pin input, WDTRES2, POCRES, LVIRES, or CLMRES signal.
In the case of the WDTRES1 signal, the oscillation stabilization time is not secured.

**2.** Non-maskable interrupt request signal or unmasked maskable interrupt request signal.

**3.** $\overline{\text{RESET}}$ pin input, WDTRES2, POCRES, or LVIRES signal.

**4.** Non-maskable interrupt request signal (NMI pin input, INTWDT2 signal) or unmasked internal interrupt request signal from peripheral functions operable in STOP mode.

**5.** The main clock ($f_X$) starts oscillating. After the oscillation stabilization time, the normal operation mode is set.
If watchdog timer 2 overflows while the oscillation stabilization time is being secured because of an abnormality (stoppage) of the main clock oscillation ($f_X$), the ring clock operation mode is set.

**6.** Non-maskable interrupt request signal (NMI pin input, INTWDT2 signal) or unmasked internal interrupt request signal from peripheral functions operable in IDLE mode.

**7.** $\overline{\text{RESET}}$ pin input, WDTRES2, POCRES, or LVIRES signal.
While the main clock ($f_X$) is oscillating, the standby mode can be released by the CLMRES signal (refer to **Note 9**).

**8.** Non-maskable interrupt request signal (NMI pin input, INTWDT2 signal) or unmasked internal interrupt request signal from peripheral functions operable in sub-IDLE mode.

**9.** If the main clock oscillation ($f_X$) is abnormal (stops), watchdog timer 1 does not count the oscillation stabilization time. When watchdog timer 2 counts the ring clock and overflows, the ring clock operation mode is set.

**10.** Non-maskable interrupt request signal (NMI pin input, INTWDT2 signal) or unmasked internal interrupt request signal from peripheral functions operable in ring HALT mode.

**Remarks 1.** WDTRES1 signal: Reset signal by watchdog timer 1 overflow

**2.** WDTRES2 signal: Reset signal by watchdog timer 2 overflow

**3.** POCRES signal: Reset signal by power-on-clear circuit

**4.** LVIRES signal: Reset signal by low-voltage detector

**5.** CLMRES signal: Reset signal by clock monitor

## 21.2 Registers

**(1) Power save control register (PSC)**

This is an 8-bit register that controls the standby function. The STP bit of this register is used to specify the standby mode. The PSC register is a special register that can be written to only in a special sequence (refer to **3.4.7 Special registers**).

This register can be read or written in 8-bit or 1-bit units.

After reset, PSC is cleared to 00H.

After reset: 00H R/W Address: FFFFF1FEH

| | <7> | 6 | <5> | <4> | 3 | 2 | <1> | 0 |
|---|---|---|---|---|---|---|---|---|
| PSC | NMI2M | 0 | NMI0M | INTM | 0 | 0 | STP | 0 |

| NMI2M | Control of releasing standby mode[Note] by INTWDT2 signal |
|---|---|
| 0 | Releasing standby mode[Note] by INTWDT2 signal enabled |
| 1 | Releasing standby mode[Note] by INTWDT2 signal disabled |

| NMI0M | Control of releasing standby mode[Note] by NMI pin input |
|---|---|
| 0 | Releasing standby mode[Note] by NMI pin input enabled |
| 1 | Releasing standby mode[Note] by NMI pin input disabled |

| INTM | Control of releasing standby mode[Note] by maskable interrupt request signals |
|---|---|
| 0 | Releasing standby mode[Note] by maskable interrupt request signals enabled |
| 1 | Releasing standby mode[Note] by maskable interrupt request signals disabled |

| STP | Standby mode[Note] setting |
|---|---|
| 0 | Normal mode |
| 1 | Standby mode[Note] |

**Note** In this case, standby mode means the IDLE/STOP mode; it does not include the HALT mode.

**Cautions 1. If the NMI2M, NMI0M, and INTM bits, and the STP bit are set to 1 at the same time, the setting of NMI2M, NMI0M, and INTM bits becomes invalid. If there is an unmasked interrupt request signal being held pending when the IDLE/STOP mode is set, set the bit corresponding to the interrupt request signal (NMI2M, NMI0M, or INTM) to 1, and then set the STP bit to 1.**

**2. When the IDLE/STOP mode is set, set the PSMR.PSM bit and then set the STP bit to 1.**

**(2) Power save mode register (PSMR)**

This is an 8-bit register that controls the operation status in the power save mode and the clock operation.

This register can be read or written in 8-bit or 1-bit units.

After reset, PSMR is cleared to 00H.

After reset: 00H R/W Address: FFFFF820H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | <0> |
|---|---|---|---|---|---|---|---|---|
| PSMR | XTSTP | 0 | 0 | 0 | 0 | 0 | 0 | PSM |

| XTSTP | Specification of subclock oscillator use |
|---|---|
| 0 | Subclock oscillator used |
| 1 | Subclock oscillator not used |

| PSM | Specification of operation in standby mode |
|---|---|
| 0 | IDLE mode |
| 1 | STOP mode |

**Cautions 1. Be sure to clear the XTSTP bit to 0 during subclock resonator connection.**

**2. Be sure to clear bits 1 to 6 of the PSMR register to 0.**

**3. The PSM bit is valid only when the PSC.STP bit is 1.**

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

The wait time until the oscillation stabilizes after the STOP mode is released is controlled by the OSTS register.

The OSTS register can be read or written in 8-bit units.

After reset, OSTS is set to 01H.

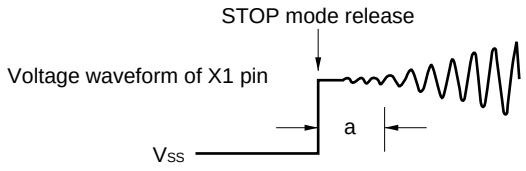
After reset: **Note** R/W Address: FFFFF6C0H

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

| OSTS2 | OSTS1 | OSTS0 | Selection of oscillation stabilization time | | | |
|---|---|---|---|---|---|---|
| | | | | $f_X$ | | |
| | | | | 4 MHz | 5 MHz | 10 MHz |
| 0 | 0 | 0 | $2^{13}/f_X$ | 2.048 ms | 1.638 ms | 0.819 ms |
| 0 | 0 | 1 | $2^{15}/f_X$ | 8.192 ms | 6.554 ms | 3.277 ms |
| 0 | 1 | 0 | $2^{16}/f_X$ | 16.38 ms | 13.11 ms | 6.554 ms |
| 0 | 1 | 1 | $2^{17}/f_X$ | 32.77 ms | 26.21 ms | 13.11 ms |
| 1 | 0 | 0 | $2^{18}/f_X$ | 65.54 ms | 52.43 ms | 26.21 ms |
| 1 | 0 | 1 | $2^{19}/f_X$ | 131.1 ms | 104.9 ms | 52.43 ms |
| 1 | 1 | 0 | $2^{20}/f_X$ | 262.1 ms | 209.7 ms | 104.9 ms |
| 1 | 1 | 1 | $2^{21}/f_X$ | 524.3 ms | 419.4 ms | 209.7 ms |

**Note** This register is set to 00H or 01H, depending on the setting of the mask option/option byte. For details, refer to **CHAPTER 28 MASK OPTION/OPTION BYTE**.

**Cautions 1. The wait time following release of the STOP mode does not include the time until the clock oscillation starts (“a” in the figure below) following release of the STOP mode, regardless of whether the STOP mode is released by reset or the occurrence of an interrupt request signal.**




**2. Be sure to clear bits 3 to 7 to 0.**

**3. The oscillation stabilization time is also inserted during external clock input.**

**Remark** $f_X$: Main clock oscillation frequency

## 21.3 HALT Mode

### 21.3.1 Setting and operation status

The HALT mode is set when a dedicated instruction (HALT) is executed in the normal operation mode.

In the HALT mode, the clock oscillator continues operating. Only clock supply to the CPU is stopped; clock supply to the other on-chip peripheral functions continues.

As a result, program execution is stopped, and the internal RAM retains the contents before the HALT mode was set. The on-chip peripheral functions that are independent of instruction processing by the CPU continue operating.

Table 21-3 shows the operation status in the HALT mode.

The average power consumption of the system can be reduced by using the HALT mode in combination with the normal operation mode for intermittent operation.

**Cautions 1. Insert five or more NOP instructions after the HALT instruction.**

**2. If the HALT instruction is executed with an unmasked interrupt request signal held pending, the system shifts to the HALT mode, but the HALT mode is immediately released by the pending interrupt request signal.**

### 21.3.2 Releasing HALT mode

The HALT mode is released by a non-maskable interrupt request signal (NMI pin input, INTWDT1, INTWDT2 signal), an unmasked maskable interrupt request signal, and reset signal ($\overline{\text{RESET}}$ pin input, WDTRES1, WDTRES2, POCRES, LVIRES, CLMRES signal).

After the HALT mode has been released, the normal operation mode is restored.

**(1) Releasing HALT mode by non-maskable interrupt request signal or unmasked maskable interrupt request signal**

The HALT mode is released by a non-maskable interrupt request signal or an unmasked maskable interrupt request signal, regardless of the priority of the interrupt request. If the HALT mode is set in an interrupt servicing routine, however, an interrupt request that is issued later is serviced as follows.

(a) If an interrupt request signal with a priority lower than that of the interrupt request currently being serviced is issued, only the HALT mode is released, and that interrupt request signal is not acknowledged. The interrupt request signal itself is retained.

(b) If an interrupt request signal with a priority higher than that of the interrupt request currently being serviced is issued (including a non-maskable interrupt request signal), the HALT mode is released and that interrupt request signal is acknowledged.

**Table 21-2. Operation After Releasing HALT Mode by Interrupt Request Signal**

| Release Source | Interrupt Enabled (EI) Status | Interrupt Disabled (DI) Status |
|---|---|---|
| Non-maskable interrupt request signal | Execution branches to the handler address | |
| Maskable interrupt request signal | Execution branches to the handler address or the next instruction is executed | The next instruction is executed |

**(2) Releasing HALT mode by reset**

The same operation as the normal reset operation is performed.

**Table 21-3. Operation Status in HALT Mode**

| Setting of HALT Mode / Item | | When CPU Is Operating with Main Clock | |
|---|---|---|---|
| | | When Subclock Is Not Used | When Subclock Is Used |
| CPU | | Stops operation | |
| ROM correction | | Stops operation | |
| Main clock oscillator | | Oscillation enabled | |
| Subclock oscillator | | – | Oscillation enabled |
| Ring-OSC ($f_R$) | | Operable | |
| Interrupt controller | | Operable | |
| 16-bit timer (TMP0) | | Operable | |
| 16-bit timers (TM00, TM01) | | Operable | |
| 8-bit timers (TM50, TM51) | | Operable | |
| Timer H (TMH0, TMH1) | | Operable | |
| Watch timer | | Operable when main clock is selected as count clock | Operable |
| Watchdog timer 1 | | Operable | |
| Watchdog timer 2 | | Operable when Ring-OSC ($f_R$) is selected as count clock | Operable |
| Serial interface | CSI00, CSI01 | Operable | |
| | CSIA0 | Operable | |
| | $I^2C0$[Note] | Operable | |
| | UART0, UART1 | Operable | |
| Key interrupt function | | Operable | |
| A/D converter | | Operable | |
| Real-time output | | Operable | |
| Clock monitor (CLM) | | Operable | |
| Power-on-clear (POC) | | Operable | |
| Low-voltage detection (LVI) | | Operable | |
| Regulator | | Operable | |
| Port function | | Retains status before HALT mode was set. | |
| External bus interface | | Refer to **2.2 Pin Status**. | |
| Internal data | | The CPU registers, statuses, data, and all other internal data such as the contents of the internal RAM are retained as they were before the HALT mode was set. | |

**Note** Only in the μPD703308Y, 70F3306Y, 70F3308Y

## 21.4 IDLE Mode

### 21.4.1 Setting and operation status

The IDLE mode is set by clearing the PSMR.PSM bit to 0 and setting the PSC.STP bit to 1 in the normal operation mode.

In the IDLE mode, the clock oscillator continues operation but clock supply to the CPU and other on-chip peripheral functions stops.

As a result, program execution stops and the contents of the internal RAM before the IDLE mode was set are retained. The CPU and other on-chip peripheral functions stop operating. However, the on-chip peripheral functions that can operate with the subclock, Ring-OSC clock, or an external clock continue operating.

Table 21-5 shows the operation status in the IDLE mode.

The IDLE mode can reduce the power consumption more than the HALT mode because it stops the operation of the on-chip peripheral functions. The main clock oscillator does not stop, so the normal operation mode can be restored without waiting for the oscillation stabilization time after the IDLE mode has been released, in the same manner as when the HALT mode is released.

**Caution Insert five or more NOP instructions after the instruction that stores data in the PSC register to set the IDLE mode.**

### 21.4.2 Releasing IDLE mode

The IDLE mode is released by a non-maskable interrupt request signal (NMI pin input, INTWDT2 signal), unmasked external interrupt request signal (INTP0 to INTP7 pin input), unmasked internal interrupt request signal from the peripheral functions operable in the IDLE mode, or reset (except WDTRES1 signal).

After the IDLE mode has been released, the normal operation mode is restored.

**(1) Releasing IDLE mode by non-maskable interrupt request signal or unmasked maskable interrupt request signal**

The IDLE mode is released by a non-maskable interrupt request signal or an unmasked maskable interrupt request signal, regardless of the priority of the interrupt request. If the IDLE mode is set in an interrupt servicing routine, however, an interrupt request that is issued later is serviced as follows.

(a) If an interrupt request signal with a priority lower than that of the interrupt request currently being serviced is issued, only the IDLE mode is released, and that interrupt request signal is not acknowledged. The interrupt request signal itself is retained.

(b) If an interrupt request signal with a priority higher than that of the interrupt request currently being serviced is issued (including a non-maskable interrupt request signal), the IDLE mode is released and that interrupt request signal is acknowledged.

**Table 21-4. Operation After Releasing IDLE Mode by Interrupt Request Signal**

| Release Source | Interrupt Enabled (EI) Status | Interrupt Disabled (DI) Status |
|---|---|---|
| Non-maskable interrupt request signal | Execution branches to the handler address | |
| Maskable interrupt request signal | Execution branches to the handler address or the next instruction is executed | The next instruction is executed |

**Caution The interrupt request signal that is disabled by setting the PSC.NMI2M, PSC.NMI0M, and PSC.INTM bits to 1 becomes invalid and IDLE mode is not released.**

**(2) Releasing IDLE mode by reset**

The same operation as the normal reset operation is performed.

**Table 21-5. Operation Status in IDLE Mode**

| Setting of IDLE Mode / Item | | When CPU Is Operating with Main Clock | |
|---|---|---|---|
| | | When Subclock Is Not Used | When Subclock Is Used |
| CPU | | Stops operation | |
| ROM correction | | Stops operation | |
| Main clock oscillator | | Oscillation enabled | |
| Subclock oscillator | | – | Oscillation enabled |
| Ring-OSC ($f_R$) | | Operable | |
| Interrupt controller | | Stops operation | |
| 16-bit timer (TMP0) | | Stops operation | |
| 16-bit timers (TM00, TM01) | | TM00: Stops operation<br>TM01: Operable when INTWT is selected as count clock and $f_{BRG}$ is selected as count clock of WT | TM00: Stops operation<br>TM01: Operable when INTWT is selected as count clock |
| 8-bit timers (TM50, TM51) | | • Operable when TI5m is selected as count clock<br>• Operable when INTTM010 is selected as count clock and TM01 is enabled in IDLE mode | |
| Timer H (TMH0) | | Stops operation | |
| Timer H (TMH1) | | Operable when $f_R$/2048 is selected as count clock | |
| Watch timer | | Operable when main clock is selected as count clock | Operable |
| Watchdog timer 1 | | Stops operation | |
| Watchdog timer 2 | | Operable when $f_R$ is selected as count clock | Operable |
| Serial interface | CSI00, CSI01 | Operable when $\overline{SCK0m}$ input clock is selected as operation clock | |
| | CSIA0 | Stops operation | |
| | $I^2C0$ Note | Stops operation | |
| | UART0 | Operable when ASCK0 is selected as count clock | |
| | UART1 | Stops operation | |
| Key interrupt function | | Operable | |
| A/D converter | | Stops operation | |
| Real-time output | | Operable when INTTM5m is selected as real-time output trigger and TM5m is enabled in IDLE mode | |
| Clock monitor (CLM) | | Operable | |
| Power-on-clear (POC) | | Operable | |
| Low-voltage detection (LVI) | | Operable | |
| Regulator | | Continues operation | |
| Port function | | Retains status before IDLE mode was set. | |
| External bus interface | | Refer to **2.2 Pin Status**. | |
| Internal data | | The CPU registers, statuses, data, and all other internal data such as the contents of the internal RAM are retained as they were before the IDLE mode was set. | |

**Note** Only in the μPD703308Y, 70F3306Y, 70F3308Y

**Remark** m = 0, 1

## 21.5 STOP Mode

### 21.5.1 Setting and operation status

The STOP mode is set when the PSMR.PSM bit is set to 1 and the PSC.STP bit is set to 1 in the normal operation mode.

In the STOP mode, the subclock oscillator continues operating but the main clock oscillator stops. Clock supply to the CPU and the on-chip peripheral functions is stopped.

As a result, program execution is stopped, and the contents of the internal RAM before the STOP mode was set are retained. The CPU and other on-chip peripheral functions stop operating. However, the on-chip peripheral functions that can operate with the subclock oscillator, Ring-OSC clock, or an external clock continue operating.

Table 21-7 shows the operation status in the STOP mode.

Because the STOP mode stops operation of the main clock oscillator, it reduces the power consumption to a level lower than the IDLE mode. If the subclock oscillator, Ring-OSC clock, and external clock are not used, the power consumption can be minimized with only leakage current flowing.

**Caution Insert five or more NOP instructions after the instruction that stores data in the PSC register to set the STOP mode.**

### 21.5.2 Releasing STOP mode

The STOP mode is released by a non-maskable interrupt request signal (NMI pin input, INTWDT2 signal), unmasked external interrupt request signal (INTP0 to INTP7 pin input), unmasked internal interrupt request signal from the peripheral functions operable in the STOP mode, or reset (except WDTRES1 signal).

After the STOP mode has been released, the normal operation mode is restored after the oscillation stabilization time has been secured.

**(1) Releasing STOP mode by non-maskable interrupt request signal or unmasked maskable interrupt request signal**

The STOP mode is released by a non-maskable interrupt request signal or an unmasked maskable interrupt request signal, regardless of the priority of the interrupt request. If the STOP mode is set in an interrupt servicing routine, however, an interrupt request that is issued later is serviced as follows.

(a) If an interrupt request signal with a priority lower than that of the interrupt request currently being serviced is issued, only the STOP mode is released, and that interrupt request signal is not acknowledged. The interrupt request signal itself is retained.

(b) If an interrupt request signal with a priority higher than that of the interrupt request currently being serviced is issued (including a non-maskable interrupt request signal), the STOP mode is released and that interrupt request signal is acknowledged.

**Table 21-6. Operation After Releasing STOP Mode by Interrupt Request Signal**

| Release Source | Interrupt Enabled (EI) Status | Interrupt Disabled (DI) Status |
|---|---|---|
| Non-maskable interrupt request signal | Execution branches to the handler address | |
| Maskable interrupt request signal | Execution branches to the handler address or the next instruction is executed | The next instruction is executed |

**Caution The interrupt request signal that is disabled by setting the PSC.NMI2M, PSC.NMI0M, and PSC.INTM bits to 1 becomes invalid and STOP mode is not released.**

**(2) Releasing STOP mode by reset**

The same operation as the normal reset operation is performed.

**Table 21-7. Operation Status in STOP Mode**

| Setting of STOP Mode / Item | | When CPU Is Operating with Main Clock | |
|---|---|---|---|
| | | When Subclock Is Not Used | When Subclock Is Used |
| CPU | | Stops operation | |
| ROM correction | | Stops operation | |
| Main clock oscillator | | Oscillation stops | |
| Subclock oscillator | | – | Oscillation enabled |
| Ring-OSC ($f_R$) | | Operable | |
| Interrupt controller | | Stops operation | |
| 16-bit timer (TMP0) | | Stops operation | |
| 16-bit timers (TM00, TM01) | | Stops operation | TM00: Stops operation<br>TM01: Operable when INTWT is selected as count clock and $f_{XT}$ is selected as count clock of WT |
| 8-bit timers (TM50, TM51) | | Operable when TI5m is selected as count clock | Operable when TI5m is selected as count clock or when INTTM010 is selected as count clock and TM01 is enabled in STOP mode |
| Timer H (TMH0) | | Stops operation | |
| Timer H (TMH1) | | Operable when $f_R$/2048 is selected as count clock | |
| Watch timer | | Stops operation | Operable when $f_{XT}$ is selected as count clock |
| Watchdog timer 1 | | Stops operation | |
| Watchdog timer 2 | | Operable when $f_R$ is selected as count clock | Operable |
| Serial interface | CSI00, CSI01 | Operable when $\overline{\text{SCK0m}}$ input clock is selected as operation clock | |
| | CSIA0 | Stops operation | |
| | $I^2C0$[Note] | Stops operation | |
| | UART0 | Operable when ASCK0 is selected as count clock | |
| | UART1 | Stops operation | |
| Key interrupt function | | Operable | |
| A/D converter | | Stops operation | |
| Real-time output | | Operable when INTTM5m is selected as real-time output trigger and TM5m is enabled in STOP mode | |
| Clock monitor (CLM) | | Stops operation | |
| Power-on-clear (POC) | | Operable | |
| Low-voltage detection (LVI) | | Operable | |
| Regulator | | Stops operation | |
| Port function | | Retains status before STOP mode was set. | |
| External bus interface | | Refer to **2.2 Pin Status**. | |
| Internal data | | The CPU registers, statuses, data, and all other internal data such as the contents of the internal RAM are retained as they were before the STOP mode was set. | |

**Note** Only in the μPD703308Y, 70F3306Y, 70F3308Y

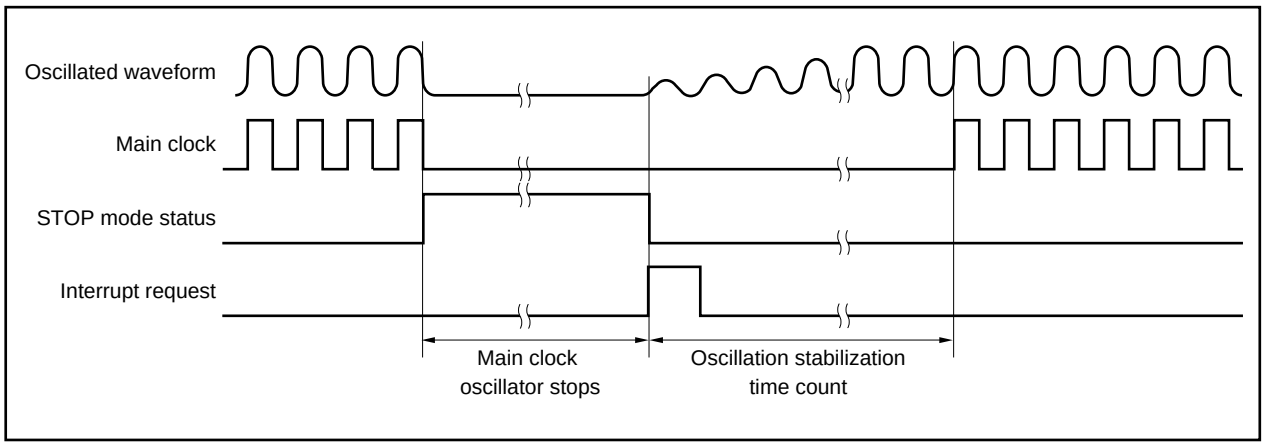
**Remark** m = 0, 1

### 21.5.3 Securing oscillation stabilization time when STOP mode is released

When the STOP mode is released, only the oscillation stabilization time set by the OSTS register elapses. If the STOP mode has been released by reset, however, the reset value of the OSTS register[Note] elapses.

The operation performed when the STOP mode is released by an interrupt request signal is shown below.

**Figure 21-2. Oscillation Stabilization Time**




**Note** The reset value of the OSTS register differs depending on the setting of the mask option/option byte. For details, refer to **CHAPTER 28 MASK OPTION/OPTION BYTE**.

**Caution For details of the OSTS register, refer to 21.2 (3) Oscillation stabilization time selection register (OSTS).**

## 21.6 Subclock Operation Mode

### 21.6.1 Setting and operation status

The subclock operation mode is set when the PCC.CK3 bit is set to 1 in the normal operation mode.

When the subclock operation mode is set, the internal system clock is changed from the main clock to the subclock.

When the PCC.MCK bit is set to 1, the operation of the main clock oscillator is stopped. As a result, the system operates only with the subclock.

Table 21-8 shows the operation status in subclock operation mode.

In the subclock operation mode, the power consumption can be reduced to a level lower than in the normal operation mode because the subclock is used as the internal system clock. In addition, the power consumption can be further reduced to the level of the STOP mode by stopping the operation of the main clock oscillator.

**Cautions 1. When manipulating the CK3 bit, do not change the set values of the PCC.CK2 to PCC.CK0 bits (using a bit manipulation instruction to manipulate the bit is recommended). For details, refer to 6.3 (1) Processor clock control register (PCC).**

**2. If the following conditions are not satisfied, change the CK2 to CK0 bits so that the conditions are satisfied and set the subclock operation mode.**

**Main clock ($f_{XX}$) > Subclock ($f_{XT}$: 32.768 kHz) × 4**

### 21.6.2 Releasing subclock operation mode

The subclock operation mode is released when the CK3 bit is cleared to 0 or by reset. If the main clock is stopped (MCK bit = 1), set the MCK bit to 1, secure the oscillation stabilization time of the main clock by software, and clear the CK3 bit to 0.

The normal operation mode is restored when the subclock operation mode is released.

**Caution When manipulating the CK3 bit, do not change the set values of the CK2 to CK0 bits (using a bit manipulation instruction to manipulate the bit is recommended). For details, refer to 6.3 (1) Processor clock control register (PCC).**

**Table 21-8. Operation Status in Subclock Operation Mode**

| Setting of Subclock Operation Mode / Item | | Operation Status | |
|---|---|---|---|
| | | When Main Clock Is Oscillating | When Main Clock Is Stopped |
| CPU | | Operable | |
| ROM correction | | Operable | |
| Subclock oscillator | | Oscillation enabled | |
| Ring-OSC ($f_R$) | | Operable | |
| Interrupt controller | | Operable | |
| 16-bit timer (TMP0) | | Operable | Stops operation |
| 16-bit timers (TM00, TM01) | | Operable | TM00: Stops operation<br>TM01: Operable when INTWT is selected as count clock and $f_{XT}$ is selected as count clock of WT |
| 8-bit timers (TM50, TM51) | | Operable | • Operable when TI5m is selected as count clock<br>• Operable when INTTM010 is selected as count clock and when TM01 is enabled in subclock operation mode |
| Timer H (TMH0) | | Operable | Stops operation |
| Timer H (TMH1) | | Operable | Operable when $f_R$/2048 is selected as count clock |
| Watch timer | | Operable | Operable when $f_{XT}$ is selected as count clock |
| Watchdog timer 1 | | Operable | Stops operation |
| Watchdog timer 2 | | Operable | |
| Serial interface | CSI00, CSI01 | Operable | Operable when $\overline{\text{SCK0m}}$ input clock is selected as operation clock |
| | CSIA0 | Operable | Stops operation |
| | $I^2C0$[Note] | Operable | Stops operation |
| | UART0 | Operable | Operable when ASCK0 is selected as count clock |
| | UART1 | Operable | Stops operation |
| Key interrupt function | | Operable | |
| A/D converter | | Operable | Stops operation |
| Real-time output | | Operable | Operable when INTTM5m is selected as real-time output trigger and TM5m is enabled in subclock operation mode |
| Clock monitor (CLM) | | Operable | Stops operation |
| Power-on-clear (POC) | | Operable | |
| Low-voltage detection (LVI) | | Operable | |
| Regulator | | Continues operation | |
| Port function | | Settable | |
| External bus interface | | Operable | |
| Internal data | | Settable | |

**Note** Only in the $\mu$PD703308Y, 70F3306Y, 70F3308Y

**Remark** m = 0, 1

## 21.7 Sub-IDLE Mode

### 21.7.1 Setting and operation status

The sub-IDLE mode is set when the PSMR.PSM bit is cleared to 0 and the PSC.STP bit is set to 1 in the subclock operation mode.

In this mode, the clock oscillator continues operation but clock supply to the CPU and the other on-chip peripheral functions is stopped.

As a result, program execution is stopped and the contents of the internal RAM before the sub-IDLE mode was set are retained. The CPU and the other on-chip peripheral functions are stopped. However, the on-chip peripheral functions that can operate with the subclock, Ring-OSC clock, or an external clock continue operating.

Table 21-10 shows the operation status in the sub-IDLE mode.

Because the sub-IDLE mode stops operation of the CPU and other on-chip peripheral functions, it can reduce the power consumption more than the subclock operation mode. If the sub-IDLE mode is set after the main clock has been stopped, the power consumption can be reduced to a level as low as that in the STOP mode.

### 21.7.2 Releasing sub-IDLE mode

The sub-IDLE mode is released by a non-maskable interrupt request signal (NMI pin input, INTWDT2 signal), unmasked external interrupt request signal (INTP0 to INTP7 pin input), unmasked internal interrupt request signal from the peripheral functions operable in the sub-IDLE mode, or reset (except WDTRES1 signal).

When the sub-IDLE mode is released by an interrupt request signal, the subclock operation mode is set. If it is released by reset, the normal operation mode is restored.

**(1) Releasing sub-IDLE mode by non-maskable interrupt request signal or unmasked maskable interrupt request signal**

The sub-IDLE mode is released by a non-maskable interrupt request signal or an unmasked maskable interrupt request signal, regardless of the priority of the interrupt request. If the sub-IDLE mode is set in an interrupt servicing routine, however, an interrupt request signal that is issued later is serviced as follows.

(a) If an interrupt request signal with a priority lower than that of the interrupt request currently being serviced is issued, only the sub-IDLE mode is released, and that interrupt request signal is not acknowledged. The interrupt request signal itself is retained.

(b) If an interrupt request signal with a priority higher than that of the interrupt request currently being serviced is issued (including a non-maskable interrupt request signal), the sub-IDLE mode is released and that interrupt request signal is acknowledged.

**Table 21-9. Operation After Releasing Sub-IDLE Mode by Interrupt Request Signal**

| Release Source | Interrupt Enabled (EI) Status | Interrupt Disabled (DI) Status |
|---|---|---|
| Non-maskable interrupt request signal | Execution branches to the handler address | |
| Maskable interrupt request signal | Execution branches to the handler address or the next instruction is executed | The next instruction is executed |

**Caution The interrupt request signal that is disabled by setting the PSC.NMI2M, PSC.NMI0M, and PSC.INTM bits to 1 becomes invalid and sub-IDLE mode is not released.**

**(2) Releasing sub-IDLE mode by reset**

The same operation as the normal reset operation is performed.

**Table 21-10. Operation Status in Sub-IDLE Mode**

| Item \ Setting of Sub-IDLE Mode | | Operation Status | |
|---|---|---|---|
| | | When Main Clock Is Oscillating | When Main Clock Is Stopped |
| CPU | | Stops operation | |
| ROM correction | | Stops operation | |
| Subclock oscillator | | Oscillation enabled | |
| Ring-OSC ($f_R$) | | Operable | |
| Interrupt controller | | Stops operation | |
| 16-bit timer (TMP0) | | Stops operation | |
| 16-bit timers (TM00, TM01) | | TM00: Stops operation<br>TM01: Operable when INTWT is selected as count clock | TM00: Stops operation<br>TM01: Operable when INTWT is selected as count clock and $f_{XT}$ is selected as count clock of WT |
| 8-bit timers (TM50, TM51) | | • Operable when TI5m is selected as count clock<br>• Operable when INTTM010 is selected as count clock and INTWT is selected as count clock of TM01 | • Operable when TI5m is selected as count clock<br>• Operable when INTTM010 is selected as count clock and when TM01 is enabled in sub-IDLE mode |
| Timer H (TMH0) | | Stops operation | |
| Timer H (TMH1) | | Operable when $f_R$/2048 is selected as count clock | |
| Watch timer | | Operable | Operable when $f_{XT}$ is selected as count clock |
| Watchdog timer 1 | | Stops operation | |
| Watchdog timer 2 | | Operable | |
| Serial interface | CSI00, CSI01 | Stops operation | Operable when $\overline{SCK0m}$ input clock is selected as operation clock |
| | CSIA0 | Stops operation | |
| | $I^2C0$ Note | Stops operation | |
| | UART0 | Operable when ASCK0 is selected as count clock | |
| | UART1 | Stops operation | |
| Key interrupt function | | Operable | |
| A/D converter | | Stops operation | |
| Real-time output | | Operable when INTTM5m is selected as real-time output trigger and TM5m is enabled in sub-IDLE mode | |
| Clock monitor (CLM) | | Operable | Stops operation |
| Power-on-clear (POC) | | Operable | |
| Low-voltage detection (LVI) | | Operable | |
| Regulator | | Stops operation | |
| Port function | | Retains status before sub-IDLE mode was set. | |
| External bus interface | | Refer to **2.2 Pin Status**. | |
| Internal data | | The CPU registers, statuses, data, and all other internal data such as the contents of the internal RAM are retained as they were before the sub-IDLE mode was set. | |

**Note** Only in the μPD703308Y, 70F3306Y, 70F3308Y

**Remark** m = 0, 1

# CHAPTER 22 RESET FUNCTION

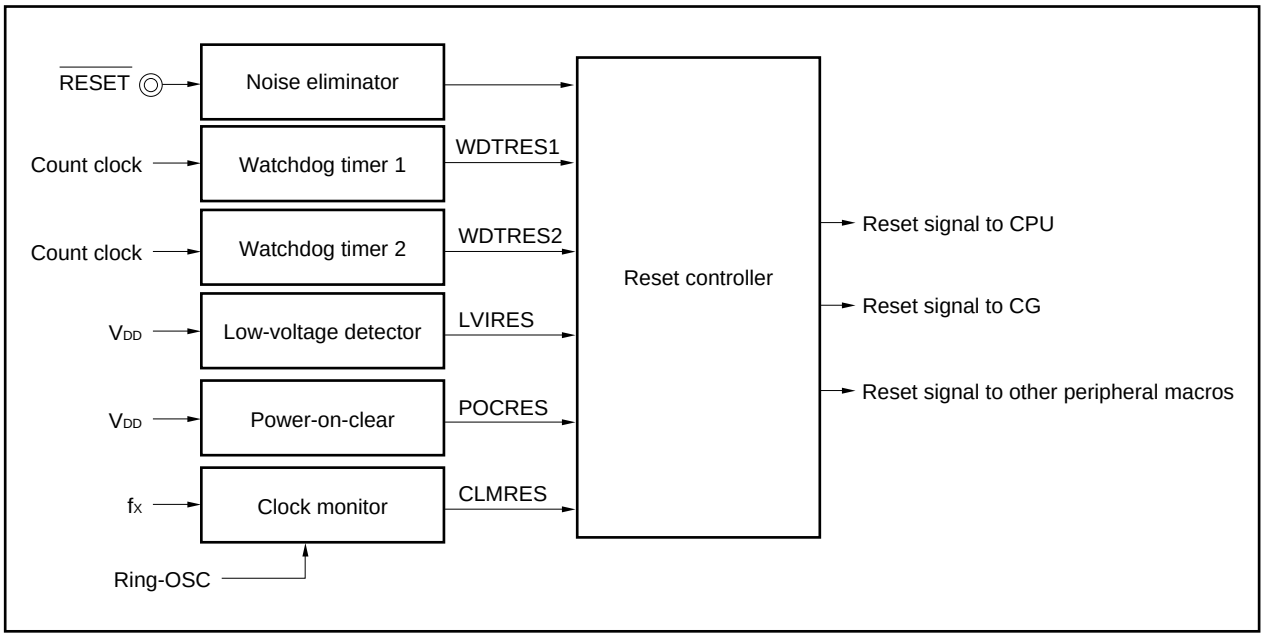
## 22.1 Overview

The following reset functions are available.

- Reset by $\overline{\text{RESET}}$ pin input
- Reset by watchdog timer 1 overflow (WDTRES1)
- Reset by watchdog timer 2 overflow (WDTRES2)
- System reset by low-voltage detector (LVI) (LVIRES)
- System reset by clock monitor (CLM) (CLMRES)
- System reset by power-on-clear (POC) (POCRES)
- Analog/digital + analog noise eliminator of $\overline{\text{RESET}}$ pin selectable
- Reset output function (P00/TOH0 pin)

## 22.2 Configuration

**Figure 22-1. Reset Block Diagram**



## 22.3 Register to Check Reset Source

**(1) Reset source flag register (RESF)**

The RESF register is a special register that can be written only by a combination of specific sequences (refer to **3.4.7 Special registers**).

The RESF register indicates the source from which a reset signal is generated.

This register can be read or written in 8-bit or 1-bit units (however, only "0" can be written to this register).

$\overline{\text{RESET}}$ pin input or reset by the POC circuit (POCRES) clears this register to 00H. The default value differs if reset is effected from a source other than the $\overline{\text{RESET}}$ pin.

After reset: 00H[Note] R/W Address: FFFFF888H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RESF | WDT1RF | 0 | 0 | WDT2RF | 0 | 0 | CLMRF | LVIRF |

| WDT1RF | Reset signal from watchdog timer 1 (WDTRES1) |
|---|---|
| 0 | Not generated |
| 1 | Generated |

| WDT2RF | Reset signal from watchdog timer 2 (WDTRES2) |
|---|---|
| 0 | Not generated |
| 1 | Generated |

| CLMRF | Reset signal from clock monitor (CLMRES) |
|---|---|
| 0 | Not generated |
| 1 | Generated |

| LVIRF | Reset signal from low-voltage detector (LVIRES) |
|---|---|
| 0 | Not generated |
| 1 | Generated |

**Note** This register is cleared to 00H when a reset is executed via the $\overline{\text{RESET}}$ pin or POC circuit.
When a reset is executed by the WDTRES1 signal, WDTRES2 signal, low-voltage detector (LVI), or clock monitor (CLM), the reset flags of this register (WDT1RF, WDT2RF, CLMRF, and LVIRF bits) are set (with the other sources retained).

**Caution Only "0" can be written to each bit of this register. If writing "0" conflicts with setting the flag (occurrence of reset), setting the flag takes precedence.**

## 22.4 Reset Sources

The following six reset sources are available.

- Reset by $\overline{\text{RESET}}$ pin input
- Reset by watchdog timer 1 overflow (WDTRES1)
- Reset by watchdog timer 2 overflow (WDTRES2)
- System reset by low-voltage detector (LVI) (LVIRES)
- System reset by clock monitor (CLM) (CLMRES)
- System reset by power-on-clear (POC) (POCRES)

### 22.4.1 Reset operation via $\overline{\text{RESET}}$ pin

When a low level is input to the $\overline{\text{RESET}}$ pin, the system is reset, and each hardware unit is initialized.

The $\overline{\text{RESET}}$ pin has a noise eliminator that can eliminate analog noise or digital + analog noise, depending on the setting of the RNZC register.

While a low level is being input to the $\overline{\text{RESET}}$ pin, the main clock oscillator stops. Therefore, the overall power consumption of the system can be reduced.

When the level of the $\overline{\text{RESET}}$ pin is changed from low to high, the reset status is released.

If the reset status is released by $\overline{\text{RESET}}$ pin input, the oscillation stabilization time elapses and then the CPU starts program execution (for the oscillation stabilization time, refer to **21.2 (3) Oscillation stabilization time selection register (OSTS)** and **CHAPTER 28 MASK OPTION/OPTION BYTE**).

**Table 22-1. Hardware Status on $\overline{\text{RESET}}$ Pin Input**

| Item | During Reset | After Reset |
|---|---|---|
| Main clock oscillator ($f_X$) | Oscillation stops | Oscillation starts |
| Subclock oscillator ($f_{XT}$) | Oscillation continues | |
| Ring-OSC ($f_R$) | Oscillation stops | Oscillation starts |
| Peripheral clock ($f_{XX}$ to $f_{XX}$/1024) | Operation stops | Operation starts after securing oscillation stabilization time |
| Internal system clock ($f_{CLK}$) | Operation stops | Operation starts after securing oscillation stabilization time (initialized to $f_{XX}$/8) |
| CPU clock ($f_{CPU}$) | Operation stops | Operation starts after securing oscillation stabilization time (initialized to $f_{XX}$/8) |
| Watchdog timer 1 clock ($f_{XW}$) | Operation stops | Operation starts |
| CPU | Initialized | Program execution starts after securing oscillation stabilization time |
| Internal RAM | Undefined if power-on reset or writing data to RAM (by CPU) and reset input conflict (data is damaged).<br>Otherwise value immediately before reset input is retained. | |
| I/O lines (P00) | Low-level output | |
| I/O lines (ports other than P00) | High impedance | |
| On-chip peripheral I/O registers | Initialized to specified status | |
| Watchdog timer 2 | Operation stops | Operation starts ($f_R$) |
| Other on-chip peripheral functions | Operation stops | Operation can be started after securing oscillation stabilization time |

**Figure 22-2. Hardware Status on $\overline{\text{RESET}}$ Pin Input**

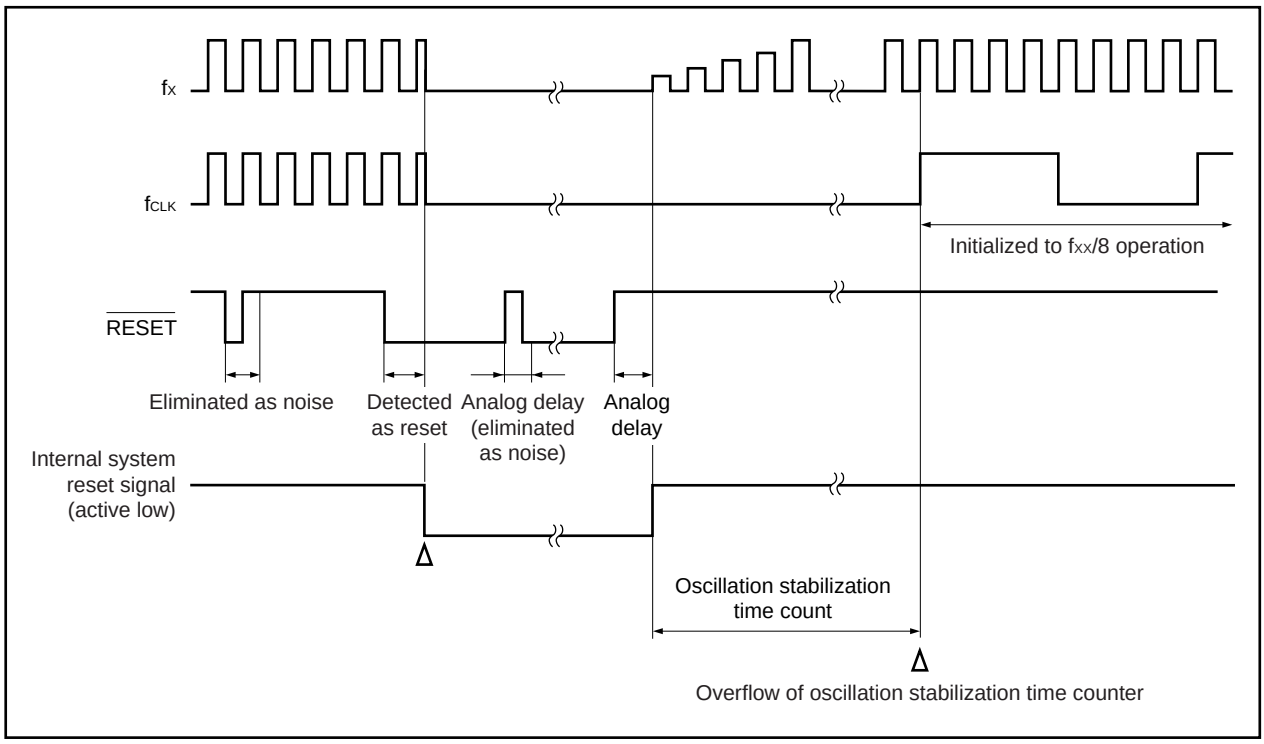
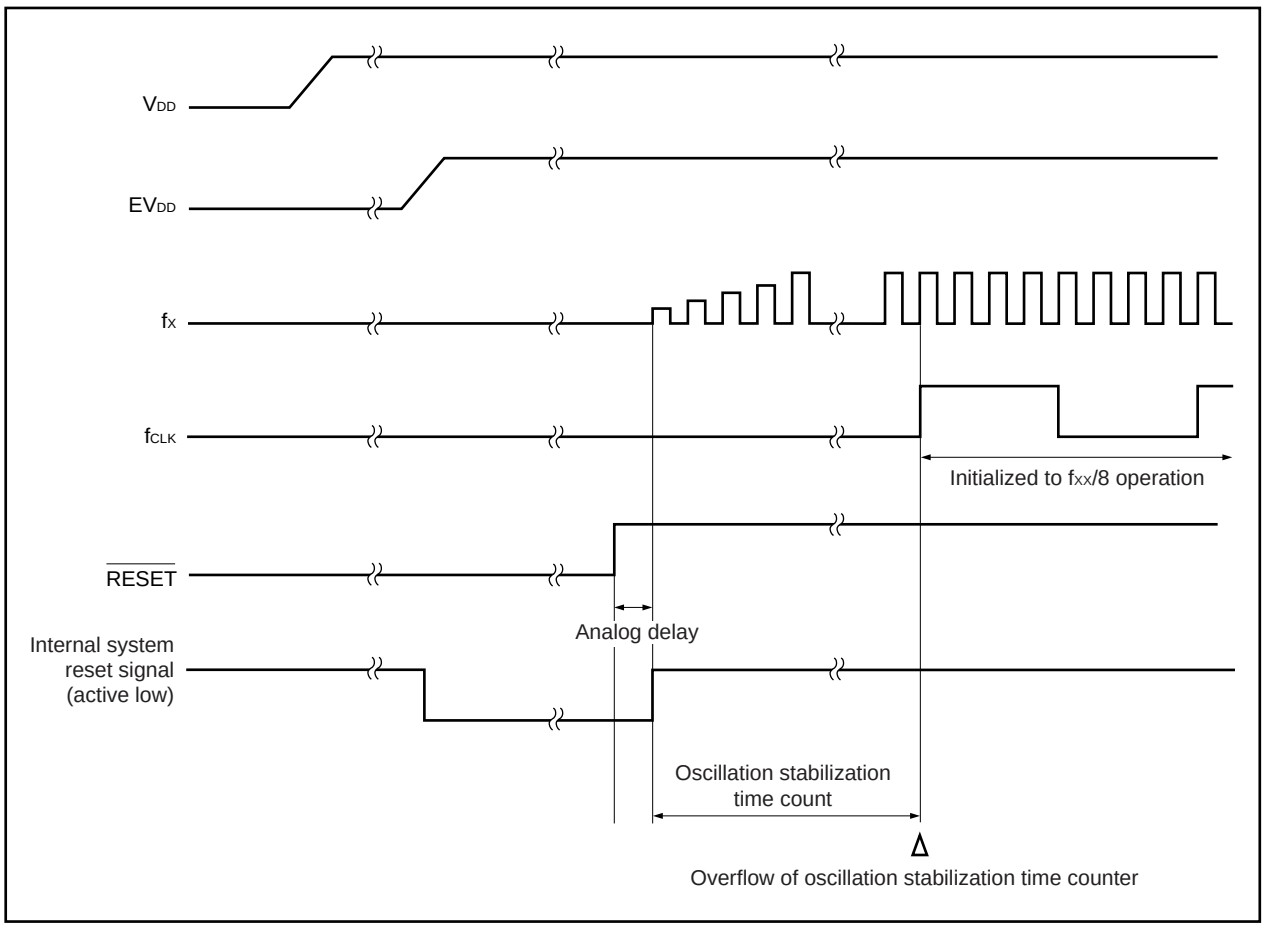



**Figure 22-3. Operation on Power Application**



**(1) Elimination of digital noise on $\overline{\text{RESET}}$ pin**

For the $\overline{\text{RESET}}$ pin of the V850ES/KF1+, an analog/digital + analog noise eliminator can be selected.

The digital noise eliminator is selected when the RNZC.RNZSEL bit = 1. The digital noise is sampled using the main clock ($f_X$), and the number of samplings can be selected from 10 or 20 by the RNZC.SMPSEL bit.

**(a) Reset noise elimination control register (RNZC)**

The RNZC register can be read or written in 8-bit units.

After reset, RNZC is cleared to 00H.

After reset: 00H R/W Address: FFFFF860H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| RNZC | 0 | 0 | 0 | 0 | 0 | 0 | SMPSEL | RNZSEL[Note] |

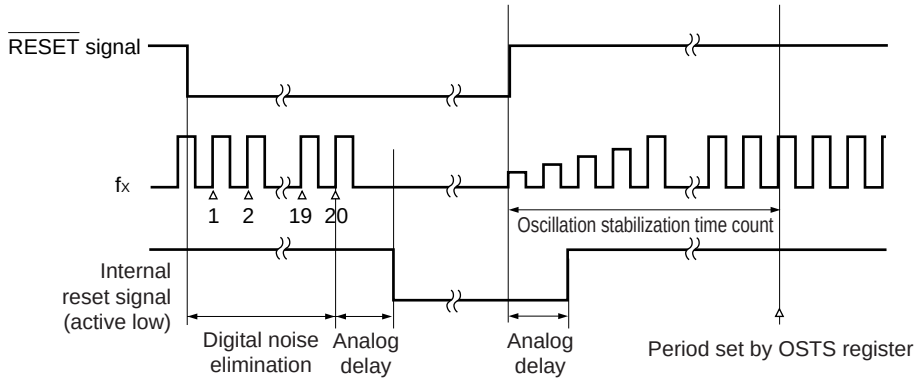
| SMPSEL | Selection of number of samplings |
|---|---|
| 0 | 20 times |
| 1 | 10 times |

| RNZSEL[Note] | Selection of noise eliminator of $\overline{\text{RESET}}$ pin |
|---|---|
| 0 | Analog noise elimination only |
| 1 | Digital + analog noise elimination |

**Note** If the sampling clock is stopped, only the analog noise is eliminated automatically, regardless of the setting of the RNZSEL bit.

**Caution The RNZC register can be set (written) only once after the reset signal is released. Even if the register is written two or more times, the first value written to it is not updated. To change the set value of the register, input the reset signal.**

**Figure 22-4. Sampling Operation Timing (20 Times)**




<1> Digital noise is eliminated when the RNZC.RNZSEL bit = 1.

<2> The $\overline{\text{RESET}}$ pin is always sampled at the rising edge of the sampling clock ($f_X$).

<3> If the $\overline{\text{RESET}}$ pin goes low and is detected as low over the entire sampling timing, it is detected as an internal reset signal. Because the analog noise eliminator is activated after digital noise has been eliminated, the internal reset signal is detected after analog delay.

<4> When the internal reset signal is detected, the RNZC register is initialized, so only the analog noise eliminator can be selected.

**(b) Operation when sampling clock is stopped**

If the sampling clock ($f_X$) stops when the digital + analog noise eliminator is selected, input to the $\overline{\text{RESET}}$ signal is not received. Therefore, only the analog noise eliminator is automatically selected.

Only the analog noise eliminator is automatically selected during the following periods.

- In STOP mode:
  Setting of STOP mode → Period to time set by the OSTS register that elapses after the STOP mode is released (by a source other than reset)
- In subclock operation mode:
  Setting of subclock operation mode (PCC.CLS bit = 0 → 1) → Period until the main clock operation mode (CLS bit = 1 → 0) is restored

**(c) Digital noise elimination width**

The digital noise elimination width ($t_{WRSL}$) is as follows where T is the sampling clock period and N is the number of samplings.

**Table 22-2. Digital Noise Elimination Width of $\overline{\text{RESET}}$ Pin**

| Digital Noise Elimination Width ($t_{WRSL}$) | | | Operation |
|---|---|---|---|
| | T = 10 MHz, N = 20 | T = 5 MHz, N = 10 | |
| $t_{WRSL} < (N-1)T$ | $t_{WRSL} < 1.9\ \mu s$ | $t_{WRSL} < 1.8\ \mu s$ | Eliminated as noise |
| $(N-1)T < t_{WRSL} < NT$ | $1.9\ \mu s \le t_{WRSL} < 2.0\ \mu s$ | $1.8\ \mu s \le t_{WRSL} < 2.0\ \mu s$ | May be eliminated as noise or detected as reset |
| $NT \le t_{WRSL}$ | $2.0\ \mu s \le t_{WRSL}$ | $2.0\ \mu s \le t_{WRSL}$ | Detected as reset |

**Remark** The noise on the $\overline{\text{RESET}}$ pin is eliminated by a value that takes the value shown in this table and the analog delay value into consideration.

### 22.4.2 Reset operation by WDTRES1 signal

If a reset operation mode in which reset is effected when watchdog timer 1 overflows is set, the system is reset when watchdog timer 1 overflows (when the WDTRES1 signal is generated), and each hardware unit is initialized to a specific status.

After watchdog timer 1 has overflowed, the system is reset for a specific duration of time ($f_{CLK}$: 12 clocks) and then automatically released from the reset status. After release of the reset status, the CPU starts program execution.

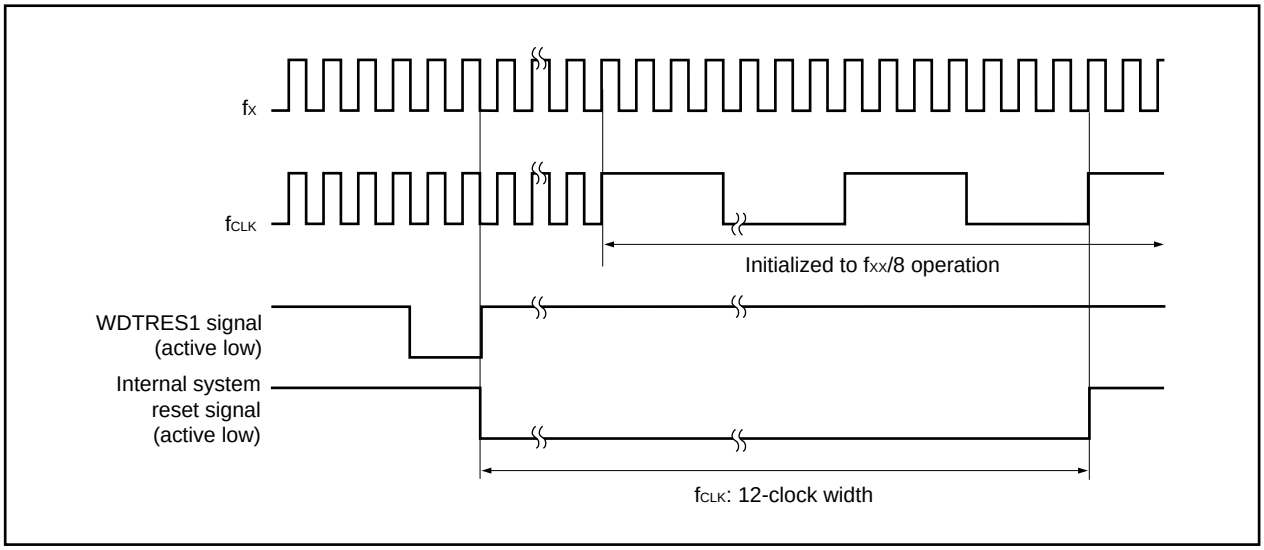
Note that, because the main clock oscillator continues operating even during the reset period, the oscillation stabilization time is not secured.

The following table shows the status of each hardware unit during the period of reset that is effected by the WDTRES1 signal and after release of the reset status.

**Table 22-3. Hardware Status on Occurrence of WDTRES1 Signal**

| Item | During Reset | After Reset |
|---|---|---|
| Main clock oscillator ($f_X$) | Oscillation continues | |
| Subclock oscillator ($f_{XT}$) | Oscillation continues | |
| Ring-OSC ($f_R$) | Oscillation continues | |
| Peripheral clock ($f_{XX}$ to $f_{XX}$/1024) | Operation stops | Operation starts |
| Internal system clock ($f_{CLK}$) | Oscillation continues (initialized to $f_{XX}$/8) | |
| CPU clock ($f_{CPU}$) | Oscillation continues (initialized to $f_{XX}$/8) | |
| Watchdog timer 1 clock ($f_{XW}$) | Operation continues | |
| Internal RAM | Undefined if writing data to RAM (by CPU) and reset input conflict (data is damaged). Otherwise value immediately before reset input is retained. | |
| I/O lines (P00) | Low-level output | |
| I/O lines (ports other than P00) | High impedance | |
| On-chip peripheral I/O registers | Initialized to specified status | |
| Watchdog timer 2 | Operation stops | Operation starts ($f_R$) |
| Other on-chip peripheral functions | Operation stops | Operation can be started |

**Figure 22-5. Timing of Reset Operation by Watchdog Timer 1**



### 22.4.3 Reset operation by WDTRES2 signal

If a reset operation mode in which reset is effected when watchdog timer 2 overflows is set, the system is reset when watchdog timer 2 overflows (when the WDTRES2 signal is generated), and each hardware unit is initialized to a specific status.

After watchdog timer 2 has overflowed, the system is reset for a specific duration of time (equivalent to analog delay) and then automatically released from the reset status. After release of the reset status, the oscillation stabilization time of the main clock oscillator is secured, and then the CPU starts program execution.

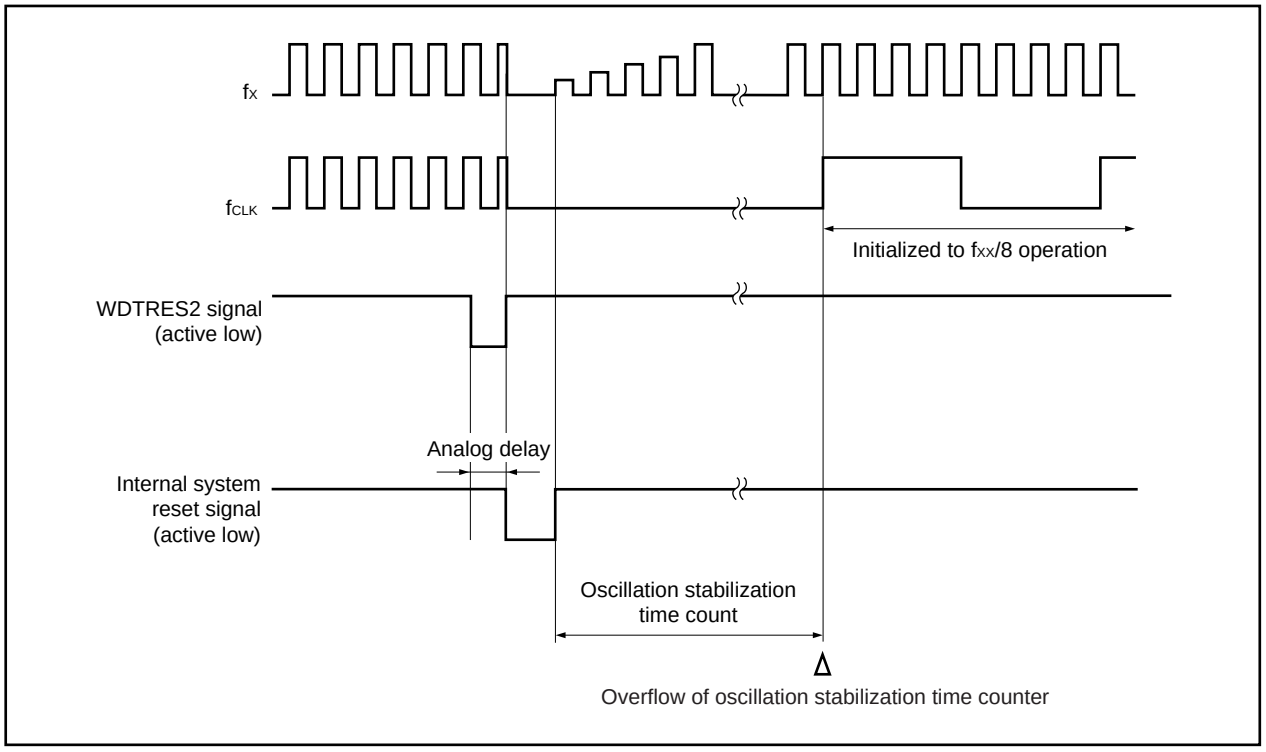
Note that, because the main clock oscillator stops during the reset period, the oscillation stabilization time must be secured. The oscillation stabilization time is determined by the default value of the OSTS register (for the oscillation stabilization time, refer to **21.2 (3) Oscillation stabilization time selection register (OSTS)** and **CHAPTER 28 MASK OPTION/OPTION BYTE**).

The status of each hardware unit during the period of reset effected by the WDTRES2 signal and after release of the reset status is the same as when reset is effected by the $\overline{\text{RESET}}$ pin input.

For details, refer to **Table 22-1 Hardware Status on $\overline{\text{RESET}}$ Pin Input**.

The following figure shows the timing of the reset operation by the WDTRES2 signal.

**Figure 22-6. Timing of Reset Operation by Watchdog Timer 2**



### 22.4.4 Power-on-clear reset operation

The supply voltage ($V_{DD}$) and detection voltage ($V_{POC}$) are compared. When $V_{DD} < V_{POC}$, the system is reset and each hardware unit is initialized to a specific status.

The detection voltage ($V_{POC}$) is 2.6 V ±0.1 V.

While $V_{DD} < V_{POC}$, the system is reset. Reset is released when $V_{DD} \geq V_{POC}$. After release of the reset status, the oscillation stabilization time of the main clock oscillator is secured, and then the CPU starts program execution.

Note that, because the main clock oscillator stops during the reset period, the oscillation stabilization time must be secured. The oscillation stabilization time is determined by the default value of the OSTS register (for the oscillation stabilization time, refer to **21.2 (3) Oscillation stabilization time selection register (OSTS)** and **CHAPTER 28 MASK OPTION/OPTION BYTE**).

The following table shows the status of each hardware unit during the period of reset effected by the POCRES signal and after release of reset.

**Table 22-4. Hardware Status During Reset Operation by Power-on-Clear**

| Item | During Reset | After Reset |
|---|---|---|
| Main clock oscillator ($f_X$) | Oscillation stops | Oscillation starts |
| Subclock oscillator ($f_{XT}$) | Oscillation continues | |
| Ring-OSC ($f_R$) | Oscillation stops | Oscillation starts |
| Peripheral clock ($f_{XX}$ to $f_{XX}$/1024) | Operation stops | Operation starts after securing oscillation stabilization time |
| Internal system clock ($f_{CLK}$) | Operation stops | Operation starts after securing oscillation stabilization time (initialized to $f_{XX}$/8) |
| CPU clock ($f_{CPU}$) | Operation stops | Operation starts after securing oscillation stabilization time (initialized to $f_{XX}$/8) |
| Watchdog timer 1 clock ($f_{XW}$) | Operation stops | Operation starts |
| CPU | Initialized | Program execution starts after securing oscillation stabilization time |
| Internal RAM | Undefined if power-on reset or writing data to RAM (by CPU) and reset input conflict (data is damaged).<br>Otherwise value immediately before reset input is retained. | |
| I/O lines (P00) | Low-level output | |
| I/O lines (ports other than P00) | High impedance | |
| On-chip peripheral I/O registers | Initialized to specified status | |
| Watchdog timer 2 | Operation stops | Operation starts ($f_R$) |
| Other on-chip peripheral functions | Operation stops | Operation can be started after securing oscillation stabilization time |

**Figure 22-7. Reset Timing by Power-on-Clear Circuit**

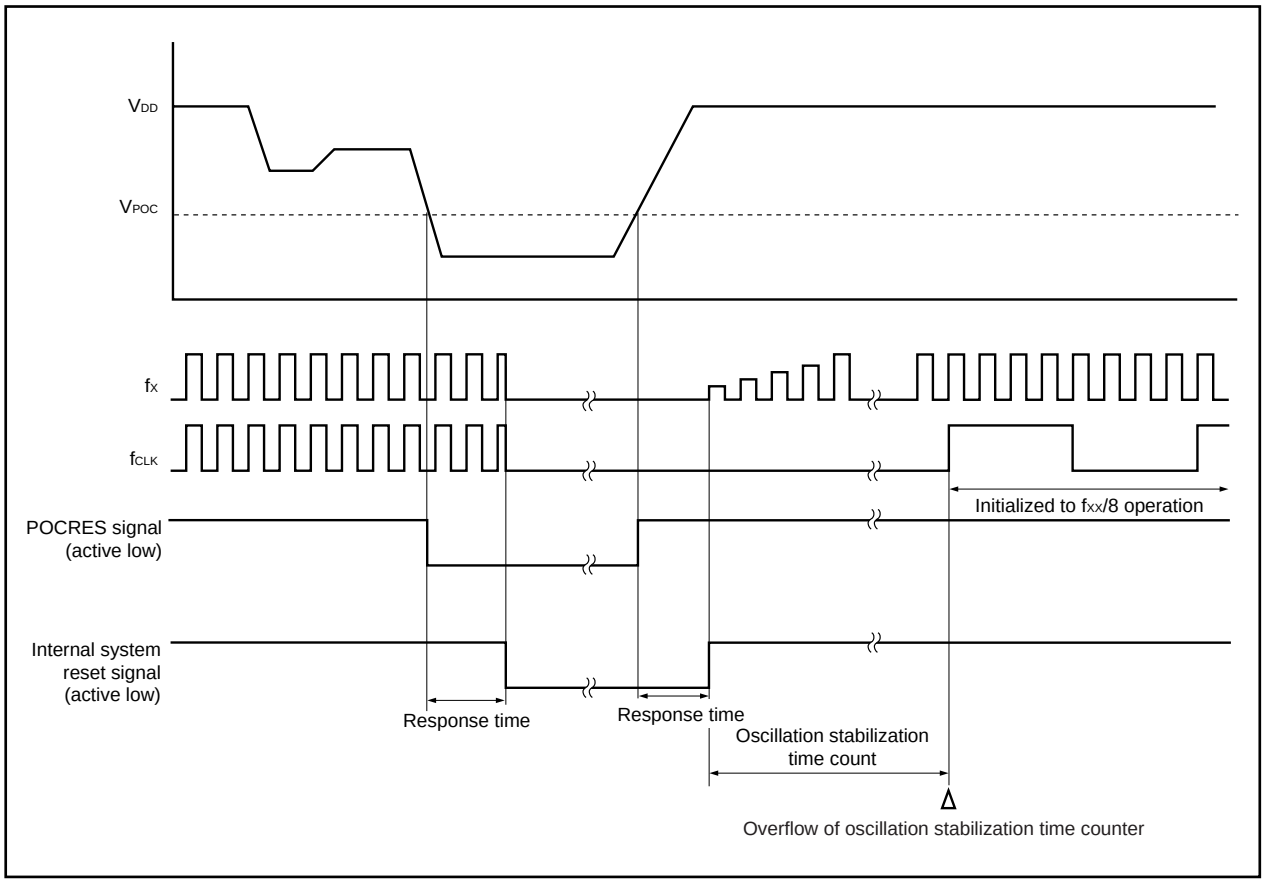
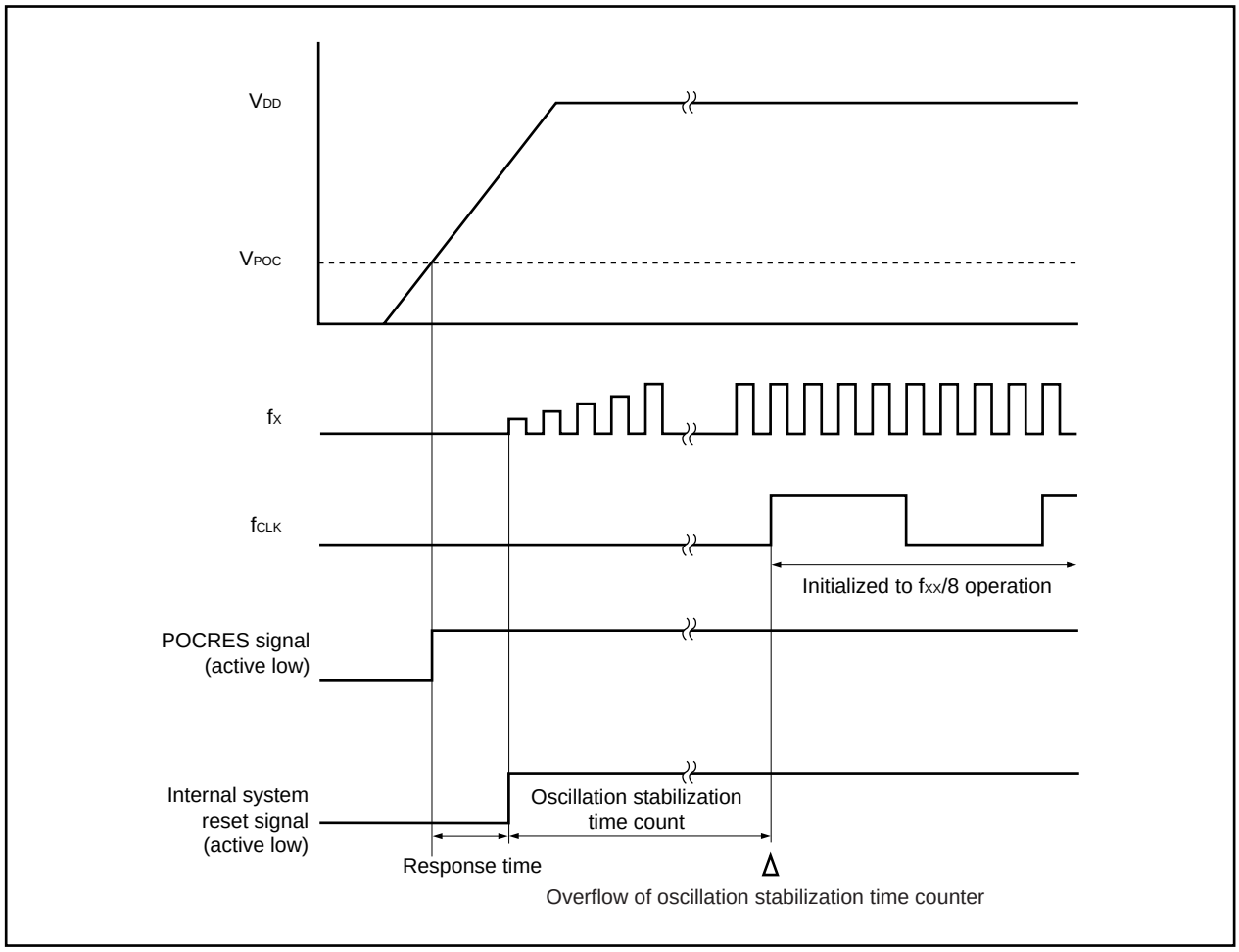


**Figure 22-8. Reset Timing on Power Application**



### 22.4.5 Reset operation by low-voltage detector

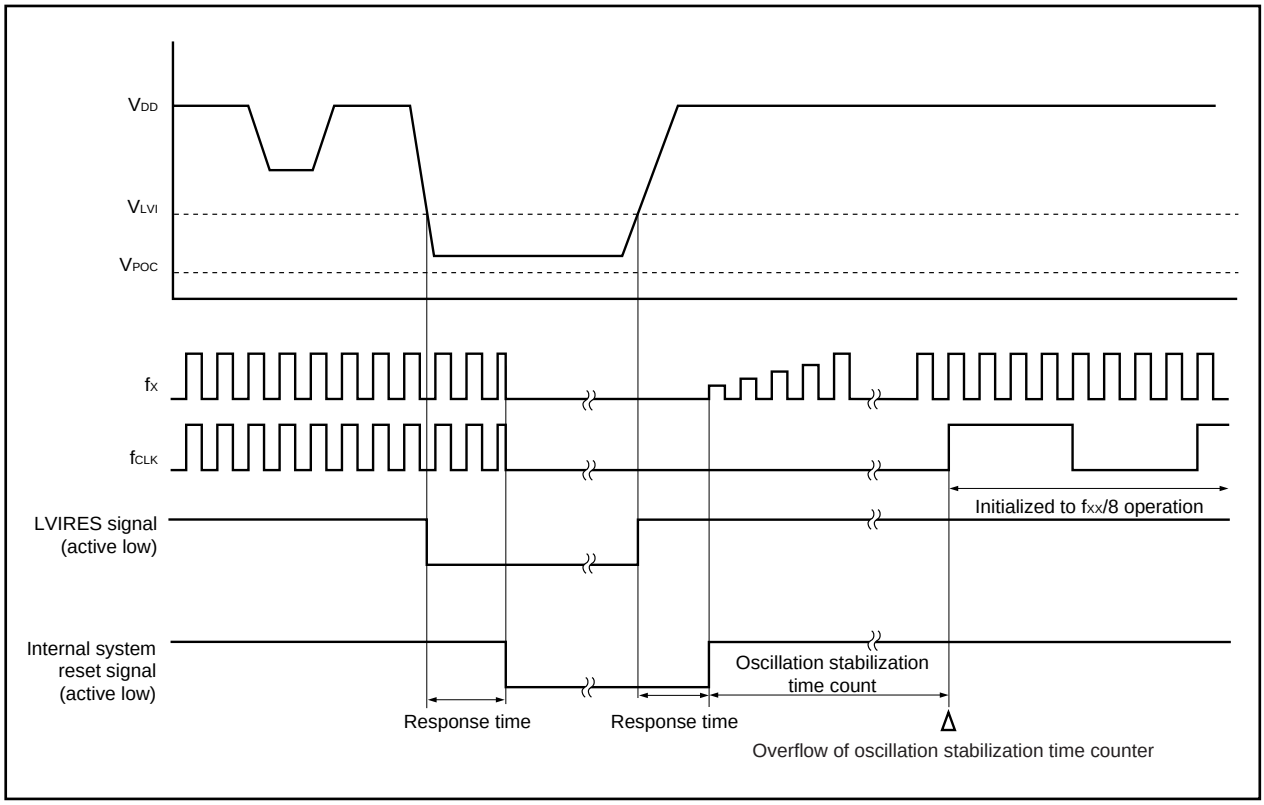
If a mode in which the internal reset signal (LVIRES) is to be generated by the low-voltage detector is set, the supply voltage ($V_{DD}$) and detection voltage ($V_{LVI}$) are compared. When $V_{DD} < V_{LVI}$, the system is reset and each hardware unit is initialized to a specific status.

While $V_{DD} < V_{LVI}$, the system is reset. Reset is released when $V_{DD} \geq V_{LVI}$. After release of the reset status, the oscillation stabilization time of the main clock oscillator is secured, and then the CPU starts program execution.

Note that, because the main clock oscillator stops during the reset period, the oscillation stabilization time must be secured. The oscillation stabilization time is determined by the default value of the OSTS register (for the oscillation stabilization time, refer to **21.2 (3) Oscillation stabilization time selection register (OSTS)** and **CHAPTER 28 MASK OPTION/OPTION BYTE**).

The status of each hardware unit during the period of reset effected by the LVIRES signal and after release of reset is the same as when reset is effected by the POCRES signal.

**Figure 22-9. Reset Timing by Low-Voltage Detector**



### 22.4.6 Reset operation by clock monitor

If the main clock is monitored using the sampling clock (Ring-OSC: $f_R$) and if it is detected that the main clock has stopped when the clock monitor operation is enabled, the system is reset and each hardware unit is initialized to a specific status.

After it is detected that the main clock stops, the system is reset for the duration of a specific time (equivalent to analog delay), and then the reset status is automatically released. After release of the reset status, the timer for oscillation stabilization does not perform its counting operation because the main clock is stopped. If watchdog timer 2, which starts by default, overflows, the CPU starts program execution with Ring-OSC ($f_R$).

The status of each hardware unit during the period of reset effected by the CLMRES signal and after release of the reset status is shown below.

For the timing of reset by the clock monitor, refer to **Figure 23-4**.

**Table 22-5. Hardware Status During Reset Operation by Clock Monitor**

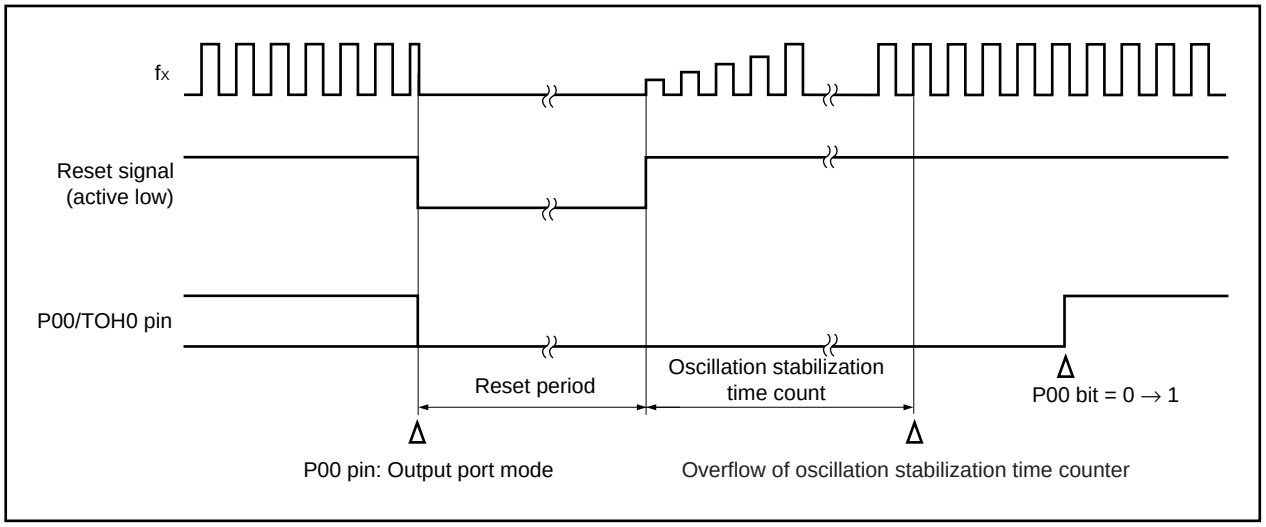
| Item | During Reset | After Reset |
|---|---|---|
| Main clock oscillator ($f_X$) | Oscillation stops | Oscillation remains stopped |
| Subclock oscillator ($f_{XT}$) | Oscillation continues | |
| Ring-OSC ($f_R$) | Oscillation stops | Oscillation starts |
| Peripheral clock ($f_{XX}$ to $f_{XX}$/1024) | Operation stops | Operation remains stopped because $f_X$ is stopped |
| Internal system clock ($f_{CLK}$) | Operation stops | Operation starts ($f_R$) after overflow of watchdog timer 2 |
| CPU clock ($f_{CPU}$) | Operation stops | Operation starts ($f_R$) after overflow of watchdog timer 2 |
| Watchdog timer 1 clock ($f_{XW}$) | Operation stops | Operation remains stopped because $f_X$ is stopped |
| CPU | Initialized | Program execution starts after overflow of watchdog timer 2 |
| Internal RAM | Undefined if writing data to RAM (by CPU) and reset input conflict (data is damaged). Otherwise value immediately before reset input is retained. | |
| I/O lines (P00) | Low-level output | |
| I/O lines (ports other than P00) | High impedance | |
| On-chip peripheral I/O registers | Initialized to specified status | |
| Watchdog timer 2 | Operation stops | Operation starts ($f_R$ only). However, WDTRES2 is not generated if watchdog timer 2 overflows before CPU execution. |
| Other on-chip peripheral functions | Operation stops | Operation cannot be started because $f_X$ is stopped.<br>However, the peripheral functions that operate on $f_{XT}$, $f_R$, or external clock can operate (for details, refer to **Table 23-2**). |

## 22.5 Reset Output Function

The P00/TOH0 pin of the V850ES/KF1+ can be used as a dummy reset output pin.

The P00 pin is set in the output port mode (PM0.PM00 bit = 0) and outputs a low level (P0.P00 bit = 0) when the reset signal is generated. To release the reset output (low-level output → high-level output), set the P00 bit to 1 by software.

**Figure 22-10. Reset Output Function**



# CHAPTER 23 CLOCK MONITOR

## 23.1 Function

The clock monitor samples the main clock by using the on-chip Ring-OSC clock and generates a reset signal (CLMRES) when oscillation of the main clock is stopped.

After reset is released, the CPU operates on Ring-OSC.

Once the operation of the clock monitor has been enabled by the CLM.CLME bit, it can be stopped only by reset.

The clock monitor automatically stops under the following conditions.

- When the oscillation stabilization time is counted after the STOP mode has been released
- When the main clock is stopped (PCC.MCK bit = 1 when subclock operates and PCC.CLS bit = 0 when main clock operates)
- When the sampling clock (Ring-OSC) is stopped
- When the CPU operates on Ring-OSC

## 23.2 Registers

**(1) Clock monitor mode register (CLM)**

The CLM register is a special register that can be written only by a combination of specific sequences (refer to **3.4.7 Special registers**).

The CLM register is used to select the operation mode of the clock monitor.

This register can be read or written in 8-bit or 1-bit units.

After reset, CLM is cleared to 00H.

After reset: 00H R/W Address: FFFFF870H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | <0> |
|---|---|---|---|---|---|---|---|---|
| CLM | 0 | 0 | 0 | 0 | 0 | 0 | 0 | CLME |

| CLME | Enable/disable of clock monitor operation |
|---|---|
| 0 | Disable clock monitor operation |
| 1 | Enable clock monitor operation |

**Caution Once the CLME bit has been set to 1, it cannot be cleared to 0 by any means other than reset.**

**(2) Ring-OSC mode register (RCM)**

The RCM register is an 8-bit register that sets the operation mode of Ring-OSC.

This register can be read or written in 8-bit or 1-bit units.

After reset, RCM is cleared to 00H.

After reset: 00H R/W Address: FFFFF80CH

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | <0> |
|---|---|---|---|---|---|---|---|---|
| RCM | 0 | 0 | 0 | 0 | 0 | 0 | 0 | RSTOP |

| RSTOP | Oscillation/stop of Ring-OSC |
|---|---|
| 0 | Ring-OSC oscillating |
| 1 | Ring-OSC stopped |

**Caution The setting of the RCM register is valid when stopping oscillation of Ring-OSC by software is enabled by the mask option/option byte. For details, refer to CHAPTER 28 MASK OPTION/ OPTION BYTE.**

## 23.3 Operation

The clock monitor start and stop conditions are as follows.

**<Monitor start condition>**

Set the CLM.CLME bit to 1

**<Monitor stop conditions>**

- When the oscillation stabilization time is counted after the STOP mode has been released
- When the main clock is stopped (PCC.MCK bit = 1 when subclock operates and PCC.CLS bit = 0 when main clock operates)
- When the sampling clock (Ring-OSC) is stopped
- When the CPU operates on Ring-OSC

**Table 23-1. Operation Status of Clock Monitor (When CLME Bit = 1, During Ring-OSC Operation)**

| Operation Mode | | Status of Main Clock | Status of Ring-OSC Clock | Status of Clock Monitor |
|---|---|---|---|---|
| Normal operation mode | | Oscillates | Oscillates[Note 1] | Operates[Note 2] |
| HALT mode | | Oscillates | Oscillates[Note 1] | Operates[Note 2] |
| IDLE mode | | Oscillates | Oscillates[Note 1] | Operates[Note 2] |
| STOP mode | | Stops | Oscillates[Note 1] | Stops |
| Subclock operation mode | MCK bit = 0 | Oscillates | Oscillates[Note 1] | Operates[Note 2] |
| Sub-IDLE mode | | Oscillates | Oscillates[Note 1] | Operates[Note 2] |
| Subclock operation mode | MCK bit = 1 | Stops | Oscillates[Note 1] | Stops |
| Sub-IDLE mode | | Stops | Oscillates[Note 1] | Stops |
| Ring clock operation mode | | Stops | Oscillates[Note 1] | Stops |
| During reset | | Stops | Stops | Stops |

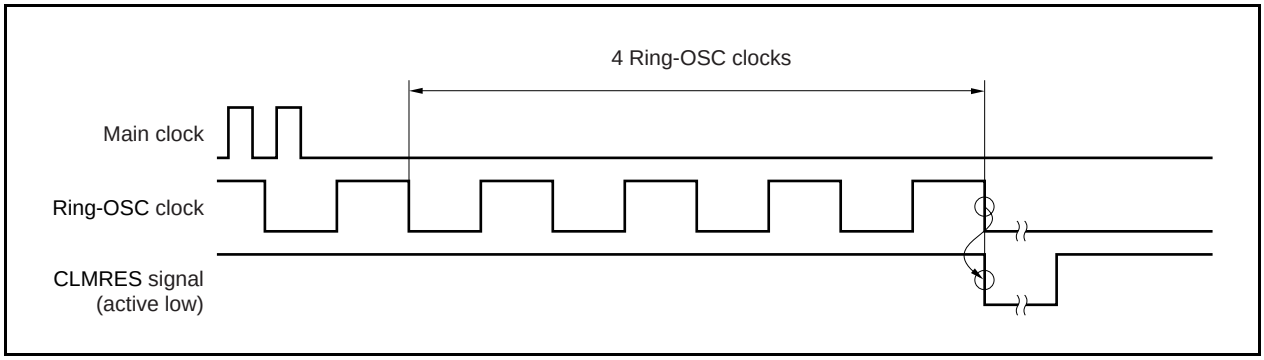
**Notes 1.** Ring-OSC can be stopped by setting the RCM.RSTOP bit to 1.
(Valid only when specified by mask option/option byte. For details, refer to **CHAPTER 28 MASK OPTION/OPTION BYTE**).

**2.** The clock monitor is stopped while Ring-OSC is stopped.

**(a) Operation when main clock oscillation is stopped**

If oscillation of the main clock is stopped when the CLME bit = 1, the CLMRES signal is generated as shown in Figure 23-1.

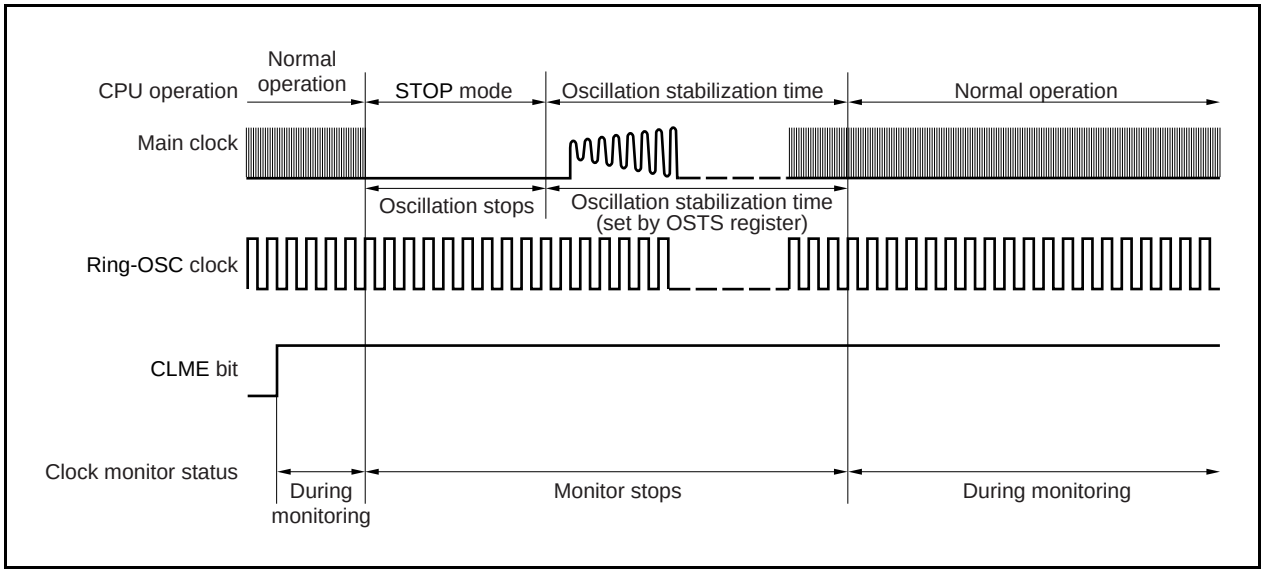
**Figure 23-1. When Oscillation of Main Clock Is Stopped**




**(b) Operation in STOP mode and after STOP mode is released**

If the STOP mode is set when the CLME bit = 1, the monitor operation is stopped in the STOP mode and while the oscillation stabilization time is being counted. The monitor operation is automatically started after the oscillation stabilization time has elapsed.

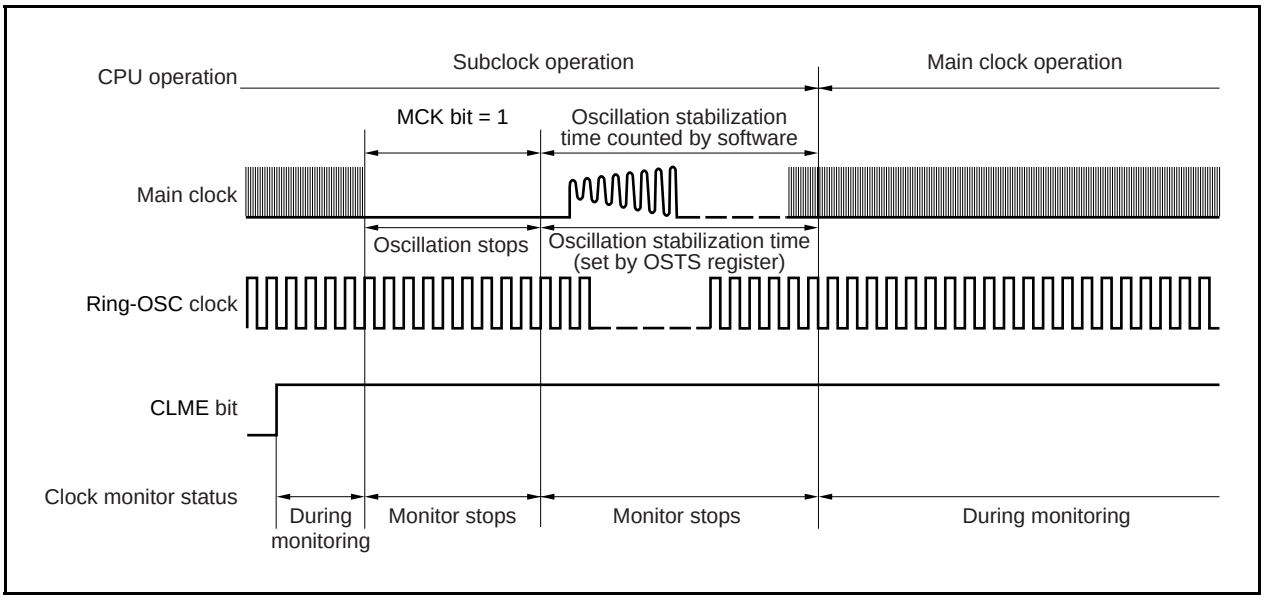
**Figure 23-2. Operation in STOP Mode and After STOP Mode Is Released**



**(c) Operation when main clock is stopped (arbitrary)**

If the main clock is stopped by setting the PCC.MCK bit to 1 while the subclock is operating (PCC.CLS bit = 1), the monitor operation is stopped until the main clock operates (CLS bit = 0). The monitor operation is automatically started when the main clock starts operating.

**Figure 23-3. Operation When Main Clock Is Stopped (Arbitrary)**




**(d) Operation when CPU operates on Ring-OSC clock (CCLS.CCLSF bit = 1)**

The monitor operation is not started even if the CLME bit is set to 1 when the CCLSF bit is 1.

## 23.4 Ring Clock Operation Mode

### 23.4.1 Setting and operation status

The ring clock operation mode is set by the clock monitor function when the main clock oscillation frequency ($f_X$) is abnormal (stopped).

In the ring clock operation mode, Ring-OSC ($f_R$) is supplied as the internal system clock ($f_{CLK}$) and CPU clock ($f_{CPU}$).

Because the operating clock is Ring-OSC ($f_R$), it is recommended to reset the system once to set it in the normal operation mode.

Because the main clock oscillator ($f_X$) is stopped, only the internal peripheral functions that can operate on the subclock, ring clock, or external clock can continue operating.

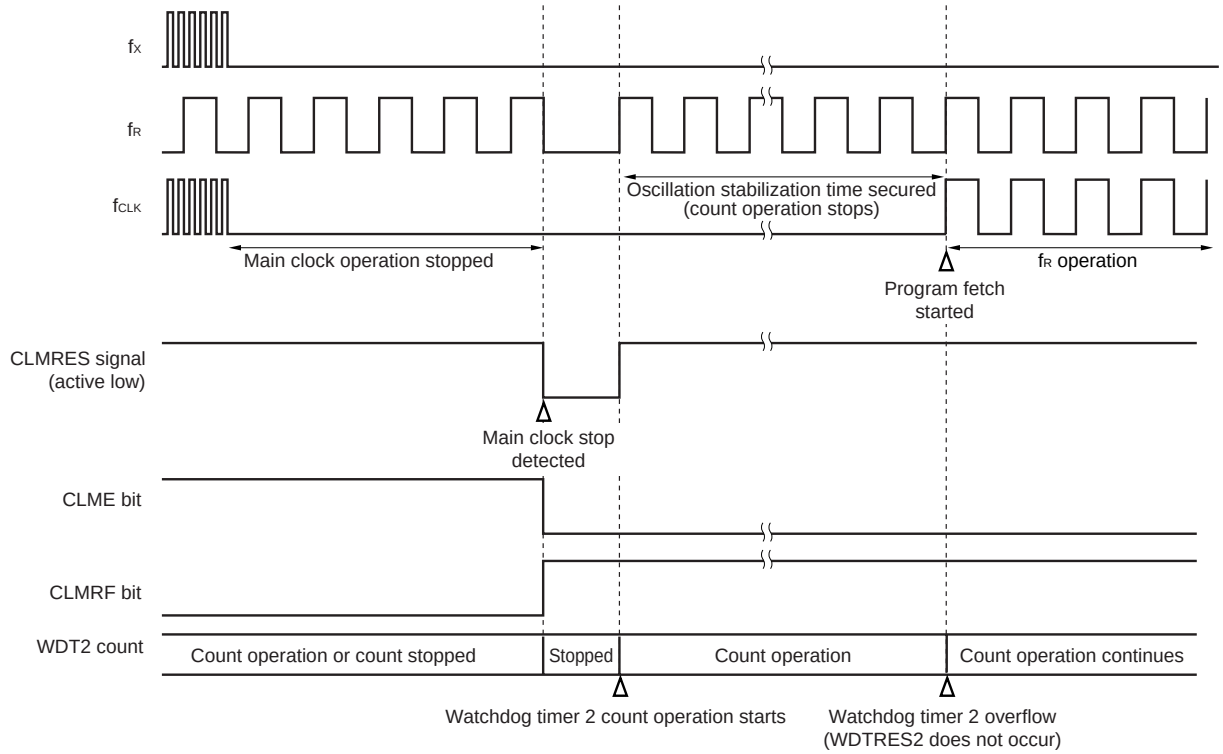
Table 23-2 shows the operation status in the ring clock operation mode.

### 23.4.2 Releasing ring clock operation mode

The ring clock operation mode is replaced by the normal operation mode in which the main clock ($f_X$) oscillates when the system is reset.

The ring clock operation mode cannot be released by software.

**Figure 23-4. Reset Timing of Clock Monitor**




**Remark** Software cannot be used to restore the normal operation mode from the ring clock operation mode. After reset (generation of the $\overline{\text{RESET}}$, WDTRES2, POCRES, or LVIRES signal), the normal operation mode can be restored only if the main clock ($f_X$) oscillates correctly.

**Table 23-2. Operation Status in Ring Clock Operation Mode**

| Setting of Ring Clock Operation Mode / Item | | Operation Status | |
|---|---|---|---|
| | | When Subclock Is Not Used | When Subclock Is Used |
| ROM correction | | Operable | |
| Interrupt controller | | Operable | |
| 16-bit timer (TMP0) | | Stops operation | |
| 16-bit timers (TM00, TM01) | | Stops operation | TM00: Stops operation<br>TM01: Operable when INTWT is selected as count clock and $f_{XT}$ is selected as count clock of WT |
| 8-bit timers (TM50, TM51) | | Operable when TI5m is selected as count clock | Operable when TI5m is selected as count clock or when INTTM010 is selected as count clock and TM01 is enabled in ring clock operation mode |
| Timer H (TMH0) | | Stops operation | |
| Timer H (TMH1) | | Operable when $f_R$/2048 is selected as count clock | |
| Watch timer | | Stops operation | Operable when $f_{XT}$ is selected as count clock |
| Watchdog timer 1 | | Stops operation | |
| Watchdog timer 2 | | Operable when $f_R$ is selected as count clock | Operable |
| Serial interface | CSI00, CSI01 | Operable when $\overline{\text{SCK0m}}$ input clock is selected as operation clock | |
| | CSIA0 | Stops operation | |
| | $I^2C0$[Note] | Stops operation | |
| | UART0 | Operable when ASCK0 is selected as count clock | |
| | UART1 | Stops operation | |
| Key interrupt function | | Operable | |
| A/D converter | | Stops operation | |
| Real-time output | | Operable when INTTM5m is selected as real-time output trigger and TM5m is enabled in ring clock operation mode | |
| Clock monitor | | Stops operation | |
| Power-on-clear | | Operable | |
| Low-voltage detector | | Operable | |
| Regulator | | Operable | |
| Port function | | Operable | |
| External bus interface | | Operable | |

**Note** Only in the μPD703308Y, 70F3306Y, 70F3308Y

**Remark** m = 0, 1

## 23.5 Ring HALT Mode

### 23.5.1 Setting and operation status

The ring HALT mode is set when a dedicated instruction (HALT instruction) is executed in the ring clock operation mode.

In the ring HALT mode, the Ring-OSC oscillator continues operating. Only clock supply to the CPU is stopped; clock supply to the other on-chip peripheral functions continues.

As a result, program execution is stopped, and the internal RAM retains the contents before the ring HALT mode was set. The on-chip peripheral functions that are independent of instruction processing by the CPU continue operating. The main clock oscillator ($f_X$) stops but the on-chip peripheral functions that can operate on the subclock ($f_{XT}$), Ring-OSC clock ($f_R$), or external clock continue operating.

Table 23-4 shows the operation status in the ring HALT mode.

**Cautions 1. Insert five or more NOP instructions after the HALT instruction.**

**2. If the HALT instruction is executed with an unmasked interrupt request signal held pending, the system shifts to the ring HALT mode, but the ring HALT mode is immediately released by the pending interrupt request signal.**

### 23.5.2 Releasing ring HALT mode

When the ring HALT mode is released by an interrupt request signal, the ring clock operation mode is set. When the ring HALT mode is released by reset, the normal operation mode is restored if the main clock ($f_X$) oscillates correctly.

**(1) Releasing ring HALT mode by non-maskable interrupt request signal or unmasked maskable interrupt request signal**

The ring HALT mode is released by a non-maskable interrupt request signal or an unmasked maskable interrupt request signal, regardless of the priority of the interrupt request. If the ring HALT mode is set in an interrupt servicing routine, however, an interrupt request that is issued later is serviced as follows.

(a) If an interrupt request signal with a priority lower than that of the interrupt request currently being serviced is issued, the ring HALT mode is released, but that interrupt request signal is not acknowledged. The interrupt request signal itself is retained.

(b) If an interrupt request signal with a priority higher than that of the interrupt request currently being serviced is issued (including a non-maskable interrupt request signal), the ring HALT mode is released and that interrupt request signal is acknowledged.

**Table 23-3. Operation After Releasing Ring HALT Mode by Interrupt Request Signal**

| Release Source | Interrupt Enabled (EI) Status | Interrupt Disabled (DI) Status |
|---|---|---|
| Non-maskable interrupt request signal | Execution branches to the handler address | |
| Maskable interrupt request signal | Execution branches to the handler address or the next instruction is executed | The next instruction is executed |

**(2) Releasing ring HALT mode by reset**

The same operation as the normal reset operation is performed.

**Table 23-4. Operation Status in Ring HALT Mode**

| Item \ Setting of Ring HALT Mode | | Operation Status | |
|---|---|---|---|
| | | When Subclock Is Not Used | When Subclock Is Used |
| CPU | | Stops operation | |
| ROM correction | | Stops operation | |
| Main clock oscillator | | Stops operation | |
| Subclock oscillator | | – | Continues operation |
| Interrupt controller | | Operable | |
| 16-bit timer (TMP0) | | Stops operation | |
| 16-bit timers (TM00, TM01) | | Stops operation | TM00: Stops operation<br>TM01: Operable when INTWT is selected as count clock and $f_{XT}$ is selected as count clock of WT |
| 8-bit timers (TM50, TM51) | | Operable when TI5m is selected as count clock | Operable when TI5m is selected as count clock or when INTTM010 is selected as count clock and TM01 is enabled in ring HALT mode |
| Timer H (TMH0) | | Stops operation | |
| Timer H (TMH1) | | Operable when $f_R$/2048 is selected as count clock | |
| Watch timer | | Stops operation | Operable when $f_{XT}$ is selected as count clock |
| Watchdog timer 1 | | Stops operation | |
| Watchdog timer 2 | | Operable when $f_R$ is selected as count clock | Operable |
| Serial interface | CSI00, CSI01 | Operable when $\overline{SCK0m}$ input clock is selected as operation clock | |
| | CSIA0 | Stops operation | |
| | $I^2C0$[Note] | Stops operation | |
| | UART0 | Operable when ASCK0 is selected as count clock | |
| | UART1 | Stops operation | |
| Key interrupt function | | Operable | |
| A/D converter | | Stops operation | |
| Real-time output | | Operable when INTTM5m is selected as real-time output trigger and TM5m is enabled in ring HALT mode | |
| Clock monitor | | Stops operation | |
| Power-on-clear | | Operable | |
| Low-voltage detector | | Operable | |
| Regulator | | Continues operation | |
| Port function | | Retains status before ring HALT mode was set. | |
| External bus interface | | Refer to **2.2 Pin Status**. | |

**Note** Only in the μPD703308Y, 70F3306Y, 70F3308Y

**Remark** m = 0, 1

# CHAPTER 24 LOW-VOLTAGE DETECTOR

## 24.1 Function

The low-voltage detector (LVI) has the following functions.

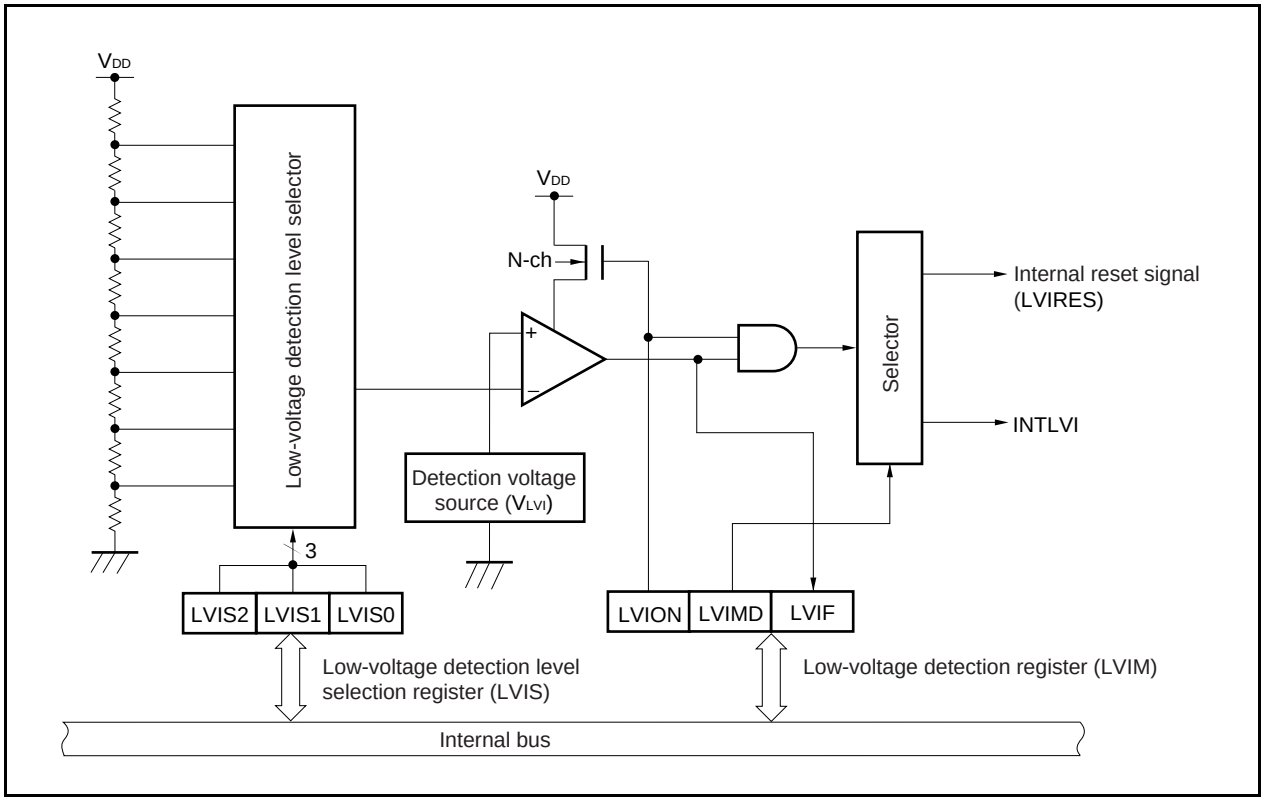
- Compares the supply voltage ($V_{DD}$) and detection voltage ($V_{LVI}$), and generates an interrupt request signal (INTLVI) or reset signal (LVIRES) when $V_{DD} < V_{LVI}$.
- Detection levels (seven levels) of supply voltage can be changed by software.
- Interrupt or reset function can be selected by software.
- Operable in STOP mode.

When the low-voltage detector is used to reset, the RESF.LVIRF bit is set to 1 if the LVIRES signal is generated. For details of the RESF register, refer to **22.3 (1) Reset source flag register (RESF)**.

## 24.2 Configuration

A block diagram of the low-voltage detector is shown below.

**Figure 24-1. Block Diagram of Low-Voltage Detector**



## 24.3 Registers

The low-voltage detector is controlled by the following two registers.

- Low-voltage detection register (LVIM)
- Low-voltage detection level selection register (LVIS)

**(1) Low-voltage detection register (LVIM)**

The LVIM register is an 8-bit register that sets the operation mode of the low-voltage detector.

The LVIM register is a special register that can be written only by a combination of specific sequences (refer to **3.4.7 Special registers**).

This register can be read or written in 8-bit or 1-bit units. If the LVION and LVIMD bits = 11, however, the LVIM register cannot be rewritten until the reset signal (LVIRES) is generated.

The LVIM register is reset to 00H by a reset source other than the low-voltage detector. The LVIM register holds its value when reset is effected by the low-voltage detector.

After reset: 00H[Note 1] R/W Address: FFFFF890H

| | <7> | 6 | 5 | 4 | 3 | 2 | <1> | <0> |
|---|---|---|---|---|---|---|---|---|
| LVIM | LVION | 0 | 0 | 0 | 0 | 0 | LVIMD | LVIF[Note 2] |

| LVION | Enable/disable low-voltage detection operation |
|---|---|
| 0 | Disable operation |
| 1 | Enable operation |

| LVIMD | Low-voltage detection operation mode selection |
|---|---|
| 0 | Generate interrupt request signal (INTLVI) when supply voltage ($V_{DD}$) < detection voltage |
| 1 | Generate internal reset signal (LVIRES) when supply voltage ($V_{DD}$) < detection voltage |

| LVIF[Note 2] | Low-voltage detection flag |
|---|---|
| 0 | Supply voltage ($V_{DD}$) > detection voltage ($V_{LVI}$), or when operation is disabled |
| 1 | Supply voltage ($V_{DD}$) < detection voltage ($V_{LVI}$) |

**Notes 1.** The LVIM register holds its value when reset is effected by the low-voltage detector.

**2.** The LVIF bit is read-only.

**Caution Be sure to clear bits 6 to 2 to 0.**

**Remark** The value of the LVIF bit is output as the interrupt request signal (INTLVI) when the LVION bit = 1 and LVIMD bit = 0.

**(2) Low-voltage detection level selection register (LVIS)**

The LVIS register is an 8-bit register that selects the low-voltage detection level.

The LVIS register can be read or written in 8-bit units. If the LVIM.LVION and LVIM.LVIMD bits = 11, however, the LVIS register cannot be rewritten until the reset signal (LVIRES) is generated.

The LVIS register is reset to 00H by a reset source other than the low-voltage detector. The LVIS register holds its value when reset is effected by the low-voltage detector.

After reset: 00H[Note] R/W Address: FFFFF891H

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| LVIS | 0 | 0 | 0 | 0 | 0 | LVIS2 | LVIS1 | LVIS0 |

| LVIS2 | LVIS1 | LVIS0 | Detection level |
|---|---|---|---|
| 0 | 0 | 0 | 4.3 V ±0.2 V |
| 0 | 0 | 1 | 4.1 V ±0.2 V |
| 0 | 1 | 0 | 3.9 V ±0.2 V |
| 0 | 1 | 1 | 3.7 V ±0.2 V |
| 1 | 0 | 0 | 3.5 V ±0.2 V |
| 1 | 0 | 1 | 3.3 V ±0.15 V |
| 1 | 1 | 0 | 3.1 V ±0.15 V |
| Other than above | | | Setting prohibited |

**Note** The LVIS register holds its value when reset is effected by the low-voltage detector.

**Caution Be sure to clear bits 7 to 3 to 0.**

## 24.4 Operation

The low-voltage detector can be used in the following two modes.

- Reset operation (LVIRES): Compares the supply voltage ($V_{DD}$) and detection voltage ($V_{LVI}$), and generates a reset signal (LVIRES) when $V_{DD} < V_{LVI}$.
- Interrupt operation (INTLVI): Compares the supply voltage ($V_{DD}$) and detection voltage ($V_{LVI}$), and generates an interrupt request signal (INTLVI) when $V_{DD} < V_{LVI}$.

### (1) Reset operation (LVIRES)

**<When starting operation>**

<1> Mask the INTLVI interrupt (LVIMK bit = 1).
<2> Set the detection voltage ($V_{LVI}$) using the LVIS.LVIS2 to LVIS.LVIS0 bits.
<3> Set the LVIM.LVION bit to 1 (enables low-voltage detector operation).
<4> Use software to instigate a wait of at least 0.2 ms.
<5> Confirm that the LVIM.LVIF bit is cleared to 0 (supply voltage ($V_{DD}$) > detection voltage ($V_{LVI}$)).
When the LVIF bit is set to 1, use software to instigate a wait until the LVIF bit is cleared to 0.
<6> Set the LVIM.LVIMD bit to 1 (generates internal reset signal (LVIRES) when supply voltage ($V_{DD}$) < detection voltage ($V_{LVI}$)).

**Caution <1> must always be executed. When the LVIMK bit = 0, an interrupt (INTLVI) may occur immediately after the processing in <3>.**

**<When stopping operation>**

The low-voltage detection operation cannot be stopped until a reset signal other than LVIRES is generated.

### (2) Interrupt operation (INTLVI)

**<When starting operation>**

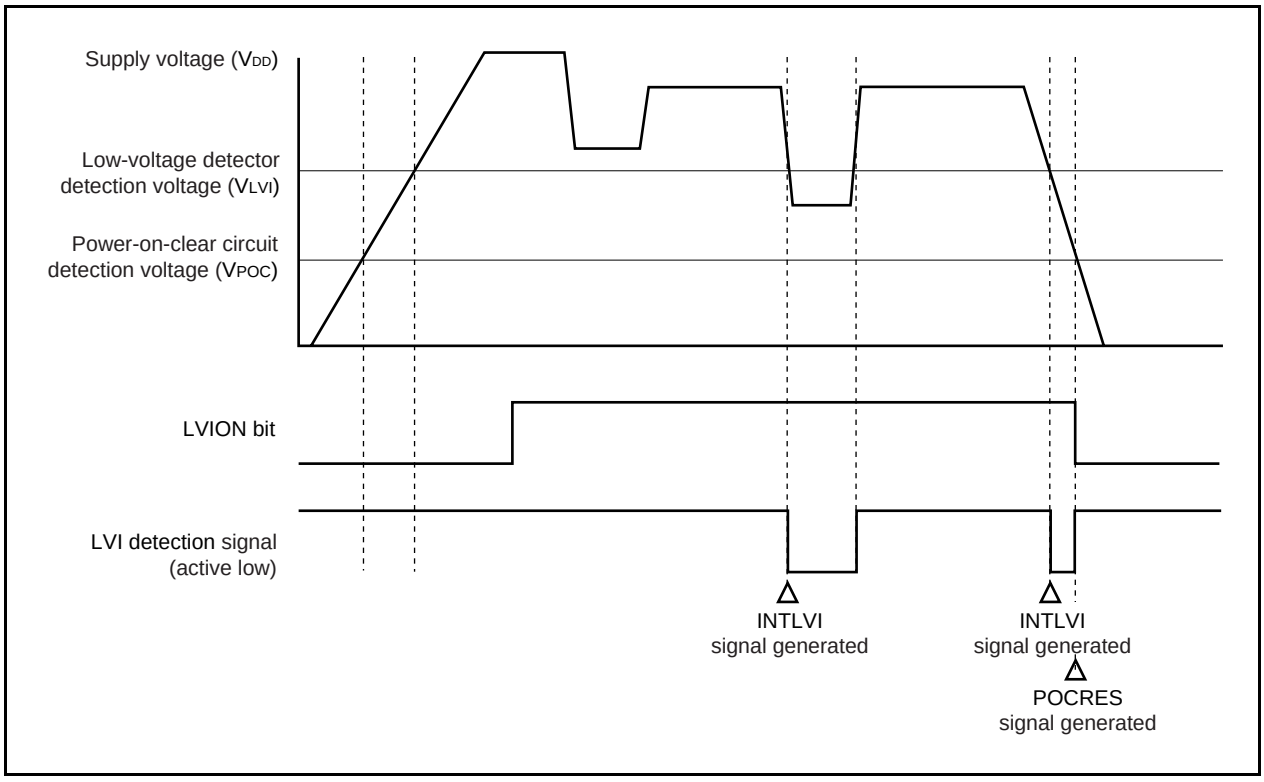
<1> Mask the INTLVI interrupt (LVIMK bit = 1).
<2> Set the detection voltage ($V_{LVI}$) using the LVIS.LVIS2 to LVIS.LVIS0 bits.
<3> Set the LVIM.LVION bit to 1 (enables low-voltage detector operation).
<4> Use software to instigate a wait of at least 0.2 ms.
<5> Confirm that the LVIM.LVIF bit is cleared to 0 (supply voltage ($V_{DD}$) > detection voltage ($V_{LVI}$)).
When the LVIF bit is set to 1, use software to instigate a wait until the LVIF bit is cleared to 0.
<6> Clear the INTLVI interrupt request flag (LVIIF bit) to 0.
<7> Release the INTLVI interrupt mask status (LVIMK bit = 0).

**Caution <1> must always be executed. When the LVIMK bit = 0, an interrupt (INTLVI) may occur immediately after the processing in <3>.**

**<When stopping operation>**

Clear the LVION bit to 0.

**Figure 24-2. Timing of INTLVI Interrupt Generation by Low-Voltage Detector**



# CHAPTER 25 POWER-ON-CLEAR CIRCUIT

## 25.1 Function

The power-on-clear (POC) circuit has the following functions.

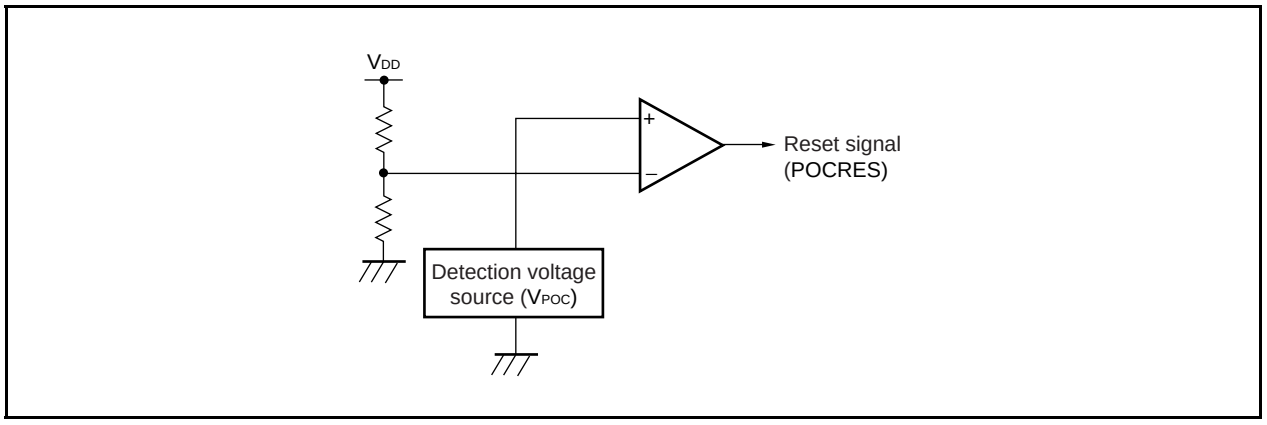
- Generates a reset signal (POCRES) upon power application.
- Compares the supply voltage ($V_{DD}$) and detection voltage ($V_{POC}$), and generates a reset signal (POCRES) when $V_{DD} < V_{POC}$ (detection voltage: $V_{POC}$ = 2.6 V ±0.1 V).

**Caution If the POCRES signal is generated by the POC circuit, the RESF register is cleared (to 00H).**

## 25.2 Configuration

A block diagram of the power-on-clear circuit is shown below.

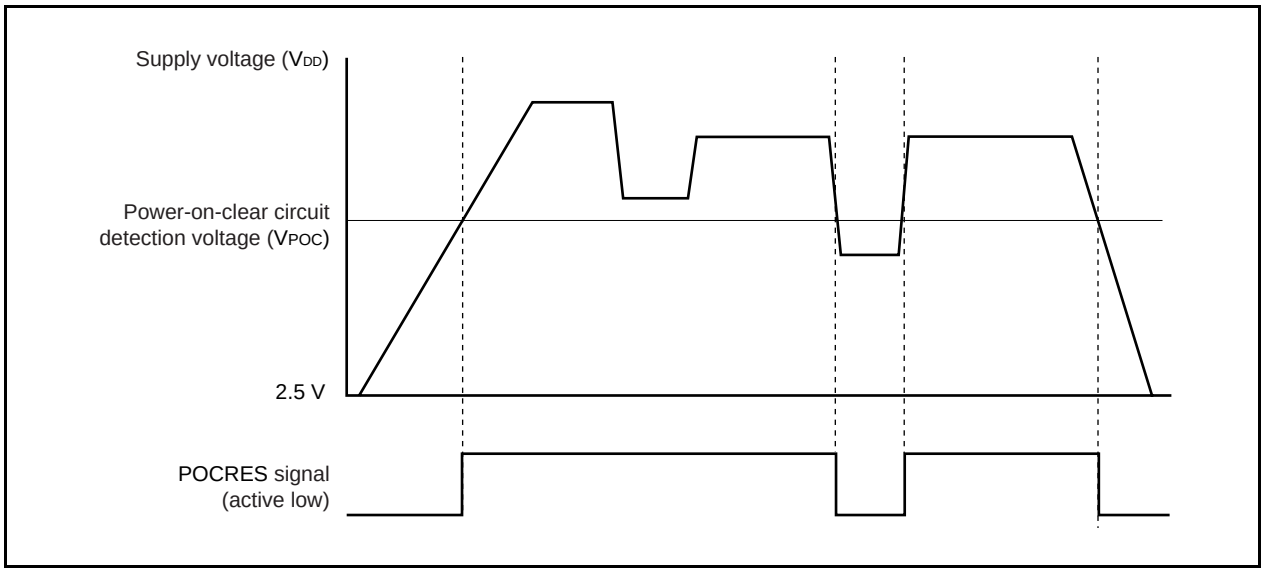
**Figure 25-1. Block Diagram of Power-on-Clear Circuit**



## 25.3 Operation

The power-on-clear circuit compares the supply voltage ($V_{DD}$) and detection voltage ($V_{POC}$), and generates a reset signal (POCRES) when $V_{DD} < V_{POC}$.

**Figure 25-2. Operation of Power-on-Clear Circuit**



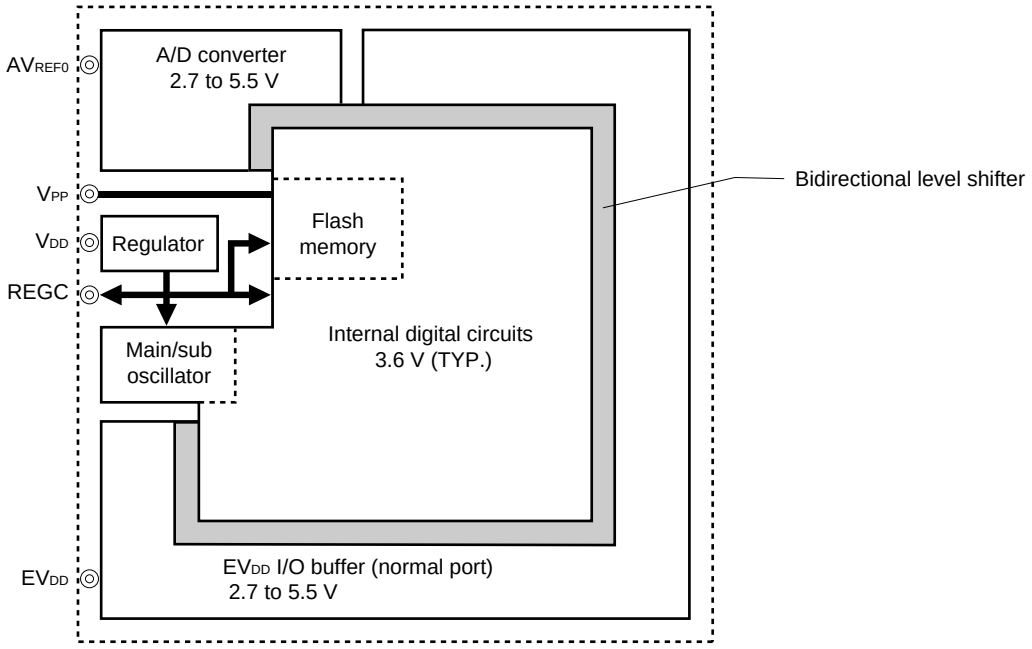
# CHAPTER 26 REGULATOR

## 26.1 Overview

The V850ES/KF1+ includes a regulator to reduce the power consumption and noise.

This regulator supplies a stepped-down $V_{DD}$ power supply voltage to the oscillator block and internal logic circuits (except the A/D converter and output buffer). The regulator output voltage is set to 3.6 V (TYP.).

**Figure 26-1. Regulator**




**Caution Use the regulator with a setting of $V_{DD} = EV_{DD} = AV_{REF0}$.**

## 26.2 Operation

The regulator stops operating in the following modes (but only when REGC = $V_{DD}$).

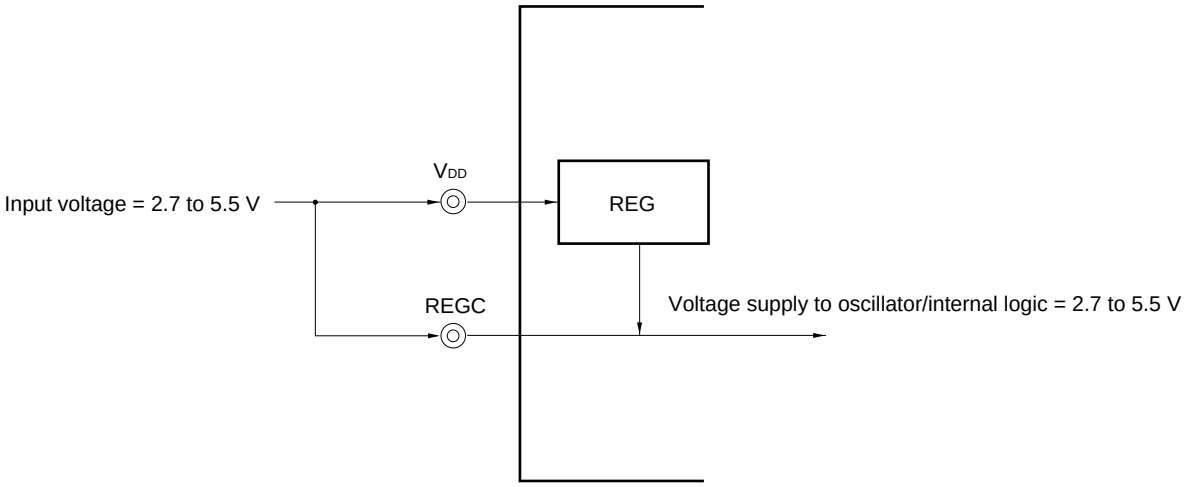
- At reset (except WDTRES1 and during oscillation stabilization time)
- In STOP mode
- In sub-IDLE mode

When using the regulator, be sure to connect a capacitor (10 $\mu$F) to the REGC pin to stabilize the regulator output.

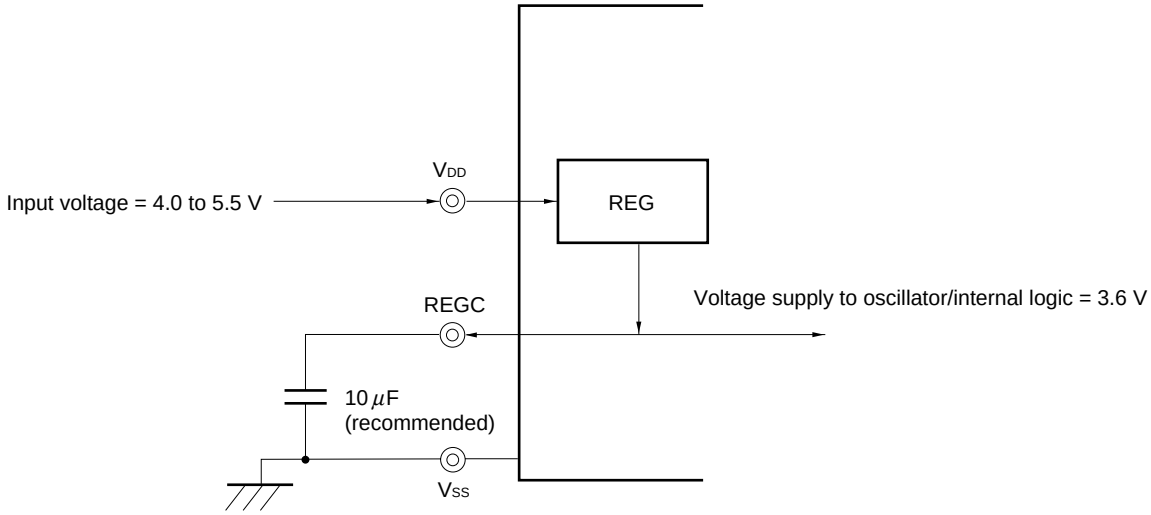
A diagram of the regulator pin connections is shown below.

**Figure 26-2. REGC Pin Connection**

**(a) When REGC = $V_{DD}$**




**(b) When connecting REGC pin to $V_{SS}$ via a capacitor**



# CHAPTER 27 ROM CORRECTION FUNCTION

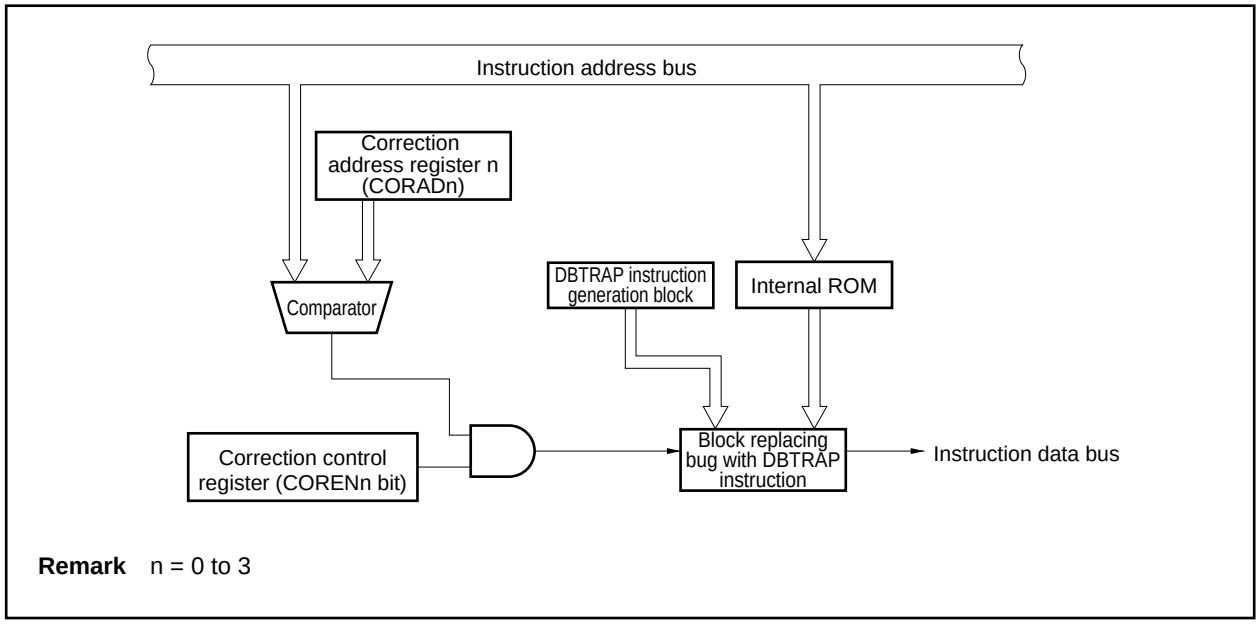
## 27.1 Overview

The ROM correction function is used to replace part of the program in the internal ROM with the program of an external memory or the internal RAM.

By using this function, program bugs found in the internal ROM can be corrected.

Up to four addresses can be specified for correction.

**Figure 27-1. Block Diagram of ROM Correction**




**Remark** n = 0 to 3

## 27.2 Control Registers

### 27.2.1 Correction address registers 0 to 3 (CORAD0 to CORAD3)

These registers are used to set the first address of the program to be corrected.

The program can be corrected at up to four places because four CORADn registers are provided.

The CORADn register can be read or written in 32-bit units. If the higher 16 bits of the CORADn register are used as the CORADnH register, and the lower 16 bits as the CORADnL register, these registers can be read or written in 16-bit units.

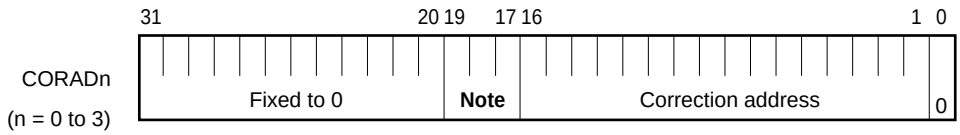
After reset, CORADn is cleared to 00000000H.

Set correction addresses in the following ranges.

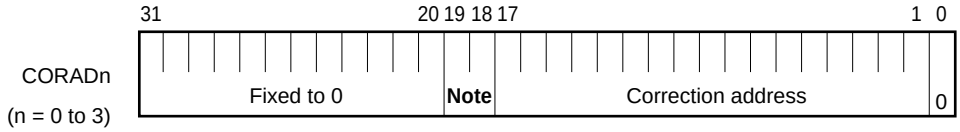
μPD70F3306, 70F3306Y (128 KB): 0000000H to 001FFFEH
μPD703308, 703308Y, 70F3308, 70F3308Y (256 KB): 0000000H to 003FFFEH

After reset: 00000000H R/W Address: Refer to **Table 27-1**

**(a) 128 KB**




**(b) 256 KB**




**Note** Be sure to clear these bits to 0.

**Table 27-1. CORADn Address**

| Address | Register Name | Address | Register Name |
|---|---|---|---|
| FFFFF840H | CORAD0 | FFFFF848H | CORAD2 |
| FFFFF840H | CORAD0L | FFFFF848H | CORAD2L |
| FFFFF842H | CORAD0H | FFFFF84AH | CORAD2H |
| FFFFF844H | CORAD1 | FFFFF84CH | CORAD3 |
| FFFFF844H | CORAD1L | FFFFF84CH | CORAD3L |
| FFFFF846H | CORAD1H | FFFFF84EH | CORAD3H |

### 27.2.2 Correction control register (CORCN)

This register disables or enables the correction operation at the address specified by the CORADn register.

Each channel can be enabled or disabled by this register.

This register can be read or written in 8-bit or 1-bit units.

After reset, CORCN is cleared to 00H.

After reset: 00H R/W Address: FFFFF880H

| | 7 | 6 | 5 | 4 | <3> | <2> | <1> | <0> |
|---|---|---|---|---|---|---|---|---|
| CORCN | 0 | 0 | 0 | 0 | COREN3 | COREN2 | COREN1 | COREN0 |

| CORENn | Correction operation enable/disable |
|---|---|
| 0 | Disabled |
| 1 | Enabled |

**Remark** n = 0 to 3

**Table 27-2. Correspondence Between CORCN Register Bits and CORADn Registers**

| CORCN Register Bit | Corresponding CORADn Register |
|---|---|
| COREN3 | CORAD3 |
| COREN2 | CORAD2 |
| COREN1 | CORAD1 |
| COREN0 | CORAD0 |

## 27.3 ROM Correction Operation and Program Flow

<1> If the address to be corrected and the fetch address of the internal ROM match, the fetch code is replaced by the DBTRAP instruction.

<2> When the DBTRAP instruction is executed, execution branches to address 00000060H.

<3> Software processing after branching causes the result of ROM correction to be judged (the fetch address and ROM correction operation are confirmed) and execution to branch to the correction software.

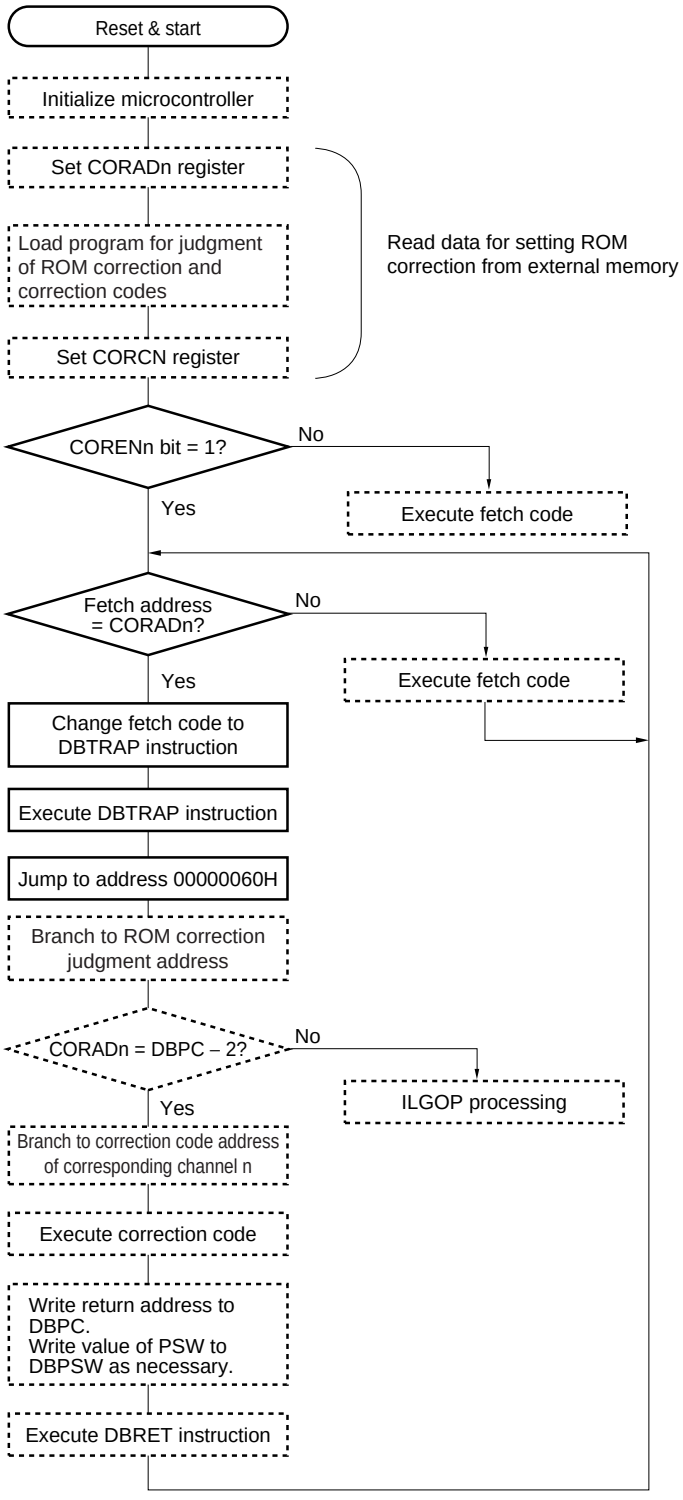
<4> After the correction software has been executed, the return address is set, and return processing is started by the DBRET instruction.

**Cautions 1. The software that performs <3> and <4> must be executed in the internal RAM.**

**2. When setting an address to be corrected to the CORADn register, clear the higher bits to 0 in accordance with the capacity of the internal ROM.**

**3. The ROM correction function cannot be used to correct the data of the internal ROM. It can only be used to correct instruction codes. If ROM correction is used to correct data, that data is replaced with the DBTRAP instruction code.**

**Figure 27-2. ROM Correction Operation and Program Flow**




**Remarks 1.** : Processing by user program (software)

: Processing by ROM correction (hardware)

**2.** n = 0 to 3

# CHAPTER 28 MASK OPTION/OPTION BYTE

## 28.1 Mask Option (Mask ROM Versions)

The mask ROM versions ($\mu$PD703308 and 703308Y) have the following mask options.

- Connection of pull-up resistor to P38 and P39 pins
- Enabling/disabling stopping Ring-OSC by software
- Shortening oscillation stabilization time of main clock oscillation after release of reset

### (1) Connection of pull-up resistor to P38 and P39 pins

| PUmn | Connection of Pull-up Resistor to Port mn |
|---|---|
| 0 | Not connected |
| 1 | Connected |

**Remark** mn = 38, 39

### (2) Enabling/disabling stopping Ring-OSC by software

| RINGSTP | Control of Stopping Ring-OSC by Software |
|---|---|
| 0 | Can be stopped by software |
| 1 | Setting invalid by software |

Depending on whether the option to enable/disable stopping of Ring-OSC by software is set or not, the operation differs as follows.

**Table 28-1. Option to Enable/Disable Stopping of Ring-OSC by Software**

| | | RINGSTP = 0 (Can Be Stopped) | RINGSTP = 1 (Setting Invalid) |
|---|---|---|---|
| Ring-OSC | | Ring-OSC: Can be stopped.<br>RCM.RSTOP bit can be set. | Ring-OSC: Cannot be stopped.<br>Setting of RSTOP bit is invalid. |
| WDT2 | Count operation | Operation can be stopped by WDTM2.WDCS24 bit. | Operation cannot be stopped. |
| | Input clock | The following clock can be selected by the WDTM2 register.<br>• Ring-OSC: $f_R$/8<br>• Subclock: $f_{XT}$ | Fixed to Ring-OSC ($f_R$/8) |
| | Operation mode | The following mode can be selected by the WDTM2 register.<br>• NMI interrupt mode (INTWDT2)<br>• Reset mode (WDTRES2) | Fixed to reset mode (WDTRES2) |

### (3) Shortening oscillation stabilization time of main clock oscillation after release of reset

| OSTS0 | Option to Shorten Oscillation Stabilization Time of Main Clock Oscillation After Release of Reset | | |
|---|---|---|---|
| | | (Default Value of OSTS Register) | Oscillation Stabilization Time |
| 0 | Shorten oscillation stabilization time. | 00H | $2^{13}/f_X$ |
| 1 | Do not shorten oscillation stabilization time. | 01H | $2^{15}/f_X$ |

## 28.2 Option Byte (Flash Memory Versions)

The flash memory versions ($\mu$PD70F3306, 70F3306Y, 70F3308, and 70F3308Y) can realize the mask options of the mask ROM version by using an option byte (except the pull-up resistor option).

The option byte is stored in address 000007AH of the internal flash memory (internal ROM area) as 8-bit data.

Address: 0000007AH

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| – | – | – | OSTS0 | – | – | – | RINGSTP |

| OSTS0[Note 1] | Option to shorten oscillation stabilization time of main clock oscillation after release of reset |
|---|---|
| 0 | Shorten oscillation stabilization time (default value of OSTS register = 00H) |
| 1 | Do not shorten oscillation stabilization time (default value of OSTS register = 01H) |

| RINGSTP[Note 2] | Option to enable/disable stopping Ring-OSC by software |
|---|---|
| 0 | Can be stopped by software |
| 1 | Cannot be stopped by software |

**Notes 1.** For details of the option, refer to **28.1 (3) Shortening oscillation stabilization time of main clock oscillation after release of reset**.

**2.** For details of the option, refer to **Table 28-1 Option to Enable/Disable Stopping of Ring-OSC by Software**.

# CHAPTER 29 FLASH MEMORY

The following products are the flash memory versions of the V850ES/KF1+.

**Caution There are differences in noise immunity and noise radiation between the flash memory and mask ROM versions. When pre-producing and application set with the flash memory version and then mass-producing it with the mask ROM version, be sure to conduct sufficient evaluation for the commercial samples (not engineering samples) of the mask ROM version.**
**For the electrical specifications related to the flash memory rewriting, refer to CHAPTER 30 ELECTRICAL SPECIFICATIONS (TARGET).**

- μPD70F3306, 70F3306Y: On-chip 128 KB flash memory
- μPD70F3308, 70F3308Y: On-chip 256 KB flash memory

Flash memory versions are commonly used in the following development environments and mass production applications.

- For altering software after the V850ES/KF1+ is soldered onto the target system.
- For data adjustment when starting mass production.
- For differentiating software according to the specification in small scale production of various models.
- For facilitating inventory management.
- For updating software after shipment.

## 29.1 Features

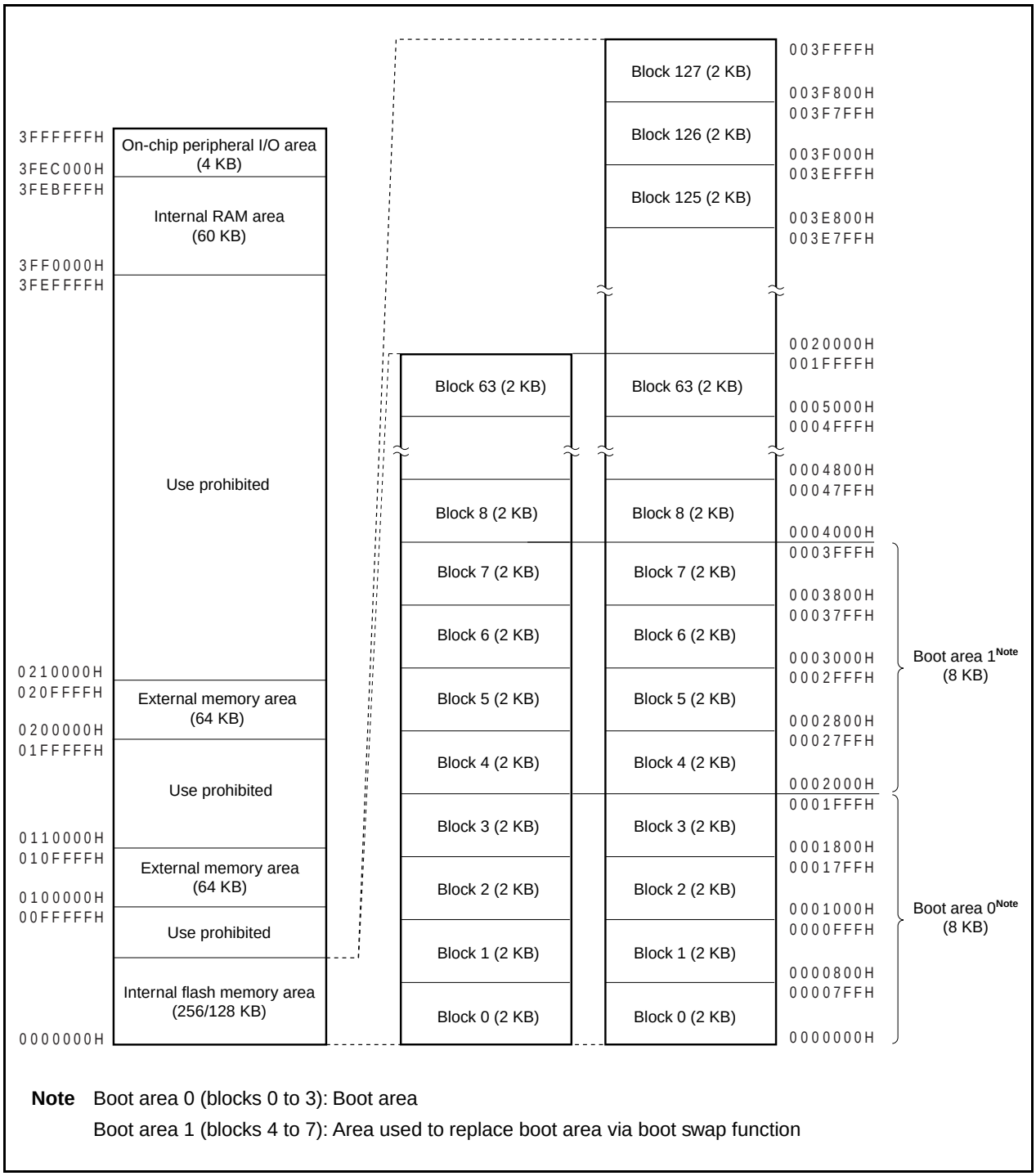
- 4-byte/1-clock access (when instruction is fetched)
- Capacity: 128/256 KB
- Write voltage: Erase/write with a single power supply
- Rewriting method
  - Rewriting by communication with dedicated flash programmer via serial interface (on-board/off-board programming)
  - Rewriting flash memory by user program (self programming)
- Flash memory write prohibit function supported (security function)
- Safe rewriting of entire flash memory area by self programming using boot swap function
- Interrupts can be acknowledged during self programming.

## 29.2 Memory Configuration

The 128/256 KB internal flash memory area is divided into 64/128 blocks and can be programmed/erased in block units. All the blocks can also be erased at once.

When the boot swap function is used, the physical memory (blocks 0 to 3) located at the addresses of boot area 0 is replaced by the physical memory (blocks 4 to 7) located at the addresses of boot area 1. For details of the boot swap function, refer to **29.5 Rewriting by Self Programming**.

**Figure 29-1. Flash Memory Mapping**




**Note** Boot area 0 (blocks 0 to 3): Boot area
Boot area 1 (blocks 4 to 7): Area used to replace boot area via boot swap function

## 29.3 Functional Outline

The internal flash memory of the V850ES/KF1+ can be rewritten by using the rewrite function of the dedicated flash programmer, regardless of whether the V850ES/KF1+ has already been mounted on the target system or not (on-board/off-board programming).

In addition, a security function that prohibits rewriting the user program written to the internal flash memory is also supported, so that the program cannot be changed by an unauthorized person.

The rewrite function using the user program (self programming) is ideal for an application where it is assumed that the program is changed after production/shipment of the target system. A boot swap function that rewrites the entire flash memory area safely is also supported. In addition, interrupt servicing is supported during self programming, so that the flash memory can be rewritten under various conditions, such as while communicating with an external device.

**Table 29-1. Rewrite Method**

| Rewrite Method | Functional Outline | Operation Mode |
|---|---|---|
| On-board programming | Flash memory can be rewritten after the device is mounted on the target system, by using a dedicated flash programmer. | Flash memory programming mode |
| Off-board programming | Flash memory can be rewritten before the device is mounted on the target system, by using a dedicated flash programmer and a dedicated program adapter board (FA series). | |
| Self programming | Flash memory can be rewritten by executing a user program that has been written to the flash memory in advance by means of on-board/off-board programming. (During self-programming, instructions cannot be fetched from or data access cannot be made to the internal flash memory area. Therefore, the rewrite program must be transferred to the internal RAM or external memory in advance). | Normal operation mode |

**Remark** The FA series is a product of Naito Densei Machida Mfg. Co., Ltd.

**Table 29-2. Basic Functions**

| Function | Functional Outline | Support (○: Supported, ×: Not supported) | |
|---|---|---|---|
| | | On-Board/Off-Board Programming | Self Programming |
| Block erasure | The contents of specified memory blocks are erased. | ○ | ○ |
| Chip erasure | The contents of the entire memory area are erased all at once. | ○ | × |
| Write | Writing to specified addresses, and a verify check to see if write level is secured are performed. | ○ | ○ |
| Verify/checksum | Data read from the flash memory is compared with data transferred from the flash programmer. | ○ | × (Can be read by user program) |
| Blank check | The erasure status of the entire memory is checked. | ○ | ○ |
| Security setting | Use of the block erase command, chip erase command, and program command can be prohibited. | ○ | × (Only values set by on-board/off-board programming can be retained) |

The following table lists the security functions. The block erase command prohibit, chip erase command prohibit, and program command prohibit functions are enabled by default after shipment, and security can be set by rewriting via on-board/off-board programming. Each security function can be used in combination with the others at the same time.

**Table 29-3. Security Functions**

| Function | Functional Outline | Rewriting Operation When Prohibited (○: Executable, ×: Not Executable) | |
|---|---|---|---|
| | | On-Board/Off-Board Programming | Self Programming |
| Block erase command prohibit | Execution of a block erase command on all blocks is prohibited. Setting of prohibition can be initialized by execution of a chip erase command. | Block erase command: × Chip erase command: ○ Program command: ○ | Can always be rewritten regardless of setting of prohibition |
| Chip erase command prohibit | Execution of block erase and chip erase commands on all the blocks is prohibited. Once prohibition is set, setting of prohibition cannot be initialized because the chip erase command cannot be executed. | Block erase command: × Chip erase command: × Program command: ○ | |
| Program command prohibit | Write and block erase commands on all the blocks are prohibited. Setting of prohibition can be initialized by execution of the chip erase command. | Block erase command: × Chip erase command: ○ Program command: × | |

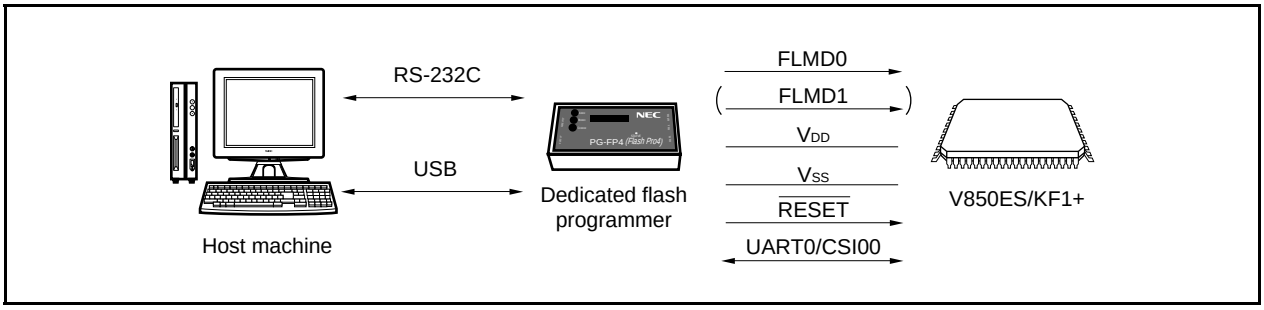
## 29.4 Rewriting by Dedicated Flash Programmer

The flash memory can be rewritten by using a dedicated flash programmer after the V850ES/KF1+ is mounted on the target system (on-board programming). The flash memory can also be rewritten before the device is mounted on the target system (off-board programming) by using a dedicated program adapter (FA series).

### 29.4.1 Programming environment

The following shows the environment required for writing programs to the flash memory of the V850ES/KF1+.

**Figure 29-2. Environment Required for Writing Programs to Flash Memory**




A host machine is required for controlling the dedicated flash programmer.

UART0 or CSI00 is used for the interface between the dedicated flash programmer and the V850ES/KF1+ to perform writing, erasing, etc. A dedicated program adapter (FA series) is required for off-board writing.

**Remark** The FA series is a product of Naito Densei Machida Mfg. Co., Ltd.

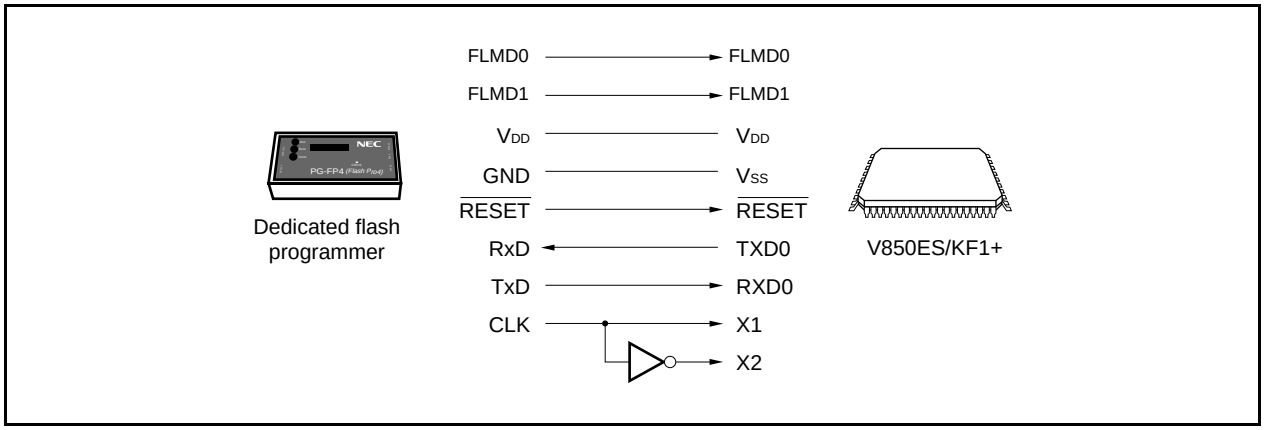
### 29.4.2 Communication mode

Communication between the dedicated flash programmer and the V850ES/KF1+ is performed by serial communication using the UART0 or CSI00 interfaces of the V850ES/KF1+.

**(1) UART0**

Transfer rate: 9,600 to 153,600 bps

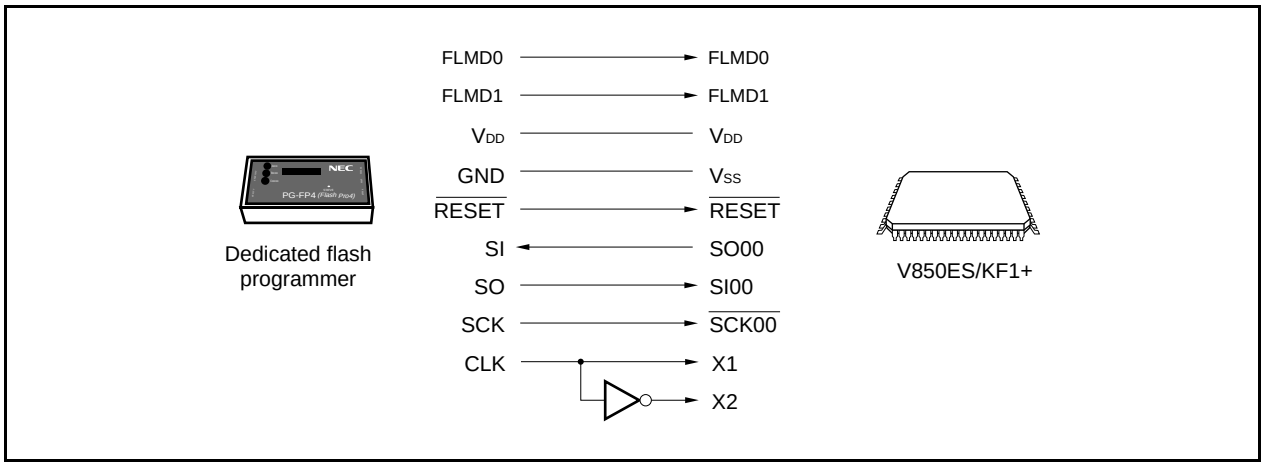
**Figure 29-3. Communication with Dedicated Flash Programmer (UART0)**




**(2) CSI00**

Serial clock: 2.4 kHz to 2.5 MHz (MSB first)

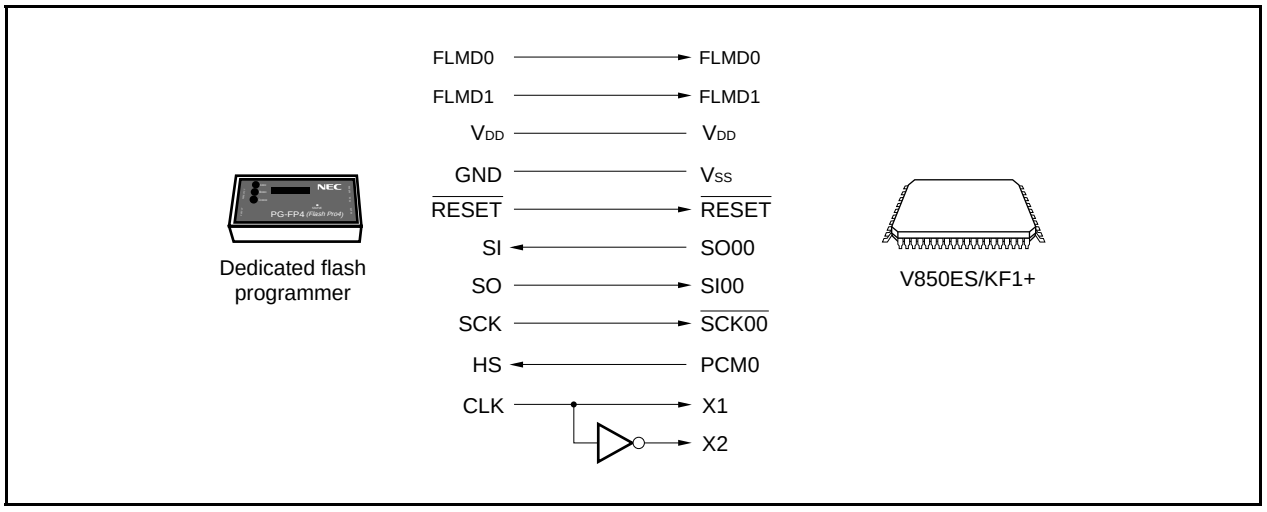
**Figure 29-4. Communication with Dedicated Flash Programmer (CSI00)**



**(3) CSI00 + HS**

Serial clock: 2.4 kHz to 2.5 MHz (MSB first)

**Figure 29-5. Communication with Dedicated Flash Programmer (CSI00 + HS)**




The dedicated flash programmer outputs the transfer clock, and the V850ES/KF1+ operates as a slave.

When the PG-FP4 is used as the dedicated flash programmer, it generates the following signals to the V850ES/KF1+. For details, refer to the **PG-FP4 User's Manual (U15260E)**.

**Table 29-4. Signal Connections of Dedicated Flash Programmer (PG-FP4)**

| PG-FP4 | | | V850ES/KF1+ | Processing for Connection | | |
|---|---|---|---|---|---|---|
| Signal Name | I/O | Pin Function | Pin Name | UART0 | CSI00 | CSI00 + HS |
| FLMD0 | Output | Write enable/disable | FLMD0 | ◎ | ◎ | ◎ |
| FLMD1 | Output | Write enable/disable | FLMD1 | ◎Note 1 | ◎Note 1 | ◎Note 1 |
| VDD | – | $V_{DD}$ voltage generation/voltage monitor | $V_{DD}$ | ◎ | ◎ | ◎ |
| GND | – | Ground | $V_{SS}$ | ◎ | ◎ | ◎ |
| CLK | Output | Clock output to V850ES/KF1+ | X1, X2 | ×Note 2 | ×Note 2 | ×Note 2 |
| $\overline{\text{RESET}}$ | Output | Reset signal | $\overline{\text{RESET}}$ | ◎ | ◎ | ◎ |
| SI/RxD | Input | Receive signal | SO00, TXD0 | ◎ | ◎ | ◎ |
| SO/TxD | Output | Transmit signal | SI00, RXD0 | ◎ | ◎ | ◎ |
| SCK | Output | Transfer clock | $\overline{\text{SCK00}}$ | × | ◎ | ◎ |
| HS | Input | Handshake signal for CSI00 + HS communication | PCM0 | × | × | ◎ |

**Notes 1.** Wire the pin as shown in Figure 29-6, or connect it to GND on board via a pull-down resistor.

**2.** Connect these pins to supply a clock from the PG-FP4 (wire as shown in Figure 29-6, or create an oscillator on board and supply the clock).

**Remark** ◎: Must be connected.

×: Does not have to be connected.

**Table 29-5. Wiring Between μPD70F3306, 70F3306Y, 70F3308, and 70F3308Y, and PG-FP4**

| Pin Configuration of Flash Programmer (PG-FP4) | | | Pin Name on FA Board | With CSI00-HS | | With CSI00 | | With UART0 | |
|---|---|---|---|---|---|---|---|---|---|
| Signal Name | I/O | Pin Function | | Pin Name | Pin No. | Pin Name | Pin No. | Pin Name | Pin No. |
| SI/RxD | Input | Receive signal | SI | P41/SO00 | 20 | P41/SO00 | 20 | P30/TXD0 | 22 |
| SO/TxD | Output | Transmit signal | SO | P40/SI00 | 19 | P40/SI00 | 19 | P31/RXD0/INTP7 | 23 |
| SCK | Output | Transfer clock | SCK | P42/$\overline{SCK00}$ | 21 | P42/$\overline{SCK00}$ | 21 | Not needed | Not needed |
| CLK | Output | Clock to V850ES/KF1+ | X1 | X1 | 12 | X1 | 12 | X1 | 12 |
| | | | X2 | X2[Note] | 13 | X2[Note] | 13 | X2[Note] | 13 |
| /RESET | Output | Reset signal | /RESET | $\overline{RESET}$ | 14 | $\overline{RESET}$ | 14 | $\overline{RESET}$ | 14 |
| FLMD0 | Input | Write voltage | FLMD0 | FLMD0 | 8 | FLMD0 | 8 | FLMD0 | 8 |
| FLMD1 | Input | Write voltage | FLMD1 | PDL5/AD5/FLMD1 | 62 | PDL5/AD5/FLMD1 | 62 | PDL5/AD5/FLMD1 | 62 |
| HS | Input | Handshake signal for CSI00 + HS communication | RESERVE/HS | PCM0/$\overline{WAIT}$ | 49 | Not needed | Not needed | Not needed | Not needed |
| VDD | – | VDD voltage generation/voltage monitor | VDD | VDD | 9 | VDD | 9 | VDD | 9 |
| | | | | EVDD | 31 | EVDD | 31 | EVDD | 31 |
| | | | | AVREF0 | 1 | AVREF0 | 1 | AVREF0 | 1 |
| GND | – | Ground | GND | VSS | 11 | VSS | 11 | VSS | 11 |
| | | | | AVSS | 2 | AVSS | 2 | AVSS | 2 |
| | | | | EVSS | 30 | EVSS | 30 | EVSS | 30 |

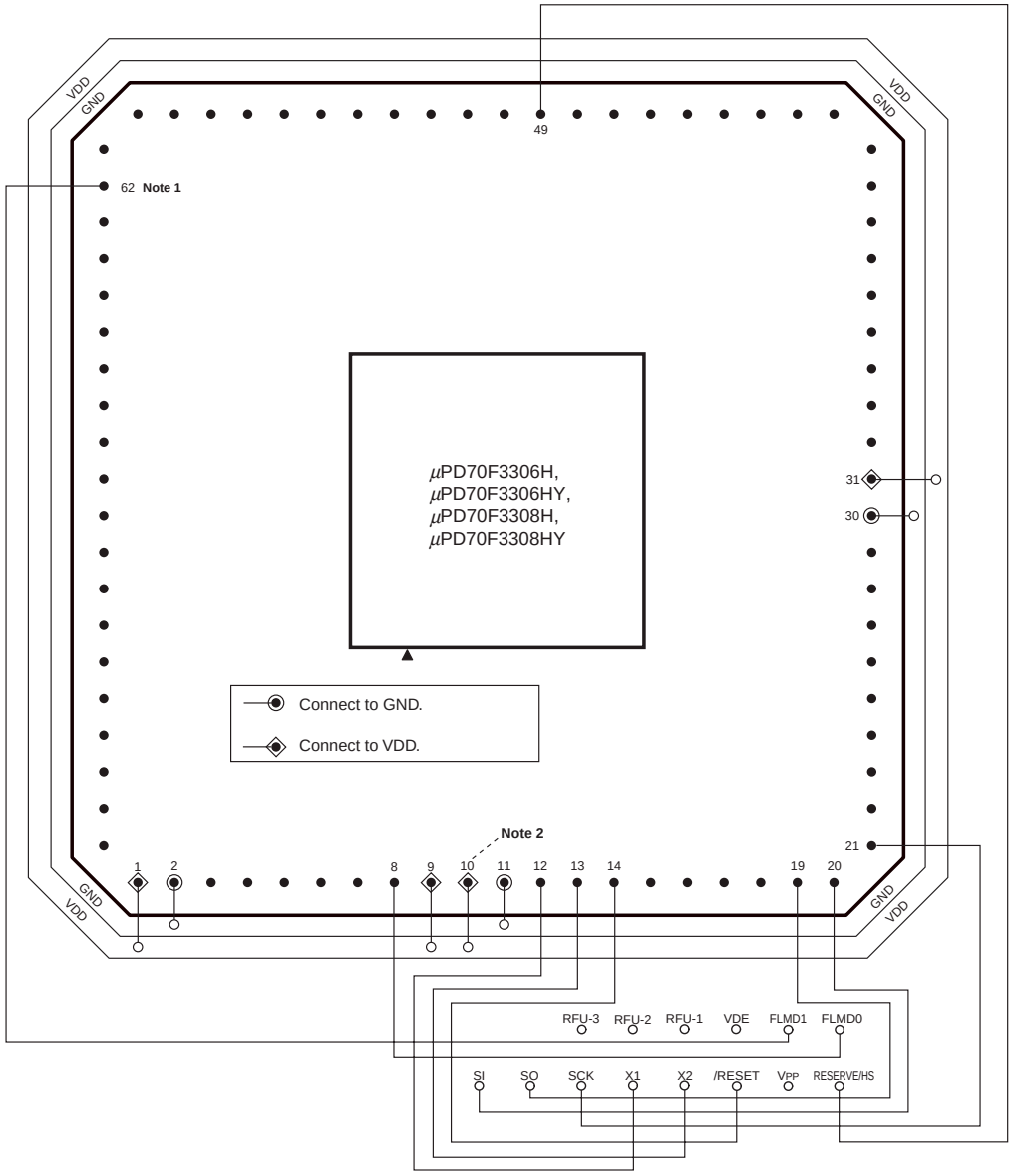
**Note** When using the clock out of the flash programmer, connect CLK of the programmer to X1, and connect its inverse signal to X2.

**Cautions 1. Be sure to connect the REGC pin in either of the following ways.**

- **Connect to GND via a 10 μF capacitor**
- **Directly connect to VDD**

**2. When connecting the REGC pin to GND via a 10 μF capacitor, the clock cannot be supplied from the CLK pin of the flash programmer.**
**Supply the clock by creating an oscillator on the board.**

**Figure 29-6. Wiring Example of V850ES/KF1+ Flash Writing Adapter (FA-80GC-8BT, FA-80GK-9EU)**




**Notes 1.** Wire the FLMD1 pin as shown in the figure, or connect it to GND on board via a pull-down resistor.

**2.** Be sure to connect the REGC pin in either of the following ways.

- Connect to GND via a 10 μF capacitor.
- Directly connect to $V_{DD}$.

When connecting the REGC pin to GND via a 10 μF capacitor, the clock cannot be supplied from the CLK pin of the flash programmer.

Supply the clock by creating an oscillator on the board.

**Remarks 1.** Handle the pins not described above in accordance with the specified handling of unused pins (refer to **2.3 Pin I/O Circuits and Recommended Connection of Unused Pins)**.

When connecting to $V_{DD}$ via a resistor, use of a resistor of 1 kΩ to 10 kΩ is recommended.

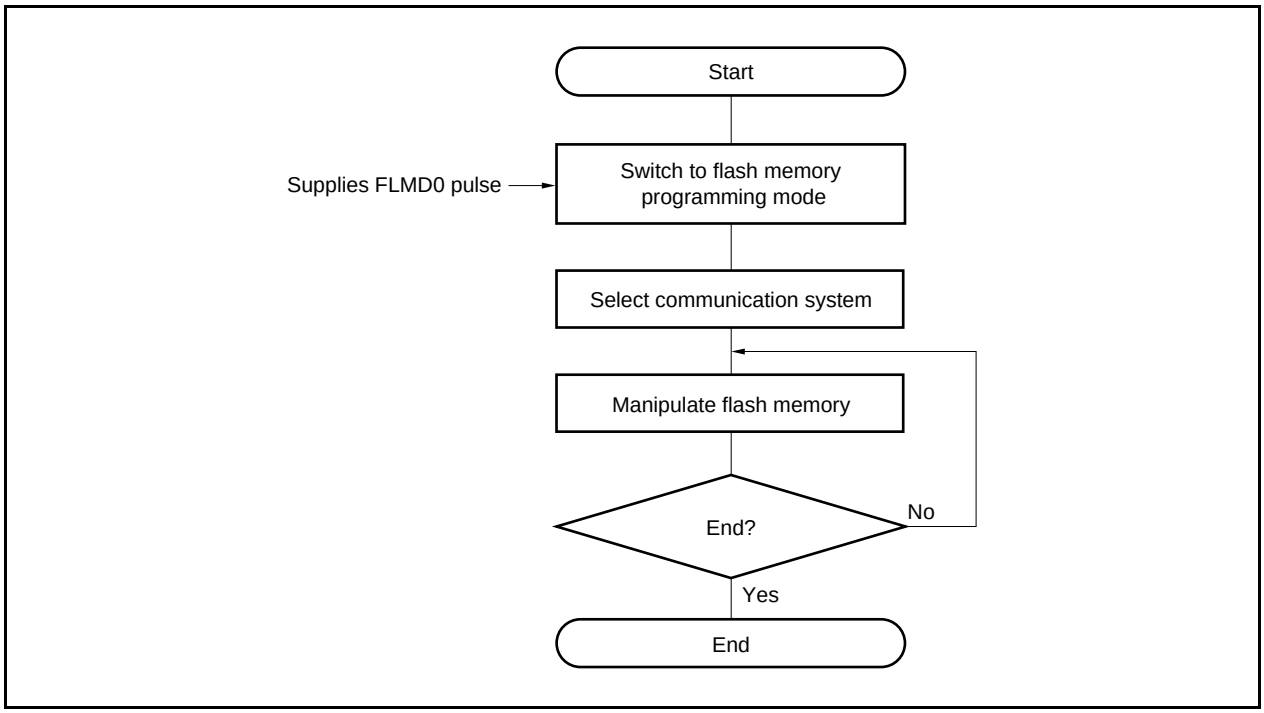
**2.** This adapter is for an 80-pin plastic TQFP (fine pitch) or 80-pin plastic QFP package.

**3.** This diagram shows the wiring when using a handshake-supporting CSI.

### 29.4.3 Flash memory control

The following shows the procedure for manipulating the flash memory.

**Figure 29-7. Procedure for Manipulating Flash Memory**

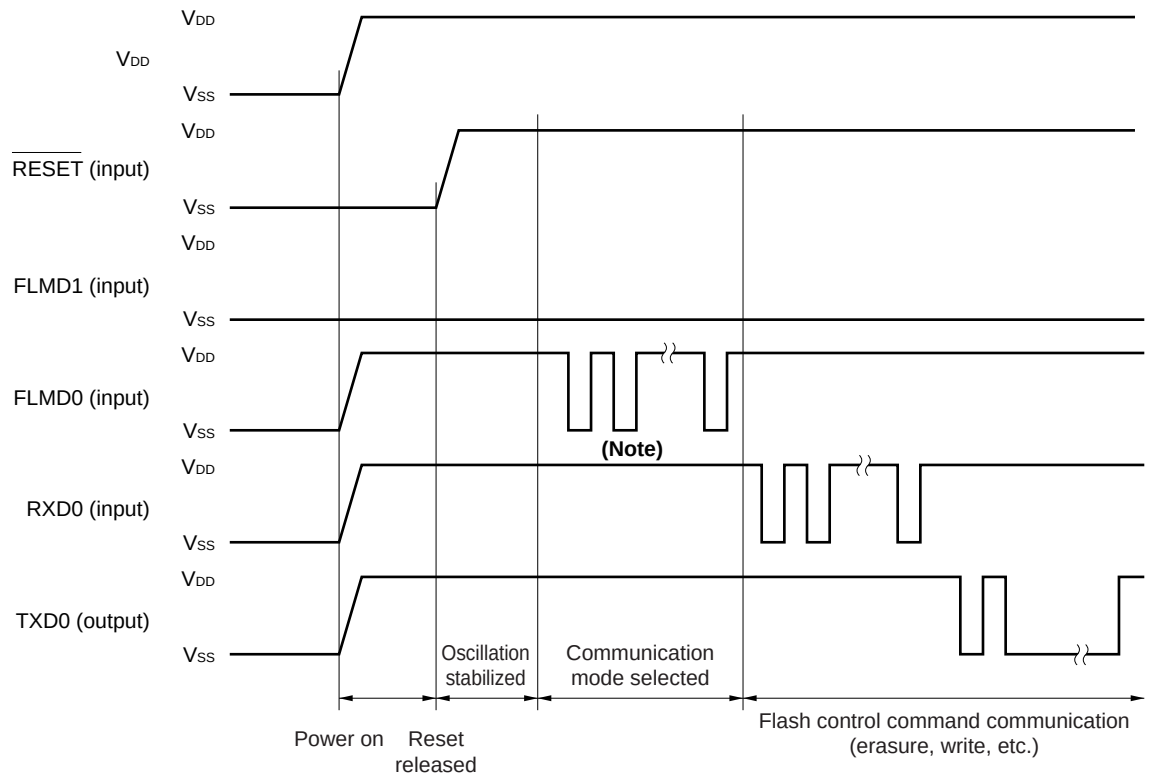


### 29.4.4 Selection of communication mode

In the V850ES/KF1+, the communication mode is selected by inputting pulses (12 pulses max.) to the FLMD0 pin after switching to the flash memory programming mode. The FLMD0 pulse is generated by the dedicated flash programmer.

The following shows the relationship between the number of pulses and the communication mode.

**Figure 29-8. Selection of Communication Mode**




**Note** The number of clocks is as follows depending on the communication mode.

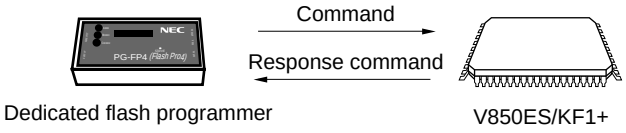
| FLMD0 Pulse | Communication Mode | Remarks |
|---|---|---|
| 0 | UART0 | Communication rate: 9600 bps (after reset), LSB first |
| 8 | CSI00 | V850ES/KF1+ performs slave operation, MSB first |
| 11 | CSI00 + HS | V850ES/KF1+ performs slave operation, MSB first |
| Other | RFU | Setting prohibited |

**Caution When UART0 is selected, the receive clock is calculated based on the reset command sent from the dedicated flash programmer after receiving the FLMD0 pulse.**

### 29.4.5 Communication commands

The V850ES/KF1+ communicates with the dedicated flash programmer by means of commands. The signals sent from the dedicated flash programmer to the V850ES/KF1+ are called “commands”. The response signals sent from the V850ES/KF1+ to the dedicated flash programmer are called “response commands”.

**Figure 29-9. Communication Commands**




The following shows the commands for flash memory control in the V850ES/KF1+. All of these commands are issued from the dedicated flash programmer, and the V850ES/KF1+ performs the processing corresponding to the commands.

**Table 29-6. Flash Memory Control Commands**

| Classification | Command Name | Support | | | Function |
|---|---|---|---|---|---|
| | | CSI00 | CSI00 + HS | UART0 | |
| Blank check | Block blank check command | ○ | ○ | ○ | Checks if the contents of the memory in the specified block have been correctly erased. |
| Erase | Chip erase command | ○ | ○ | ○ | Erases the contents of the entire memory. |
| | Block erase command | ○ | ○ | ○ | Erases the contents of the memory of the specified block. |
| Write | Write command | ○ | ○ | ○ | Writes the specified address range, and executes a contents verify check. |
| Verify | Verify command | ○ | ○ | ○ | Compares the contents of memory in the specified address range with data transferred from the flash programmer. |
| | Checksum command | ○ | ○ | ○ | Reads the checksum in the specified address range. |
| System setting, control | Silicon signature command | ○ | ○ | ○ | Reads silicon signature information. |
| | Security setting command | ○ | ○ | ○ | Disables the chip erase command, enables the block erase command, and disables the write command. |

### 29.4.6 Pin connection

When performing on-board writing, mount a connector on the target system to connect to the dedicated flash programmer. Also, incorporate a function on-board to switch from the normal operation mode to the flash memory programming mode.

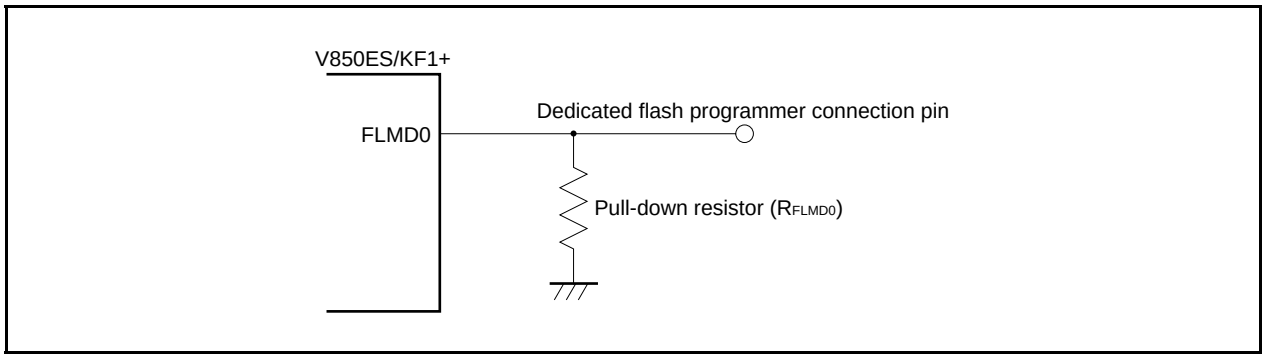
In the flash memory programming mode, all the pins not used for flash memory programming become the same status as that immediately after reset. Therefore, pin handling is required when the external device does not acknowledge the status immediately after a reset.

**(1) FLMD0 pin**

In the normal operation mode, input a voltage of $V_{SS}$ level to the FLMD0 pin. In the flash memory programming mode, supply a write voltage of $V_{DD}$ level to the FLMD0 pin.

Because the FLMD0 pin serves as a write protection pin in the self programming mode, a voltage of $V_{DD}$ level must be supplied to the FLMD0 pin via port control, etc., before writing to the flash memory. For details, refer to **29.5.5 (1) FLMD0 pin**.

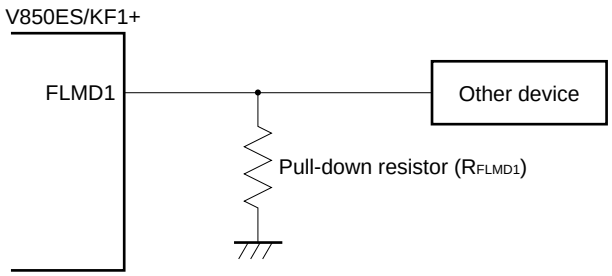
**Figure 29-10. FLMD0 Pin Connection Example**



**(2) FLMD1 pin**

When 0 V is input to the FLMD0 pin, the FLMD1 pin does not function. When $V_{DD}$ is supplied to the FLMD0 pin, the flash memory programming mode is entered, so 0 V must be input to the FLMD1 pin. The following shows an example of the connection of the FLMD1 pin.

**Figure 29-11. FLMD1 Pin Connection Example**




**Caution If the $V_{DD}$ signal is input to the FLMD1 pin from another device during on-board writing and immediately after reset, isolate this signal.**

**Table 29-7. Relationship Between FLMD0 and FLMD1 Pins and Operation Mode When Reset Is Released**

| FLMD0 | FLMD1 | Operation Mode |
|---|---|---|
| 0 | don't care | Normal operation mode |
| $V_{DD}$ | 0 | Flash memory programming mode |
| $V_{DD}$ | $V_{DD}$ | Setting prohibited |

**(3) Serial interface pin**

The following shows the pins used by each serial interface.

**Table 29-8. Pins Used by Serial Interfaces**

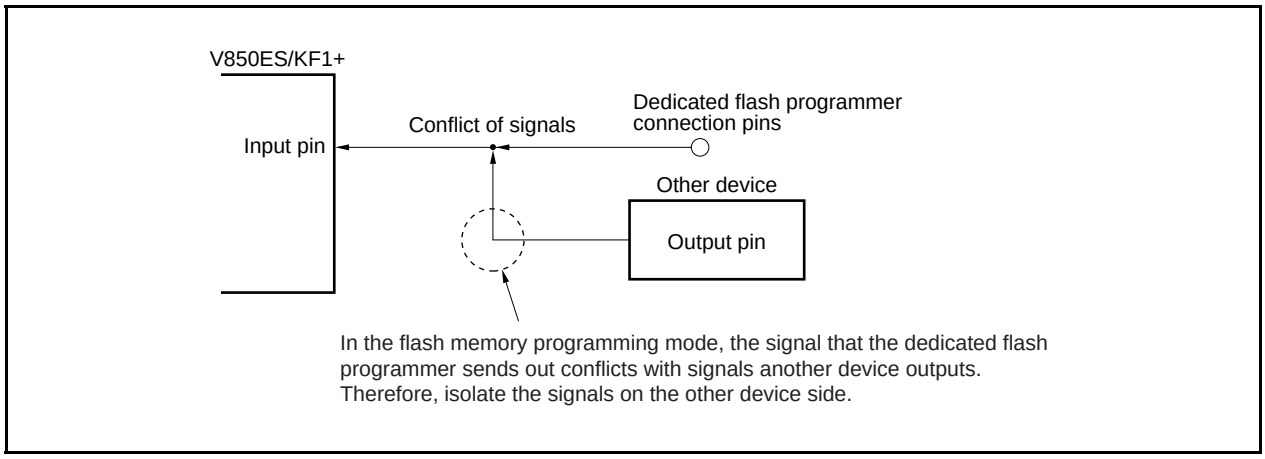
| Serial Interface | Pins Used |
|---|---|
| UART0 | TXD0, RXD0 |
| CSI00 | SO00, SI00, $\overline{\text{SCK00}}$ |
| CSI00 + HS | SO00, SI00, $\overline{\text{SCK00}}$, PCM0 |

When connecting a dedicated flash programmer to a serial interface pin that is connected to another device on-board, care should be taken to avoid conflict of signals and malfunction of the other device.

**(a) Conflict of signals**

When the dedicated flash programmer (output) is connected to a serial interface pin (input) that is connected to another device (output), a conflict of signals occurs. To avoid the conflict of signals, isolate the connection to the other device or set the other device to the output high-impedance status.

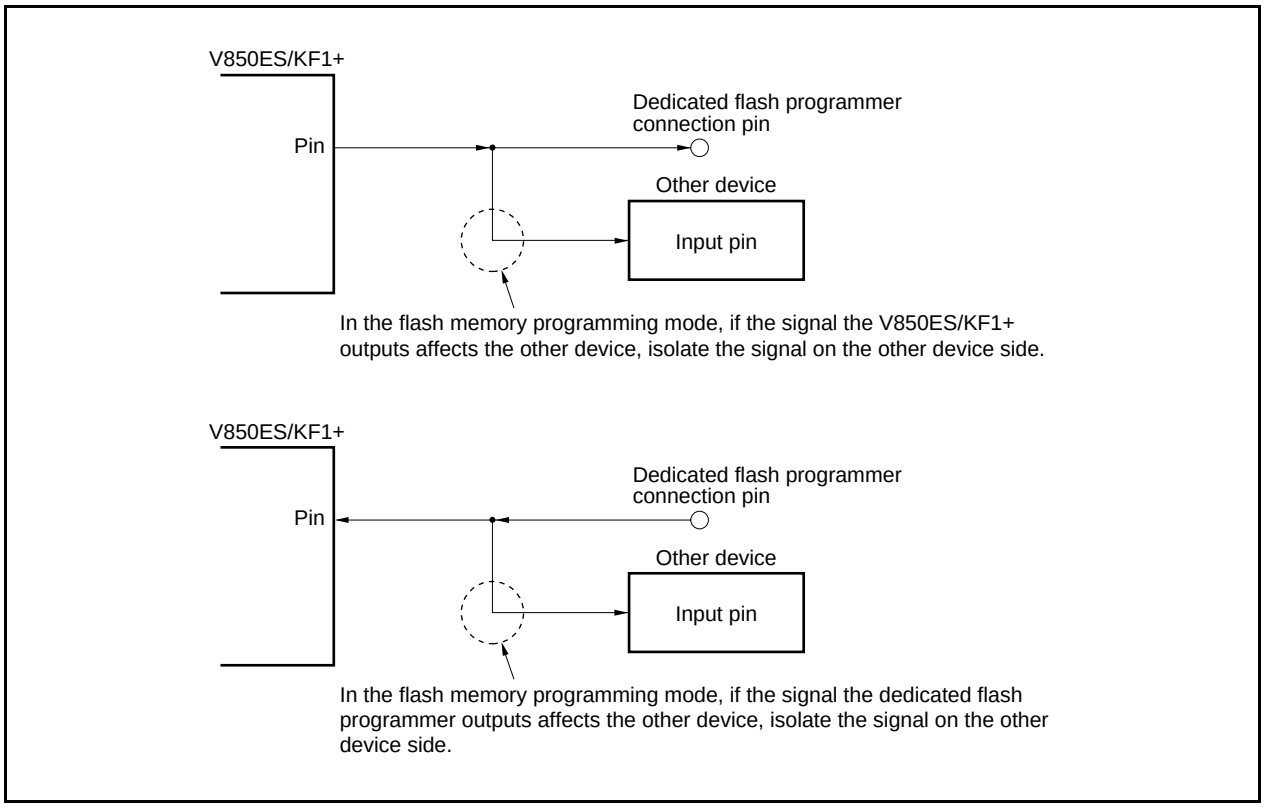
**Figure 29-12. Conflict of Signals (Serial Interface Input Pin)**



**(b) Malfunction of other device**

When the dedicated flash programmer (output or input) is connected to a serial interface pin (input or output) that is connected to another device (input), the signal is output to the other device, causing the device to malfunction. To avoid this, isolate the connection to the other device.

**Figure 29-13. Malfunction of Other Device**

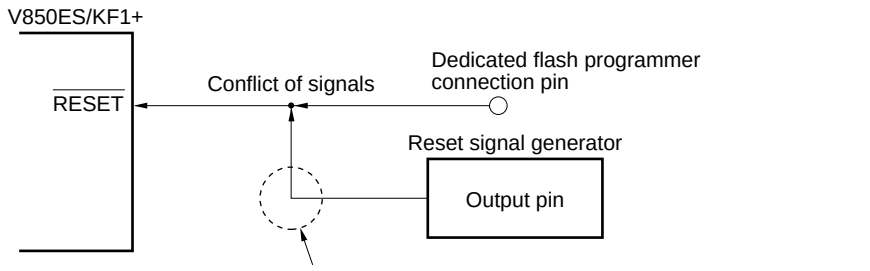


**(4) $\overline{\text{RESET}}$ pin**

When the reset signals of the dedicated flash programmer are connected to the $\overline{\text{RESET}}$ pin that is connected to the reset signal generator on-board, a conflict of signals occurs. To avoid the conflict of signals, isolate the connection to the reset signal generator.

When a reset signal is input from the user system in the flash memory programming mode, the programming operation will not be performed correctly. Therefore, do not input signals other than the reset signals from the dedicated flash programmer.

**Figure 29-14. Conflict of Signals ($\overline{\text{RESET}}$ Pin)**




In the flash memory programming mode, the signal the reset signal generator outputs conflicts with the signal the dedicated flash programmer outputs. Therefore, isolate the signals on the reset signal generator side.

**(5) Port pins (including NMI)**

When the system shifts to the flash memory programming mode, all the pins that are not used for flash memory programming are in the same status as that immediately after reset. If the external device connected to each port does not recognize the status of the port immediately after reset, pins require appropriate processing, such as connecting to $V_{DD}$ via a resistor or connecting to $V_{SS}$ via a resistor.

**(6) Other signal pins**

Connect X1, X2, XT1, XT2, and REGC in the same status as that in the normal operation mode.

**(7) Power supply**

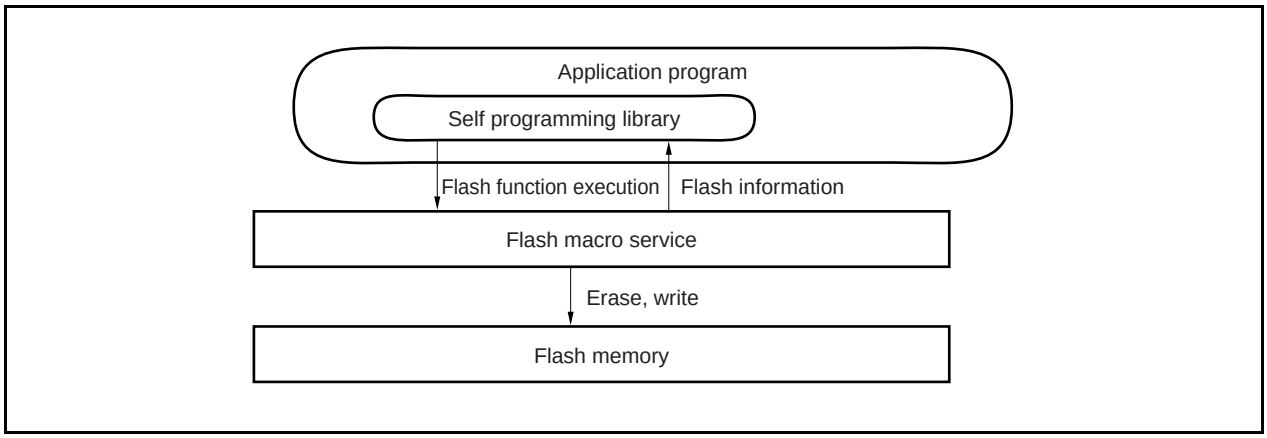
Supply the same power ($V_{DD}$, $V_{SS}$, $EV_{DD}$, $EV_{SS}$, $AV_{SS}$, $AV_{REF0}$) as in normal operation mode.

## 29.5 Rewriting by Self Programming

### 29.5.1 Overview

The V850ES/KF1+ supports a flash macro service that allows the user program to rewrite the internal flash memory by itself. By using this interface and a self programming library that is used to rewrite the flash memory with a user application program, the flash memory can be rewritten by a user application transferred in advance to the internal RAM or external memory. Consequently, the user program can be upgraded and constant data can be rewritten in the field.

**Figure 29-15. Concept of Self Programming**

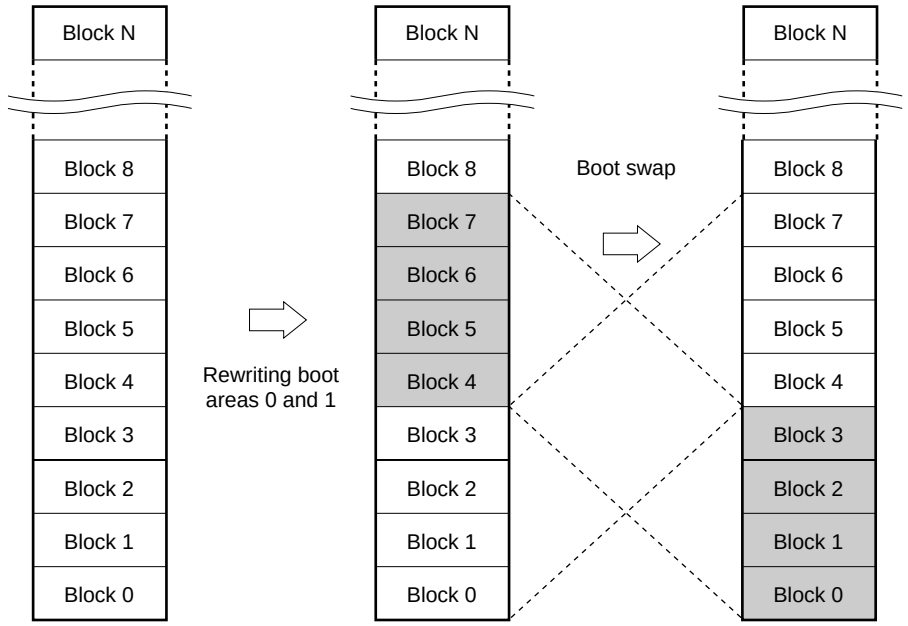


### 29.5.2 Features

**(1) Secure self programming (boot swap function)**

The V850ES/KF1+ supports a boot swap function that can exchange the physical memory (blocks 0 to 3) of boot area 0 with the physical memory (blocks 4 to 7) of boot area 1. By writing the start program to be rewritten to boot area 1 in advance and then swapping the physical memory, the entire area can be safely rewritten even if a power failure occurs during rewriting because the correct user program always exists in boot area 0.

**Figure 29-16. Rewriting Entire Memory Area (Boot Swap)**




**Remark** $\mu$PD70F3306, 70F3306Y: N = 63
$\mu$PD70F3308, 70F3308Y: N = 127

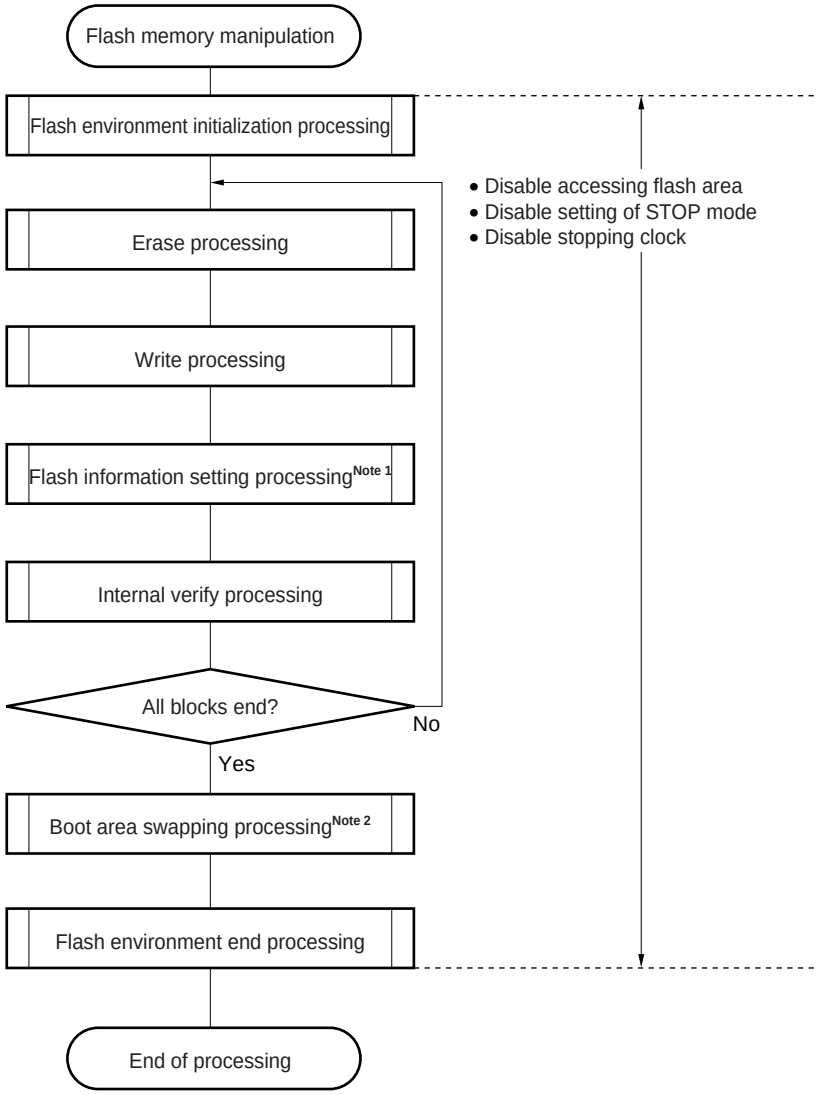
**(2) Interrupt support**

Instructions cannot be fetched from the flash memory during self programming. Conventionally, therefore, a user handler written to the flash memory could not be used even if an interrupt occurred. With the V850ES/KF1+, a user handler can be registered to an entry RAM area by using a library function, so that interrupt servicing can be performed by internal RAM or external memory execution.

### 29.5.3 Standard self programming flow

The entire processing to rewrite the flash memory by flash self programming is illustrated below.

**Figure 29-17. Standard Self Programming Flow**




**Notes 1.** If a security setting is not performed, flash information setting processing does not have to be executed.

**2.** If boot swap is not used, flash information setting processing and boot area swap processing do not have to be executed.

### 29.5.4 Flash functions

**Table 29-9. Flash Function List**

| Function Name | Outline | Support |
|---|---|---|
| FlashEnv | Initialization of flash control macro | √ |
| FlashBlockErase | Erasure of only specified one block | √ |
| FlashWordRead | Reading data from specified address | √ |
| FlashWordWrite | Writing from specified address | √ |
| FlashBlockIVerify | Internal verification of specified block | √ |
| FlashBlockBlankCheck | Blank check of specified block | √ |
| FlashFLMDCheck | Check of FLMD pin | √ |
| FlashGetInfo | Reading of flash information | √ |
| FlashSetInfo | Setting of flash information | √ |
| FlashBootSwap | Swapping of boot area | √ |

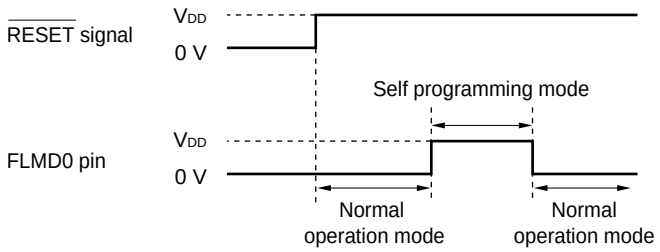
### 29.5.5 Pin processing

**(1) FLMD0 pin**

The FLMD0 pin is used to set the operation mode when reset is released and to protect the flash memory from being written during self rewriting. It is therefore necessary to keep the voltage applied to the FLMD0 pin at 0 V when reset is released and a normal operation is executed. It is also necessary to apply a voltage of $V_{DD}$ level to the FLMD0 pin during the self programming mode period via port control before the memory is rewritten.

When self programming has been completed, the voltage on the FLMD0 pin must be returned to 0 V.

**Figure 29-18. Mode Change Timing**




**Caution Make sure that the FLMD0 pin is at 0 V when reset is released.**

### 29.5.6 Internal resources used

The following table lists the internal resources used for self programming. These internal resources can also be used freely for purposes other than self programming.

**Table 29-10. Internal Resources Used**

| Resource Name | Description |
|---|---|
| Entry RAM area (internal RAM/external RAM size[Note]) | Routines and parameters used for the flash macro service are located in this area. The entry program and default parameters are copied by calling a library initialization function. |
| Stack area (stack size[Note]) | An extension of the stack used by the user is used by the library (can be used in both the internal RAM and external RAM). |
| Library code (code size[Note]) | Program entity of library (can be used anywhere other than the flash memory block to be manipulated). |
| Application program | Executed as user application.<br>Calls flash functions. |
| Maskable interrupt | Can be used in user application execution status or self programming status. To use this interrupt in the self programming status, the interrupt servicing start address must be registered in advance by a registration function. |
| NMI interrupt | Can be used in user application execution status or self programming status. To use this interrupt in the self programming status, the interrupt servicing start address must be registered in advance by a registration function. |
| TM50, TM51 | Because TM50 and TM51 are used in the flash macro service, do not use them in the self programming status.<br>When using TM50 and TM51 after self programming, set them again. |

**Note** For the capacity to be used, refer to the **V850 Series Flash Memory Self Programming (Single Power Supply Flash Memory) User's Manual** (under preparation).

**Absolute Maximum Ratings ($T_A$ = 25°C) (1/2)**

| Parameter | Symbol | Conditions | Ratings | Unit |
|---|---|---|---|---|
| Supply voltage | $V_{DD}$ | $V_{DD} = EV_{DD} = AV_{REF0}$ | −0.3 to +6.5 | V |
| | $AV_{REF0}$ | $V_{DD} = EV_{DD} = AV_{REF0}$ | −0.3 to +6.5 | V |
| | $EV_{DD}$ | $V_{DD} = EV_{DD} = AV_{REF0}$ | −0.3 to +6.5 | V |
| | $V_{SS}$ | $V_{SS} = EV_{SS} = AV_{SS}$ | −0.3 to +0.3 | V |
| | $AV_{SS}$ | $V_{SS} = EV_{SS} = AV_{SS}$ | −0.3 to +0.3 | V |
| | $EV_{SS}$ | $V_{SS} = EV_{SS} = AV_{SS}$ | −0.3 to +0.3 | V |
| Input voltage | $V_{I1}$ | P00 to P06, P30 to P35, P38, P39, P40 to P42, P50 to P55, P90, P91, P96 to P99, P913 to P915, PCM0 to PCM3, PCS0, PCS1, PCT0, PCT1, PCT4, PCT6, PDL0 to PDL15, $\overline{\text{RESET}}$, FLMD0 | −0.3 to $EV_{DD}$ + 0.3[Note] | V |
| | $V_{I2}$ | X1, X2, XT1, XT2 | −0.3 to $V_{DD}$ + 0.3[Note] | V |
| Analog input voltage | $V_{IAN}$ | P70 to P77 | −0.3 to $AV_{REF0}$ + 0.3[Note] | V |

**Note** Be sure not to exceed the absolute maximum ratings (MAX. value) of each supply voltage.

**Absolute Maximum Ratings ($T_A$ = 25°C) (2/2)**

| Parameter | Symbol | Conditions | | Ratings | Unit |
|---|---|---|---|---|---|
| Output current, low | $I_{OL}$ | P00 to P06, P30 to P35, P40 to P42, P50 to P55, P90, P91, P96 to P99, P913 to P915, PCM0 to PCM3, PCS0, PCS1, PCT0, PCT1, PCT4, PCT6, PDL0 to PDL15 | Per pin | 20 | mA |
| | | P38, P39 | | 30 | mA |
| | | P00 to P06, P30 to P35, P38, P39, P40 to P42 | Total of all pins: 70 mA | 35 | mA |
| | | P50 to P55, P90, P91, P96 to P99, P913 to P915, PCM0 to PCM3, PCS0, PCS1, PCT0, PCT1, PCT4, PCT6, PDL0 to PDL15 | | 35 | |
| Output current, high | $I_{OH}$ | Per pin | | −10 | mA |
| | | P00 to P06, P30 to P35, P40 to P42 | Total of all pins: −60 mA | −30 | |
| | | P50 to P55, P90, P91, P96 to P99, P913 to P915, PCM0 to PCM3, PCS0, PCS1, PCT0, PCT1, PCT4, PCT6, PDL0 to PDL15 | | −30 | mA |
| Operating ambient temperature | $T_A$ | Normal operation mode | | −40 to +85 | °C |
| | | Flash memory programming mode | | T.B.D. | °C |
| Storage temperature | $T_{stg}$ | μPD703308, 703308Y | | −65 to +150 | °C |
| | | μPD70F3306, 70F3306Y, 70F3308, 70F3308Y | | −40 to +125 | °C |

**Cautions 1. Do not directly connect the output (or I/O) pins of IC products to each other, or to $V_{DD}$, $V_{CC}$, and GND. Open-drain pins or open-collector pins, however, can be directly connected to each other. Direct connection of the output pins between an IC product and an external circuit is possible, if the output pins can be set to the high-impedance state and the output timing of the external circuit is designed to avoid output conflict.**

**2. 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. The ratings and conditions indicated for DC characteristics and AC characteristics represent the quality assurance range during normal operation.**

**Capacitance ($T_A$ = 25°C, $V_{DD}$ = $EV_{DD}$ = $AV_{REF0}$ = $V_{SS}$ = $EV_{SS}$ = $AV_{SS}$ = 0 V)**

| Parameter | Symbol | Conditions | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Input capacitance | $C_I$ | $f_X$ = 1 MHz Unmeasured pins returned to 0 V | P70 to P77 | | | 15 | pF |
| I/O capacitance | $C_{IO}$ | | **Note** | | | 15 | pF |
| | | | P38, P39 | | | 20 | pF |

**Note** P00 to P06, P30 to P35, P40 to P42, P50 to P55, P90, P91, P96 to P99, P913 to P915, PCM0 to PCM3, PCS0, PCS1, PCT0, PCT1, PCT4, PCT6, PDL0 to PDL15

**Remark** $f_X$: Main clock oscillation frequency

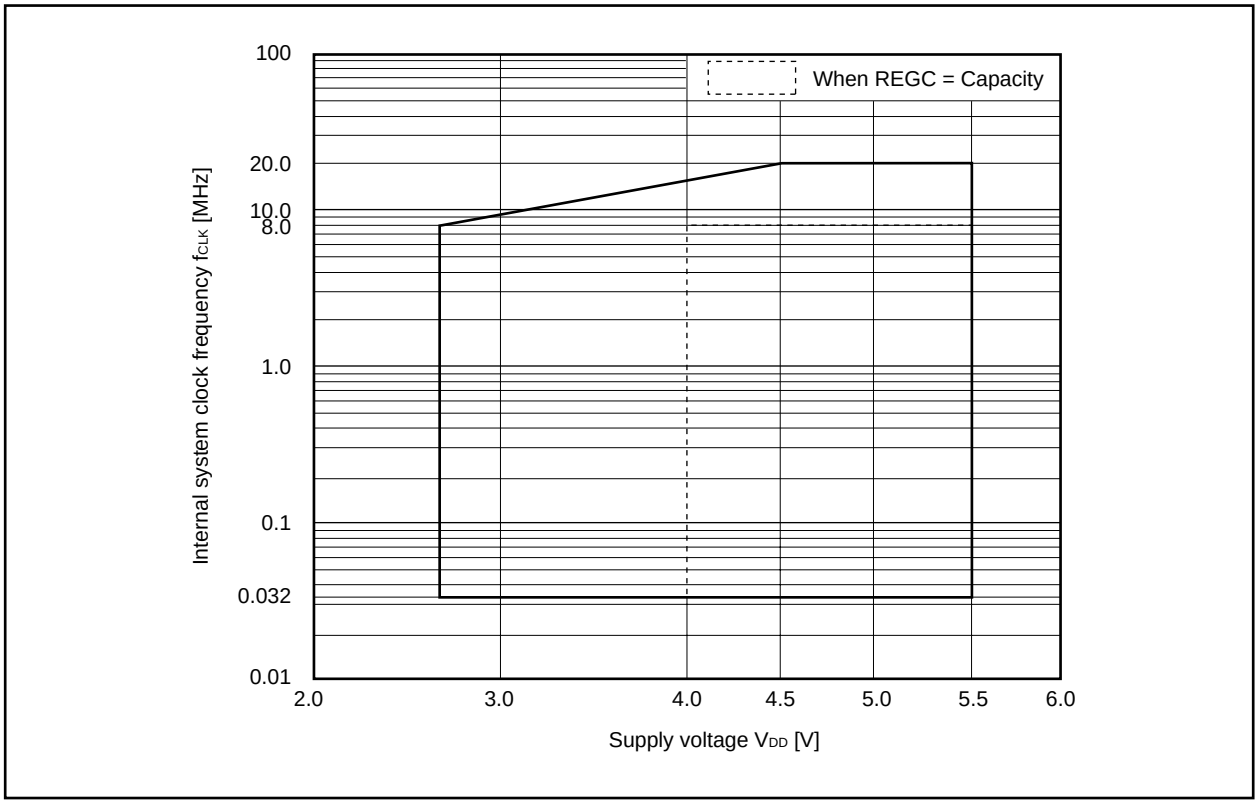
### Operating Conditions

**($T_A$ = –40 to +85°C, $V_{DD}$ = $EV_{DD}$ = $AV_{REF0}$ = 2.7 to 5.5 V, $V_{SS}$ = $EV_{SS}$ = $AV_{SS}$ = 0 V, $C_L$ = 50 pF)**

| Parameter | Symbol | Conditions | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Internal system clock frequency | $f_{CLK}$ | In PLL mode | REGC = $V_{DD}$ = 4.5 to 5.5 V | 0.25 | | 20 | MHz |
| | | | REGC = $V_{DD}$ = 4.0 to 5.5 V | 0.25 | | 16 | MHz |
| | | | REGC = Capacity, $V_{DD}$ = 4.0 to 5.5 V | 0.25 | | 8 Note | MHz |
| | | | REGC = $V_{DD}$ = 2.7 to 5.5 V | 0.25 | | 8 Note | MHz |
| | | In clock-through mode | REGC = $V_{DD}$ = 4.0 to 5.5 V | 0.0625 | | 10 | MHz |
| | | | REGC = Capacity, $V_{DD}$ = 4.0 to 5.5 V | 0.0625 | | 8 Note | MHz |
| | | | REGC = $V_{DD}$ = 2.7 to 5.5 V | 0.0625 | | 8 Note | MHz |
| | | Operating with subclock | REGC = $V_{DD}$ = 2.7 to 5.5 V | | 32.768 | | kHz |
| | | Operating with on-chip ring clock | REGC = $V_{DD}$ = 2.7 to 5.5 V | 120 | 240 | 480 | kHz |

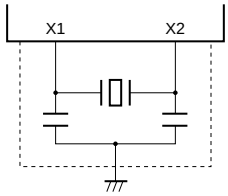
**Note** These values may change after evaluation.

### Internal System Clock Frequency vs. Supply Voltage



### Main Clock Oscillator Characteristics

**(1) Crystal resonator, ceramic resonator ($T_A$ = –40 to +85°C, $V_{DD}$ = 2.7 to 5.5 V, $V_{SS}$ = 0 V)**

| Recommended Circuit | Parameter | Conditions | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
|  | Oscillation frequency ($f_X$)Note 1 | PLL mode | REGC = $V_{DD}$ = 4.5 to 5.5 V | 2 | | 5 | MHz |
| | | | REGC = $V_{DD}$ = 4.0 to 5.5 V | 2 | | 4 | MHz |
| | | | REGC = Capacity, $V_{DD}$ = 4.0 to 5.5 V | 2 | | 2Note 2 | MHz |
| | | | REGC = $V_{DD}$ = 2.7 to 5.5 V | 2 | | 2.5 | MHz |
| | | Clock-through mode | $V_{DD}$ = 2.7 to 5.5 V | 2 | | 10 | MHz |
| | Oscillation stabilization timeNote 3 | After reset is released | When OSTS0Note 4 = 0 | | $2^{13}/f_X$ | | s |
| | | | When OSTS0Note 4 = 1 | | $2^{15}/f_X$ | | s |
| | | After STOP mode is released | | | **Note 5** | | s |

**Notes 1.** Indicates only oscillator characteristics.

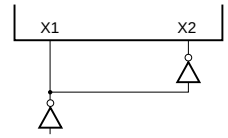
**2.** This value may change after evaluation.

**3.** Time required to stabilize the resonator after reset or STOP mode is released.

**4.** Set by mask option/option byte (refer to **CHAPTER 28**).

**5.** The value differs depending on the OSTS register settings.

**(2) External clock ($T_A$ = –40 to +85°C, $V_{DD}$ = 2.7 to 5.5 V, $V_{SS}$ = 0 V)**

| Recommended Circuit | Parameter | Conditions | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
|  | Input frequency ($f_X$) | PLL modeNote | REGC = $V_{DD}$ = 4.5 to 5.5 V | 2 | | 5 | MHz |
| | | | REGC = $V_{DD}$ = 4.0 to 5.5 V | 2 | | 4 | MHz |
| | | | REGC = $V_{DD}$ = 2.7 to 5.5 V | 2 | | 2.5 | MHz |
| | | Clock-through modeNote | $V_{DD}$ = 2.7 to 5.5 V | 2 | | 10 | MHz |

**Note** Make sure that the duty ratio of the input waveform is within 50% ±5%.

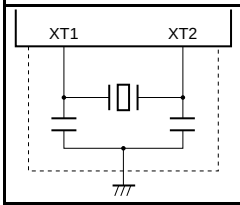
**Cautions 1. When using the main clock oscillator, wire as follows in the area enclosed by the broken lines in the above figures to avoid an adverse effect from wiring capacitance.**

- **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 clock is stopped and the device is operating on the subclock, wait until the oscillation stabilization time has been secured by the program before switching back to the main clock.**

### Subclock Oscillator Characteristics

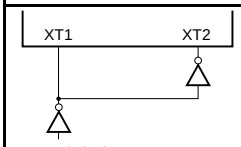
**(1) Crystal resonator ($T_A$ = –40 to +85°C, $V_{DD}$ = 2.7 to 5.5 V, $V_{SS}$ = 0 V)**

| Recommended Circuit | Parameter | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
|  | Oscillation frequency ($f_{XT}$)[Note 1] | | 32 | 32.768 | 35 | kHz |
| | Oscillation stabilization time[Note 2] | | | 10 | | s |

**Notes 1.** Indicates only oscillator characteristics.

**2.** Time required from when $V_{DD}$ reaches oscillation voltage range (2.7 V (MIN.)) to when the crystal resonator stabilizes.

**(2) External clock ($T_A$ = –40 to +85°C, $V_{DD}$ = 2.7 to 5.5 V, $V_{SS}$ = 0 V)**

| Recommended Circuit | Parameter | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
|  | Input frequency ($f_{XT}$) | REGC = $V_{DD}$ | 32 | | 35 | kHz |

**Cautions 1. When using the subclock oscillator, wire as follows in the area enclosed by the broken lines in the above figure 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. The subclock oscillator is designed as a low-amplitude circuit for reducing power consumption, and is more prone to malfunction due to noise than the main clock oscillator. Particular care is therefore required with the wiring method when the subclock is used.**

**3. Make sure that the duty ratio of the input waveform is within 50% ±5%.**

**Ring-OSC Characteristics ($T_A$ = –40 to +85°C, $V_{DD}$ = 2.7 to 5.5 V, $V_{SS}$ = 0 V)**

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Ring-OSC frequency | $f_R$ | | 120 | 240 | 480 | kHz |

**PLL Characteristics ($T_A$ = –40 to +85°C, $V_{DD}$ = 2.7 to 5.5 V, $V_{SS}$ = 0 V)**

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Input frequency | $f_X$ | | 2 | | 5 | MHz |
| Output frequency | $f_{XX}$ | | 8 | | 20 | MHz |
| Lock time | $t_{PLL}$ | After $V_{DD}$ reaches 2.7 V (MIN.) | | | 200 | μs |

**DC Characteristics**

**($T_A$ = –40 to +85°C, $V_{DD}$ = $EV_{DD}$ = $AV_{REF0}$ = 2.7 to 5.5 V, $V_{SS}$ = $EV_{SS}$ = $AV_{SS}$ = 0 V) (1/5)**

| Parameter | Symbol | Conditions | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Output current, high | $I_{OH1}$ | Per pin for P00 to P06, P30 to P35, P40 to P42, P50 to P55, P90, P91, P96 to P99, P913 to P915, PCM0 to PCM3, PCS0, PCS1, PCT0, PCT1, PCT4, PCT6, PDL0 to PDL15 | | | | –5.0 | mA |
| | | Total of P00 to P06, P30 to P35, P40 to P42 | $EV_{DD}$ = 4.0 to 5.5 V | | | –30 | mA |
| | | | $EV_{DD}$ = 2.7 to 5.5 V | | | –15 | mA |
| | | Total of P50 to P55, P90, P91, P96 to P99, P913 to P915, PCM0 to PCM3, PCS0, PCS1, PCT0, PCT1, PCT4, PCT6, PDL0 to PDL15 | $EV_{DD}$ = 4.0 to 5.5 V | | | –30 | mA |
| | | | $EV_{DD}$ = 2.7 to 5.5 V | | | –15 | mA |
| Output current, low | $I_{OL1}$ | Per pin for P00 to P06, P30 to P35, P40 to P42, P50 to P55, P90, P91, P96 to P99, P913 to P915, PCM0 to PCM3, PCS0, PCS1, PCT0, PCT1, PCT4, PCT6, PDL0 to PDL15 | | | | 10 | mA |
| | | Per pin for P38, P39 | $EV_{DD}$ = 4.0 to 5.5 V | | | 15 | mA |
| | | | $EV_{DD}$ = 2.7 to 5.5 V | | | 8 | mA |
| | | Total of P00 to P06, P30 to P35, P40 to P42 | | | | 30 | mA |
| | | Total of P38, P39, P50 to P55, P90, P91, P96 to P99, P913 to P915, PCM0 to PCM3, PCS0, PCS1, PCT0, PCT1, PCT4, PCT6, PDL0 to PDL15 | | | | 30 | mA |
| Input voltage, high | $V_{IH1}$ | **Note 1** | | 0.7$EV_{DD}$ | | $EV_{DD}$ | V |
| | $V_{IH2}$ | **Note 2** | | 0.8$EV_{DD}$ | | $EV_{DD}$ | V |
| | $V_{IH3}$ | P70 to P77 | | 0.7$AV_{REF0}$ | | $AV_{REF0}$ | V |
| | $V_{IH4}$ | X1, X2, XT1, XT2 | | $V_{DD}$ – 0.5 | | $V_{DD}$ | V |
| Input voltage, low | $V_{IL1}$ | **Note 1** | | $EV_{SS}$ | | 0.3$EV_{DD}$ | V |
| | $V_{IL2}$ | **Note 2** | | $EV_{SS}$ | | 0.2$EV_{DD}$ | V |
| | $V_{IL3}$ | P70 to P77 | | $AV_{SS}$ | | 0.3$AV_{REF0}$ | V |
| | $V_{IL4}$ | X1, X2, XT1, XT2 | | $V_{SS}$ | | 0.4 | V |

**Notes 1.** P00, P01, P30, P41, P98, PCM0 to PCM3, PCS0, PCS1, PCT0, PCT1, PCT4, PCT6, PDL0 to PDL15 and their alternate-function pins.

**2.** $\overline{\text{RESET}}$, P02 to P06, P31 to P35, P38, P39, P40, P42, P50 to P55, P90, P91, P96, P97, P99, P913 to P915 and their alternate-function pins.

**DC Characteristics**

**($T_A$ = –40 to +85°C, $V_{DD}$ = $EV_{DD}$ = $AV_{REF0}$ = 2.7 to 5.5 V, $V_{SS}$ = $EV_{SS}$ = $AV_{SS}$ = 0 V) (2/5)**

| Parameter | Symbol | Conditions | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Output voltage, high | $V_{OH1}$ | **Note 1** | $I_{OH}$ = –2.0 mA, $EV_{DD}$ = 4.0 to 5.5 V | $EV_{DD}$ – 1.0 | | $EV_{DD}$ | V |
| | | **Note 2** | $I_{OH}$ = –0.1 mA, $EV_{DD}$ = 2.7 to 5.5 V | $EV_{DD}$ – 0.5 | | $EV_{DD}$ | V |
| Output voltage, low | $V_{OL1}$ | **Note 3** | $I_{OL}$ = 2.0 mA[Note 4] | 0 | | 0.8 | V |
| | $V_{OL2}$ | P38, P39 | $I_{OL}$ = 15 mA, $EV_{DD}$ = 4.0 to 5.5 V | 0 | | 2.0 | V |
| | | | $I_{OL}$ = 8 mA, $EV_{DD}$ = 3.0 to 5.5 V | 0 | | 1.0 | V |
| | | | $I_{OL}$ = 5 mA, $EV_{DD}$ = 2.7 to 5.5 V | 0 | | 1.0 | V |
| Input leakage current, high | $I_{LIH}$ | $V_{IN}$ = $V_{DD}$ | | | | 3.0 | μA |
| Input leakage current, low | $I_{LIL}$ | $V_{IN}$ = 0 V | | | | –3.0 | μA |
| Output leakage current, high | $I_{LOH}$ | $V_O$ = $V_{DD}$ | | | | 3.0 | μA |
| Output leakage current, low | $I_{LOL}$ | $V_O$ = 0 V | | | | –3.0 | μA |
| Pull-up resistor | $R_L$ | $V_{IN}$ = 0 V | | 10 | 30 | 100 | kΩ |

**Notes 1.** Total of P00 to P06, P30 to P35, P40 to P42 and their alternate-function pins: $I_{OH}$ = –30 mA, total of P50 to P55, P90, P91, P96 to P99, P913 to P915, PCM0 to PCM3, PCS0, PCS1, PCT0, PCT1, PCT4, PCT6, PDL0 to PDL15 and their alternate-function pins: $I_{OH}$ = –30 mA.

**2.** Total of P00 to P06, P30 to P35, P40 to P42 and their alternate-function pins: $I_{OH}$ = –15 mA, total of P50 to P55, P90, P91, P96 to P99, P913 to P915, PCM0 to PCM3, PCS0, PCS1, PCT0, PCT1, PCT4, PCT6, PDL0 to PDL15 and their alternate-function pins: $I_{OH}$ = –15 mA.

**3.** Total of P00 to P06, P30 to P35, P40 to P42 and their alternate-function pins: $I_{OL}$ = 30 mA, total of P38, P39, P50 to P55, P90, P91, P96 to P99, P913 to P915, PCM0 to PCM3, PCS0, PCS1, PCT0, PCT1, PCT4, PCT6, PDL0 to PDL15 and their alternate-function pins: $I_{OL}$ = 30 mA.

**4.** Refer to $I_{OL1}$ for $I_{OL}$ of P38 and P39.

**DC Characteristics**

**($T_A$ = –40 to +85°C, $V_{DD}$ = $EV_{DD}$ = $AV_{REF0}$ = 2.7 to 5.5 V, $V_{SS}$ = $EV_{SS}$ = $AV_{SS}$ = 0 V) (3/5)**

| Parameter | Symbol | Conditions | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Supply current[Note 1] ($\mu$PD70F3308, 70F3308Y) | $I_{DD1}$ | Normal operation mode All peripheral functions operating | $f_{XX}$ = 20 MHz ($f_X$ = 5 MHz) (in PLL mode) REGC = $V_{DD}$ = 5 V ±10% | | 55 | 75 | mA |
| | | | $f_{XX}$ = T.B.D. (in clock-through mode) REGC = $V_{DD}$ = 3 V ±10% | | T.B.D. | T.B.D. | mA |
| | $I_{DD2}$ | HALT mode All peripheral functions operating | $f_{XX}$ = 20 MHz ($f_X$ = 5 MHz) (in PLL mode) REGC = $V_{DD}$ = 5 V ±10% | | 29 | 43 | mA |
| | | | $f_{XX}$ = T.B.D. (in clock-through mode) REGC = $V_{DD}$ = 3 V ±10% | | T.B.D. | T.B.D. | mA |
| | $I_{DD3}$ | IDLE mode Watch timer operating, ring oscillation stopped | $f_{XX}$ = 5 MHz (when PLL mode off) REGC = $V_{DD}$ = 5 V ±10% | | 2.1 | 3.3 | mA |
| | | | $f_X$ = 8 MHz (in clock-through mode) REGC = $V_{DD}$ = 3 V ±10% | | T.B.D. | T.B.D. | mA |
| | $I_{DD4}$ | Subclock operation mode ($f_{XT}$ = 32.768 kHz) Main oscillation stopped, ring oscillation stopped | | | 250 | 420 | $\mu$A |
| | $I_{DD5}$ | Sub-IDLE mode ($f_{XT}$ = 32.768 kHz) Main oscillation stopped, ring oscillation stopped | | | 20 | 75 | $\mu$A |
| | $I_{DD6}$ | STOP mode | Sub-oscillation operating, ring oscillation operating | | 34 | 103 | $\mu$A |
| | | | Sub-oscillation stopped (XT1 = $V_{SS}$), ring oscillation operating | | 17.5 | 63.5 | $\mu$A |
| | | | Sub-oscillation stopped (XT1 = $V_{SS}$), ring oscillation stopped | | 3.5 | 35.5 | $\mu$A |
| | $I_{DD7}$[Note 2] | Ring clock operation mode ($f_{XX}$ = 240 kHz) Main oscillation stopped, sub-oscillation stopped | | | 4 | 11 | mA |
| | $I_{DD8}$[Note 2] | Ring HALT mode ($f_{XX}$ = 240 kHz) Main oscillation stopped, sub-oscillation stopped | | | T.B.D. | T.B.D. | mA |
| | $I_{DD9}$ | Flash memory erase/write | $f_{XX}$ = 20 MHz ($f_X$ = 5 MHz) (in PLL mode) REGC = $V_{DD}$ = 5 V ±10% | | 65 | 90 | mA |
| | | | $f_{XX}$ = T.B.D. (in clock-through mode) REGC = $V_{DD}$ = 3 V ±10% | | T.B.D. | T.B.D. | mA |

**Notes 1.** Total current of $V_{DD}$ and $EV_{DD}$ (all ports stopped). $AV_{REF0}$ is not included.

**2.** The supply current of the main clock oscillator is not included since the main clock oscillator is stopped because of an abnormality.

**Remark** $f_{XX}$: Main clock frequency

$f_X$: Main clock oscillation frequency

$f_{XT}$: Subclock frequency

**DC Characteristics**

**($T_A$ = –40 to +85°C, $V_{DD}$ = $EV_{DD}$ = $AV_{REF0}$ = 2.7 to 5.5 V, $V_{SS}$ = $EV_{SS}$ = $AV_{SS}$ = 0 V) (4/5)**

| Parameter | Symbol | Conditions | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Supply current[Note 1] ($\mu$PD70F3306, 70F3306Y) | $I_{DD1}$ | Normal operation mode All peripheral functions operating | $f_{XX}$ = 20 MHz ($f_X$ = 5 MHz) (in PLL mode) REGC = $V_{DD}$ = 5 V ±10% | | 51 | 70 | mA |
| | | | $f_{XX}$ = T.B.D. (in clock-through mode) REGC = $V_{DD}$ = 3 V ±10% | | T.B.D. | T.B.D. | mA |
| | $I_{DD2}$ | HALT mode All peripheral functions operating | $f_{XX}$ = 20 MHz ($f_X$ = 5 MHz) (in PLL mode) REGC = $V_{DD}$ = 5 V ±10% | | 25 | 38 | mA |
| | | | $f_{XX}$ = T.B.D. (in clock-through mode) REGC = $V_{DD}$ = 3 V ±10% | | T.B.D. | T.B.D. | mA |
| | $I_{DD3}$ | IDLE mode Watch timer operating, ring oscillation stopped | $f_{XX}$ = 5 MHz (when PLL mode off) REGC = $V_{DD}$ = 5 V ±10% | | 1.8 | 2.9 | mA |
| | | | $f_{XX}$ = T.B.D. (in clock-through mode) REGC = $V_{DD}$ = 3 V ±10% | | T.B.D. | T.B.D. | mA |
| | $I_{DD4}$ | Subclock operation mode ($f_{XT}$ = 32.768 kHz) Main oscillation stopped, ring oscillation stopped | | | 240 | 400 | $\mu$A |
| | $I_{DD5}$ | Sub-IDLE mode ($f_{XT}$ = 32.768 kHz) Main oscillation stopped, ring oscillation stopped | | | 20 | 75 | $\mu$A |
| | $I_{DD6}$ | STOP mode | Sub-oscillation operating, ring oscillation operating | | 34 | 103 | $\mu$A |
| | | | Sub-oscillation stopped (XT1 = $V_{SS}$), ring oscillation operating | | 17.5 | 63.5 | $\mu$A |
| | | | Sub-oscillation stopped (XT1 = $V_{SS}$), ring oscillation stopped | | 3.5 | 35.5 | $\mu$A |
| | $I_{DD7}$[Note 2] | Ring clock operation mode ($f_{XX}$ = 240 kHz) Main oscillation stopped, sub-oscillation stopped | | | 3.5 | 10.5 | mA |
| | $I_{DD8}$[Note 2] | Ring HALT mode ($f_{XX}$ = 240 kHz) Main oscillation stopped, sub-oscillation stopped | | | T.B.D. | T.B.D. | mA |
| | $I_{DD9}$ | Flash memory erase/write | $f_{XX}$ = 20 MHz ($f_X$ = 5 MHz) (in PLL mode) REGC = $V_{DD}$ = 5 V ±10% | | 61 | 85 | mA |
| | | | $f_{XX}$ = T.B.D. (in clock-through mode) REGC = $V_{DD}$ = 3 V ±10% | | T.B.D. | T.B.D. | mA |

**Notes 1.** Total current of $V_{DD}$ and $EV_{DD}$ (all ports stopped). $AV_{REF0}$ is not included.

**2.** The supply current of the main clock oscillator is not included since the main clock oscillator is stopped because of an abnormality.

**Remark** $f_{XX}$: Main clock frequency

$f_X$: Main clock oscillation frequency

$f_{XT}$: Subclock frequency

**DC Characteristics**

**($T_A$ = –40 to +85°C, $V_{DD}$ = $EV_{DD}$ = $AV_{REF0}$ = 2.7 to 5.5 V, $V_{SS}$ = $EV_{SS}$ = $AV_{SS}$ = 0 V) (5/5)**

| Parameter | Symbol | Conditions | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Supply current[Note 1] ($\mu$PD703308, 703308Y) | $I_{DD1}$ | Normal operation mode All peripheral functions operating | $f_{XX}$ = 20 MHz ($f_X$ = 5 MHz) (in PLL mode) REGC = $V_{DD}$ = 5 V ±10% | | 42 | 60 | mA |
| | | | $f_{XX}$ = T.B.D. (in clock-through mode) REGC = $V_{DD}$ = 3 V ±10% | | T.B.D. | T.B.D. | mA |
| | $I_{DD2}$ | HALT mode All peripheral functions operating | $f_{XX}$ = 20 MHz ($f_X$ = 5 MHz) (in PLL mode) REGC = $V_{DD}$ = 5 V ±10% | | 29 | 40 | mA |
| | | | $f_{XX}$ = T.B.D. (in clock-through mode) REGC = $V_{DD}$ = 3 V ±10% | | T.B.D. | T.B.D. | mA |
| | $I_{DD3}$ | IDLE mode Watch timer operating, ring oscillation stopped | $f_X$ = 5 MHz (when PLL mode off) REGC = $V_{DD}$ = 5 V ±10% | | 1.7 | 2.7 | mA |
| | | | $f_X$ = T.B.D. (in clock-through mode) REGC = $V_{DD}$ = 3 V ±10% | | T.B.D. | T.B.D. | mA |
| | $I_{DD4}$ | Subclock operation mode ($f_{XT}$ = 32.768 kHz) Main oscillation stopped, ring oscillation stopped | | | 100 | 220 | $\mu$A |
| | $I_{DD5}$ | Sub-IDLE mode ($f_{XT}$ = 32.768 kHz) Main oscillation stopped, ring oscillation stopped | | | 20 | 75 | $\mu$A |
| | $I_{DD6}$ | STOP mode | Sub-oscillation operating, ring oscillation operating | | 34 | 103 | $\mu$A |
| | | | Sub-oscillation stopped (XT1 = $V_{SS}$), ring oscillation operating | | 17.5 | 63.5 | $\mu$A |
| | | | Sub-oscillation stopped (XT1 = $V_{SS}$), ring oscillation stopped | | 3.5 | 35.5 | $\mu$A |
| | $I_{DD7}$[Note 2] | Ring clock operation mode ($f_{XX}$ = 240 kHz) Main oscillation stopped, sub-oscillation stopped | | | 3 | 9.5 | mA |
| | $I_{DD8}$[Note 2] | Ring HALT mode ($f_{XX}$ = 240 kHz) Main oscillation stopped, sub-oscillation stopped | | | T.B.D. | T.B.D. | $\mu$A |

**Notes 1.** Total current of $V_{DD}$ and $EV_{DD}$ (all ports stopped). $AV_{REF0}$ is not included.

**2.** The supply current of the main clock oscillator is not included since the main clock oscillator is stopped because of an abnormality.

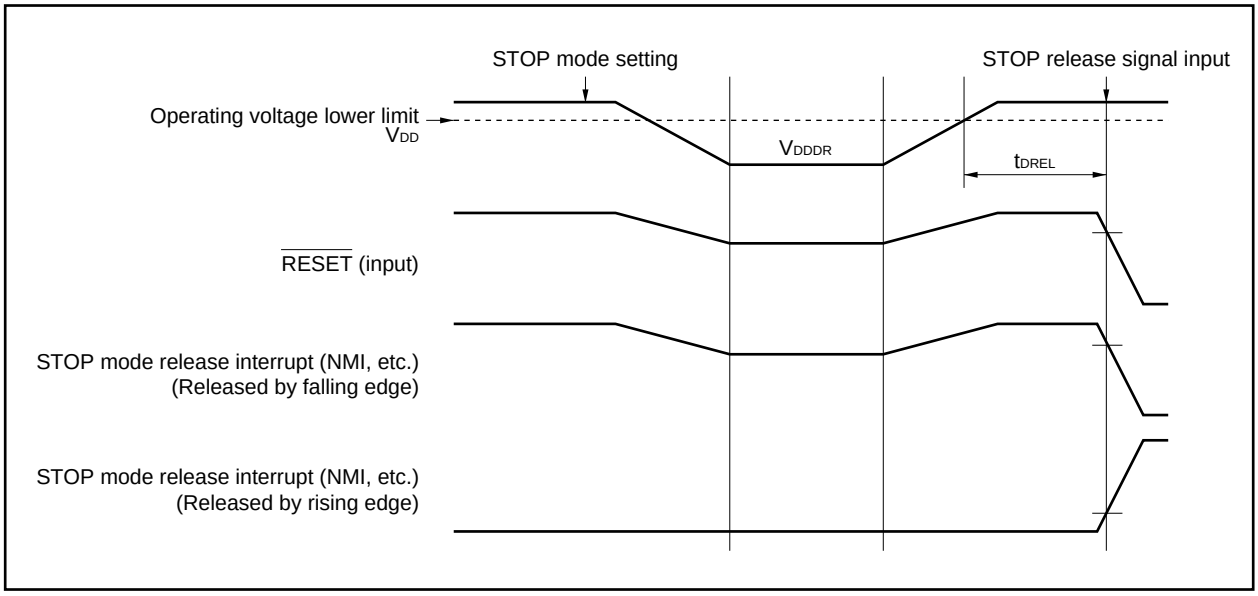
**Remark** $f_{XX}$: Main clock frequency
$f_X$: Main clock oscillation frequency
$f_{XT}$: Subclock frequency

### Data Retention Characteristics

**STOP Mode ($T_A$ = –40 to +85°C)**

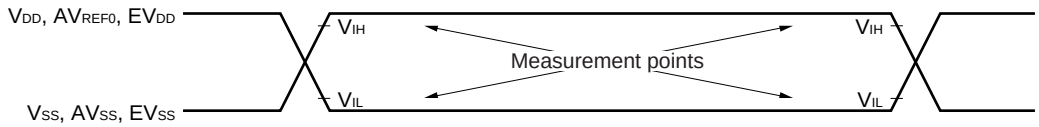
| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Data retention voltage | $V_{DDDR}$ | STOP mode | 2.0 | | 5.5 | V |
| STOP release signal input time | $t_{DREL}$ | | 0 | | | μs |

**Caution Shifting to STOP mode and restoring from STOP mode must be performed within the rated operating range.**

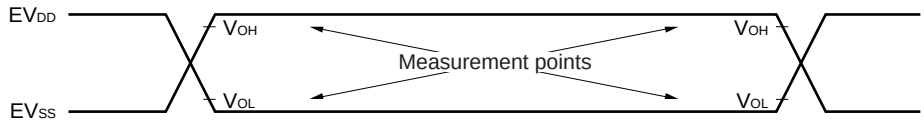


### AC Characteristics

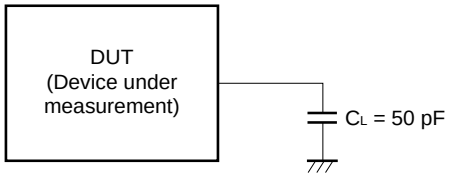
### AC Test Input Measurement Points




### AC Test Output Measurement Points




### Load Conditions



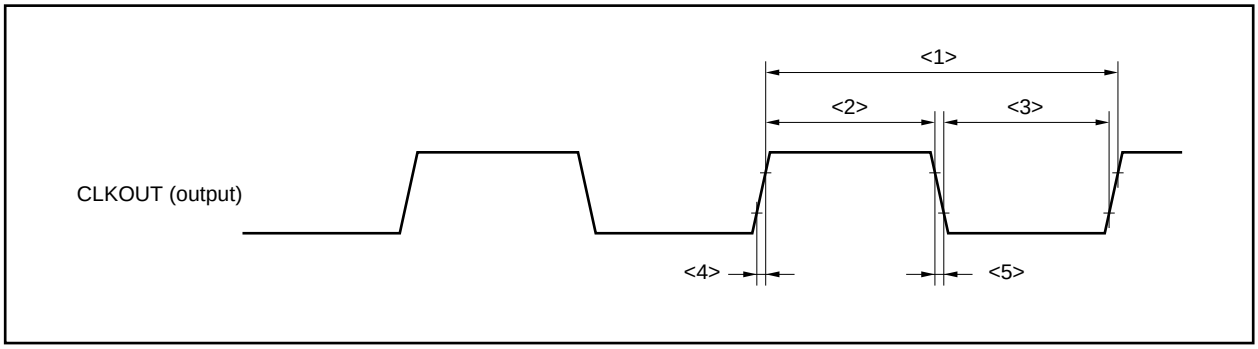

**Caution If the load capacitance exceeds 50 pF due to the circuit configuration, bring the load capacitance of the device to 50 pF or less by inserting a buffer or by some other means.**

### CLKOUT Output Timing

**($T_A$ = –40 to +85°C, $V_{DD}$ = $EV_{DD}$ = $AV_{REF0}$ = 2.7 to 5.5 V, $V_{SS}$ = $EV_{SS}$ = $AV_{SS}$ = 0 V, $C_L$ = 50 pF)**

| Parameter | Symbol | | Conditions | MIN. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Output cycle | $t_{CYK}$ | <1> | | 50 ns | 30.6 μs | |
| High-level width | $t_{WKH}$ | <2> | $V_{DD}$ = 4.0 to 5.5 V | $t_{CYK}/2 - 17$ | | ns |
| | | | $V_{DD}$ = 2.7 to 5.5 V | $t_{CYK}/2 - 26$ | | ns |
| Low-level width | $t_{WKL}$ | <3> | $V_{DD}$ = 4.0 to 5.5 V | $t_{CYK}/2 - 17$ | | ns |
| | | | $V_{DD}$ = 2.7 to 5.5 V | $t_{CYK}/2 - 26$ | | ns |
| Rise time | $t_{KR}$ | <4> | $V_{DD}$ = 4.0 to 5.5 V | | 17 | ns |
| | | | $V_{DD}$ = 2.7 to 5.5 V | | 26 | ns |
| Fall time | $t_{KF}$ | <5> | $V_{DD}$ = 4.0 to 5.5 V | | 17 | ns |
| | | | $V_{DD}$ = 2.7 to 5.5 V | | 26 | ns |

### Clock Timing



## Bus Timing

### (1) Read/write cycle

### (a) Read/write cycle (CLKOUT asynchronous)

**($T_A$ = –40 to +85°C, $V_{DD}$ = $EV_{DD}$ = $AV_{REF0}$ = 4.0 to 5.5 V, $V_{SS}$ = $EV_{SS}$ = $AV_{SS}$ = 0 V, $C_L$ = 50 pF) (1/2)**

| Parameter | Symbol | | Conditions | MIN. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Address setup time (to ASTB↓) | $t_{SAST}$ | <6> | | $(0.5 + t_{ASW})T - 23$ | | ns |
| Address hold time (from ASTB↓) | $t_{HSTA}$ | <7> | | $(0.5 + t_{ASW})T - 15$ | | ns |
| Delay time from $\overline{RD}$↓ to address float | $t_{FRDA}$ | <8> | | | 16 | ns |
| Data input setup time from address | $t_{SAID}$ | <9> | | | $(2 + n + t_{ASW} + t_{AHW})T - 40$ | ns |
| Data input setup time from $\overline{RD}$↓ | $t_{SRID}$ | <10> | | | $(1 + n + t_{ASW} + t_{AHW})T - 25$ | ns |
| Delay time from ASTB↓ to $\overline{RD}$, $\overline{WRm}$↓ | $t_{DSTRDWR}$ | <11> | | $(0.5 + t_{AHW})T - 20$ | | ns |
| Data input hold time (from $\overline{RD}$↑) | $t_{HRDID}$ | <12> | | 0 | | ns |
| Address output time from $\overline{RD}$↑ | $t_{DRDA}$ | <13> | | $(1 + i)T - 16$ | | ns |
| Delay time from $\overline{RD}$, $\overline{WRm}$↑ to ASTB↑ | $t_{DRDWRST}$ | <14> | | $0.5T - 10$ | | ns |
| Delay time from $\overline{RD}$↑ to ASTB↓ | $t_{DRDST}$ | <15> | | $(1.5 + i + t_{ASW})T - 10$ | | ns |
| $\overline{RD}$, $\overline{WRm}$ low-level width | $t_{WRDWRL}$ | <16> | | $(1 + n)T - 10$ | | ns |
| ASTB high-level width | $t_{WSTH}$ | <17> | | $(1 + t_{ASW})T - 25$ | | ns |
| Data output time from $\overline{WRm}$↓ | $t_{DWROD}$ | <18> | | | 20 | ns |
| Data output setup time (to $\overline{WRm}$↑) | $t_{SODWR}$ | <19> | | $(1 + n)T - 25$ | | ns |
| Data output hold time (from $\overline{WRm}$↑) | $t_{HWROD}$ | <20> | | $T - 15$ | | ns |
| $\overline{WAIT}$ setup time (to address) | $t_{SAWT1}$ | <21> | $n \geq 1$ | | $(1.5 + t_{ASW} + t_{AHW})T - 45$ | ns |
| | $t_{SAWT2}$ | <22> | | | $(1.5 + n + t_{ASW} + t_{AHW})T - 45$ | ns |
| $\overline{WAIT}$ hold time (from address) | $t_{HAWT1}$ | <23> | $n \geq 1$ | $(0.5 + n + t_{ASW} + t_{AHW})T$ | | ns |
| | $t_{HAWT2}$ | <24> | | $(1.5 + n + t_{ASW} + t_{AHW})T$ | | ns |
| $\overline{WAIT}$ setup time (to ASTB↓) | $t_{SSTWT1}$ | <25> | $n \geq 1$ | | $(1 + t_{AHW})T - 32$ | ns |
| | $t_{SSTWT2}$ | <26> | | | $(1 + n + t_{AHW})T - 32$ | ns |
| $\overline{WAIT}$ hold time (from ASTB↓) | $t_{HSTWT1}$ | <27> | $n \geq 1$ | $(n + t_{AHW})T$ | | ns |
| | $t_{HSTWT2}$ | <28> | | $(1 + n + t_{AHW})T$ | | ns |

**Remarks 1.** $t_{ASW}$: Number of address setup wait clocks
$t_{AHW}$: Number of address hold wait clocks

**2.** $T = 1/f_{CPU}$ ($f_{CPU}$: CPU operating clock frequency)

**3.** n: Number of wait clocks inserted in the bus cycle
The sampling timing changes when a programmable wait is inserted.

**4.** m = 0, 1

**5.** i: Number of idle states inserted after a read cycle (0 or 1)

**6.** The values in the above specifications are values for when clocks with a 1:1 duty ratio are input from X1.

**($T_A$ = –40 to +85°C, $V_{DD}$ = $EV_{DD}$ = $AV_{REF0}$ = 2.7 to 5.5 V, $V_{SS}$ = $EV_{SS}$ = $AV_{SS}$ = 0 V, $C_L$ = 50 pF) (2/2)**

| Parameter | Symbol | | Conditions | MIN. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Address setup time (to ASTB↓) | $t_{SAST}$ | <6> | | $(0.5 + t_{ASW})T - 42$ | | ns |
| Address hold time (from ASTB↓) | $t_{HSTA}$ | <7> | | $(0.5 + t_{ASW})T - 30$ | | ns |
| Delay time from $\overline{RD}$↓ to address float | $t_{FRDA}$ | <8> | | | 32 | ns |
| Data input setup time from address | $t_{SAID}$ | <9> | | | $(2 + n + t_{ASW} + t_{AHW})T - 72$ | ns |
| Data input setup time from $\overline{RD}$↓ | $t_{SRID}$ | <10> | | | $(1 + n + t_{ASW} + t_{AHW})T - 40$ | ns |
| Delay time from ASTB↓ to $\overline{RD}$, $\overline{WRm}$↓ | $t_{DSTRDWR}$ | <11> | | $(0.5 + t_{AHW})T - 35$ | | ns |
| Data input hold time (from $\overline{RD}$↑) | $t_{HRDID}$ | <12> | | 0 | | ns |
| Address output time from $\overline{RD}$↑ | $t_{DRDA}$ | <13> | | $(1 + i)T - 32$ | | ns |
| Delay time from $\overline{RD}$, $\overline{WRm}$↑ to ASTB↑ | $t_{DRDWRST}$ | <14> | | $0.5T - 20$ | | ns |
| Delay time from $\overline{RD}$↑ to ASTB↓ | $t_{DRDST}$ | <15> | | $(1.5 + i + t_{ASW})T - 20$ | | ns |
| $\overline{RD}$, $\overline{WRm}$ low-level width | $t_{WRDWRL}$ | <16> | | $(1 + n)T - 20$ | | ns |
| ASTB high-level width | $t_{WSTH}$ | <17> | | $(1 + t_{ASW})T - 50$ | | ns |
| Data output time from $\overline{WRm}$↓ | $t_{DWROD}$ | <18> | | | 35 | ns |
| Data output setup time (to $\overline{WRm}$↑) | $t_{SODWR}$ | <19> | | $(1 + n)T - 40$ | | ns |
| Data output hold time (from $\overline{WRm}$↑) | $t_{HWROD}$ | <20> | | $T - 30$ | | ns |
| $\overline{WAIT}$ setup time (to address) | $t_{SAWT1}$ | <21> | $n \geq 1$ | | $(1.5 + t_{ASW} + t_{AHW})T - 80$ | ns |
| | $t_{SAWT2}$ | <22> | | | $(1.5 + n + t_{ASW} + t_{AHW})T - 80$ | ns |
| $\overline{WAIT}$ hold time (from address) | $t_{HAWT1}$ | <23> | $n \geq 1$ | $(0.5 + n + t_{ASW} + t_{AHW})T$ | | ns |
| | $t_{HAWT2}$ | <24> | | $(1.5 + n + t_{ASW} + t_{AHW})T$ | | ns |
| $\overline{WAIT}$ setup time (to ASTB↓) | $t_{SSTWT1}$ | <25> | $n \geq 1$ | | $(1 + t_{AHW})T - 60$ | ns |
| | $t_{SSTWT2}$ | <26> | | | $(1 + n + t_{AHW})T - 60$ | ns |
| $\overline{WAIT}$ hold time (from ASTB↓) | $t_{HSTWT1}$ | <27> | $n \geq 1$ | $(n + t_{AHW})T$ | | ns |
| | $t_{HSTWT2}$ | <28> | | $(1 + n + t_{AHW})T$ | | ns |

**Caution Set the following in accordance with the usage conditions of the CPU operating clock frequency (k = 0, 1).**

- **70 ns < $1/f_{CPU}$ < 84 ns**
  **Set an address setup wait (AWC.ASWk bit = 1).**
- **62.5 ns < $1/f_{CPU}$ < 70 ns**
  **Set an address setup wait (ASWk bit = 1) and address hold wait (AWC.AHWk bit = 1).**

**Remarks 1.** $t_{ASW}$: Number of address setup wait clocks
$t_{AHW}$: Number of address hold wait clocks

**2.** $T = 1/f_{CPU}$ ($f_{CPU}$: CPU operating clock frequency)

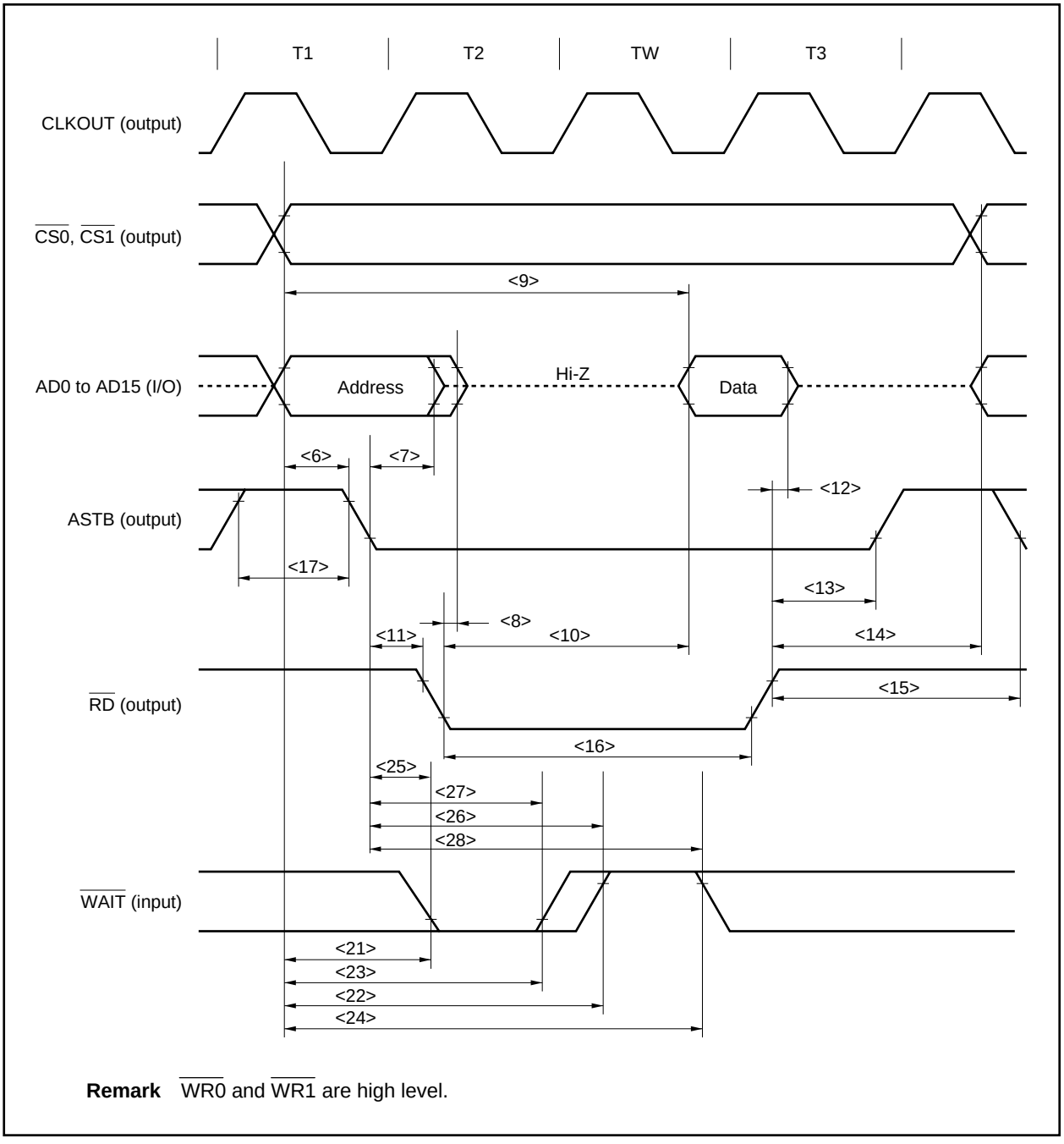
**3.** n: Number of wait clocks inserted in the bus cycle
The sampling timing changes when a programmable wait is inserted.

**4.** m = 0, 1

**5.** i: Number of idle states inserted after a read cycle (0 or 1)

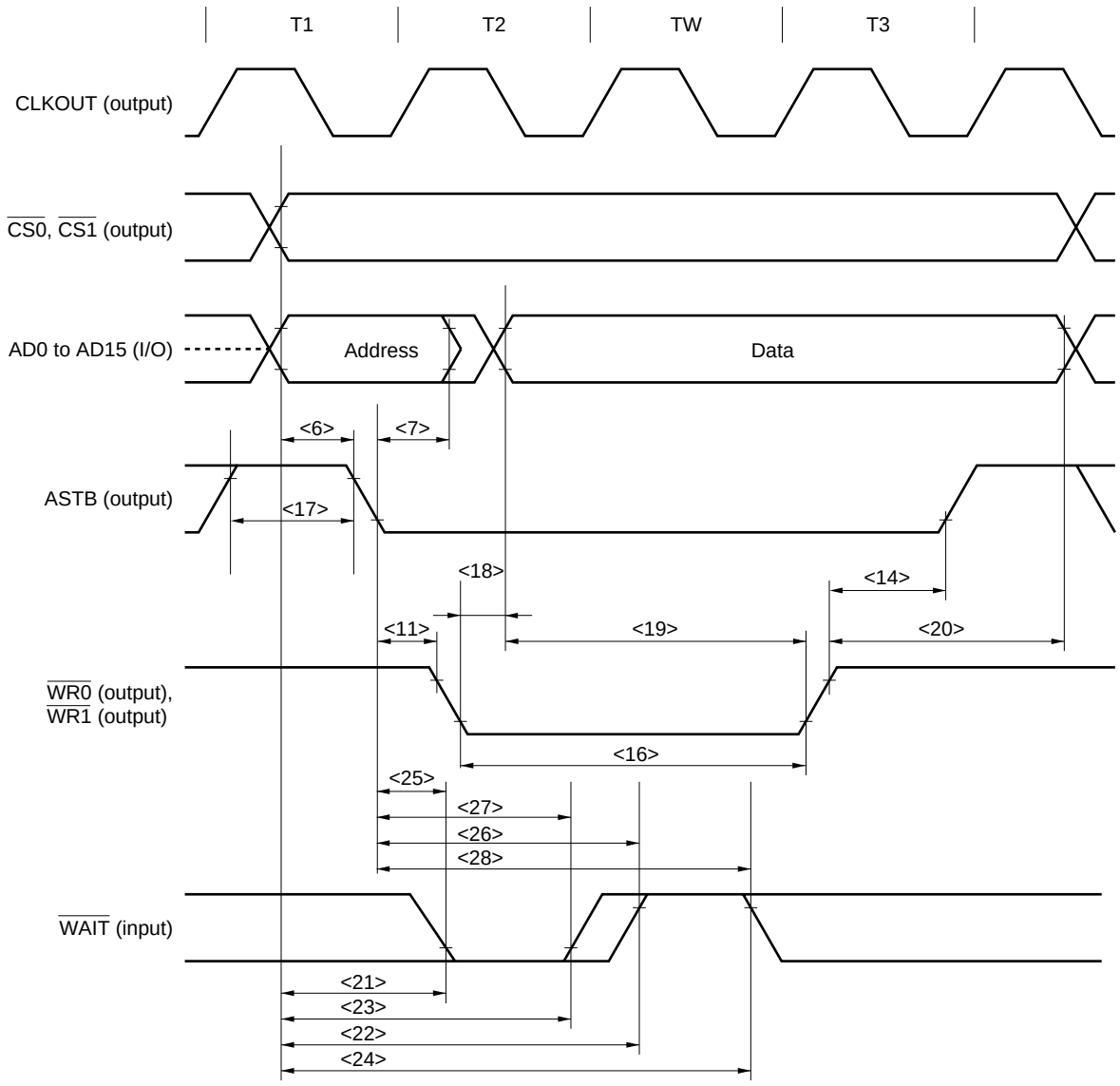
**6.** The values in the above specifications are values for when clocks with a 1:1 duty ratio are input from X1.

**Read Cycle (CLKOUT Asynchronous)**




**Remark** WR0 and WR1 are high level.

**Write Cycle (CLKOUT Asynchronous)**




**Remark** $\overline{WR0}$ and $\overline{WR1}$ are high level.

### (b) Read/write cycle (CLKOUT synchronous)

**($T_A$ = –40 to +85°C, $V_{DD}$ = $EV_{DD}$ = $AV_{REF0}$ = 4.0 to 5.5 V, $V_{SS}$ = $EV_{SS}$ = $AV_{SS}$ = 0 V, $C_L$ = 50 pF) (1/2)**

| Parameter | Symbol | | Conditions | MIN. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Delay time from CLKOUT↑ to address | $t_{DKA}$ | <29> | | 0 | 19 | ns |
| Delay time from CLKOUT↑ to address float | $t_{FKA}$ | <30> | | 0 | 14 | ns |
| Delay time from CLKOUT↓ to ASTB | $t_{DKST}$ | <31> | | 0 | 23 | ns |
| Delay time from CLKOUT↑ to $\overline{RD}$, $\overline{WRm}$ | $t_{DKRDWR}$ | <32> | | –22 | 0 | ns |
| Data input setup time (to CLKOUT↑) | $t_{SIDK}$ | <33> | | 15 | | ns |
| Data input hold time (from CLKOUT↑) | $t_{HKID}$ | <34> | | 0 | | ns |
| Data output delay time from CLKOUT↑ | $t_{DKOD}$ | <35> | | | 19 | ns |
| $\overline{WAIT}$ setup time (to CLKOUT↓) | $t_{SWTK}$ | <36> | | 15 | | ns |
| $\overline{WAIT}$ hold time (from CLKOUT↓) | $t_{HKWT}$ | <37> | | 0 | | ns |

**Remarks 1.** m = 0, 1

**2.** The values in the above specifications are values for when clocks with a 1:1 duty ratio are input from X1.

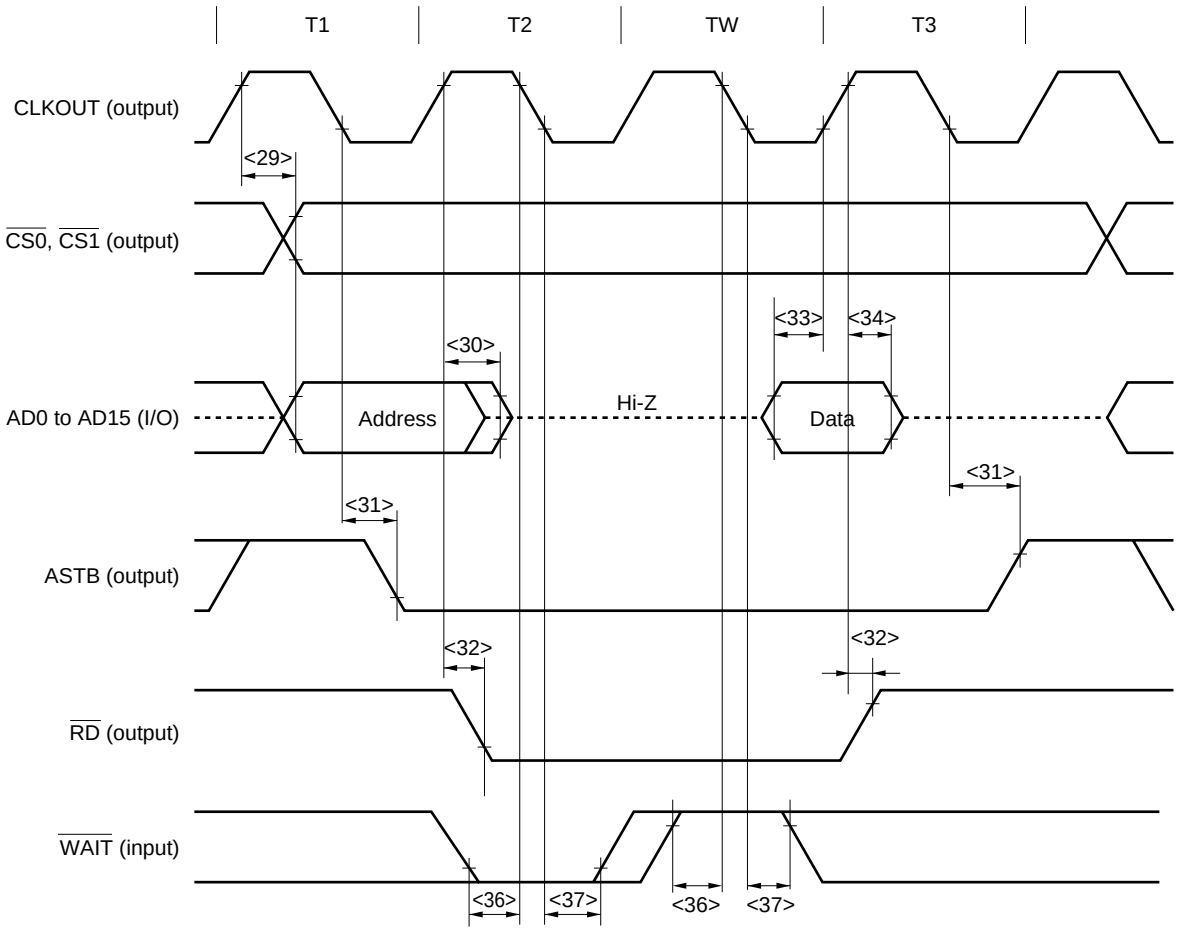
**($T_A$ = –40 to +85°C, $V_{DD}$ = $EV_{DD}$ = $AV_{REF0}$ = 2.7 to 5.5 V, $V_{SS}$ = $EV_{SS}$ = $AV_{SS}$ = 0 V, $C_L$ = 50 pF) (2/2)**

| Parameter | Symbol | | Conditions | MIN. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Delay time from CLKOUT↑ to address | $t_{DKA}$ | <29> | | 0 | 19 | ns |
| Delay time from CLKOUT↑ to address float | $t_{FKA}$ | <30> | | 0 | 18 | ns |
| Delay time from CLKOUT↓ to ASTB | $t_{DKST}$ | <31> | | 0 | 55 | ns |
| Delay time from CLKOUT↑ to $\overline{RD}$, $\overline{WRm}$ | $t_{DKRDWR}$ | <32> | | –22 | 0 | ns |
| Data input setup time (to CLKOUT↑) | $t_{SIDK}$ | <33> | | 30 | | ns |
| Data input hold time (from CLKOUT↑) | $t_{HKID}$ | <34> | | 0 | | ns |
| Data output delay time from CLKOUT↑ | $t_{DKOD}$ | <35> | | | 19 | ns |
| $\overline{WAIT}$ setup time (to CLKOUT↓) | $t_{SWTK}$ | <36> | | 25 | | ns |
| $\overline{WAIT}$ hold time (from CLKOUT↓) | $t_{HKWT}$ | <37> | | 0 | | ns |

**Remarks 1.** m = 0, 1

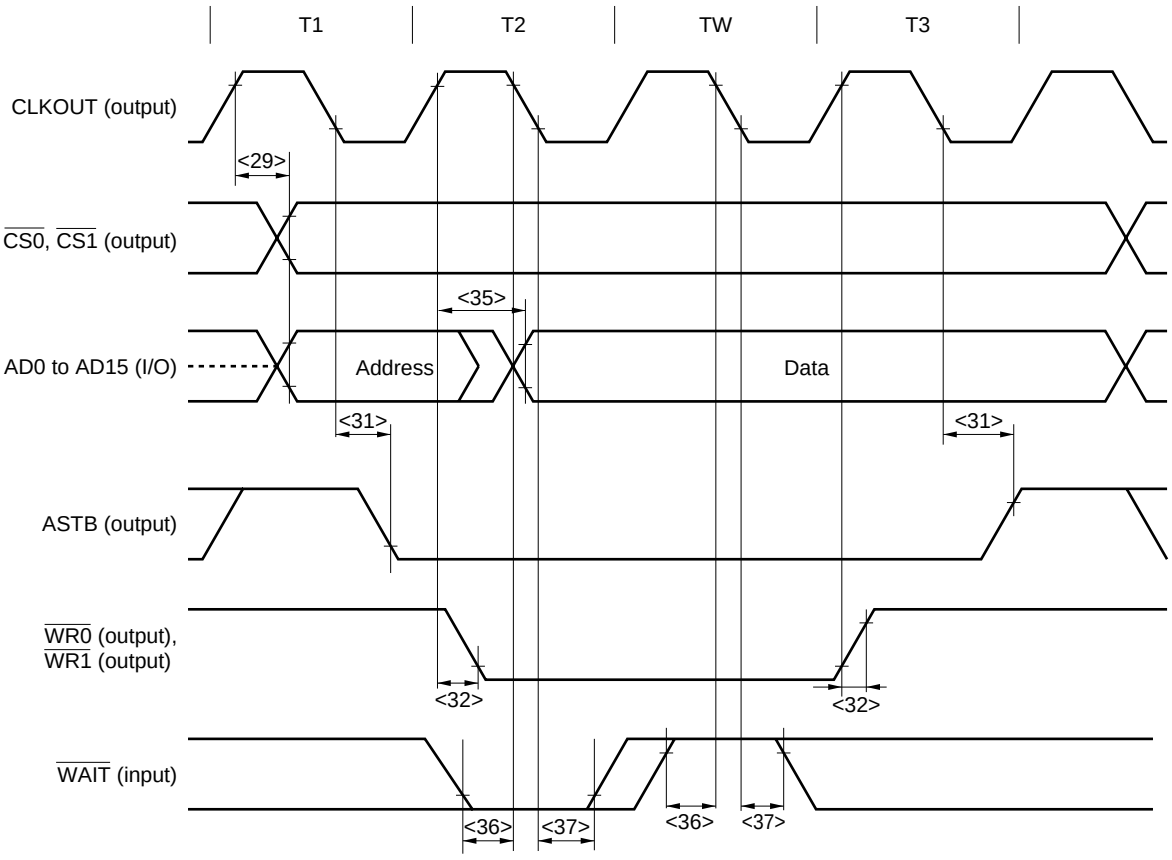
**2.** The values in the above specifications are values for when clocks with a 1:1 duty ratio are input from X1.

**Read Cycle (CLKOUT Synchronous)**




**Remark** WR0 and WR1 are high level.

**Write Cycle (CLKOUT Synchronous)**




**Remark** RD is high level.

### (2) Bus hold

#### (a) CLKOUT asynchronous

**($T_A$ = –40 to +85°C, $V_{DD}$ = $EV_{DD}$ = $AV_{REF0}$ = 4.0 to 5.5 V, $V_{SS}$ = $EV_{SS}$ = $AV_{SS}$ = 0 V, $C_L$ = 50 pF) (1/2)**

| Parameter | Symbol | | Conditions | MIN. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{\text{HLDRQ}}$ high-level width | $t_{WHQH}$ | <78> | | T + 10 | | ns |
| $\overline{\text{HLDAK}}$ low-level width | $t_{WHAL}$ | <79> | | T – 15 | | ns |
| Delay time from $\overline{\text{HLDAK}}\uparrow$ to bus output | $t_{DHAC}$ | <80> | | –40 | | ns |
| Delay time from $\overline{\text{HLDRQ}}\downarrow$ to $\overline{\text{HLDAK}}\downarrow$ | $t_{DHQHA1}$ | <81> | | | (2n + 7.5)T + 40 | ns |
| Delay time from $\overline{\text{HLDRQ}}\uparrow$ to $\overline{\text{HLDAK}}\uparrow$ | $t_{DHQHA2}$ | <82> | | 0.5T | 1.5T + 40 | ns |

**Remarks 1.** T = 1/$f_{CPU}$ ($f_{CPU}$: CPU operating clock frequency)

**2.** n: Number of wait clocks inserted in the bus cycle
The sampling timing changes when a programmable wait is inserted.

**3.** The values in the above specifications are values for when clocks with a 1:1 duty ratio are input from X1.

**($T_A$ = –40 to +85°C, $V_{DD}$ = $EV_{DD}$ = $AV_{REF0}$ = 2.7 to 5.5 V, $V_{SS}$ = $EV_{SS}$ = $AV_{SS}$ = 0 V, $C_L$ = 50 pF) (2/2)**

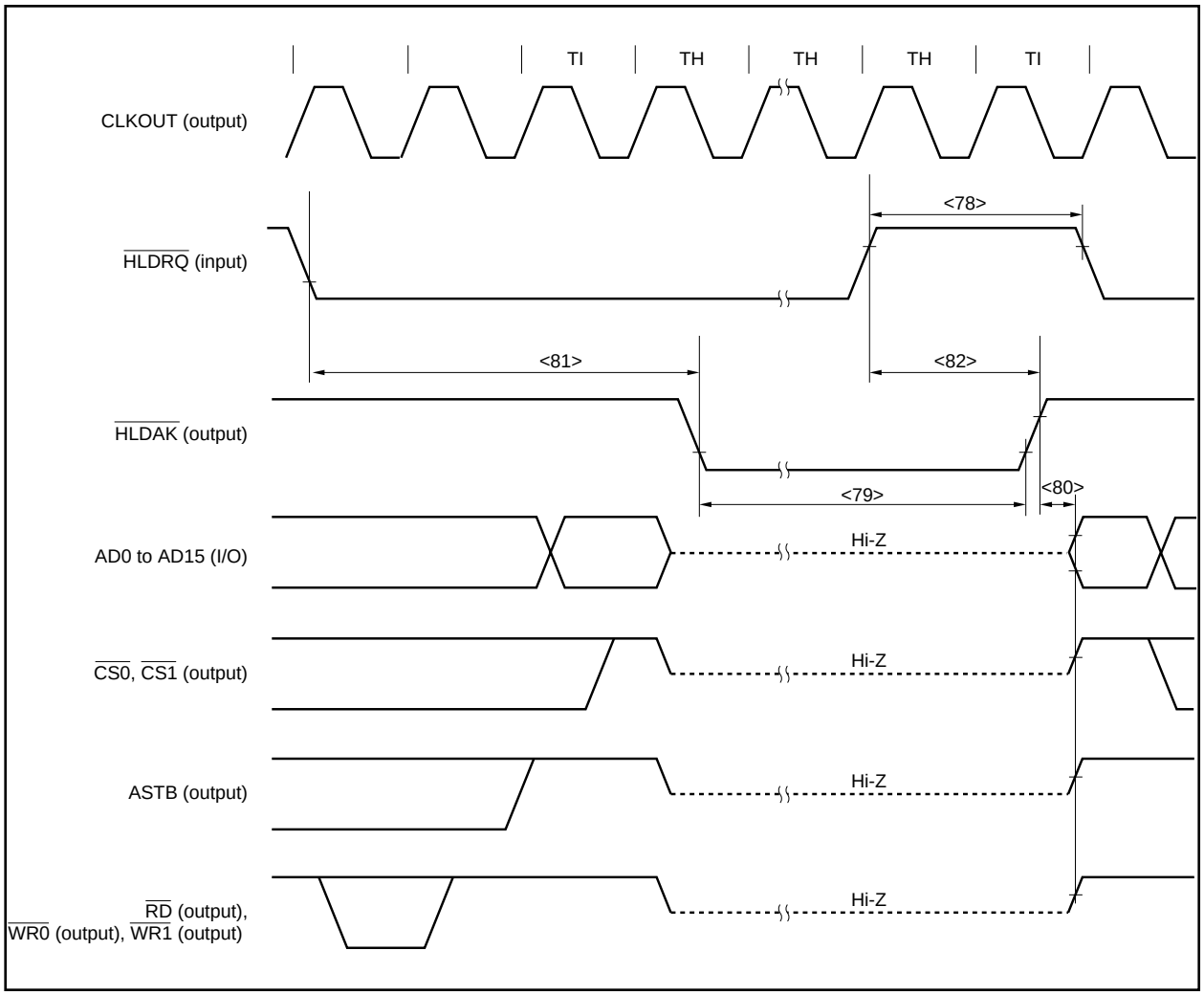
| Parameter | Symbol | | Conditions | MIN. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{\text{HLDRQ}}$ high-level width | $t_{WHQH}$ | <78> | | T + 10 | | ns |
| $\overline{\text{HLDAK}}$ low-level width | $t_{WHAL}$ | <79> | | T – 15 | | ns |
| Delay time from $\overline{\text{HLDAK}}\uparrow$ to bus output | $t_{DHAC}$ | <80> | | –80 | | ns |
| Delay time from $\overline{\text{HLDRQ}}\downarrow$ to $\overline{\text{HLDAK}}\downarrow$ | $t_{DHQHA1}$ | <81> | | | (2n + 7.5)T + 70 | ns |
| Delay time from $\overline{\text{HLDRQ}}\uparrow$ to $\overline{\text{HLDAK}}\uparrow$ | $t_{DHQHA2}$ | <82> | | 0.5T | 1.5T + 70 | ns |

**Remarks 1.** T = 1/$f_{CPU}$ ($f_{CPU}$: CPU operating clock frequency)

**2.** n: Number of wait clocks inserted in the bus cycle
The sampling timing changes when a programmable wait is inserted.

**3.** The values in the above specifications are values for when clocks with a 1:1 duty ratio are input from X1.

### Bus Hold (CLKOUT Asynchronous)



### (b) CLKOUT synchronous

**($T_A$ = –40 to +85°C, $V_{DD}$ = $EV_{DD}$ = $AV_{REF0}$ = 4.0 to 5.5 V, $V_{SS}$ = $EV_{SS}$ = $AV_{SS}$ = 0 V, $C_L$ = 50 pF) (1/2)**

| Parameter | Symbol | | Conditions | MIN. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{\text{HLDRQ}}$ setup time (to CLKOUT↓) | $t_{SHQK}$ | <83> | | 15 | | ns |
| $\overline{\text{HLDRQ}}$ hold time (from CLKOUT↓) | $t_{HKHQ}$ | <84> | | 0 | | ns |
| Delay time from CLKOUT↑ to bus float | $t_{DKF}$ | <85> | | | 20 | ns |
| Delay time from CLKOUT↑ to $\overline{\text{HLDAK}}$ | $t_{DKHA}$ | <86> | | | 20 | ns |

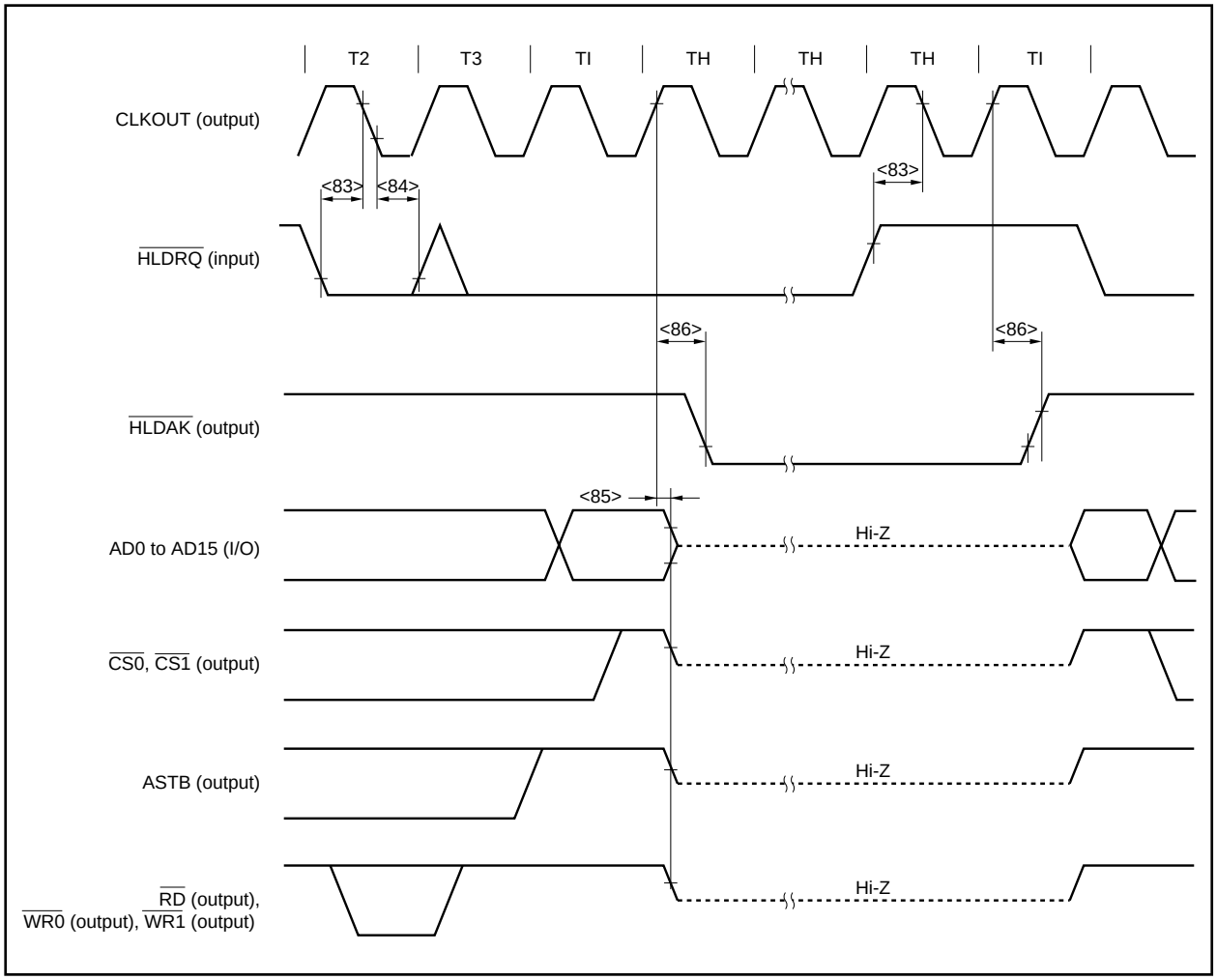
**Remark** The values in the above specifications are values for when clocks with a 1:1 duty ratio are input from X1.

**($T_A$ = –40 to +85°C, $V_{DD}$ = $EV_{DD}$ = $AV_{REF0}$ = 2.7 to 5.5 V, $V_{SS}$ = $EV_{SS}$ = $AV_{SS}$ = 0 V, $C_L$ = 50 pF) (2/2)**

| Parameter | Symbol | | Conditions | MIN. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{\text{HLDRQ}}$ setup time (to CLKOUT↓) | $t_{SHQK}$ | <83> | | 25 | | ns |
| $\overline{\text{HLDRQ}}$ hold time (from CLKOUT↓) | $t_{HKHQ}$ | <84> | | 0 | | ns |
| Delay time from CLKOUT↑ to bus float | $t_{DKF}$ | <85> | | | 40 | ns |
| Delay time from CLKOUT↑ to $\overline{\text{HLDAK}}$ | $t_{DKHA}$ | <86> | | | 40 | ns |

**Remark** The values in the above specifications are values for when clocks with a 1:1 duty ratio are input from X1.

### Bus Hold (CLKOUT Synchronous)



**Basic Operation**

**(1) Reset/external interrupt timing**

**($T_A$ = –40 to +85°C, $V_{DD}$ = $EV_{DD}$ = $AV_{REF0}$ = 2.7 to 5.5 V, $V_{SS}$ = $EV_{SS}$ = $AV_{SS}$ = 0 V, $C_L$ = 50 pF)**

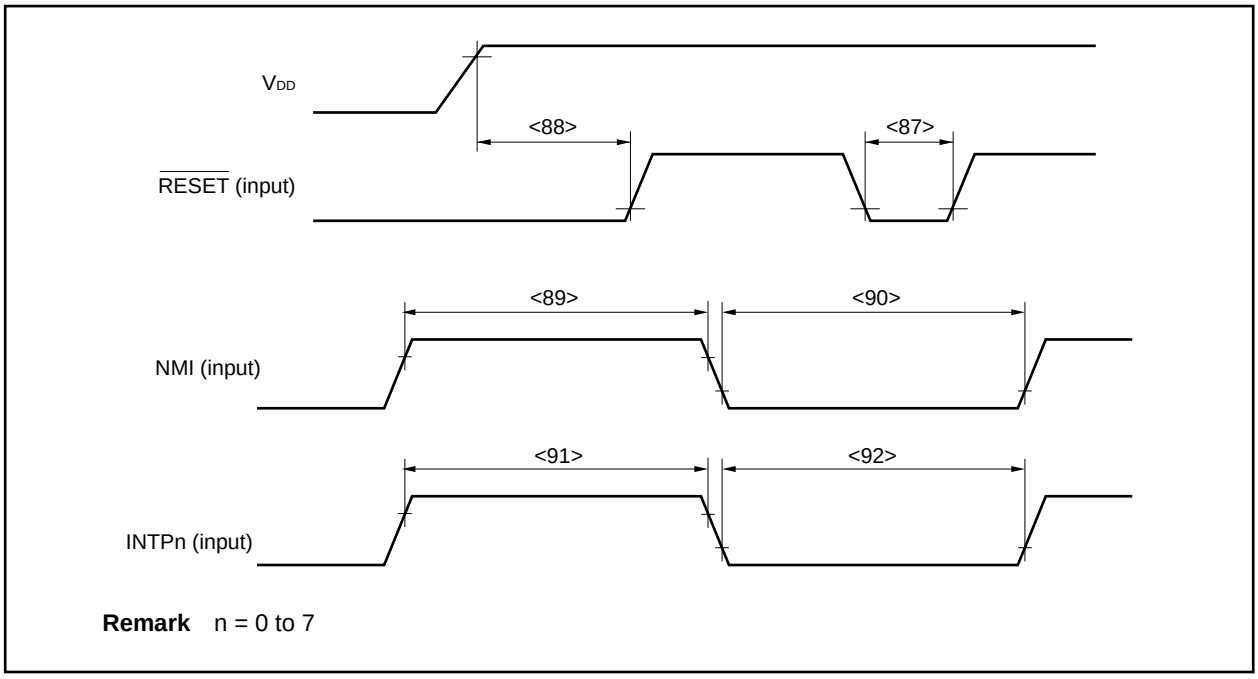
| Parameter | Symbol | | Conditions | | MIN. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| $\overline{\text{RESET}}$ low-level width[Note] | $t_{WRSL1}$ | <87> | Reset in power-on status | When digital noise elimination not selected | 2 | | μs |
| | | | | When digital noise elimination selected | Nr × $t_{RSMP}$ + 2 | | μs |
| | $t_{WRSL2}$ | <88> | Power-on reset | | 3 | | ms |
| NMI high-level width | $t_{WNIH}$ | <89> | Analog noise elimination | | 1 | | μs |
| NMI low-level width | $t_{WNIL}$ | <90> | Analog noise elimination | | 1 | | μs |
| INTPn high-level width | $t_{WITH}$ | <91> | n = 0 to 7 (analog noise elimination) | | 600 | | ns |
| | | | n = 3 (when digital noise elimination selected) | | Ni × $t_{ISMP}$ + 200 | | ns |
| INTPn low-level width | $t_{WITL}$ | <92> | n = 0 to 7 (analog noise elimination) | | 600 | | ns |
| | | | n = 3 (when digital noise elimination selected) | | Ni × $t_{ISMP}$ + 200 | | ns |

**Note** The $\overline{\text{RESET}}$ low-level width is when the $\overline{\text{RESET}}$ pin input is valid (when POCRES is invalid).

**Remarks 1.** Nr: Number of samplings
$t_{RSMP}$: Digital noise elimination sampling clock cycle of $\overline{\text{RESET}}$ pin
Ni: Number of samplings
$t_{ISMP}$: Digital noise elimination sampling clock cycle of INTP3 pin

**2.** The above specification shows the pulse width that is accurately detected as a valid edge. If a pulse narrower than the above specification is input, therefore, it may also be detected as a valid edge.

**Reset/Interrupt**




**Remark** n = 0 to 7

**Timer Timing**

**($T_A$ = –40 to +85°C, $V_{DD}$ = $EV_{DD}$ = $AV_{REF0}$ = 2.7 to 5.5 V, $V_{SS}$ = $EV_{SS}$ = $AV_{SS}$ = 0 V, $C_L$ = 50 pF)**

| Parameter | Symbol | | Conditions | MIN. | MAX. | Unit |
|---|---|---|---|---|---|---|
| TI0n high-level width | $t_{TI0H}$ | <93> | REGC = $V_{DD}$ = 4.0 to 5.5 V | $2T_{smp0}$ + 100 Note 1 | | ns |
| | | | REGC = Capacity, $V_{DD}$ = 4.0 to 5.5 V, REGC = $V_{DD}$ = 2.7 to 5.5 V | $2T_{smp0}$ + 200 Note 1 | | ns |
| TI0n low-level width | $t_{TI0L}$ | <94> | REGC = $V_{DD}$ = 4.0 to 5.5 V | $2T_{smp0}$ + 100 Note 1 | | ns |
| | | | REGC = Capacity, $V_{DD}$ = 4.0 to 5.5 V, REGC = $V_{DD}$ = 2.7 to 5.5 V | $2T_{smp0}$ + 200 Note 1 | | ns |
| TI5m high-level width | $t_{TI5H}$ | <95> | REGC = $V_{DD}$ = 4.0 to 5.5 V | 50 | | ns |
| | | | REGC = Capacity, $V_{DD}$ = 4.0 to 5.5 V, REGC = $V_{DD}$ = 2.7 to 5.5 V | 100 | | ns |
| TI5m low-level width | $t_{TI5L}$ | <96> | REGC = $V_{DD}$ = 4.0 to 5.5 V | 50 | | ns |
| | | | REGC = Capacity, $V_{DD}$ = 4.0 to 5.5 V, REGC = $V_{DD}$ = 2.7 to 5.5 V | 100 | | ns |
| TIP0m high-level width | $t_{TIPH}$ | <97> | REGC = $V_{DD}$ = 4.0 to 5.5 V | $np \times T_{smpp}$ + 100 Note 2 | | ns |
| | | | REGC = Capacity, $V_{DD}$ = 4.0 to 5.5 V, REGC = $V_{DD}$ = 2.7 to 5.5 V | $np \times T_{smpp}$ + 200 Note 2 | | ns |
| TIP0m low-level width | $t_{TIPL}$ | <98> | REGC = $V_{DD}$ = 4.0 to 5.5 V | $np \times T_{smpp}$ + 100 Note 2 | | ns |
| | | | REGC = Capacity, $V_{DD}$ = 4.0 to 5.5 V, REGC = $V_{DD}$ = 2.7 to 5.5 V | $np \times T_{smpp}$ + 200 Note 2 | | ns |

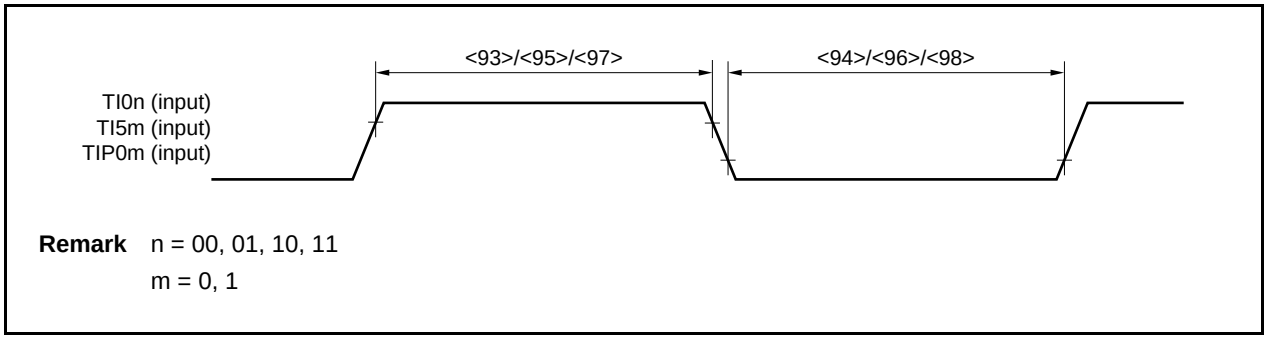
**Notes 1.** $T_{smp0}$: Timer 0 count clock cycle
However, $T_{smp0} = 4/f_{XX}$ when TI0n is used as an external clock.

**2.** $T_{smpp}$: Digital noise elimination sampling clock cycle of TIP0m pin
If TIP00 is used as an external event count input or an external trigger input, however, $T_{smpp}$ = 0 (digital noise is not eliminated).

**Remarks 1.** n = 00, 01, 10, 11
m = 0, 1

**2.** The above specification shows the pulse width that is accurately detected as a valid edge. If a pulse narrower than the above specification is input, therefore, it may also be detected as a valid edge.

**Timer Input Timing**




**Remark** n = 00, 01, 10, 11
m = 0, 1

### UART Timing

**($T_A$ = –40 to +85°C, $V_{DD}$ = $EV_{DD}$ = $AV_{REF0}$ = 2.7 to 5.5 V, $V_{SS}$ = $EV_{SS}$ = $AV_{SS}$ = 0 V, $C_L$ = 50 pF)**

| Parameter | Symbol | Conditions | MIN. | MAX. | Unit |
|---|---|---|---|---|---|
| Transmit rate | | | | 312.5 | kbps |
| ASCK0 frequency | | REGC = $V_{DD}$ = 4.0 to 5.5 V | | 12 | MHz |
| | | REGC = Capacity, $V_{DD}$ = 4.0 to 5.5 V, REGC = $V_{DD}$ = 2.7 to 5.5 V | | 6 | MHz |

### CSI0 Timing

#### (1) Master mode

**($T_A$ = –40 to +85°C, $V_{DD}$ = $EV_{DD}$ = $AV_{REF0}$ = 2.7 to 5.5 V, $V_{SS}$ = $EV_{SS}$ = $AV_{SS}$ = 0 V, $C_L$ = 50 pF)**

| Parameter | Symbol | | Conditions | MIN. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{SCK0n}$ cycle time | $t_{KCY1}$ | <99> | REGC = $V_{DD}$ = 4.0 to 5.5 V | 200 | | ns |
| | | | REGC = Capacity, $V_{DD}$ = 4.0 to 5.5 V, REGC = $V_{DD}$ = 2.7 to 5.5 V | 400 | | ns |
| $\overline{SCK0n}$ high-/low-level width | $t_{KH1}$, $t_{KL1}$ | <100> | | $t_{KCY1}$/2 – 30 | | ns |
| SI0n setup time (to $\overline{SCK0n}$) | $t_{SIK1}$ | <101> | REGC = $V_{DD}$ = 4.0 to 5.5 V | 30 | | ns |
| | | | REGC = Capacity, $V_{DD}$ = 4.0 to 5.5 V, REGC = $V_{DD}$ = 2.7 to 5.5 V | 50 | | ns |
| SI0n hold time (from $\overline{SCK0n}$) | $t_{KSI1}$ | <102> | REGC = $V_{DD}$ = 5 V ±10% | 30 | | ns |
| | | | REGC = Capacity, $V_{DD}$ = 4.0 to 5.5 V, REGC = $V_{DD}$ = 2.7 to 5.5 V | 50 | | ns |
| Delay time from $\overline{SCK0n}$ to SO0n output | $t_{KSO1}$ | <103> | REGC = $V_{DD}$ = 4.0 to 5.5 V | | 30 | ns |
| | | | REGC = Capacity, $V_{DD}$ = 4.0 to 5.5 V, REGC = $V_{DD}$ = 2.7 to 5.5 V | | 60 | ns |

**Remark** n = 0, 1

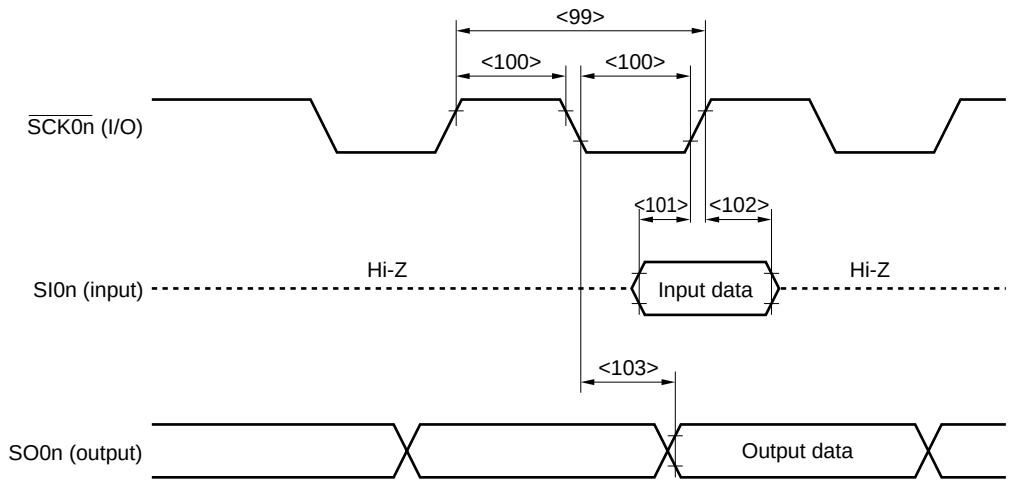
#### (2) Slave mode

**($T_A$ = –40 to +85°C, $V_{DD}$ = $EV_{DD}$ = $AV_{REF0}$ = 2.7 to 5.5 V, $V_{SS}$ = $EV_{SS}$ = $AV_{SS}$ = 0 V, $C_L$ = 50 pF)**

| Parameter | Symbol | | Conditions | MIN. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{SCK0n}$ cycle time | $t_{KCY2}$ | <99> | REGC = $V_{DD}$ = 4.0 to 5.5 V | 200 | | ns |
| | | | REGC = Capacity, $V_{DD}$ = 4.0 to 5.5 V, REGC = $V_{DD}$ = 2.7 to 5.5 V | 400 | | ns |
| $\overline{SCK0n}$ high-/low-level width | $t_{KH2}$, $t_{KL2}$ | <100> | REGC = $V_{DD}$ = 4.0 to 5.5 V | 45 | | ns |
| | | | REGC = Capacity, $V_{DD}$ = 4.0 to 5.5 V, REGC = $V_{DD}$ = 2.7 to 5.5 V | 90 | | ns |
| SI0n setup time (to $\overline{SCK0n}$) | $t_{SIK2}$ | <101> | REGC = $V_{DD}$ = 4.0 to 5.5 V | 30 | | ns |
| | | | REGC = Capacity, $V_{DD}$ = 4.0 to 5.5 V, REGC = $V_{DD}$ = 2.7 to 5.5 V | 60 | | ns |
| SI0n hold time (from $\overline{SCK0n}$) | $t_{KSI2}$ | <102> | REGC = $V_{DD}$ = 4.0 to 5.5 V | 30 | | ns |
| | | | REGC = Capacity, $V_{DD}$ = 4.0 to 5.5 V, REGC = $V_{DD}$ = 2.7 to 5.5 V | 60 | | ns |
| Delay time from $\overline{SCK0n}$ to SO0n output | $t_{KSO2}$ | <103> | REGC = $V_{DD}$ = 4.0 to 5.5 V | | 50 | ns |
| | | | REGC = Capacity, $V_{DD}$ = 4.0 to 5.5 V, REGC = $V_{DD}$ = 2.7 to 5.5 V | | 100 | ns |

**Remark** n = 0, 1

**CSI0 Timing**




**Remarks 1.** When transmit/receive type 1 (CSICn.CKPn, CSICn.DAPn bits = 00)

**2.** n = 0, 1

### CSIA Timing

#### (1) Master mode

**($T_A$ = –40 to +85°C, $V_{DD}$ = $EV_{DD}$ = $AV_{REF0}$ = 2.7 to 5.5 V, $V_{SS}$ = $EV_{SS}$ = $AV_{SS}$ = 0 V, $C_L$ = 50 pf)**

| Parameter | Symbol | | Conditions | MIN. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{SCKA0}$ cycle time | $t_{KCY3}$ | <104> | REGC = $V_{DD}$ = 4.0 to 5.5 V | 500 | | ns |
| | | | REGC = Capacity, $V_{DD}$ = 4.0 to 5.5 V, REGC = $V_{DD}$ = 2.7 to 5.5 V | 1000 | | ns |
| $\overline{SCKA0}$ high-/low-level width | $t_{KH3}$, $t_{KL3}$ | <105> | | $t_{KCY3}/2 - 30$ | | ns |
| SIA0 setup time (to $\overline{SCKA0}$↑) | $t_{SIK3}$ | <106> | REGC = $V_{DD}$ = 4.0 to 5.5 V | 30 | | ns |
| | | | REGC = Capacity, $V_{DD}$ = 4.0 to 5.5 V, REGC = $V_{DD}$ = 2.7 to 5.5 V | 60 | | ns |
| SIA0 hold time (from $\overline{SCKA0}$↑) | $t_{KSI3}$ | <107> | REGC = $V_{DD}$ = 4.0 to 5.5 V | 30 | | ns |
| | | | REGC = Capacity, $V_{DD}$ = 4.0 to 5.5 V, REGC = $V_{DD}$ = 2.7 to 5.5 V | 60 | | ns |
| Delay time from $\overline{SCKA0}$↓ to SOA0 output | $t_{KSO3}$ | <108> | REGC = $V_{DD}$ = 4.0 to 5.5 V | | 30 | ns |
| | | | REGC = Capacity, $V_{DD}$ = 4.0 to 5.5 V, REGC = $V_{DD}$ = 2.7 to 5.5 V | | 60 | ns |

#### (2) Slave mode

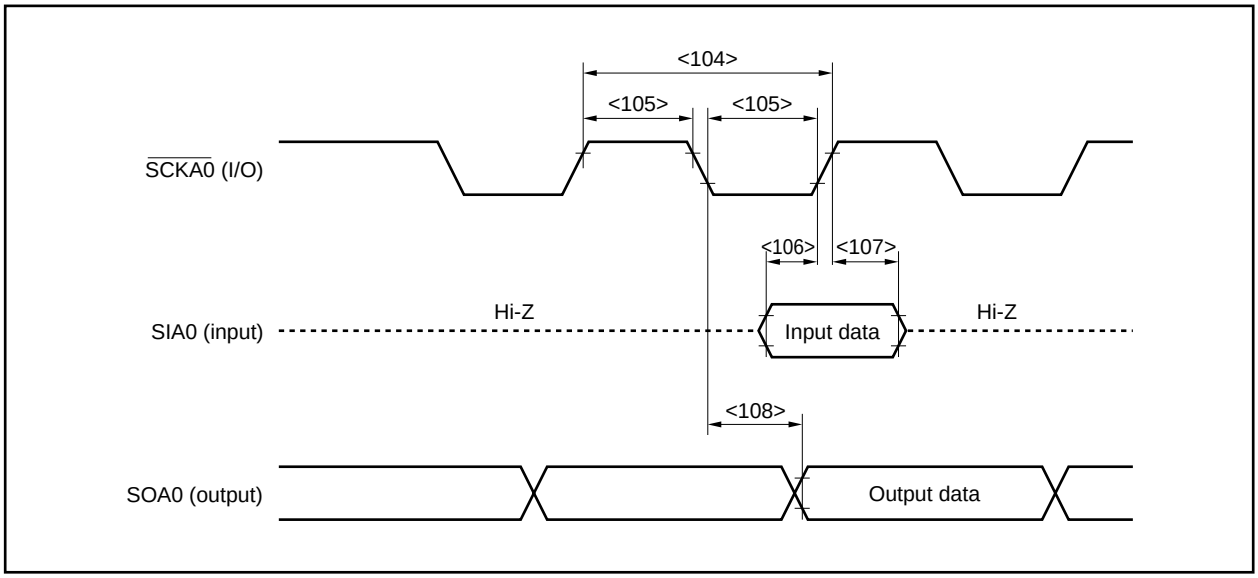
**($T_A$ = –40 to +85°C, $V_{DD}$ = $EV_{DD}$ = $AV_{REF0}$ = 2.7 to 5.5 V, $V_{SS}$ = $EV_{SS}$ = $AV_{SS}$ = 0 V, $C_L$ = 50 pf)**

| Parameter | Symbol | | Conditions | MIN. | MAX. | Unit |
|---|---|---|---|---|---|---|
| $\overline{SCKA0}$ cycle time | $t_{KCY4}$ | <104> | REGC = $V_{DD}$ = 4.0 to 5.5 V | 840 | | ns |
| | | | REGC = Capacity, $V_{DD}$ = 4.0 to 5.5 V, REGC = $V_{DD}$ = 2.7 to 5.5 V | 1700 | | ns |
| $\overline{SCKA0}$ high-/low-level width | $t_{KH4}$, $t_{KL4}$ | <105> | | $t_{KCY4}/2 - 30$ | | ns |
| SIA0 setup time (to $\overline{SCKA0}$↑) | $t_{SIK4}$ | <106> | REGC = $V_{DD}$ = 4.0 to 5.5 V | 50 | | ns |
| | | | REGC = Capacity, $V_{DD}$ = 4.0 to 5.5 V, REGC = $V_{DD}$ = 2.7 to 5.5 V | 100 | | ns |
| SIA0 hold time (from $\overline{SCKA0}$↑) | $t_{KSI4}$ | <107> | REGC = $V_{DD}$ = 4.0 to 5.5 V | $t_{CY} \times 2 + 15$[Note] | | ns |
| | | | REGC = Capacity, $V_{DD}$ = 4.0 to 5.5 V, REGC = $V_{DD}$ = 2.7 to 5.5 V | $t_{CY} \times 2 + 30$[Note] | | ns |
| Delay time from $\overline{SCKA0}$↓ to SOA0 output | $t_{KSO4}$ | <108> | REGC = $V_{DD}$ = 4.0 to 5.5 V | | $t_{CY} \times 2 + 30$[Note] | ns |
| | | | REGC = Capacity, $V_{DD}$ = 4.0 to 5.5 V, REGC = $V_{DD}$ = 2.7 to 5.5 V | | $t_{CY} \times 2 + 60$[Note] | ns |

**Note** $t_{CY}$: Internal clock output cycle

$f_{XX}$ (CSIS0.CKSA01, CSIS0.CKSA00 bits = 00), $f_{XX}/2$ (CKSA01, CKSA00 bits = 01)

$f_{XX}/2^2$ (CKSA01, CKSA00 bits = 10), $f_{XX}/2^3$ (CKSA01, CKSA00 bits = 11)


<104>
<105>
<105>
SCKA0 (I/O)
<106>
<107>
Hi-Z
SIA0 (input)
Input data
Hi-Z
<108>
SOA0 (output)
Output data

**$I^2C$ Bus Mode ($\mu$PD703308Y, 70F3306Y, 70F3308Y Only)**

**($T_A$ = –40 to +85°C, $V_{DD}$ = $EV_{DD}$ = $AV_{REF0}$ = 2.7 to 5.5 V, $V_{SS}$ = $EV_{SS}$ = $AV_{SS}$ = 0 V, $C_L$ = 50 pF)**

| Parameter | | Symbol | | Normal Mode | | High-Speed Mode | | Unit |
|---|---|---|---|---|---|---|---|---|
| | | | | MIN. | MAX. | MIN. | MAX. | |
| SCL0 clock frequency | | $f_{CLK}$ | | 0 | 100 | 0 | 400 | kHz |
| Bus free time (Between start and stop conditions) | | $t_{BUF}$ | <109> | 4.7 | – | 1.3 | – | $\mu$s |
| Hold time[Note 1] | | $t_{HD:STA}$ | <110> | 4.0 | – | 0.6 | – | $\mu$s |
| SCL0 clock low-level width | | $t_{LOW}$ | <111> | 4.7 | – | 1.3 | – | $\mu$s |
| SCL0 clock high-level width | | $t_{HIGH}$ | <112> | 4.0 | – | 0.6 | – | $\mu$s |
| Setup time for start/restart conditions | | $t_{SU:STA}$ | <113> | 4.7 | – | 0.6 | – | $\mu$s |
| Data hold time | CBUS compatible master | $t_{HD:DAT}$ | <114> | 5.0 | – | – | – | $\mu$s |
| | $I^2C$ mode | | | 0[Note 2] | – | 0[Note 2] | 0.9[Note 3] | $\mu$s |
| Data setup time | | $t_{SU:DAT}$ | <115> | 250 | – | 100[Note 4] | – | ns |
| SDA0 and SCL0 signal rise time | | $t_R$ | <116> | – | 1000 | 20 + 0.1Cb[Note 5] | 300 | ns |
| SDA0 and SCL0 signal fall time | | $t_F$ | <117> | – | 300 | 20 + 0.1Cb[Note 5] | 300 | ns |
| Stop condition setup time | | $t_{SU:STO}$ | <118> | 4.0 | – | 0.6 | – | $\mu$s |
| Pulse width of spike suppressed by input filter | | $t_{SP}$ | <119> | – | – | 0 | 50 | ns |
| Capacitance load of each bus line | | Cb | | – | 400 | – | 400 | pF |

**Notes 1.** At the start condition, the first clock pulse is generated after the hold time.

**2.** The system requires a minimum of 300 ns hold time internally for the SDA0 signal (at $V_{IHmin.}$ of SCL0 signal) in order to occupy the undefined area at the falling edge of SCL0.

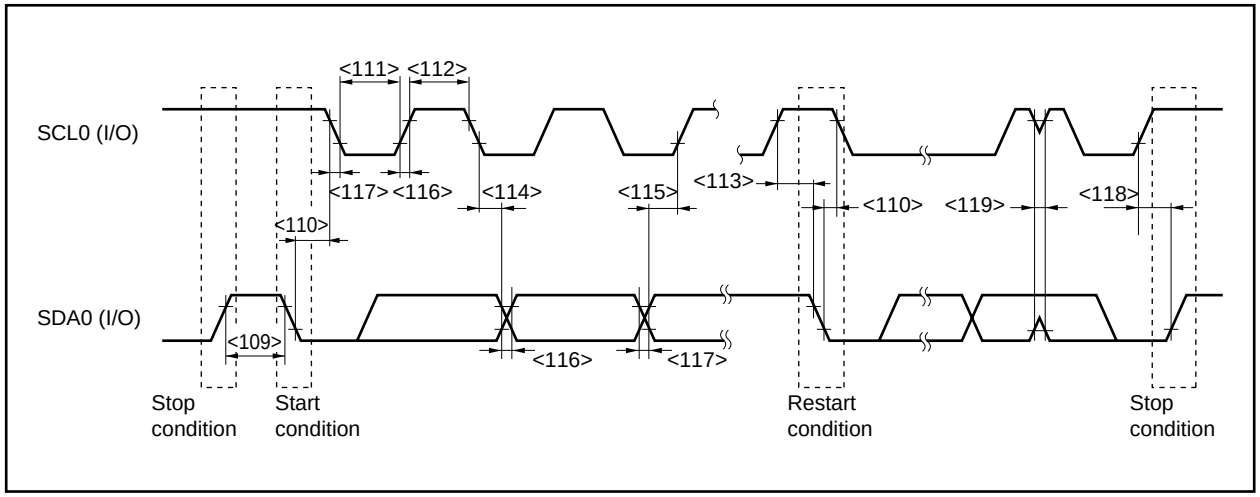
**3.** If the system does not extend the SCL0 signal low hold time ($t_{LOW}$), only the maximum data hold time ($t_{HD:DAT}$) needs to be satisfied.

**4.** The high-speed mode $I^2C$ bus can be used in the normal-mode $I^2C$ bus system. In this case, set the high-speed mode $I^2C$ bus so that it meets the following conditions.

- If the system does not extend the SCL0 signal's low state hold time:
  $t_{SU:DAT} \geq 250$ ns
- If the system extends the SCL0 signal's low state hold time:
  Transmit the following data bit to the SDA0 line prior to the SCL0 line release ($t_{Rmax.}$ + $t_{SU:DAT}$ = 1000 + 250 = 1250 ns: Normal mode $I^2C$ bus specification).

**5.** Cb: Total capacitance of one bus line (unit: pF)

**I²C Bus Mode (μPD703308Y, 70F3306Y, 70F3308Y Only)**



### A/D Converter

**($T_A$ = –40 to +85°C, $V_{DD}$ = $EV_{DD}$ = $AV_{REF0}$ = 2.7 to 5.5 V, $V_{SS}$ = $EV_{SS}$ = $AV_{SS}$ = 0 V)**

| Parameter | Symbol | Conditions | | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Resolution | | | | 10 | 10 | 10 | bit |
| Overall error[Note 1] | AINL | 4.0 ≤ $AV_{REF0}$ ≤ 5.5 V | | | ±0.2 | ±0.4 | %FSR |
| | | 2.7 ≤ $AV_{REF0}$ ≤ 4.0 V | | | ±0.3 | ±0.6 | %FSR |
| Conversion time | $t_{CONV}$ | 4.5 ≤ $AV_{REF0}$ ≤ 5.5 V | High-speed mode | 3.0 | | 100 | μs |
| | | | Normal mode | 14.0 | | 100 | μs |
| | | 4.0 ≤ $AV_{REF0}$ ≤ 4.5 V | High-speed mode | 4.8 | | 100 | μs |
| | | | Normal mode | 14.0 | | 100 | μs |
| | | 2.85 ≤ $AV_{REF0}$ ≤ 4.0 V | High-speed mode | 6.0 | | 100 | μs |
| | | | Normal mode | 17.0 | | 100 | μs |
| | | 2.7 ≤ $AV_{REF0}$ ≤ 2.85 V | High-speed mode | 14.0 | | 100 | μs |
| | | | Normal mode | 17.0 | | 100 | μs |
| Zero-scale error[Note 1] | $E_{ZS}$ | 4.0 ≤ $AV_{REF0}$ ≤ 5.5 V | | | | ±0.4 | %FSR |
| | | 2.7 ≤ $AV_{REF0}$ ≤ 4.0 V | | | | ±0.6 | %FSR |
| Full-scale error[Note 1] | $E_{fs}$ | 4.0 ≤ $AV_{REF0}$ ≤ 5.5 V | | | | ±0.4 | %FSR |
| | | 2.7 ≤ $AV_{REF0}$ ≤ 4.0 V | | | | ±0.6 | %FSR |
| Non-linearity error[Note 2] | ILE | 4.0 ≤ $AV_{REF0}$ ≤ 5.5 V | | | | ±2.5 | LSB |
| | | 2.7 ≤ $AV_{REF0}$ ≤ 4.0 V | | | | ±4.5 | LSB |
| Differential linearity error[Note 2] | DLE | 4.0 ≤ $AV_{REF0}$ ≤ 5.5 V | | | | ±1.5 | LSB |
| | | 2.7 ≤ $AV_{REF0}$ ≤ 4.0 V | | | | ±2.0 | LSB |
| Analog input voltage | $V_{IAN}$ | | | 0 | | $AV_{REF0}$ | V |
| $AV_{REF0}$ current | $IA_{REF0}$ | When using A/D converter | | | 1.3 | 2.5 | mA |
| | | When not using A/D converter | | | 1.0 | T.B.D. | μA |

**Notes 1.** Excluding quantization error (±0.05 %FSR).

**2.** Excluding quantization error (±0.5 LSB).

**Remark** LSB: Least Significant Bit
FSR: Full Scale Range

### Power-on-Clear Circuit Characteristics

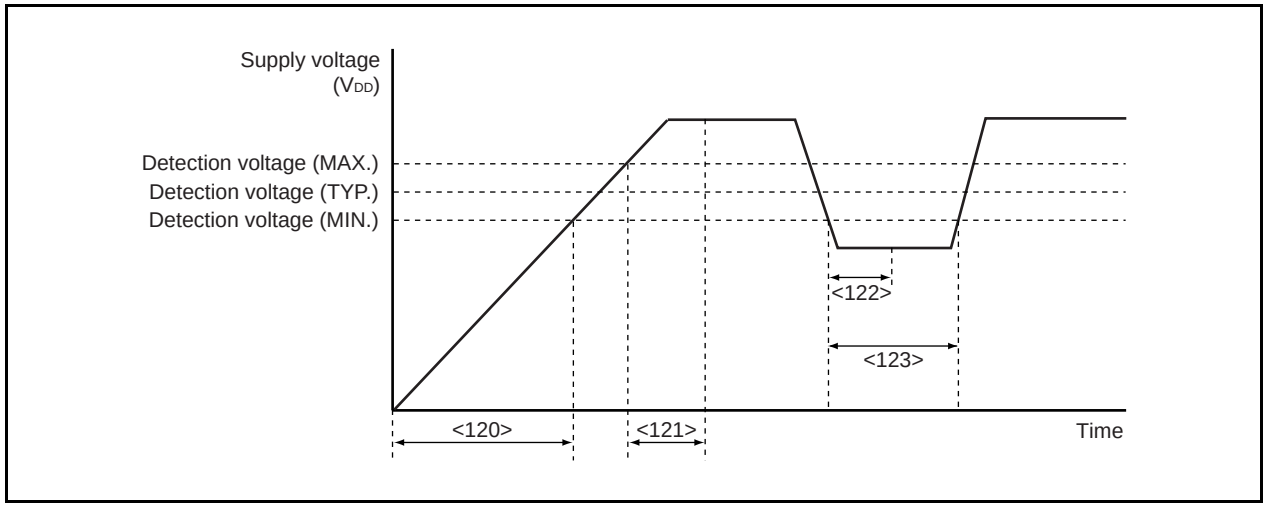
**($T_A$ = –40 to +85°C)**

| Parameter | Symbol | | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Detection voltage | $V_{POC}$ | | | 2.5 | 2.6 | 2.7 | V |
| Power supply rise time | $t_{PTH}$ | <120> | $V_{DD}$ = 0 → 2.5 V | 3 | | | μs |
| Response time 1[Note 1] | $t_{PTHD}$ | <121> | After voltage reaches detection voltage (MAX.) on power application | | | 3.0 | ms |
| Response time 2[Note 2] | $t_{PD}$ | <122> | When power supply drops | | | 1.0 | ms |
| Minimum pulse width | $t_{PW}$ | <123> | | 0.2 | | | ms |

**Notes 1.** Time from when the detection voltage ($V_{POC}$) is detected until the reset signal (POCRES) is released

**2.** Time from when the detection voltage ($V_{POC}$) is detected until the reset signal (POCRES) is generated

### Power-on-Clear Circuit Timing



**Low-Voltage Detector Characteristics**

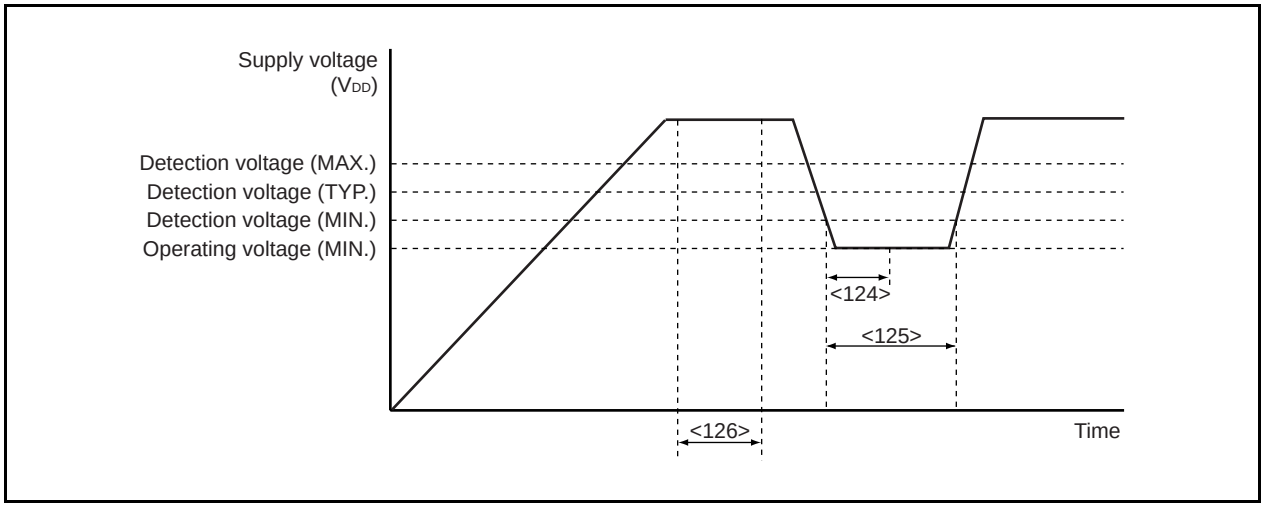
**($T_A$ = –40 to +85°C)**

| Parameter | Symbol | | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Detection voltage | $V_{LVI}$ | | | 4.1 | 4.3 | 4.5 | V |
| | | | | 3.9 | 4.1 | 4.3 | V |
| | | | | 3.7 | 3.9 | 4.1 | V |
| | | | | 3.5 | 3.7 | 3.9 | V |
| | | | | 3.3 | 3.5 | 3.7 | V |
| | | | | 3.15 | 3.3 | 3.45 | V |
| | | | | 2.95 | 3.1 | 3.25 | V |
| Response time[Note 1] | $t_{LD}$ | <124> | | | 0.2 | 2.0 | ms |
| Minimum pulse width | $t_{LW}$ | <125> | | 0.2 | | | ms |
| Operation stabilization wait time[Note 2] | $t_{WAIT1}$ | <126> | | | 0.1 | 0.2 | ms |

**Notes 1.** Time from when the detection voltage ($V_{LVI}$) is detected until an interrupt request signal (INTLVI) or reset signal (LVIRES) is generated

**2.** Time from when the LVIM.LVION bit = 1 until operation is stabilized

**Low-Voltage Detector Timing**



### Flash Memory Programming Characteristics

**($T_A$ = –10 to +65°C, $V_{DD}$ = $EV_{DD}$ = $AV_{REF0}$ = 2.7 to 5.5 V, $V_{SS}$ = $EV_{SS}$ = $AV_{SS}$ = 0 V)**

#### (1) Basic characteristics

| Parameter | Symbol | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|
| Programming operation frequency | $f_{CPU}$ | REGC = $V_{DD}$ = 4.5 to 5.5 V | 2 | | 20 | MHz |
| | | REGC = $V_{DD}$ = 4.0 to 5.5 V | 2 | | 16 | MHz |
| | | REGC = Capacity, $V_{DD}$ = 4.0 to 5.5 V | 2 | | 8 Note 1 | MHz |
| | | REGC = $V_{DD}$ = 2.7 to 5.5 V | 2 | | 8 Note 1 | MHz |
| Supply voltage | $V_{DD}$ | | 2.7 | | 5.5 | V |
| Overall erase time | $t_{ERA}$ | | | T.B.D. | | s |
| Write time | $t_{WHB}$ | | | T.B.D. | | s |
| Number of rewrites | $C_{ERWR}$ | **Note 2** | 100 | | | Times |

**Notes 1.** These values may change after evaluation.

**2.** When writing initially to shipped products, it is also counted as one rewrite for “write only”.

Example (P: Write, E: Erase)

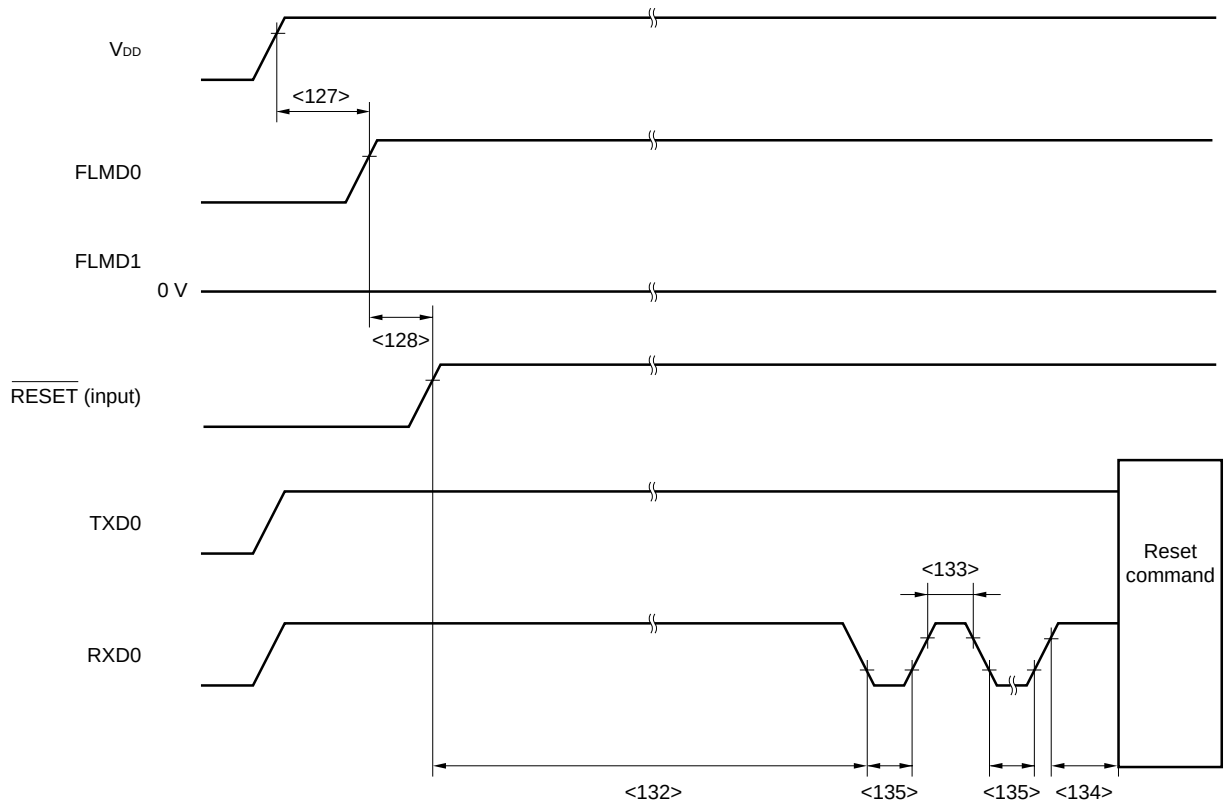
Shipped product ⟶P→E→P→E→P: 3 rewrites

Shipped product → E→P→E→P→E→P: 3 rewrites

#### (2) Serial write operation characteristics

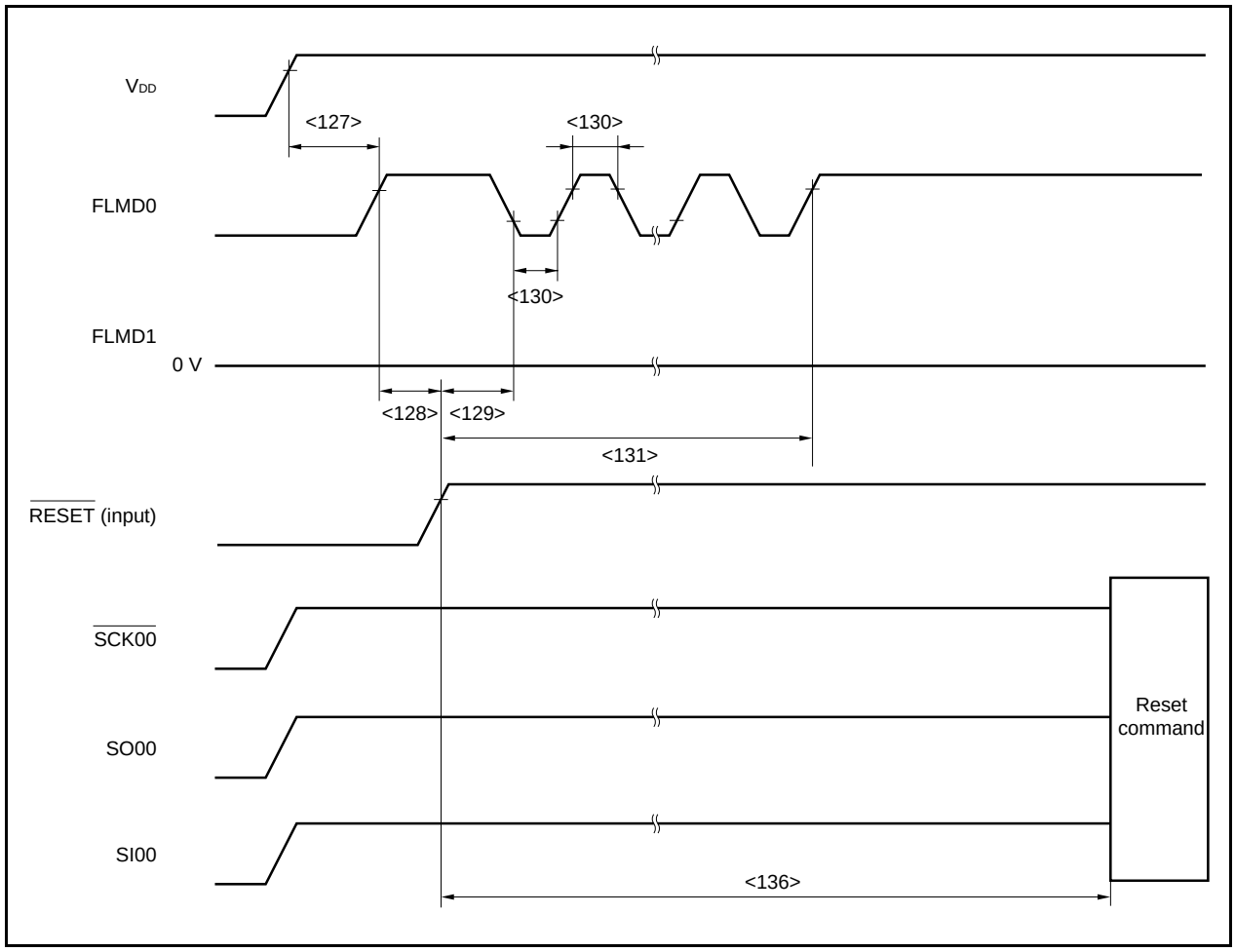
| Parameter | Symbol | | Conditions | MIN. | TYP. | MAX. | Unit |
|---|---|---|---|---|---|---|---|
| Setup time from $V_{DD}$↑ to FLMD0↑ | $t_{DP}$ | <127> | | T.B.D. | | | μs |
| Release time from FLMD0↑ to $\overline{\text{RESET}}$↑ | $t_{PR}$ | <128> | | T.B.D. | | | ms |
| FLMD0 pulse input start time from $\overline{\text{RESET}}$↑ (after securing oscillation stabilization time) | $t_{RP}$ | <129> | | T.B.D. | | | ms |
| FLMD0 pulse high-/low-level width | $t_{PW}$ | <130> | | T.B.D. | | T.B.D. | μs |
| FLMD0 pulse input end time from $\overline{\text{RESET}}$↑ (after securing oscillation stabilization time) | $t_{RPE}$ | <131> | | | | T.B.D. | ms |
| 1st low data input time from $\overline{\text{RESET}}$↑ (after securing oscillation stabilization time) | $t_{R1}$ | <132> | When UART communication is selected | T.B.D. | | | s |
| Time from 1st low data input to 2nd low data input | $t_{12}$ | <133> | When UART communication is selected | T.B.D. | | | s |
| Time from 2nd low data input to reset command input | $t_{2C}$ | <134> | When UART communication is selected | T.B.D. | | | s |
| Low data input width | $t_{L1}$/$t_{L2}$ | <135> | When UART communication is selected | | 9600 | | bps |
| Time from $\overline{\text{RESET}}$↑ (after securing oscillation stabilization time) to reset command input | $t_{RC}$ | <136> | When CSI or CSI-HS communication is selected | T.B.D. | | | s |

**Serial Write Operation Timing (UART)**

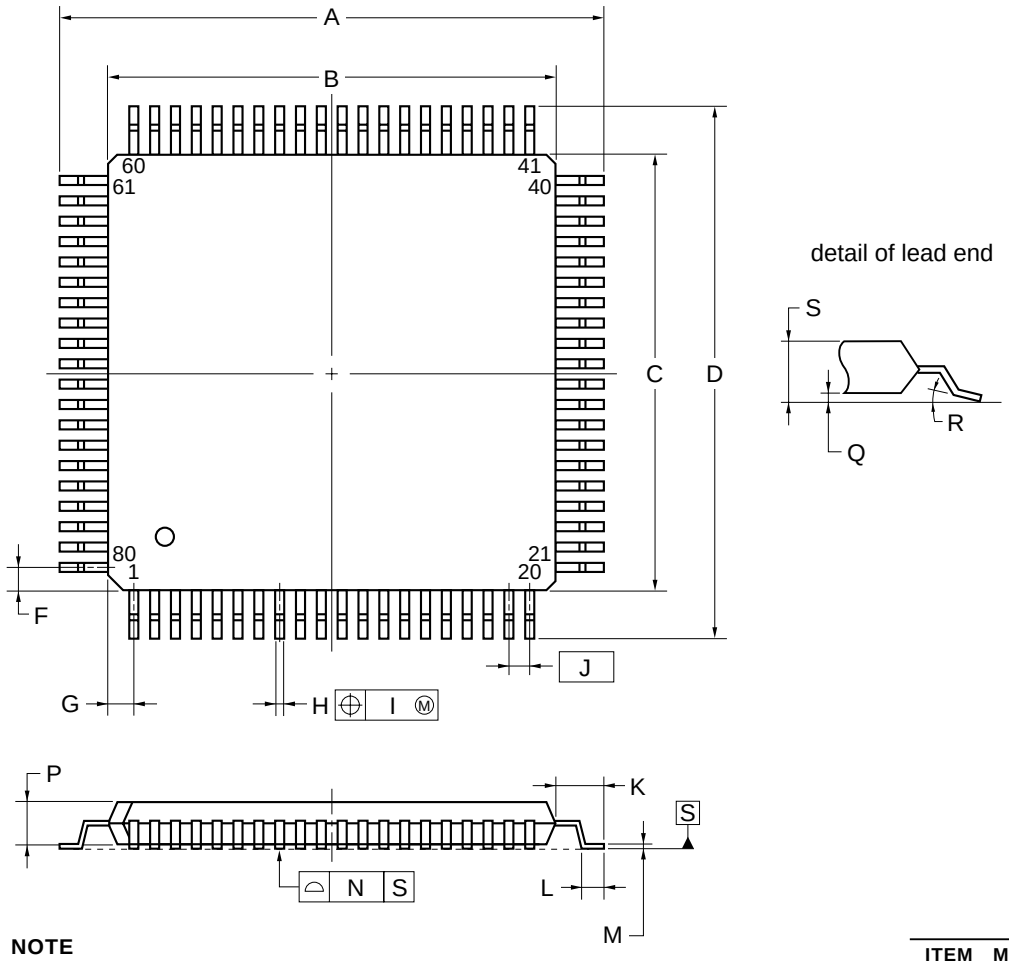



**Remark** The FLMD0 pulse does not have to be input for UART0 communication.

**Serial Write Operation Timing (CSI or CSI-HS)**



## 80-PIN PLASTIC QFP (14x14)



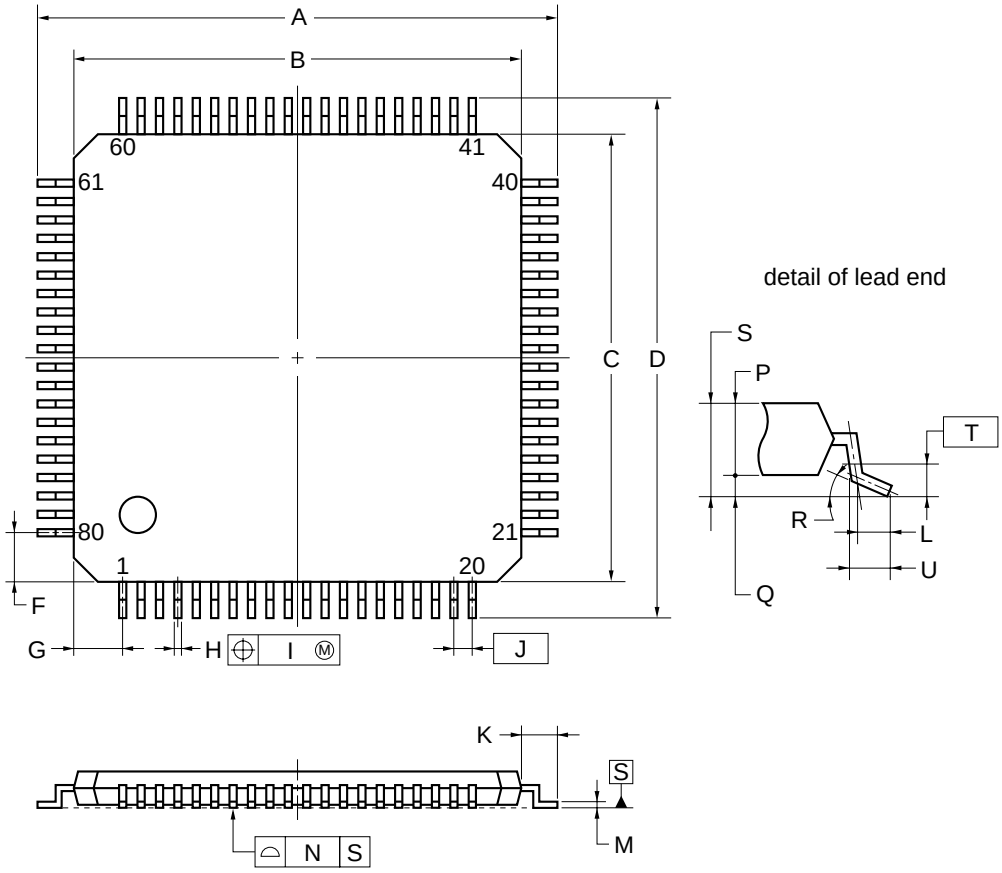

**NOTE**

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**

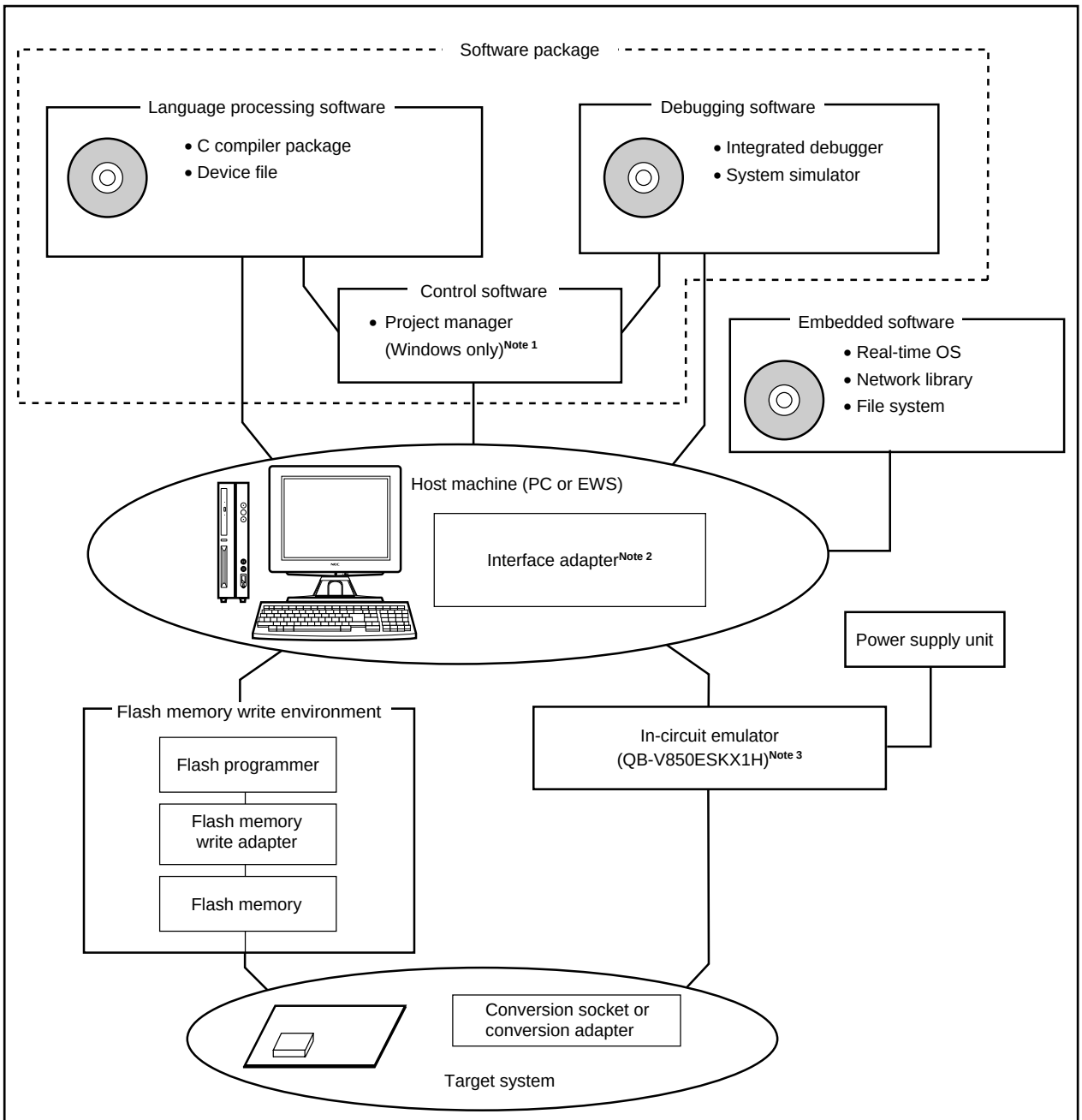
# APPENDIX A DEVELOPMENT TOOLS

The following development tools are available for the development of systems that employ the V850ES/KF1+. Figure A-1 shows the development tool configuration.

- **Support for PC98-NX series**
  Unless otherwise specified, products supported by IBM PC/AT™ compatibles are compatible with PC98-NX series computers. When using PC98-NX series computers, refer to the explanation for IBM PC/AT compatibles.

- **Windows™**
  Unless otherwise specified, “Windows” means the following OSs.
  - Windows 98, 2000
  - Windows Me
  - Windows XP
  - Windows NT™ Ver. 4.0

**Figure A-1. Development Tool Configuration**




**Notes 1.** The project manager PM plus is included in the C compiler package.
The PM plus is only used for Windows.

**2.** QB-V850ESKX1H supports USB only.

**3.** QB-V850ESKX1H is supplied with ID850QB, a device file, and power supply unit. Any other products are sold separately.

## A.1 Software Package

| | |
|---|---|
| SP850<br>V850 Series software package | Development tools (software) common to the V850 Series are combined in this package. |
| | Part number: μS××××SP850 |

**Remark** ×××× in the part number differs depending on the host machine and OS used.

μS××××SP850

| ×××× | Host Machine | OS | Supply Medium |
|---|---|---|---|
| AB17 | PC-9800 series,<br>IBM PC/AT compatibles | Windows (Japanese version) | CD-ROM |
| BB17 | | Windows (English version) | |

## A.2 Language Processing Software

| | |
|---|---|
| CA850<br>C compiler package | This compiler converts programs written in C language into object codes executable with a microcontroller. This compiler is started from project manager PM plus. |
| | Part number: μS××××CA703000 |
| DF703308<br>Device file | This file contains information peculiar to the device.<br>This device file should be used in combination with a tool (CA850, SM850, and ID850QB).<br>The corresponding OS and host machine differ depending on the tool to be used. |

**Remark** ×××× in the part number differs depending on the host machine and OS used.

μS××××CA703000

| ×××× | Host Machine | OS | Supply Medium |
|---|---|---|---|
| AB17 | PC-9800 series,<br>IBM PC/AT compatibles | Windows (Japanese version) | CD-ROM |
| BB17 | | Windows (English version) | |
| 3K17 | SPARCstation™ | SunOS™ (Rel. 4.1.4),<br>Solaris™ (Rel. 2.5.1) | |

## A.3 Control Software

| | |
|---|---|
| PM plus<br>Project manager | This is control software designed to enable efficient user program development in the Windows environment. All operations used in development of a user program, such as starting the editor, building, and starting the debugger, can be performed from the PM plus.<br>**<Caution>**<br>The PM plus is included in the C compiler package CA850.<br>It can only be used in Windows. |

## A.4 Debugging Tools (Hardware)

### A.4.1 When using in-circuit emulator QB-V850ESKX1H

| | |
|---|---|
| QB-V850ESKX1H[Notes 1, 2]<br>In-circuit emulator | The in-circuit emulator serves to debug hardware and software when developing application systems using a V850ES/KF1+ product. It corresponds to the integrated debugger ID850QB. This emulator should be used in combination with a power supply unit and emulation probe. Use USB to connect this emulator to the host machine. |
| Emulation probe for GC package[Note 2]<br>(part number pending) | This probe is used to connect the in-circuit emulator and target system, and is designed for an 80-pin plastic QFP (GC-8BT type). |
| Emulation probe for GK package[Note 2]<br>(part number pending) | This probe is used to connect the in-circuit emulator and target system, and is designed for an 80-pin plastic TQFP (GK-9EU type). |

**Notes 1.** QB-V850ESKX1H is supplied with a power supply unit. It is also supplied with integrated debugger ID850QB and a device file as control software.

**2.** Under development

## A.5 Debugging Tools (Software)

| | |
|---|---|
| SM plus[Note]<br>System simulator | This is a system simulator for the V850 Series. The SM plus is Windows-based software.<br>It is used to perform debugging at the C source level or assembler level while simulating the operation of the target system on a host machine.<br>Use of the SM plus allows the execution of application logical testing and performance testing on an independent basis from hardware development, thereby providing higher development efficiency and software quality.<br>It should be used in combination with the device file (sold separately). |
| | Part number: μS××××SM703100 |
| ID850QB<br>Integrated debugger<br>(supporting in-circuit emulator QB-V850ESKX1H) | This debugger supports the in-circuit emulators for the V850 Series. The ID850QB is Windows-based software.<br>It has improved C-compatible debugging functions and can display the results of tracing with the source program using an integrating window function that associates the source program, disassemble display, and memory display with the trace result.<br>It should be used in combination with the device file (sold separately). |

**Note** Under development

**Remark** ×××× in the part number differs depending on the host machine and OS used.

μS××××SM703100

| ×××× | Host Machine | OS | Supply Medium |
|---|---|---|---|
| AB17 | PC-9800 series,<br>IBM PC/AT compatibles | Windows (Japanese version) | CD-ROM |
| BB17 | | Windows (English version) | |

## A.6 Embedded Software

| | |
|---|---|
| RX850, RX850 Pro<br>Real-time OS | The RX850 and RX850 Pro are real-time OSs conforming to μITRON 3.0 specifications.<br>A tool (configurator) for generating multiple information tables is supplied.<br>RX850 Pro has more functions than RX850. |
| | Part number: μS××××RX703000-ΔΔΔΔ (RX850)<br>μS××××RX703100-ΔΔΔΔ (RX850 Pro) |
| V850mini-NET[Note] (provisional name)<br>(Network library) | This is a network library conforming to RFC.<br>It is a lightweight TCP/IP of compact design, requiring only a small memory.<br>In addition to the TCP/IP standard set, an HTTP server, SMTP client, and POP client are also supported. |
| RX-FS850<br>(File system) | This is a FAT file system function.<br>It is a file system that supports the CD-ROM file system function.<br>This file system is used with the real-time OS RX850 Pro. |

**Note** Under development

**Caution To purchase the RX850 or RX850 Pro, first fill in the purchase application form and sign the user agreement.**

**Remark** ×××× and ΔΔΔΔ in the part number differ depending on the host machine and OS used.

μS××××RX703000-ΔΔΔΔ
μS××××RX703100-ΔΔΔΔ

| ΔΔΔΔ | Product Outline | Maximum Number for Use in Mass Production |
|---|---|---|
| 001 | Evaluation object | Do not use for mass-produced product. |
| 100K | Mass-production object | 0.1 million units |
| 001M | | 1 million units |
| 010M | | 10 million units |
| S01 | Source program | Object source program for mass production |

| ×××× | Host Machine | OS | Supply Medium |
|---|---|---|---|
| AB17 | PC-9800 series,<br>IBM PC/AT compatibles | Windows (Japanese version) | CD-ROM |
| BB17 | | Windows (English version) | |
| 3K17 | SPARCstation | Solaris (Rel. 2.5.1) | |

## A.7 Flash Memory Writing Tools

| | |
|---|---|
| Flashpro IV<br>(part number: PG-FP4)<br>Flash programmer | Flash programmer dedicated to microcontrollers with on-chip flash memory. |
| FA-80GC-8BT-A<br>Flash memory writing adapter | Flash memory writing adapter used connected to the Flashpro IV.<br>• FA-80GC-8BT-A: For 80-pin plastic QFP (GC-8BT type) |
| FA-80GK-9EU-A<br>Flash memory writing adapter | Flash memory writing adapter used connected to the Flashpro IV.<br>• FA-80GK-9EU-A: For 80-pin plastic TQFP (GK-9EU type) |

**Remark** FA-80GC-8BT-A and FA-80GK-9EU-A are products of Naito Densei Machida Mfg. Co., Ltd.
TEL: +81-45-475-4191 Naito Densei Machida Mfg. Co., Ltd.

# APPENDIX B INSTRUCTION SET LIST

## B.1 Conventions

### (1) Register symbols used to describe operands

| Register Symbol | Explanation |
|---|---|
| reg1 | General-purpose registers: Used as source registers. |
| reg2 | General-purpose registers: Used mainly as destination registers. Also used as source register in some instructions. |
| reg3 | General-purpose registers: Used mainly to store the remainders of division results and the higher 32 bits of multiplication results. |
| bit#3 | 3-bit data for specifying the bit number |
| immX | X bit immediate data |
| dispX | X bit displacement data |
| regID | System register number |
| vector | 5-bit data that specifies the trap vector (00H to 1FH) |
| cccc | 4-bit data that shows the condition codes |
| sp | Stack pointer (r3) |
| ep | Element pointer (r30) |
| listX | X item register list |

### (2) Register symbols used to describe opcodes

| Register Symbol | Explanation |
|---|---|
| R | 1-bit data of a code that specifies reg1 or regID |
| r | 1-bit data of the code that specifies reg2 |
| w | 1-bit data of the code that specifies reg3 |
| d | 1-bit displacement data |
| I | 1-bit immediate data (indicates the higher bits of immediate data) |
| i | 1-bit immediate data |
| cccc | 4-bit data that shows the condition codes |
| CCCC | 4-bit data that shows the condition codes of Bcond instruction |
| bbb | 3-bit data for specifying the bit number |
| L | 1-bit data that specifies a program register in the register list |

### (3) Register symbols used in operations

| Register Symbol | Explanation |
|---|---|
| ← | Input for |
| GR [ ] | General-purpose register |
| SR [ ] | System register |
| zero-extend (n) | Expand n with zeros until word length. |
| sign-extend (n) | Expand n with signs until word length. |
| load-memory (a, b) | Read size b data from address a. |
| store-memory (a, b, c) | Write data b into address a in size c. |
| load-memory-bit (a, b) | Read bit b of address a. |
| store-memory-bit (a, b, c) | Write c to bit b of address a. |
| saturated (n) | Execute saturated processing of n (n is a 2's complement).<br>If, as a result of calculations,<br>n ≥ 7FFFFFFFH, let it be 7FFFFFFFH.<br>n ≤ 80000000H, let it be 80000000H. |
| result | Reflects the results in a flag. |
| Byte | Byte (8 bits) |
| Halfword | Halfword (16 bits) |
| Word | Word (32 bits) |
| + | Addition |
| – | Subtraction |
| ll | Bit concatenation |
| × | Multiplication |
| ÷ | Division |
| % | Remainder from division results |
| AND | Logical product |
| OR | Logical sum |
| XOR | Exclusive OR |
| NOT | Logical negation |
| logically shift left by | Logical shift left |
| logically shift right by | Logical shift right |
| arithmetically shift right by | Arithmetic shift right |

### (4) Register symbols used in execution clock

| Register Symbol | Explanation |
|---|---|
| i | If executing another instruction immediately after executing the first instruction (issue). |
| r | If repeating execution of the same instruction immediately after executing the first instruction (repeat). |
| l | If using the results of instruction execution in the instruction immediately after the execution (latency). |

### (5) Register symbols used in flag operations

| Identifier | Explanation |
|---|---|
| (Blank) | No change |
| 0 | Clear to 0 |
| X | Set or cleared in accordance with the results. |
| R | Previously saved values are restored. |

### (6) Condition codes

| Condition Name (cond) | Condition Code (cccc) | Condition Formula | Explanation |
|---|---|---|---|
| V | 0 0 0 0 | OV = 1 | Overflow |
| NV | 1 0 0 0 | OV = 0 | No overflow |
| C/L | 0 0 0 1 | CY = 1 | Carry<br>Lower (Less than) |
| NC/NL | 1 0 0 1 | CY = 0 | No carry<br>Not lower (Greater than or equal) |
| Z | 0 0 1 0 | Z = 1 | Zero |
| NZ | 1 0 1 0 | Z = 0 | Not zero |
| NH | 0 0 1 1 | (CY or Z) = 1 | Not higher (Less than or equal) |
| H | 1 0 1 1 | (CY or Z) = 0 | Higher (Greater than) |
| S/N | 0 1 0 0 | S = 1 | Negative |
| NS/P | 1 1 0 0 | S = 0 | Positive |
| T | 0 1 0 1 | – | Always (Unconditional) |
| SA | 1 1 0 1 | SAT = 1 | Saturated |
| LT | 0 1 1 0 | (S xor OV) = 1 | Less than signed |
| GE | 1 1 1 0 | (S xor OV) = 0 | Greater than or equal signed |
| LE | 0 1 1 1 | ((S xor OV) or Z) = 1 | Less than or equal signed |
| GT | 1 1 1 1 | ((S xor OV) or Z) = 0 | Greater than signed |

## B.2 Instruction Set (in Alphabetical Order)

(1/6)

| Mnemonic | Operand | Opcode | Operation | | Execution Clock i | Execution Clock r | Execution Clock l | CY | OV | S | Z | SAT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ADD | reg1,reg2 | rrrrr001110RRRRR | GR[reg2]←GR[reg2]+GR[reg1] | | 1 | 1 | 1 | × | × | × | × | |
| | imm5,reg2 | rrrrr010010iiiii | GR[reg2]←GR[reg2]+sign-extend(imm5) | | 1 | 1 | 1 | × | × | × | × | |
| ADDI | imm16,reg1,reg2 | rrrrr110000RRRRR<br>iiiiiiiiiiiiiiii | GR[reg2]←GR[reg1]+sign-extend(imm16) | | 1 | 1 | 1 | × | × | × | × | |
| AND | reg1,reg2 | rrrrr001010RRRRR | GR[reg2]←GR[reg2]AND GR[reg1] | | 1 | 1 | 1 | | 0 | × | × | |
| ANDI | imm16,reg1,reg2 | rrrrr110110RRRRR<br>iiiiiiiiiiiiiiii | GR[reg2]←GR[reg1]AND zero-extend(imm16) | | 1 | 1 | 1 | | 0 | × | × | |
| Bcond | disp9 | ddddd1011dddcccc<br>**Note 1** | if conditions are satisfied<br>then PC←PC+sign-extend(disp9) | When conditions are satisfied | 2 Note 2 | 2 Note 2 | 2 Note 2 | | | | | |
| | | | | When conditions are not satisfied | 1 | 1 | 1 | | | | | |
| BSH | reg2,reg3 | rrrrr11111100000<br>wwwww01101000010 | GR[reg3]←GR[reg2] (23 : 16) ll GR[reg2] (31 : 24) ll GR[reg2] (7 : 0) ll GR[reg2] (15 : 8) | | 1 | 1 | 1 | × | 0 | × | × | |
| BSW | reg2,reg3 | rrrrr11111100000<br>wwwww01101000000 | GR[reg3]←GR[reg2] (7 : 0) ll GR[reg2] (15 : 8) ll GR [reg2] (23 : 16) ll GR[reg2] (31 : 24) | | 1 | 1 | 1 | × | 0 | × | × | |
| CALLT | imm6 | 0000001000iiiiii | CTPC←PC+2(return PC)<br>CTPSW←PSW<br>adr←CTBP+zero-extend(imm6 logically shift left by 1)<br>PC←CTBP+zero-extend(Load-memory(adr,Halfword)) | | 4 | 4 | 4 | | | | | |
| CLR1 | bit#3,disp16[reg1] | 10bbb111110RRRRR<br>dddddddddddddddd | adr←GR[reg1]+sign-extend(disp16)<br>Z flag←Not(Load-memory-bit(adr,bit#3))<br>Store-memory-bit(adr,bit#3,0) | | 3 Note 3 | 3 Note 3 | 3 Note 3 | | | | × | |
| | reg2,[reg1] | rrrrr111111RRRRR<br>0000000011100100 | adr←GR[reg1]<br>Z flag←Not(Load-memory-bit(adr,reg2))<br>Store-memory-bit(adr,reg2,0) | | 3 Note 3 | 3 Note 3 | 3 Note 3 | | | | × | |
| CMOV | cccc,imm5,reg2,reg3 | rrrrr111111iiiii<br>wwwww011000cccc0 | if conditions are satisfied<br>then GR[reg3]←sign-extended(imm5)<br>else GR[reg3]←GR[reg2] | | 1 | 1 | 1 | | | | | |
| | cccc,reg1,reg2,reg3 | rrrrr111111RRRR<br>wwwww011001cccc0 | if conditions are satisfied<br>then GR[reg3]←GR[reg1]<br>else GR[reg3]←GR[reg2] | | 1 | 1 | 1 | | | | | |
| CMP | reg1,reg2 | rrrrr001111RRRRR | result←GR[reg2]–GR[reg1] | | 1 | 1 | 1 | × | × | × | × | |
| | imm5,reg2 | rrrrr010011iiiii | result←GR[reg2]–sign-extend(imm5) | | 1 | 1 | 1 | × | × | × | × | |
| CTRET | | 0000011111100000<br>0000000101000100 | PC←CTPC<br>PSW←CTPSW | | 3 | 3 | 3 | R | R | R | R | R |
| DBRET | | 0000011111100000<br>0000000101000110 | PC←DBPC<br>PSW←DBPSW | | 3 | 3 | 3 | R | R | R | R | R |

(2/6)

| Mnemonic | Operand | Opcode | Operation | Execution Clock | | | Flags | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | i | r | l | CY | OV | S | Z | SAT |
| DBTRAP | | 1111100001000000 | DBPC←PC+2 (restored PC)<br>DBPSW←PSW<br>PSW.NP←1<br>PSW.EP←1<br>PSW.ID←1<br>PC←00000060H | 3 | 3 | 3 | | | | | |
| DI | | 0000011111100000<br>0000000101100000 | PSW.ID←1 | 1 | 1 | 1 | | | | | |
| DISPOSE | imm5,list12 | 0 0 0 0 0 1 1 0 0 1 i i i i i L<br>L L L L L L L L L L L 0 0 0 0 0 | sp←sp+zero-extend(imm5 logically shift left by 2)<br>GR[reg in list12]←Load-memory(sp,Word)<br>sp←sp+4<br>repeat 2 steps above until all regs in list12 is loaded | n+1 Note 4 | n+1 Note 4 | n+1 Note 4 | | | | | |
| | imm5,list12,[reg1] | 0 0 0 0 0 1 1 0 0 1 i i i i i L<br>L L L L L L L L L L L R R R R R<br>**Note 5** | sp←sp+zero-extend(imm5 logically shift left by 2)<br>GR[reg in list12]←Load-memory(sp,Word)<br>sp←sp+4<br>repeat 2 steps above until all regs in list12 is loaded<br>PC←GR[reg1] | n+3 Note 4 | n+3 Note 4 | n+3 Note 4 | | | | | |
| DIV | reg1,reg2,reg3 | rrrrr111111RRRRR<br>wwwww01011000000 | GR[reg2]←GR[reg2]÷GR[reg1]<br>GR[reg3]←GR[reg2]%GR[reg1] | 35 | 35 | 35 | | × | × | × | |
| DIVH | reg1,reg2 | rrrrr000010RRRRR | GR[reg2]←GR[reg2]÷GR[reg1] Note 6 | 35 | 35 | 35 | | × | × | × | |
| | reg1,reg2,reg3 | rrrrr111111RRRRR<br>wwwww01010000000 | GR[reg2]←GR[reg2]÷GR[reg1] Note 6<br>GR[reg3]←GR[reg2]%GR[reg1] | 35 | 35 | 35 | | × | × | × | |
| DIVHU | reg1,reg2,reg3 | rrrrr111111RRRRR<br>wwwww01010000010 | GR[reg2]←GR[reg2]÷GR[reg1] Note 6<br>GR[reg3]←GR[reg2]%GR[reg1] | 34 | 34 | 34 | | × | × | × | |
| DIVU | reg1,reg2,reg3 | rrrrr111111RRRRR<br>wwwww01011000010 | GR[reg2]←GR[reg2]÷GR[reg1]<br>GR[reg3]←GR[reg2]%GR[reg1] | 34 | 34 | 34 | | × | × | × | |
| EI | | 1000011111100000<br>0000000101100000 | PSW.ID←0 | 1 | 1 | 1 | | | | | |
| HALT | | 0000011111100000<br>0000000100100000 | Stop | 1 | 1 | 1 | | | | | |
| HSW | reg2,reg3 | rrrrr11111100000<br>wwwww01101000100 | GR[reg3]←GR[reg2](15 : 0) ll GR[reg2] (31 : 16) | 1 | 1 | 1 | × | 0 | × | × | |
| JARL | disp22,reg2 | rrrrr11110ddddddd<br>ddddddddddddddd0<br>**Note 7** | GR[reg2]←PC+4<br>PC←PC+sign-extend(disp22) | 2 | 2 | 2 | | | | | |
| JMP | [reg1] | 00000000011RRRRR | PC←GR[reg1] | 3 | 3 | 3 | | | | | |
| JR | disp22 | 0000011110ddddddd<br>ddddddddddddddd0<br>**Note 7** | PC←PC+sign-extend(disp22) | 2 | 2 | 2 | | | | | |
| LD.B | disp16[reg1],reg2 | rrrrr111000RRRRR<br>ddddddddddddddddd | adr←GR[reg1]+sign-extend(disp16)<br>GR[reg2]←sign-extend(Load-memory(adr,Byte)) | 1 | 1 | Note 11 | | | | | |
| LD.BU | disp16[reg1],reg2 | rrrrr11110bRRRRR<br>ddddddddddddddd1<br>**Notes 8, 10** | adr←GR[reg1]+sign-extend(disp16)<br>GR[reg2]←zero-extend(Load-memory(adr,Byte)) | 1 | 1 | Note 11 | | | | | |

(3/6)

| Mnemonic | Operand | Opcode | Operation | | Execution Clock | | | Flags | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | i | r | l | CY | OV | S | Z | SAT |
| LD.H | disp16[reg1],reg2 | rrrrr111001RRRRR<br>ddddddddddddddd0<br>**Note 8** | adr←GR[reg1]+sign-extend(disp16)<br>GR[reg2]←sign-extend(Load-memory(adr,Halfword)) | | 1 | 1 | Note 11 | | | | | |
| LDSR | reg2,regID | rrrrr111111RRRRR<br>0000000000100000<br>**Note 12** | SR[regID]←GR[reg2] | Other than regID = PSW | 1 | 1 | 1 | | | | | |
| | | | | regID = PSW | 1 | 1 | 1 | × | × | × | × | × |
| LD.HU | disp16[reg1],reg2 | rrrrr111111RRRRR<br>ddddddddddddddd1<br>**Note 8** | adr←GR[reg1]+sign-exend(disp16)<br>GR[reg2]←zero-extend(Load-memory(adr,Halfword) | | 1 | 1 | Note 11 | | | | | |
| LD.W | disp16[reg1],reg2 | rrrrr111001RRRRR<br>ddddddddddddddd1<br>**Note 8** | adr←GR[reg1]+sign-exend(disp16)<br>GR[reg2]←Load-memory(adr,Word) | | 1 | 1 | Note 11 | | | | | |
| MOV | reg1,reg2 | rrrrr000000RRRRR | GR[reg2]←GR[reg1] | | 1 | 1 | 1 | | | | | |
| | imm5,reg2 | rrrrr010000iiiii | GR[reg2]←sign-extend(imm5) | | 1 | 1 | 1 | | | | | |
| | imm32,reg1 | 00000110001RRRRR<br>iiiiiiiiiiiiiiii<br>llllllllllllllll | GR[reg1]←imm32 | | 2 | 2 | 2 | | | | | |
| MOVEA | imm16,reg1,reg2 | rrrrr110001RRRRR<br>iiiiiiiiiiiiiiii | GR[reg2]←GR[reg1]+sign-extend(imm16) | | 1 | 1 | 1 | | | | | |
| MOVHI | imm16,reg1,reg2 | rrrrr110010RRRRR<br>iiiiiiiiiiiiiiii | GR[reg2]←GR[reg1]+(imm16 ll $0^{16}$) | | 1 | 1 | 1 | | | | | |
| MUL | reg1,reg2,reg3 | rrrrr111111RRRRR<br>wwwww01000100000 | GR[reg3] ll GR[reg2]←GR[reg2]xGR[reg1]<br>**Note 14** | | 1 | 4 | 5 | | | | | |
| | imm9,reg2,reg3 | rrrrr111111iiiii<br>wwwww01001IIII00<br>**Note 13** | GR[reg3] ll GR[reg2]←GR[reg2]xsign-extend(imm9) | | 1 | 4 | 5 | | | | | |
| MULH | reg1,reg2 | rrrrr000111RRRRR | GR[reg2]←GR[reg2]Note 6xGR[reg1]Note 6 | | 1 | 1 | 2 | | | | | |
| | imm5,reg2 | rrrrr010111iiiii | GR[reg2]←GR[reg2]Note 6xsign-extend(imm5) | | 1 | 1 | 2 | | | | | |
| MULHI | imm16,reg1,reg2 | rrrrr110111RRRRR<br>iiiiiiiiiiiiiiii | GR[reg2]←GR[reg1]Note 6ximm16 | | 1 | 1 | 2 | | | | | |
| MULU | reg1,reg2,reg3 | rrrrr111111RRRRR<br>wwwww01000100010 | GR[reg3] ll GR[reg2]←GR[reg2]xGR[reg1]<br>**Note 14** | | 1 | 4 | 5 | | | | | |
| | imm9,reg2,reg3 | rrrrr111111iiiii<br>wwwww01001IIII10<br>**Note 13** | GR[reg3] ll GR[reg2]←GR[reg2]xzero-extend(imm9) | | 1 | 4 | 5 | | | | | |
| NOP | | 0000000000000000 | Pass at least one clock cycle doing nothing. | | 1 | 1 | 1 | | | | | |
| NOT | reg1,reg2 | rrrrr000001RRRRR | GR[reg2]←NOT(GR[reg1]) | | 1 | 1 | 1 | | 0 | × | × | |
| NOT1 | bit#3,disp16[reg1] | 01bbb111110RRRRR<br>dddddddddddddddd | adr←GR[reg1]+sign-extend(disp16)<br>Z flag←Not(Load-memory-bit(adr,bit#3))<br>Store-memory-bit(adr,bit#3,Z flag) | | 3 Note 3 | 3 Note 3 | 3 Note 3 | | | | × | |
| | reg2,[reg1] | rrrrr111111RRRRR<br>0000000011100010 | adr←GR[reg1]<br>Z flag←Not(Load-memory-bit(adr,reg2))<br>Store-memory-bit(adr,reg2,Z flag) | | 3 Note 3 | 3 Note 3 | 3 Note 3 | | | | × | |

(4/6)

| Mnemonic | Operand | Opcode | Operation | Execution Clock | | | Flags | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | i | r | l | CY | OV | S | Z | SAT |
| OR | reg1,reg2 | rrrrr001000RRRRR | GR[reg2]←GR[reg2]OR GR[reg1] | 1 | 1 | 1 | | 0 | × | × | |
| ORI | imm16,reg1,reg2 | rrrrr110100RRRRR<br>iiiiiiiiiiiiiiii | GR[reg2]←GR[reg1]OR zero-extend(imm16) | 1 | 1 | 1 | | 0 | × | × | |
| PREPARE | list12,imm5 | 0000011110iiiiiL<br>LLLLLLLLLLL00001 | Store-memory(sp–4,GR[reg in list12],Word)<br>sp←sp–4<br>repeat 1 step above until all regs in list12 is stored<br>sp←sp-zero-extend(imm5) | n+1<br>Note 4 | n+1<br>Note 4 | n+1<br>Note 4 | | | | | |
| | list12,imm5,<br>sp/imm[Note 15] | 0000011110iiiiiL<br>LLLLLLLLLLLff011<br>imm16/imm32<br>**Note 16** | Store-memory(sp–4,GR[reg in list12],Word)<br>sp←sp+4<br>repeat 1 step above until all regs in list12 is stored<br>sp←sp-zero-extend (imm5)<br>ep←sp/imm | n+2<br>Note 4<br>Note 17 | n+2<br>Note 4<br>Note 17 | n+2<br>Note 4<br>Note 17 | | | | | |
| RETI | | 0000011111100000<br>0000000101000000 | if PSW.EP=1<br>then PC ←EIPC<br>PSW ←EIPSW<br>else if PSW.NP=1<br>then PC ←FEPC<br>PSW ←FEPSW<br>else PC ←EIPC<br>PSW ←EIPSW | 3 | 3 | 3 | R | R | R | R | R |
| SAR | reg1,reg2 | rrrrr111111RRRRR<br>0000000010100000 | GR[reg2]←GR[reg2]arithmetically shift right<br>by GR[reg1] | 1 | 1 | 1 | × | 0 | × | × | |
| | imm5,reg2 | rrrrr010101iiiii | GR[reg2]←GR[reg2]arithmetically shift right<br>by zero-extend (imm5) | 1 | 1 | 1 | × | 0 | × | × | |
| SASF | cccc,reg2 | rrrrr1111110cccc<br>0000001000000000 | if conditions are satisfied<br>then GR[reg2]←(GR[reg2]Logically shift left by 1)<br>OR 00000001H<br>else GR[reg2]←(GR[reg2]Logically shift left by 1)<br>OR 00000000H | 1 | 1 | 1 | | | | | |
| SATADD | reg1,reg2 | rrrrr000110RRRRR | GR[reg2]←saturated(GR[reg2]+GR[reg1]) | 1 | 1 | 1 | × | × | × | × | × |
| | imm5,reg2 | rrrrr010001iiiii | GR[reg2]←saturated(GR[reg2]+sign-extend(imm5)) | 1 | 1 | 1 | × | × | × | × | × |
| SATSUB | reg1,reg2 | rrrrr000101RRRRR | GR[reg2]←saturated(GR[reg2]–GR[reg1]) | 1 | 1 | 1 | × | × | × | × | × |
| SATSUBI | imm16,reg1,reg2 | rrrrr110011RRRRR<br>iiiiiiiiiiiiiiii | GR[reg2]←saturated(GR[reg1]–sign-extend(imm16)) | 1 | 1 | 1 | × | × | × | × | × |
| SATSUBR | reg1,reg2 | rrrrr000100RRRRR | GR[reg2]←saturated(GR[reg1]–GR[reg2]) | 1 | 1 | 1 | × | × | × | × | × |
| SETF | cccc,reg2 | rrrrr1111110cccc<br>0000000000000000 | If conditions are satisfied<br>then GR[reg2]←00000001H<br>else GR[reg2]←00000000H | 1 | 1 | 1 | | | | | |

(5/6)

| Mnemonic | Operand | Opcode | Operation | Execution Clock | | | Flags | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | i | r | l | CY | OV | S | Z | SAT |
| SET1 | bit#3,disp16[reg1] | 00bbb111110RRRRR<br>dddddddddddddddd | adr←GR[reg1]+sign-extend(disp16)<br>Z flag←Not (Load-memory-bit(adr,bit#3))<br>Store-memory-bit(adr,bit#3,1) | 3<br>Note 3 | 3<br>Note 3 | 3<br>Note 3 | | | | × | |
| | reg2,[reg1] | rrrrr111111RRRRR<br>0000000011100000 | adr←GR[reg1]<br>Z flag←Not(Load-memory-bit(adr,reg2))<br>Store-memory-bit(adr,reg2,1) | 3<br>Note 3 | 3<br>Note 3 | 3<br>Note 3 | | | | × | |
| SHL | reg1,reg2 | rrrrr111111RRRRR<br>0000000011000000 | GR[reg2]←GR[reg2] logically shift left by GR[reg1] | 1 | 1 | 1 | × | 0 | × | × | |
| | imm5,reg2 | rrrrr010110iiiii | GR[reg2]←GR[reg2] logically shift left<br>by zero-extend(imm5) | 1 | 1 | 1 | × | 0 | × | × | |
| SHR | reg1,reg2 | rrrrr111111RRRRR<br>0000000010000000 | GR[reg2]←GR[reg2] logically shift right by GR[reg1] | 1 | 1 | 1 | × | 0 | × | × | |
| | imm5,reg2 | rrrrr010100iiiii | GR[reg2]←GR[reg2] logically shift right<br>by zero-extend(imm5) | 1 | 1 | 1 | × | 0 | × | × | |
| SLD.B | disp7[ep],reg2 | rrrrr0110ddddddd | adr←ep+zero-extend(disp7)<br>GR[reg2]←sign-extend(Load-memory(adr,Byte)) | 1 | 1 | Note 9 | | | | | |
| SLD.BU | disp4[ep],reg2 | rrrrr0000110dddd<br>**Note 18** | adr←ep+zero-extend(disp4)<br>GR[reg2]←zero-extend(Load-memory(adr,Byte)) | 1 | 1 | Note 9 | | | | | |
| SLD.H | disp8[ep],reg2 | rrrrr1000ddddddd<br>**Note 19** | adr←ep+zero-extend(disp8)<br>GR[reg2]←sign-extend(Load-memory(adr,Halfword)) | 1 | 1 | Note 9 | | | | | |
| SLD.HU | disp5[ep],reg2 | rrrrr0000111dddd<br>**Notes 18, 20** | adr←ep+zero-extend(disp5)<br>GR[reg2]←zero-extend(Load-memory(adr,Halfword)) | 1 | 1 | Note 9 | | | | | |
| SLD.W | disp8[ep],reg2 | rrrrr1010ddddddd0<br>**Note 21** | adr←ep+zero-extend(disp8)<br>GR[reg2]←Load-memory(adr,Word) | 1 | 1 | Note 9 | | | | | |
| SST.B | reg2,disp7[ep] | rrrrr0111ddddddd | adr←ep+zero-extend(disp7)<br>Store-memory(adr,GR[reg2],Byte) | 1 | 1 | 1 | | | | | |
| SST.H | reg2,disp8[ep] | rrrrr1001ddddddd<br>**Note 19** | adr←ep+zero-extend(disp8)<br>Store-memory(adr,GR[reg2],Halfword) | 1 | 1 | 1 | | | | | |
| SST.W | reg2,disp8[ep] | rrrrr1010ddddddd1<br>**Note 21** | adr←ep+zero-extend(disp8)<br>Store-memory(adr,GR[reg2],Word) | 1 | 1 | 1 | | | | | |
| ST.B | reg2,disp16[reg1] | rrrrr111010RRRRR<br>dddddddddddddddd | adr←GR[reg1]+sign-extend(disp16)<br>Store-memory(adr,GR[reg2],Byte) | 1 | 1 | 1 | | | | | |
| ST.H | reg2,disp16[reg1] | rrrrr111011RRRRR<br>ddddddddddddddd0<br>**Note 8** | adr←GR[reg1]+sign-extend(disp16)<br>Store-memory (adr,GR[reg2], Halfword) | 1 | 1 | 1 | | | | | |
| ST.W | reg2,disp16[reg1] | rrrrr111011RRRRR<br>ddddddddddddddd1<br>**Note 8** | adr←GR[reg1]+sign-extend(disp16)<br>Store-memory (adr,GR[reg2], Word) | 1 | 1 | 1 | | | | | |
| STSR | regID,reg2 | rrrrr111111RRRRR<br>0000000001000000 | GR[reg2]←SR[regID] | 1 | 1 | 1 | | | | | |

(6/6)

| Mnemonic | Operand | Opcode | Operation | Execution Clock | | | Flags | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | i | r | l | CY | OV | S | Z | SAT |
| SUB | reg1,reg2 | rrrrr001101RRRRR | GR[reg2]←GR[reg2]–GR[reg1] | 1 | 1 | 1 | × | × | × | × | |
| SUBR | reg1,reg2 | rrrrr001100RRRRR | GR[reg2]←GR[reg1]–GR[reg2] | 1 | 1 | 1 | × | × | × | × | |
| SWITCH | reg1 | 00000000010RRRRR | adr←(PC+2) + (GR [reg1] logically shift left by 1)<br>PC←(PC+2) + (sign-extend<br>(Load-memory (adr,Halfword))<br>logically shift left by 1 | 5 | 5 | 5 | | | | | |
| SXB | reg1 | 00000000101RRRRR | GR[reg1]←sign-extend<br>(GR[reg1] (7 : 0)) | 1 | 1 | 1 | | | | | |
| SXH | reg1 | 00000000111RRRRR | GR[reg1]←sign-extend<br>(GR[reg1] (15 : 0)) | 1 | 1 | 1 | | | | | |
| TRAP | vector | 0 0 0 0 0 1 1 1 1 1 1 i i i i i<br>0000000100000000 | EIPC ←PC+4 (Restored PC)<br>EIPSW ←PSW<br>ECR.EICC ←Interrupt code<br>PSW.EP ←1<br>PSW.ID ←1<br>PC ←00000040H<br>(when vector is 00H to 0FH)<br>00000050H<br>(when vector is 10H to 1FH) | 3 | 3 | 3 | | | | | |
| TST | reg1,reg2 | rrrrr001011RRRRR | result←GR[reg2] AND GR[reg1] | 1 | 1 | 1 | | 0 | × | × | |
| TST1 | bit#3,disp16[reg1] | 11bbb111110RRRRR<br>ddddddddddddddddd | adr←GR[reg1]+sign-extend(disp16)<br>Z flag←Not (Load-memory-bit (adr,bit#3)) | 3<br>Note 3 | 3<br>Note 3 | 3<br>Note 3 | | | | × | |
| | reg2, [reg1] | rrrrr111111RRRRR<br>0000000011100110 | adr←GR[reg1]<br>Z flag←Not (Load-memory-bit (adr,reg2)) | 3<br>Note 3 | 3<br>Note 3 | 3<br>Note 3 | | | | × | |
| XOR | reg1,reg2 | rrrrr001001RRRRR | GR[reg2]←GR[reg2] XOR GR[reg1] | 1 | 1 | 1 | | 0 | × | × | |
| XORI | imm16,reg1,reg2 | rrrrr110101RRRRR<br>i i i i i i i i i i i i i i i i | GR[reg2]←GR[reg1] XOR zero-extend (imm16) | 1 | 1 | 1 | | 0 | × | × | |
| ZXB | reg1 | 00000000100RRRRR | GR[reg1]←zero-extend (GR[reg1] (7 : 0)) | 1 | 1 | 1 | | | | | |
| ZXH | reg1 | 00000000110RRRRR | GR[reg1]←zero-extend (GR[reg1] (15 : 0)) | 1 | 1 | 1 | | | | | |

**Notes 1.** ddddddddd: Higher 8 bits of disp9.

**2.** 3 if there is an instruction that rewrites the contents of the PSW immediately before.

**3.** If there is no wait state (3 + the number of read access wait states).

**4.** n is the total number of list12 load registers. (According to the number of wait states. Also, if there are no wait states, n is the total number of list12 registers. If n = 0, same operation as when n = 1)

**5.** RRRRR: other than 00000.

**6.** The lower halfword data only are valid.

**7.** ddddddddddddddddddddd: The higher 21 bits of disp22.

**8.** ddddddddddddddd: The higher 15 bits of disp16.

**9.** According to the number of wait states (1 if there are no wait states).

**10.** b: bit 0 of disp16.

**11.** According to the number of wait states (2 if there are no wait states).

**Notes 12.** In this instruction, for convenience of mnemonic description, the source register is made reg2, but the reg1 field is used in the opcode. Therefore, the meaning of register specification in the mnemonic description and in the opcode differs from other instructions.

rrrrr = regID specification

RRRRR = reg2 specification

**13.** iiiii: Lower 5 bits of imm9.

IIII: Higher 4 bits of imm9.

**14.** Do not specify the same register for general-purpose registers reg1 and reg3.

**15.** sp/imm: specified by bits 19 and 20 of the sub-opcode.

**16.** ff = 00: Load sp in ep.

01: Load sign expanded 16-bit immediate data (bits 47 to 32) in ep.

10: Load 16-bit logically left shifted 16-bit immediate data (bits 47 to 32) in ep.

11: Load 32-bit immediate data (bits 63 to 32) in ep.

**17.** If imm = imm32, n + 3 clocks.

**18.** rrrrr: Other than 00000.

**19.** ddddddd: Higher 7 bits of disp8.

**20.** dddd: Higher 4 bits of disp5.

**21.** dddddd: Higher 6 bits of disp8.

# APPENDIX C REGISTER INDEX

(1/7)

| Symbol | Name | Unit | Page |
|---|---|---|---|
| ADCR | A/D conversion result register | ADC | 398 |
| ADCRH | A/D conversion result register H | ADC | 398 |
| ADIC | Interrupt control register | INTC | 592 |
| ADM | A/D converter mode register | ADC | 394 |
| ADS | Analog input channel specification register | ADC | 397 |
| ADTC0 | Automatic data transfer address count register 0 | CSIA | 483 |
| ADTI0 | Automatic data transfer interval specification register 0 | CSIA | 489 |
| ADTP0 | Automatic data transfer address point specification register 0 | CSIA | 487 |
| ASICL0 | LIN operation control register 0 | UART | 426 |
| ASIF0 | Asynchronous serial interface transmit status register 0 | UART | 424 |
| ASIF1 | Asynchronous serial interface transmit status register 1 | UART | 424 |
| ASIM0 | Asynchronous serial interface mode register 0 | UART | 421 |
| ASIM1 | Asynchronous serial interface mode register 1 | UART | 421 |
| ASIS0 | Asynchronous serial interface status register 0 | UART | 423 |
| ASIS1 | Asynchronous serial interface status register 1 | UART | 423 |
| AWC | Address wait control register | BCU | 162 |
| BCC | Bus cycle control register | BCU | 163 |
| BRGC0 | Baud rate generator control register 0 | UART | 447 |
| BRGC1 | Baud rate generator control register 1 | UART | 447 |
| BRGCA0 | Divisor selection register 0 | CSIA | 487 |
| BRGIC | Interrupt control register | INTC | 592 |
| BSC | Bus size configuration register | BCU | 152 |
| CCLS | CPU operation clock status register | CG | 178 |
| CKSR0 | Clock select register 0 | UART | 446 |
| CKSR1 | Clock select register 1 | UART | 446 |
| CLM | Clock monitor mode register | CLM | 650 |
| CMP00 | 8-bit timer H compare register 00 | TMH | 340 |
| CMP01 | 8-bit timer H compare register 01 | TMH | 340 |
| CMP10 | 8-bit timer H compare register 10 | TMH | 340 |
| CMP11 | 8-bit timer H compare register 11 | TMH | 340 |
| CORAD0 | Correction address register 0 | ROMC | 669 |
| CORAD0H | Correction address register 0H | ROMC | 669 |
| CORAD0L | Correction address register 0L | ROMC | 669 |
| CORAD1 | Correction address register 1 | ROMC | 669 |
| CORAD1H | Correction address register 1H | ROMC | 669 |
| CORAD1L | Correction address register 1L | ROMC | 669 |
| CORAD2 | Correction address register 2 | ROMC | 669 |
| CORAD2H | Correction address register 2H | ROMC | 669 |
| CORAD2L | Correction address register 2L | ROMC | 669 |
| CORAD3 | Correction address register 3 | ROMC | 669 |
| CORAD3H | Correction address register 3H | ROMC | 669 |

(2/7)

| Symbol | Name | Unit | Page |
|---|---|---|---|
| CORAD3L | Correction address register 3L | ROMC | 669 |
| CORCN | Correction control register | ROMC | 670 |
| CR000 | 16-bit timer capture/compare register 000 | TM0 | 272 |
| CR001 | 16-bit timer capture/compare register 001 | TM0 | 274 |
| CR010 | 16-bit timer capture/compare register 010 | TM0 | 272 |
| CR011 | 16-bit timer capture/compare register 011 | TM0 | 274 |
| CR5 | 16-bit timer compare register 5 | TM5 | 321 |
| CR50 | 8-bit timer compare register 50 | TM5 | 321 |
| CR51 | 8-bit timer compare register 51 | TM5 | 321 |
| CRC00 | Capture/compare control register 00 | TM0 | 277 |
| CRC01 | Capture/compare control register 01 | TM0 | 277 |
| CSI0IC0 | Interrupt control register | INTC | 592 |
| CSI0IC1 | Interrupt control register | INTC | 592 |
| CSIA0B0 | CSIA0 buffer RAMn (n = 0 to F) | CSIA | 489 |
| CSIA0B0H | CSIA0 buffer RAMnH (n = 0 to F) | CSIA | 489 |
| CSIA0B0L | CSIA0 buffer RAMnL (n = 0 to F) | CSIA | 489 |
| CSIAIC0 | Interrupt control register | INTC | 592 |
| CSIC0 | Clocked serial interface clock selection register 0 | CSI0 | 459 |
| CSIC1 | Clocked serial interface clock selection register 1 | CSI0 | 459 |
| CSIM00 | Clocked serial interface mode register 00 | CSI0 | 457 |
| CSIM01 | Clocked serial interface mode register 01 | CSI0 | 457 |
| CSIMA0 | Serial operation mode specification register 0 | CSIA | 484 |
| CSIS0 | Serial status register 0 | CSIA | 485 |
| CSIT0 | Serial trigger register 0 | CSIA | 486 |
| CTBP | CALLT base pointer | CPU | 53 |
| CTPC | CALLT execution status saving register | CPU | 52 |
| CTPSW | CALLT execution status saving register | CPU | 52 |
| DBPC | Exception/debug trap status saving register | CPU | 53 |
| DBPSW | Exception/debug trap status saving register | CPU | 53 |
| DWC0 | Data wait control register 0 | BCU | 160 |
| ECR | Interrupt source register | CPU | 50 |
| EIPC | Interrupt status saving register | CPU | 49 |
| EIPSW | Interrupt status saving register | CPU | 49 |
| FEPC | NMI status saving register | CPU | 50 |
| FEPSW | NMI status saving register | CPU | 50 |
| IIC0 | IIC shift register 0 | $I^2C$ | 528 |
| IICC0 | IIC control register 0 | $I^2C$ | 516 |
| IICCL0 | IIC clock selection register 0 | $I^2C$ | 526 |
| IICF0 | IIC flag register 0 | $I^2C$ | 524 |
| IICIC0 | Interrupt control register | INTC | 592 |
| IICS0 | IIC status register 0 | $I^2C$ | 521 |
| IICX0 | IIC function expansion register 0 | $I^2C$ | 527 |
| IMR0 | Interrupt mask register 0 | INTC | 594 |
| IMR0H | Interrupt mask register 0H | INTC | 594 |

(3/7)

| Symbol | Name | Unit | Page |
|---|---|---|---|
| IMR0L | Interrupt mask register 0L | INTC | 594 |
| IMR1 | Interrupt mask register 1 | INTC | 594 |
| IMR1H | Interrupt mask register 1H | INTC | 594 |
| IMR1L | Interrupt mask register 1L | INTC | 594 |
| IMR3 | Interrupt mask register 3 | INTC | 594 |
| IMR3L | Interrupt mask register 3L | INTC | 594 |
| INTF0 | External interrupt falling edge specification register 0 | INTC | 601 |
| INTF3 | External interrupt falling edge specification register 3 | INTC | 602 |
| INTF9H | External interrupt falling edge specification register 9H | INTC | 603 |
| INTR0 | External interrupt rising edge specification register 0 | INTC | 601 |
| INTR3 | External interrupt rising edge specification register 3 | INTC | 602 |
| INTR9H | External interrupt rising edge specification register 9H | INTC | 603 |
| ISPR | In-service priority register | INTC | 595 |
| KRIC | Interrupt control register | INTC | 592 |
| KRM | Key return mode register | KR | 616 |
| LVIIC | Interrupt control register | INTC | 592 |
| LVIM | Low-voltage detection register | LVI | 660 |
| LVIS | Low-voltage detection level selection register | LVI | 661 |
| NFC | Digital noise elimination control register | INTC | 599 |
| OSTS | Oscillation stabilization time selection register | Standby | 622 |
| P0 | Port 0 register | Port | 88 |
| P0NFC | TIP00 noise elimination control register | TMP | 267 |
| P1NFC | TIP01 noise elimination control register | TMP | 267 |
| P3 | Port 3 register | Port | 91 |
| P3H | Port 3 register H | Port | 91 |
| P3L | Port 3 register L | Port | 91 |
| P4 | Port 4 register | Port | 96 |
| P5 | Port 5 register | Port | 98 |
| P7 | Port 7 register | Port | 101 |
| P9 | Port 9 register | Port | 103 |
| P9H | Port 9 register H | Port | 103 |
| P9L | Port 9 register L | Port | 103 |
| PC | Program counter | CPU | 47 |
| PCC | Processor clock control register | CG | 174 |
| PCM | Port CM register | Port | 108 |
| PCS | Port CS register | Port | 110 |
| PCT | Port CT register | Port | 112 |
| PDL | Port DL register | Port | 115 |
| PDLH | Port DL register H | Port | 115 |
| PDLL | Port DL register L | Port | 115 |
| PF3H | Port 3 function register H | Port | 93 |
| PF4 | Port 4 function register | Port | 97 |
| PF5 | Port 5 function register | Port | 99 |
| PF9H | Port 9 function register H | Port | 105 |

(4/7)

| Symbol | Name | Unit | Page |
|---|---|---|---|
| PFC3 | Port 3 function control register | Port | 93 |
| PFC5 | Port 5 function control register | Port | 100 |
| PFC9 | Port 9 function control register | Port | 106 |
| PFC9H | Port 9 function control register H | Port | 106 |
| PFC9L | Port 9 function control register L | Port | 106 |
| PFCE3 | Port 3 function control expansion register | Port | 93 |
| PFM | Power fail comparison mode register | ADC | 400 |
| PFT | Power fail comparison threshold register | ADC | 400 |
| PIC0 | Interrupt control register | INTC | 592 |
| PIC1 | Interrupt control register | INTC | 592 |
| PIC2 | Interrupt control register | INTC | 592 |
| PIC3 | Interrupt control register | INTC | 592 |
| PIC4 | Interrupt control register | INTC | 592 |
| PIC5 | Interrupt control register | INTC | 592 |
| PIC6 | Interrupt control register | INTC | 592 |
| PIC7 | Interrupt control register | INTC | 592 |
| PLLCTL | PLL control register | CG | 180, 389 |
| PM0 | Port 0 mode register | Port | 88 |
| PM3 | Port 3 mode register | Port | 91 |
| PM3H | Port 3 mode register H | Port | 91 |
| PM3L | Port 3 mode register L | Port | 91 |
| PM4 | Port 4 mode register | Port | 96 |
| PM5 | Port 5 mode register | Port | 98 |
| PM9 | Port 9 mode register | Port | 103 |
| PM9H | Port 9 mode register H | Port | 103 |
| PM9L | Port 9 mode register L | Port | 103 |
| PMC0 | Port 0 mode control register | Port | 89 |
| PMC3 | Port 3 mode control register | Port | 92 |
| PMC3H | Port 3 mode control register H | Port | 92 |
| PMC3L | Port 3 mode control register L | Port | 92 |
| PMC4 | Port 4 mode control register | Port | 96 |
| PMC5 | Port 5 mode control register | Port | 99 |
| PMC9 | Port 9 mode control register | Port | 104 |
| PMC9H | Port 9 mode control register H | Port | 104 |
| PMC9L | Port 9 mode control register L | Port | 104 |
| PMCCM | Port CM mode control register | Port | 109 |
| PMCCS | Port CS mode control register | Port | 111 |
| PMCCT | Port CT mode control register | Port | 113 |
| PMCDL | Port DL mode control register | Port | 116 |
| PMCDLH | Port DL mode control register H | Port | 116 |
| PMCDLL | Port DL mode control register L | Port | 116 |
| PMCM | Port CM mode register | Port | 108 |
| PMCS | Port CS mode register | Port | 110 |

(5/7)

| Symbol | Name | Unit | Page |
|---|---|---|---|
| PMCT | Port CT mode register | Port | 112 |
| PMDL | Port DL mode register | Port | 115 |
| PMDLH | Port DL mode register H | Port | 115 |
| PMDLL | Port DL mode register L | Port | 115 |
| PRCMD | Command register | CPU | 76 |
| PRM00 | Prescaler mode register 00 | TM0 | 280 |
| PRM01 | Prescaler mode register 01 | TM0 | 280 |
| PRSCM | Interval timer BRG compare register | CG | 364 |
| PRSM | Interval timer BRG mode register | CG | 363 |
| PSC | Power save control register | Standby | 620 |
| PSMR | Power save mode register | Standby | 621 |
| PSW | Program status word | CPU | 51 |
| PU0 | Pull-up resistor option register 0 | Port | 89 |
| PU3 | Pull-up resistor option register 3 | Port | 94 |
| PU4 | Pull-up resistor option register 4 | Port | 97 |
| PU5 | Pull-up resistor option register 5 | Port | 100 |
| PU9 | Pull-up resistor option register 9 | Port | 107 |
| PU9H | Pull-up resistor option register 9H | Port | 107 |
| PU9L | Pull-up resistor option register 9L | Port | 107 |
| PUCM | Pull-up resistor option register CM | Port | 109 |
| PUCS | Pull-up resistor option register CS | Port | 111 |
| PUCT | Pull-up resistor option register CT | Port | 113 |
| PUDL | Pull-up resistor option register DL | Port | 116 |
| PUDLL | Pull-up resistor option register DLL | Port | 116 |
| PUDLH | Pull-up resistor option register DLH | Port | 116 |
| RCM | Ring-OSC mode register | CG | 178, 651 |
| r0 to r31 | General-purpose registers | CPU | 47 |
| RESF | Reset source flag register | Reset | 637 |
| RNZC | Reset noise elimination control register | Reset | 640 |
| RTBH0 | Real-time output buffer register H0 | RTP | 383 |
| RTBL0 | Real-time output buffer register L0 | RTP | 383 |
| RTPC0 | Real-time output port control register 0 | RTP | 385 |
| RTPM0 | Real-time output port mode register 0 | RTP | 384 |
| RXB0 | Receive buffer register 0 | UART | 425 |
| RXB1 | Receive buffer register 1 | UART | 425 |
| SELCNT0 | Selector operation control register 0 | UART | 427 |
| SELCNT1 | Selector operation control register 1 | TM0 | 281 |
| SIO00 | Serial I/O shift register 0 | CSI0 | 464 |
| SIO00L | Serial I/O shift register 0L | CSI0 | 464 |
| SIO01 | Serial I/O shift register 1 | CSI0 | 464 |
| SIO01L | Serial I/O shift register 1L | CSI0 | 464 |
| SIOA0 | Serial I/O shift register A0 | CSIA | 483 |
| SIRB0 | Clocked serial interface receive buffer register 0 | CSI0 | 460 |

(6/7)

| Symbol | Name | Unit | Page |
|---|---|---|---|
| SIRB0L | Clocked serial interface receive buffer register 0L | CSI0 | 460 |
| SIRB1 | Clocked serial interface receive buffer register 1 | CSI0 | 460 |
| SIRB1L | Clocked serial interface receive buffer register 1L | CSI0 | 460 |
| SIRBE0 | Clocked serial interface read-only receive buffer register 0 | CSI0 | 461 |
| SIRBE0L | Clocked serial interface read-only receive buffer register 0L | CSI0 | 461 |
| SIRBE1 | Clocked serial interface read-only receive buffer register 1 | CSI0 | 461 |
| SIRBE1L | Clocked serial interface read-only receive buffer register 1L | CSI0 | 461 |
| SOTB0 | Clocked serial interface transmit buffer register 0 | CSI0 | 462 |
| SOTB0L | Clocked serial interface transmit buffer register 0L | CSI0 | 462 |
| SOTB1 | Clocked serial interface transmit buffer register 1 | CSI0 | 462 |
| SOTB1L | Clocked serial interface transmit buffer register 1L | CSI0 | 462 |
| SOTBF0 | Clocked serial interface initial transmit buffer register 0 | CSI0 | 463 |
| SOTBF0L | Clocked serial interface initial transmit buffer register 0L | CSI0 | 463 |
| SOTBF1 | Clocked serial interface initial transmit buffer register 1 | CSI0 | 463 |
| SOTBF1L | Clocked serial interface initial transmit buffer register 1L | CSI0 | 463 |
| SREIC0 | Interrupt control register | INTC | 592 |
| SREIC1 | Interrupt control register | INTC | 592 |
| SRIC0 | Interrupt control register | INTC | 592 |
| SRIC1 | Interrupt control register | INTC | 592 |
| STIC0 | Interrupt control register | INTC | 592 |
| STIC1 | Interrupt control register | INTC | 592 |
| SVA0 | Slave address register 0 | $I^2C$ | 528 |
| SYS | System status register | CPU | 77 |
| TCL50 | Timer clock selection register 50 | TM5 | 322 |
| TCL51 | Timer clock selection register 51 | TM5 | 322 |
| TM00 | 16-bit timer counter 00 | TM0 | 272 |
| TM01 | 16-bit timer counter 01 | TM0 | 272 |
| TM0IC00 | Interrupt control register | INTC | 592 |
| TM0IC01 | Interrupt control register | INTC | 592 |
| TM0IC10 | Interrupt control register | INTC | 592 |
| TM0IC11 | Interrupt control register | INTC | 592 |
| TM5 | 16-bit timer counter 5 | TM5 | 335 |
| TM50 | 8-bit timer counter 50 | TM5 | 320 |
| TM51 | 8-bit timer counter 51 | TM5 | 320 |
| TM5IC0 | Interrupt control register | INTC | 592 |
| TM5IC1 | Interrupt control register | INTC | 592 |
| TMC00 | 16-bit timer mode control register 00 | TM0 | 275 |
| TMC01 | 16-bit timer mode control register 01 | TM0 | 275 |
| TMC50 | 8-bit timer mode control register 50 | TM5 | 323 |
| TMC51 | 8-bit timer mode control register 51 | TM5 | 323 |
| TMCYC0 | 8-bit timer H carrier control register 0 | TMH | 344 |
| TMCYC1 | 8-bit timer H carrier control register 1 | TMH | 344 |
| TMHIC0 | Interrupt control register | INTC | 592 |
| TMHIC1 | Interrupt control register | INTC | 592 |

(7/7)

| Symbol | Name | Unit | Page |
|---|---|---|---|
| TMHMD0 | 8-bit timer H mode register 0 | TMH | 341 |
| TMHMD1 | 8-bit timer H mode register 1 | TMH | 341 |
| TOC00 | 16-bit timer output control register 00 | TM0 | 278 |
| TOC01 | 16-bit timer output control register 01 | TM0 | 278 |
| TP0CCIC0 | Interrupt control register | INTC | 592 |
| TP0CCIC1 | Interrupt control register | INTC | 592 |
| TP0CCR0 | TMP0 capture/compare register 0 | TMP | 191 |
| TP0CCR1 | TMP0 capture/compare register 1 | TMP | 193 |
| TP0CNT | TMP0 counter read buffer register | TMP | 195 |
| TP0CTL0 | TMP0 control register 0 | TMP | 185 |
| TP0CTL1 | TMP0 control register 1 | TMP | 186 |
| TP0IOC0 | TMP0 I/O control register 0 | TMP | 187 |
| TP0IOC1 | TMP0 I/O control register 1 | TMP | 188 |
| TP0IOC2 | TMP0 I/O control register 2 | TMP | 189 |
| TP0OPT0 | TMP0 option register 0 | TMP | 190 |
| TP0OVIC | Interrupt control register | INTC | 592 |
| TXB0 | Transmit buffer register 0 | UART | 425 |
| TXB1 | Transmit buffer register 1 | UART | 425 |
| VSWC | System wait control register | CPU | 78 |
| WDCS | Watchdog timer clock selection register | WDT | 374 |
| WDT1IC | Interrupt control register | INTC | 592 |
| WDTE | Watchdog timer enable register | WDT | 380 |
| WDTM1 | Watchdog timer mode register 1 | WDT | 375, 597 |
| WDTM2 | Watchdog timer mode register 2 | WDT | 379 |
| WTIC | Interrupt control register | INTC | 592 |
| WTIIC | Interrupt control register | INTC | 592 |
| WTM | Watch timer operation mode register | WT | 367 |

# X-ON Electronics

Largest Supplier of Electrical and Electronic Components

***Click to view similar products for*** 8-bit Microcontrollers - MCU ***category:***

***Click to view products by*** Renesas ***manufacturer:***

Other Similar products are found below :

CY8C20524-12PVXIT CY8C28433-24PVXIT MB95F012KPFT-G-SNE2 MB95F013KPMC-G-SNE2 MB95F263KPF-G-SNE2 MB95F264KPFT-G-SNE2 MB95F398KPMC-G-SNE2 MB95F478KPMC2-G-SNE2 MB95F562KPF-G-SNE2 MB95F564KPF-G-SNE2 MB95F634KPMC-G-SNE2 MB95F636KWQN-G-SNE1 MB95F696KPMC-G-SNE2 MB95F698KPMC1-G-SNE2 MB95F698KPMC2-G-SNE2 MB95F698KPMC-G-SNE2 MB95F818KPMC1-G-SNE2 MC908JK1ECDWER MC9S08PA32AVLD MC9S08PT60AVLD R5F1076CMSPV0 R5F5631ECDFBV0 C8051F389-B-GQ C8051F392-A-GMR ISD-ES1600_USB_PROG 901015X SC705C8AE0VFBE STM8TL53G4U6 PIC16F877-04/P-B R5F10Y17ASP#30 CY8C3MFIDOCK-125 403708R MB95F354EPF-G-SNE2 MB95F564KPFT-G-SNE2 MB95F564KWQN-G-SNE1 MB95F636KP-G-SH-SNE2 MB95F636KPMC-G-SNE2 MB95F694KPMC-G-SNE2 MB95F778JPMC1-G-SNE2 MB95F818KPMC-G-SNE2 MC908QY8CDWER MC9S08PT16AVLD MC9S08PT32AVLH MC9S08PT60AVLC MC9S08PT60AVLH C8051F500-IQR LC87F0G08AUJA-AH CP8361BT STM8S207C6T3 CG8421AF